IBBT: Informed Batch Belief Trees for Motion Planning Under Uncertainty

Dongliang Zheng[1] and Panagiotis Tsiotras[2]

***Abstract*— In this work, we propose the Informed Batch Belief Trees (IBBT) algorithm for motion planning under motion and sensing uncertainties. The original stochastic motion planning problem is divided into a *deterministic* motion planning problem and a graph search problem. First, we solve the deterministic planning problem using Rapidly-exploring Random Graph (RRG) to construct a nominal trajectory graph. Then, an informed cost-to-go heuristic for the original problem is computed based on the nominal trajectory graph. Finally, we grow a belief tree by searching the graph using the proposed heuristic. IBBT interleaves batch state sampling, nominal trajectory graph construction, heuristic computing, and searching over the graph to find belief space motion plans. IBBT is an anytime, incremental algorithm. With an increasing number of batches of samples added to the graph, the algorithm finds improved plans. IBBT is efficient by reusing results between sequential iterations. The belief tree search is an ordered search guided by an informed heuristic. We test IBBT in different planning environments. Our numerical investigation confirms that IBBT finds non-trivial motion plans and is faster compared with previous similar methods.**

I. Introduction

For safe and reliable autonomous robot operation in real-world environments, consideration of various uncertainties becomes necessary. These uncertainties may arise from an inaccurate motion model, actuation or sensor noise, partial sensing, or the presence of other agents operating in the same environment. In this paper, we study the safe motion planning problem for robot systems with nontrivial dynamics, motion uncertainty, and state-dependent measurement uncertainty in an environment with non-convex obstacles.

Planning under uncertainty is referred to as belief space planning (BSP), where the state of the robot is characterized by a probability distribution function (pdf) over all possible states. This pdf is commonly referred to as the *belief* or information state [1], [2]. A BSP problem can be formulated as a partially observable Markov decision process (POMDP) problem [3]. Solving POMDPs for continuous state, control, and observation spaces, is, however, intractable. Existing methods based on discretization are resolution-limited [4], [5]. Optimization over the entire discretized belief space to find a path is computationally expensive and does not scale well to large-scale problems.

This work has been supported by NSF award IIS-2008695 and by Ford through the Georgia Tech/Ford Alliance program.

[1]Dongliang Zheng is a Ph.D. student with the School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA. Email: dzheng@gatech.edu

[2]Panagiotis Tsiotras is a Professor with the School of Aerospace Engineering and Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA 30332, USA. Email: tsiotras@gatech.edu

Planning in infinite-dimensional distributional (e.g., belief) spaces can become more tractable by using sampling-based methods [6]. For example, belief roadmap methods [7] build a belief roadmap to reduce estimation uncertainty; the rapidly-exploring random belief trees (RRBT) algorithm [8] has been proposed to grow a tree in the belief space. Owing to their advantages in avoiding local minima, dealing with nonconvex obstacles and high-dimensional state spaces, along with their anytime property, sampling-based methods have gained increased attention in the robotics community [9], [10], [11], [12], [13], [14].

Robot safety under uncertainty can be also formulated as a chance-constrained optimization problem [15], [16], [17]. In addition to minimizing the cost function, one also wants the robot not to collide with obstacles, with high probability. By approximating the chance constraints as deterministic constraints, references [15], [16], [17] solve the problem using an optimization-based framework. However, those approaches lack scalability with respect to problem complexity [18], and the explicit representation of the obstacles is usually required.

In this paper, we focus on sampling-based approaches similar to [8], [9], [19]. One challenge of sampling-based algorithms for planning under uncertainty is the lack of the optimal substructure property, which has been discussed in [8], [20], [21]. The lack of the optimal substructure property is further explained by the lack of total ordering on paths based on cost. Specifically, it is not enough to only minimize the usual cost function — explicitly finding paths that reduce the uncertainty of the robot is also important (see Figure 1(a)). The RRBT algorithm proposed in [8] overcomes the lack of optimal substructure property by introducing a partial ordering of belief nodes and by keeping all non-dominated nodes in the belief tree. Note that without this partial ordering, the methods in [9], [10], [11], [19] may not be able to find a solution, even if one exists. Minimizing the cost and checking the chance constraints can only guarantee that the existing paths in the tree satisfy the chance constraints. Without searching for paths that explicitly reduce state uncertainty, it may be difficult for future paths to satisfy the chance constraints.

In this paper, we propose the Informed Batch Belief Tree (IBBT) algorithm, which improves over the RRBT algorithm with the introduction of *batch sampling* and *ordered graph search guided by an informed heuristic*. Firstly, IBBT uses the partial ordering of belief nodes as in [8]. Compared to [9], [10], [11], [19], IBBT is able to find sophisticated plans that visit and revisit the information-rich region to gain

information. Secondly, RRBT uses unordered search like RRT* while IBBT uses batch sampling and ordered search. RRBT adds one sample each time to the graph randomly. As shown in [22] and [23], ordered searches such as FMT* and BIT* perform better than RRT*. Thirdly, RRBT only uses the cost-to-come to guide the belief tree search while IBBT introduces a cost-to-go heuristic and uses the total path cost heuristic for informed belief tree search. After adding a sample, RRBT performs an exhaustive graph search. Thus all non-dominated belief nodes are added to the belief tree. With batch sampling and informed graph search, IBBT avoids adding unnecessary belief nodes. Thus, IBBT is able to find the initial solution in a shorter time and has better cost-time performance compared to RRBT.

II. Related Works

In [7], the problem of finding the minimum estimation uncertainty path for a robot from a starting position to a goal is studied by building a roadmap. In [8] and [24], it was noted that the true *a priori* probability distribution of the state should be used for motion planning instead of assuming maximum likelihood observations [7], [25]. A linear-quadratic Gaussian (LQG) controller along with the RRT algorithm [26] was used for motion planning in [24]. To achieve asymptotic optimality, the authors in [8] incrementally construct a graph and search over the graph to find all non-dominated belief nodes. Given the current graph, the Pareto frontier of belief nodes at each vertex is saved, where the Pareto frontier is defined by considering both the path cost and the node uncertainty.

In [10] it is shown that high-frequency replanning is able to better react to uncertainty during plan execution. Monte Carlo simulation and importance sampling are used in [11] to compute the collision probability. Moving obstacles are considered in [18]. In [27], state dependence of the collision probability is considered and incorporated with chance-constrained RRT* [9], [28]. In [29], a roadmap search method is proposed to deal with localization uncertainty; however, solutions for which the robot needs to revisit a position to gain information are ruled out. Distributionally robust RRT is proposed in [19], [30], where moment-based ambiguity sets of distributions are used to enforce chance constraints instead of assuming Gaussian distributions. Similarly, a moment-based approach that considers non-Gaussian state distributions is studied in [31]. In [32], the Wasserstein distance is used as a metric for Gaussian belief space planning. The algorithm is compared with RRBT. However, from the simulation results, RRBT usually finds better (lower cost) plans and thus has a better convergence performance.

Other works that are not based on sampling-based methods formulate the chance-constrained motion planning problem as an optimization problem [15], [16], [17]. In those methods, the explicit representation of the obstacles is usually required. The obstacles may be represented by convex constraints or polynomial constraints. The chance constraints are then approximated as deterministic constraints and the optimization problem is solved by convex [16] or nonlinear programming [17]. Differential dynamic programming has also been used to solve motion planning under uncertainty [2], [33], [34]. These algorithms find a locally optimal trajectory in the neighborhood of a given reference trajectory. The algorithms iteratively linearize the system dynamics along the reference trajectory and solve an LQG problem to find the next reference trajectory.

III. Problem formulation

We consider the problem of planning for a robot with nontrivial dynamics, model uncertainty, measurement uncertainty from sensor noise, and obstacle constraints. The state-space $\mathscr{X}$ is decomposed into free space $\mathscr{X}_{\text{free}}$ and obstacle space $\mathscr{X}_{\text{obs}}$. The motion planning problem is given by

$$\underset{u_k}{\arg\min}\ \mathbb{E}\left[\sum_{k=0}^{N-1} J(x_k,u_k)\right], \tag{1}$$

$$\text{s.t.}\quad x_0 \sim \mathcal{N}(\bar{x}_s, P_0),\quad \bar{x}_N = \bar{x}_g, \tag{2}$$

$$P(x_k \in \mathscr{X}_{\text{obs}}) < \delta,\quad k=0,\cdots,N, \tag{3}$$

$$x_{k+1} = f(x_k,u_k,w_k),\quad k=0,\ldots,N-1, \tag{4}$$

$$y_k = h(x_k,v_k),\quad k=0,\ldots,N-1, \tag{5}$$

where (4) and (5) are the motion and sensing models, respectively. Furthermore, $x_k \in \mathbb{R}^{n_x}$ is the state, $u_k \in \mathbb{R}^{n_u}$ is the control input, and $y_k \in \mathbb{R}^{n_y}$ is the measurement at time step $k = 0,1,\ldots,N-1$, where the steps of the noise processes $w_k \in \mathbb{R}^{n_w}$ and $v_k \in \mathbb{R}^{n_y}$ are i.i.d standard Gaussian random vectors, respectively. We assume that $(w_k)_{k=0}^{N-1}$ and $(v_k)_{k=0}^{N-1}$ are independent. In (1), $J(x_k,u_k)$ is the cost at step k. We assume $J(x_k,u_k)$ is a general convex function with respect to x_k and u_k. For example, it can be a quadratic function given by $x_k^\top Q_k x_k + u_k^\top R_k u_k$, where Q_k and R_k are cost penalty parameters. Expression (2) is the boundary condition for the motion planning problem. The initial state x_0 is Gaussian distributed and the terminal mean $\bar{x}_N$ is fixed. Condition (3) is a chance constraint that enforces the safety of the robot. δ specifies the maximum probability that state x_k collides with obstacles for every time step k.

Similar to [8], the motion plan considered in this paper is formed by a nominal trajectory and a feedback controller. Specifically, we will use a `Connect` function that returns a nominal trajectory and the corresponding feedback gains between two states $\bar{x}^a$ and $\bar{x}^b$,

$$(\bar{X}^{a,b}, \bar{U}^{a,b}, K^{a,b}) = \texttt{Connect}(\bar{x}^a, \bar{x}^b), \tag{6}$$

where $\bar{X}^{a,b}$ and $\bar{U}^{a,b}$ are the sequences of the states and the controls respectively of the nominal trajectory, and $K^{a,b}$ is a sequence of the corresponding feedback control gains. The nominal trajectory can be obtained by solving a deterministic optimal control problem with boundary conditions $\bar{x}^a$ and $\bar{x}^b$, and system dynamics $\bar{x}_{k+1} = f(\bar{x}_k, \bar{u}_k, 0)$. The feedback gains can be computed using, for example, finite-time LQR design [20].

A Kalman filter is used for online state estimation, which gives the estimate[1] $\hat{x}_k$ of $(x_k - \bar{x}_k)$. Thus, the control at time

[1] Note non-standard notation.

k is given by

$$u_k = \bar{u}_k + K_k \hat{x}_k. \tag{7}$$

With the introduction of the `Connect` function, the optimal motion planning problem (1)-(5) is reformulated as one of finding the sequence of intermediate states $(\bar{x}^0, \bar{x}^1, \cdots, \bar{x}^\ell)$. The final control is given by

$$(u_k)_{k=0}^{N-1} = (\texttt{Connect}(\bar{x}^0, \bar{x}^1), \cdots, \texttt{Connect}(\bar{x}^{\ell-1}, \bar{x}^\ell)). \tag{8}$$

The remaining problem is to find the optimal sequence of intermediate states and enforce the chance constraints (3).

IV. Covariance Propagation

We assume that the system given by (4) and (5) is locally well approximated by its linearization along the nominal trajectory. This is a common assumption as the system will stay close to the nominal trajectory using the feedback controller [20], [21]. Define

$$\check{x}_k = x_k - \bar{x}_k, \tag{9a}$$
$$\check{u}_k = u_k - \bar{u}_k, \tag{9b}$$
$$\check{y}_k = y_k - h(\bar{x}_k, 0). \tag{9c}$$

By linearizing along $(\bar{x}_k, \bar{u}_k)$, the error dynamics is

$$\begin{aligned} \check{x}_k &= A_{k-1}\check{x}_{k-1} + B_{k-1}\check{u}_{k-1} + G_{k-1}w_{k-1}, \\ \check{y}_k &= C_k \check{x}_k + D_k v_k. \end{aligned} \tag{10}$$

We will consider this linear time-varying system hereafter. A Kalman filter is used for estimating $\check{x}_k$ and is given by

$$\hat{x}_k = \hat{x}_{k^-} + L_k(\check{y}_k - C_k \hat{x}_{k^-}), \tag{11}$$
$$\hat{x}_{k^-} = A_{k-1}\hat{x}_{k-1} + B_{k-1}\check{u}_{k-1}, \tag{12}$$

where,

$$L_k = \tilde{P}_{k^-} C_k^\mathsf{T} (C_k \tilde{P}_{k^-} C_k^\mathsf{T} + D_k D_k^\mathsf{T})^{-1}, \tag{13a}$$
$$\tilde{P}_k = (I - L_k C_k)\tilde{P}_{k^-}, \tag{13b}$$
$$\tilde{P}_{k^-} = A_{k-1}\tilde{P}_{k-1}A_{k-1}^\mathsf{T} + G_{k-1}G_{k-1}^\mathsf{T}, \tag{13c}$$

and L_k is the Kalman gain. The covariances of $\check{x}_k$, $\hat{x}_k$ and $\tilde{x}_k \triangleq \check{x}_k - \hat{x}_k$ are denoted as $P_k = \mathbb{E}[\check{x}_k \check{x}_k^\mathsf{T}]$, $\hat{P}_k = \mathbb{E}[\hat{x}_k \hat{x}_k^\mathsf{T}]$ and $\tilde{P}_k = \mathbb{E}[\tilde{x}_k \tilde{x}_k^\mathsf{T}]$, respectively. Note that the covariance of x_k is also given by P_k and the estimation error covariance $\tilde{P}_k$ is computed from (13b). From (10)-(12), it can be verified that $\mathbb{E}[\check{x}_k] = \mathbb{E}[\hat{x}_k] = \mathbb{E}[\hat{x}_{k^-}]$. Since $\mathbb{E}[\check{x}_0] = 0$, by choosing $\mathbb{E}[\hat{x}_0] = 0$, we have $\mathbb{E}[\hat{x}_k] = 0$ for $k = 0, \cdots, N$. Using (11) and (12) we also have that

$$\begin{aligned} \hat{P}_k &= \mathbb{E}[\hat{x}_k \hat{x}_k^\mathsf{T}] = \mathbb{E}[\hat{x}_{k^-} \hat{x}_{k^-}^\mathsf{T}] + L_k(C_k \tilde{P}_{k^-} C_k^\mathsf{T} + D_k D_k^\mathsf{T}) L_k^\mathsf{T} \\ &= (A_{k-1} + B_{k-1}K_{k-1})\hat{P}_{k-1}(A_{k-1} + B_{k-1}K_{k-1})^\mathsf{T} + L_k C_k \tilde{P}_{k^-}. \end{aligned} \tag{14}$$

Using the fact that $\mathbb{E}[\hat{x}_k \tilde{x}_k^\mathsf{T}] = 0$, it can be verified that $P_k = \hat{P}_k + \tilde{P}_k$. Thus, given the feedback gains K_k and the Kalman filter gain L_k, we can predict the covariances of the state estimation error and the state along the trajectory, which also provides the state distributions in the case of a Gaussian distribution.

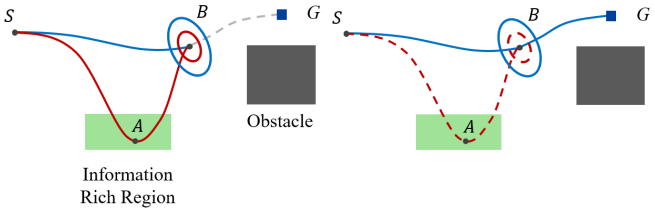


Fig. 1: (a) RRBT: Two paths reach the same point B. The red path detours to an information-rich region to reduce uncertainty. Both paths are explored and preserved in the belief tree in RRBT as it finds all non-dominated belief nodes. (b) IBBT avoids exploring unnecessary belief nodes. If the blue path $\overline{BG}$ satisfies the chance constraint, the whole blue path $\overline{SBG}$ satisfies the chance constraint and has a lower cost than the red path $\overline{SABG}$. The operation of finding more paths reaching B with less uncertainty (but a larger cost), including the red one, becomes redundant.

V. Informed Batch Belief Tree Algorithm

A. Motivation

The motivation of IBBT is shown in Figure 1. The ellipses show the covariance/uncertainty of the node. Two paths reach point B in Figure 1(a). The red path reaches B with a large cost but with low uncertainty. The blue path reaches B with a small cost but with high uncertainty. In this case, the blue path cannot dominate the red path, as it will incur a high probability of chance constraint violation for future segments of the path. Thus, in RRBT, both paths are preserved in the belief tree. More specifically, RRBT will find all non-dominated belief nodes by exhaustively searching the graph.

IBBT avoids exhaustive graph search and hence avoids adding unnecessary belief nodes. In Figure 1(b), if the blue path $\overline{BG}$ (starting anywhere inside the blue ellipse) satisfies the chance constraint, the blue path $\overline{SBG}$ will be the solution of the problem since it satisfies the chance constraints and has a lower cost than $\overline{SABG}$. The operation of searching the current graph to find more paths reaching B with less uncertainty (but a higher cost), including the red one, becomes redundant. Here, we assume that the cost of the nominal trajectory, $\sum_{k=0}^{N-1} J(\bar{x}_k, \bar{u}_k)$, makes most of the cost in (1). That is, for the path $\overline{BG}$, starting from the red ellipse and the blue ellipse will incur a similar cost. Reducing the uncertainty at node B is mainly for satisfying the chance constraint of the future trajectory. Such an assumption can also be found, for example, in [35].

RRBT performs an exhaustive search to find all non-dominated nodes whenever a vertex is added to the graph. Specifically, RRBT will spend a lot of effort finding nodes with low uncertainty but a high cost-to-come. Such nodes are only necessary if they are indeed part of the optimal path. If the blue path in Figure 1(b) is the solution, we do not need to search for other non-dominated nodes (red ellipse). However, since we do not know if the future blue path $\overline{BG}$ will satisfy the chance constraint or not, the red node may still be needed. Thus, IBBT searches the graph and adds belief nodes to the belief tree only when necessary. This is done by batch sampling and using an informed heuristic.

B. Nominal Trajectory Graph

The stochastic motion planning problem (1)-(5) is divided into a simpler deterministic planning problem and a belief tree search problem. The deterministic planning problem is given by

$$\arg\min_{\bar{u}_k} \sum_{k=0}^{N-1} J(\bar{x}_k, \bar{u}_k), \tag{15}$$

$$\text{s.t.} \quad \bar{x}_0 = \bar{x}_s, \ \bar{x}_N = \bar{x}_g, \tag{16}$$

$$\bar{x}_k \notin \mathscr{X}_{\text{obs}}, \ k = 0, \cdots, N, \tag{17}$$

$$\bar{x}_{k+1} = f(\bar{x}_k, \bar{u}_k, 0), \tag{18}$$

and is solved by the Rapidly-exploring Random Graph (RRG) algorithm [6]. RRG adds batches of samples and maintains a graph of nominal trajectories. Similarly, the PRM algorithm [36] may be used in place of RRG. The nominal trajectory graph is given by $G = (V, E)$. Where V is the vertex set and E is the edge set. Each vertex is state sampled in the state space of the robot. Each edge is a nominal trajectory from the `Connect` function introduced in Section III. An edge is added to the graph only if the corresponding nominal trajectory is obstacle-free.

C. IBBT

The Informed Batch Belief Tree algorithm repeatedly performs two main operations: It first builds a graph of nominal trajectories to explore the state space of the robot, and then it searches over this graph to grow a belief tree in the belief space. The IBBT algorithm is given by Algorithm 1 and Algorithm 2. Additional variables are needed to define a belief tree. A belief node n is defined by a state covariance $n.P$, an estimation error covariance $n.\tilde{P}$, a cost-to-come $n.c$, a heuristic cost-to-go $n.h$, and a parent node index $n.\text{parent}$. A vertex v is defined by a state $v.\bar{x}$, a set of belief nodes $v.N$, and a vertex cost $v.h$.

The graph search given by Algorithm 2 repeats two primitive procedures to grow a belief tree: *Belief node selection* which selects the best node in the belief queue for expansion; *Belief propagation* which propagates the selected belief node to its neighbor vertices to generate new belief nodes.

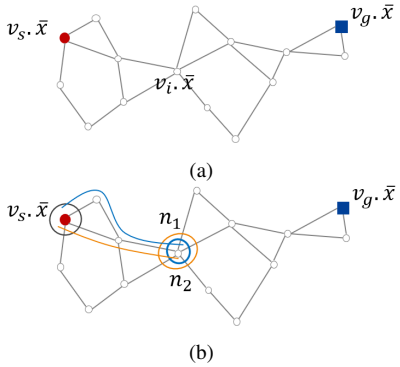


Fig. 2: (a) Nominal trajectory graph. Each edge is computed by solving a deterministic optimal control problem with edge cost given by (15). (b) Two belief nodes are shown at vertex v_i.

Algorithm 1: Informed Batch Belief Tree

$n.P \leftarrow P_0$; $n.\tilde{P} \leftarrow \tilde{P}_0$; $n.c \leftarrow 0$; $n.h \leftarrow \text{Inf}$;
  $n.\text{parent} \leftarrow \text{null}$; `// initialize belief n`
$v_s.N \leftarrow \{n\}$; $v_g.N \leftarrow \emptyset$;
$v_s.\bar{x} \leftarrow \bar{x}_s$; $v_g.\bar{x} \leftarrow \bar{x}_g$;
$v_s.h \leftarrow \text{Inf}$; $v_g.h \leftarrow 0$;
$V \leftarrow \{v_s, v_g\}$; $E \leftarrow \emptyset$; $G \leftarrow (V, E)$;
$Q \leftarrow \{n\}$; $Cost \leftarrow \text{Inf}$;
repeat
  $(G, V_{\text{new}}) = \texttt{RRG}(G, m)$;
  $G = \texttt{VI}(G)$;
  foreach $v_{\text{new}} \in V_{\text{new}}$ **do**
    foreach v_{neighbor} of v_{new} **do**
      $Q \leftarrow Q \cup v_{\text{neighbor}}.N$; `// update queue`
  `Prune`$(Q, Cost)$;
  $Q \leftarrow Q \cup v_g.N$;
  $(G, Q, \text{flag}) = \texttt{GraphSearch}(G, Q)$;
  if flag **then**
    $Cost = \min\{n.c | \forall n \in v_g.N\}$;
until Stop;
return G, flag;

Algorithm 2: Graph Search

flag $\leftarrow$ **False**;
while $Q \neq \emptyset$ **do**
  $n \leftarrow \texttt{Pop}(Q)$;
  if $v(n).\bar{x} = \bar{x}_g$ **then**
    flag $\leftarrow$ **True** ; `// solution found`
    return G, Q, flag;
  foreach v_{neighbor} of $v(n)$ **do**
    $n_{\text{new}} \leftarrow \texttt{Propagate}(e_{\text{neighbor}}, n)$;
    succ, $G \leftarrow \texttt{Append}(G, v_{\text{neighbor}}, n_{\text{new}})$;
    if succ **then**
      $Q \leftarrow Q \cup \{n_{\text{new}}\}$;
return G, Q, flag;

The metric to rank the belief nodes in the belief queue is vital for efficient graph search. Here, we propose an informed and admissible heuristic for efficient search. Using the results of the nominal trajectory graph, we compute the cost-to-go for all vertices. A nominal trajectory graph is shown in Figure 2(a). Every edge in the graph is computed by solving a deterministic optimal control problem with edge cost given by (15). We compute the cost-to-go $v_i.h$ using value iteration for every vertex in the graph. $v_i.h$ is the true cost-to-go for the nominal trajectory graph. Since $J(x_k, u_k)$ is a convex function, via Jensen's inequality [37], we have $J(\bar{x}_k, \bar{u}_k) \leq \mathbb{E}[J(x_k, u_k)]$. Thus, $\sum_{k=0}^{N-1} J(\bar{x}_k, \bar{u}_k) \leq \mathbb{E}\left[\sum_{k=0}^{N-1} J(x_k, u_k)\right]$, $\sum_{k=0}^{N-1} J(\bar{x}_k, \bar{u}_k)$ is an underestimate of the actual cost, and $v_i.h$ is an admissible cost-to-go heuristic for the belief tree search problem. On the other hand,

$\sum_{k=0}^{N-1} J(\bar{x}_k, \bar{u}_k)$ is close to $\mathbb{E}\left[\sum_{k=0}^{N-1} J(x_k, u_k)\right]$ compared to other ad hoc heuristics, which makes $v_i.h$ an informed heuristic. We call our algorithm 'informed' based on this informed heuristic. We first solve a simpler problem and use the results of the simpler problem to guide the solving of the original problem. This is different from [23] which defines an informed set for geometric planning.

The nodes in the belief node queue are ranked based on the total heuristic cost $n.f = n.c + n.h$. All belief nodes at the same vertex have the same heuristic cost-to-go and $n.h = v.h$. In Figure 2(b), two belief nodes n_1, n_2 are shown at vertex v_i. Their total heuristic costs are $n_1.f = n_1.c + v_i.h$ and $n_2.f = n_2.c + v_i.h$, respectively. The partial ordering of belief nodes is defined as follows. Let n_a and n_b be two belief nodes of the same vertex v. We use $n_a < n_b$ to denote that belief node n_b is dominated by n_a. $n_a < n_b$ is true if

$$(n_a.f < n_b.f) \wedge (n_a.P < n_b.P) \wedge (n_a.\tilde{P} < n_b.\tilde{P}). \qquad (19)$$

In this case, n_a is better than n_b since it traces back a path that reaches v with a smaller cost and less uncertainty compared with n_b. Next, we summarize some primitive procedures used in the IBBT algorithm.

RRG: The `RRG` procedure adds m new samples to the current nominal trajectory graph and forms a new denser graph. The m samples constitute one batch. RRG-D returns the updated graph and the newly added vertex set V_{new}.

Pop: `Pop`(Q) selects the best belief node in terms of the lowest cost $n.f$ from belief queue Q and removes it from Q.

Propagate: The `Propagate` procedure implements three operations: covariance propagation, chance constraint evaluation, and cost calculation. `Propagate`(e,n) performs the covariance propagation using (13a)-(14). It takes an edge e and a belief node n at the starting vertex of the edge as inputs. Chance constraints are evaluated using the state covariance P_k along the edge. We use Monte Carlo samples to approximate the probability of collision [14]. If there are no chance constraint violations, a new belief n_{new} is returned, which is the final belief at the end vertex of the edge. Otherwise, the procedure returns no belief. The cost-to-come of n_{new} is the sum of $n.c$ and the cost of edge e by applying the controller (7) associated with e.

Append Belief: The function `Append`(G, v, n_{new}) decides if the new belief n_{new} should be added to vertex v or not. If n_{new} is not dominated by any existing belief nodes in $v.N$, n_{new} is added to $v.N$. Note that adding n_{new} means extending the current belief tree such that n_{new} becomes a leaf node of the current belief tree. Next, we also check if any existing belief node in $v.N$ is dominated by n_{new}. If an existing belief is dominated, its descendant and the node itself are pruned.

Prune Node Queue: The function `Prune`$(Q, Cost)$ removes nodes in Q whose total heuristic cost is greater than $Cost$. $Cost$ is the cost of the current solution found.

Value Iteration: The function `VI`(G) computes the cost-to-go for all vertices in G using value iteration. The value iteration is done using the nominal trajectory graph. For vertices whose cost-to-go values are computed in the last iteration (before calling this function), their values are reused for initialization for faster convergence.

In Algorithm 1, Line 1-5 initializes the graph and the belief tree. The initial condition is given by the starting state $\bar{x}_s$, state covariance P_0, and estimation error covariance $\tilde{P}_0$. The goal state is $\bar{x}_g$. In Line 6, queue Q is initialized with the initial node n, and the cost of the current solution is set as infinity. In Line 8, `RRG` adds m new samples to the nominal trajectory graph. V_{new} is the set of newly added vertices. Based on the nominal trajectory graph, cost-to-go for all vertices in G is computed using value iteration (Line 9). Line 10-12 update the belief node queue after batch sampling. For every vertex that has an outgoing edge towards v_{new}, all the belief nodes at that vertex are added to the queue.

In Algorithm 2, the belief n is propagated outwards to all the neighbor vertices of $v(n)$ to grow the belief tree in Line 7-11. $v(n)$ refers to the vertex associated with n. v_{neighbor} is a neighbor of $v(n)$ when there is an edge e_{neighbor} from $v(n)$ to v_{neighbor} in the graph. The new belief n_{new} is added to the $v_{\text{neighbor}}.N$ and Q if the belief tree extension is successful. Then, n is marked as the parent node of n_{new}. Note that each belief node traces back a unique path from the initial belief node. For every belief node in the belief tree, we already found a feasible path (i.e., it satisfies the chance constraints) to this node. Algorithm 2 terminates when the belief node at $\bar{x}_g$ is selected for expansion (Line 4-6, Algorithm 2) or Q is empty. In the first case, the best solution is found. In the second case, no solution exists given the current graph.

VI. Experimental Results

In this section, we test the IBBT algorithm for different motion planning problems and compare the results with the RRBT algorithm [8] and the Monte Carlo Motion Planning (MCMP) algorithm [11]. MCMP solves a deterministic motion planning problem with inflated obstacles to compute an approximately optimal trajectory using FMT* [22], computes the collision probability of this trajectory using Monte Carlo simulations, adjusts the inflation factor of the obstacles, and repeats the process until a satisfactory solution is found or iteration limit reached. In our implementation, we used the chance constraint parameter $\delta = 0.1$ for all examples. In IBBT, the number of samples for each batch is $m = 20$. The parameters for the MCMP algorithm are $I_{\max} = 4$, $I_{\min} = 0$, $r = 10$. We also terminate the algorithm if $I_{\max} - I_{\min} < 0.05$. Please refer to [11] for more details.

Two environment models shown in Figure 3 are considered. The first model shown in the first two figures consists of rectangle obstacles and information-rich regions. The blue regions are the information-rich regions. When the robot is in this region, the measurement noise is small. Specifically, the measurement matrix $D_k = 0.01 I_4$ when the robot is in an information-rich region, otherwise $D_k = I_4$. Note that this makes the measurement model (10) state-dependent.

The second model shown in the last two figures of Figure 3 consists circle obstacles and has landmarks (shown as black stars) for robot localization. The robot observes a landmark

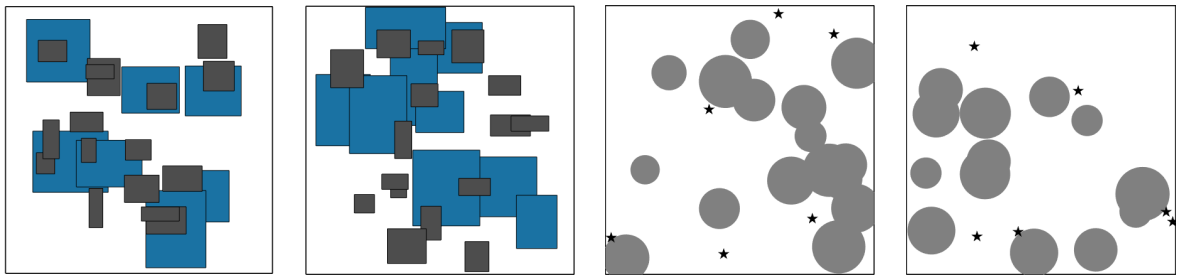


Fig. 3: Different planning environments

TABLE I: Comparison of IBBT, RRBT, and MCMP.

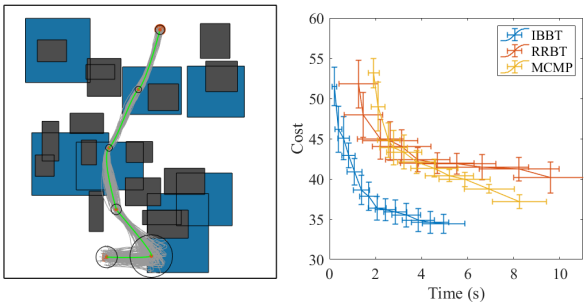
	First solution						Improved solution					
	Time			Cost			Time			Cost		
	IBBT	RRBT	MCMP	IBBT	RRBT	MCMP	IBBT	RRBT	MCMP	IBBT	RRBT	MCMP
DI-IR	**0.3240**	0.6677	0.9402	**34.10**	39.76	34.92	**0.9817**	1.9497	1.1623	**28.91**	35.83	32.15
DI-LM	**0.3681**	1.0513	1.5729	**31.98**	37.36	32.61	**1.4326**	4.0112	1.8093	**27.01**	34.11	28.50
Dubins-IR	**0.8753**	2.4415	2.6501	**28.18**	28.22	29.22	**2.3075**	7.0539	5.2650	**21.50**	23.88	22.96

DI-IR: Double integrator with information-rich regions. DI-LM: Double integrator with landmarks.

only if it is in the line of sight (the line between the robot and the landmark is obstacle-free). Each observed landmark provides a noisy position measurement of the robot. The robot receives more accurate position measurements when it is closer to the landmarks. Let L be the observed landmark set when the robot is at state x. The number of observed landmarks is ℓ. Let the Euclidean distance between the position of the robot and the j^{th} landmarks $L_j \in L$ be given by d_j. Then, the j^{th} position measurement corresponding to landmark L_j is ${}^j y = [x^{(1)} \; x^{(2)}]^\top + \eta_p d_j^2 v_p, \quad j = 1, 2, \cdots, \ell$, where parameter $\eta_p = 0.01$, and v_p is a two-dimensional standard Gaussian random vector. Thus, the total measurement vector y is a 2ℓ dimensional vector, composed of ℓ position measurements. The dimension of the measurement varies and depends on the location of the robot.

For each environment, the obstacles are generated randomly. The number, location, and size (width, length, radius) are uniformly sampled from the corresponding intervals. The information-rich regions are sampled similarly. The location of the landmarks is randomly sampled in the free space. We generate 10 environments for each environment model for testing. For each environment, 20 start-goal queries are sampled. Therefore, 400 planning problems are used to test IBBT, RRBT, and MCMP. The algorithms are tested using a double integrator model and a Dubins vehicle model.

The averaged results of the three algorithms are given in Table I. The double integrator and the Dubins vehicle are tested in the information-rich region environment. The double integrator is also tested in the landmark environment. IBBT is able to find the first solution faster. The first solutions returned by IBBT also have lower costs on average. After finding the initial solution, all algorithms are able to improve their current solution when more samples are added to the graph. IBBT shows better cost vs. time performance. Note that RRBT and IBBT are anytime algorithms while MCMP is not. The results of MCMP are based on knowing the number of samples needed. If the number of samples needed is not known, MCMP may need to restart and the performance will be worse. Figure 4 gives one example of the DI-IR problem. For the environment and start-goal shown in Figure 4(a), each algorithm is run 20 times to solve this problem. One solution from IBBT is also given in Figure 4(a). The cost vs. time results are given in Figure 4(b).



(a) (b)

Fig. 4: (a) IBBT planning result. The robot starts at the bottom of the map. Gray lines are Monte Carlo simulations of the plan. (b) Comparison results for the planning problem shown in (a).

VII. Conclusion

We developed an online, anytime, incremental algorithm for motion planning under uncertainties. The algorithm considers a robot that is partially observable, has motion uncertainty, and operates in a continuous domain. The algorithm interleaves between batch sampling, building a nominal trajectory graph, and graph searching to grow a belief tree. The proposed informed heuristic along with ordered search makes the belief tree search efficient. We have tested the IBBT algorithm in various randomly generated environments. IBBT finds non-trivial motion plans and provides better solutions using a smaller amount of time compared to previous methods. Future work includes studying the asymptotic property of IBBT, extending IBBT to consider non-Gaussian noises and replanning in changing environments.

REFERENCES

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

[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] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] B. D. Luders, S. Karaman, and J. P. How, "Robust sampling-based motion planning with asymptotic optimality guarantees," in *AIAA Guidance, Navigation, and Control*, p. 5097, 2013.

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

[11] L. Janson, E. Schmerling, and M. Pavone, "Monte Carlo motion planning for robot trajectory optimization under uncertainty," in *Robotics Research*, pp. 343–361, Springer, Cham., 2018.

[12] Y. Kantaros, B. Schlotfeldt, N. Atanasov, and G. Pappas, "Sampling-based planning for non-myopic multi-robot information gathering," *Autonomous Robots*, vol. 45, pp. 1029–1046, June 2021.

[13] A. R. Pedram, R. Funada, and T. Tanaka, "Gaussian belief space path planning for minimum sensing navigation," *IEEE Transactions on Robotics*, vol. 39, pp. 2040 – 2059, June 2023.

[14] D. Zheng, J. Ridderhof, Z. Zhang, P. Tsiotras, and A. A. Agha-mohammadi, "CS-BRM: A probabilistic roadmap for consistent belief space planning with reachability guarantees," *IEEE Transactions on Robotics*, vol. 40, pp. 1630–1649, Jan 2024.

[15] L. Blackmore, M. Ono, and B. C. Williams, "Chance-constrained optimal path planning with obstacles," *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1080–1094, 2011.

[16] M. P. Vitus and C. J. Tomlin, "Closed-loop belief space planning for linear, Gaussian systems," in *IEEE International Conference on Robotics and Automation*, pp. 2152–2159, May 2011.

[17] A. Wang, A. Jasour, and B. C. Williams, "Non-Gaussian chance-constrained trajectory planning for autonomous vehicles under agent uncertainty," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6041–6048, 2020.

[18] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How, "Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns," *Autonomous Robots*, vol. 35, no. 1, pp. 51–76, 2013.

[19] T. Summers, "Distributionally robust sampling-based motion planning under uncertainty," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 6518–6523, October 2018.

[20] 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.

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

[22] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 883–921, 2015.

[23] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Batch Informed Trees (BIT*): Informed asymptotically optimal anytime search," *The International Journal of Robotics Research*, vol. 39, no. 5, pp. 543–567, 2020.

[24] 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.

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

[26] S. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[27] W. Liu and M. H. Ang, "Incremental sampling-based algorithm for risk-aware planning under motion uncertainty," in *IEEE International Conference on Robotics and Automation*, pp. 2051–2058, May 2014.

[28] B. Luders, M. Kothari, and J. How, "Chance constrained RRT for probabilistic robustness to environmental uncertainty," in *AIAA Guidance, Navigation, and Control*, p. 8160, 2010.

[29] 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.

[30] S. Safaoui, B. J. Gravell, V. Renganathan, and T. H. Summers, "Risk-averse RRT* planning with nonlinear steering and tracking controllers for nonlinear robotic systems under uncertainty," *arXiv preprint arXiv:2103.05572*, 2021.

[31] A. Wang, A. Jasour, and B. Williams, "Moment state dynamical systems for nonlinear chance-constrained motion planning," *arXiv preprint arXiv:2003.10379*, 2020.

[32] 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), pp. 11029–11035, 2022.

[33] K. Sun and V. Kumar, "Belief space planning for mobile robots with range sensors using iLQG," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1902–1909, 2021.

[34] S. Rahman and S. L. Waslander, "Uncertainty-constrained differential dynamic programming in belief space for vision based robots," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3112–3119, 2021.

[35] B. Ichter, A. A. Schmerling, E.and Agha-mohammadi, and M. Pavone, "Real-time stochastic kinodynamic motion planning via multiobjective search on GPUs," in *IEEE International Conference on Robotics and Automation*, pp. 5019–5026, May 2017.

[36] 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.

[37] W. Feller, *An introduction to probability theory and its applications*, vol. 2. John Wiley and Sons, 1991.