Rapidly-exploring Random Belief Trees for Motion Planning Under Uncertainty

In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2011).

Rapidly-exploring Random Belief Trees for Motion Planning Under Uncertainty

Adam Bry, Nicholas Roy
Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract— **In this paper we address the problem of motion planning in the presence of state uncertainty, also known as planning in belief space. The work is motivated by planning domains involving nontrivial dynamics, spatially varying measurement properties, and obstacle constraints. To make the problem tractable, we restrict the motion plan to a nominal trajectory stabilized with a linear estimator and controller. This allows us to predict distributions over future states given a candidate nominal trajectory. Using these distributions to ensure a bounded probability of collision, the algorithm incrementally constructs a graph of trajectories through state space, while efficiently searching over candidate paths through the graph at each iteration. This process results in a search tree in belief space that provably converges to the optimal path. We analyze the algorithm theoretically and also provide simulation results demonstrating its utility for balancing information gathering to reduce uncertainty and finding low cost paths.**

I. Introduction

Advances in system identification, control, and planning algorithms increasingly make it possible for autonomous flying vehicles to utilize the full scope of their natural dynamics. Quad-rotors capable of agile flight through tight obstacles, helicopters that perform extended aerobatic sequences, and fixed-wing vehicles that mimic the perching behavior of birds, have all been reported in the literature [1], [2], [3]. Simultaneously, advances in LIDAR and computer vision algorithms have made autonomous flight through obstacle-rich environments possible without the use of an external sensor system [4], [5]. Currently, there is growing interest in extending the highly dynamic maneuvers that have been demonstrated when accurate state information is always available to autonomous vehicles that operate in unstructured environments using only on-board sensors.

Consider a fixed-wing UAV equipped with a LIDAR, navigating through the urban canyon or under the canopy in a forest. The ability of the vehicle to estimate its position is state dependent as obstacles and environmental features come in and out of view of the LIDAR. An aggressive bank angle may make localization with a planar LIDAR difficult or impossible altogether. Motion blur and other speed effects can cause similar problems for camera-based sensing if the vehicle executes maneuvers with high velocity or angular rates. These problems are compounded by the severe size and weight restrictions that are imposed on the sensors and on-board computer by the physics of flight. Further, failure to reason intelligently about the uncertainty while flying around obstacles could lead to an unacceptable probability of crashing.

The contribution of this paper is a novel algorithm that extends a class of recently proposed incremental sampling-based algorithms to handle state-dependent stochasticity in both dynamics and measurements [6]. The key idea is in leveraging the fact that the incremental sampling approach allows us to enumerate all possible paths through an environment to find paths that optimally trade off information gathering, avoiding obstacles, and quickly reaching the goal. To ensure the plan is suitable for systems with nontrivial dynamics, we evaluate paths with a probabilistic distribution over all possible trajectories that may be realized while following a path with a closed-loop controller. The algorithm proceeds by incrementally constructing a graph of feedback stabilized trajectories through state space, while efficiently searching over candidate paths through the graph as new samples are added. We provide a pruning technique that exploits specific properties of uncertainty propagation for eliminating possible paths and terminating search after each sample is added. In the limit, this process results in a tree in belief space (the space of probability distributions over states) that contains the optimal path in terms of minimum cost with a bounded probability of collision or “chance-constraint”.

Both sampling-based algorithms and linear control and estimation schemes have been shown to scale well with dimensionality, so the algorithm we present should extend easily to more complicated systems. Additionally, the algorithm has the powerful property offered by incremental sampling algorithms of quickly exploring the space and then provably converging to the optimal solution. This is particularly desirable for stochastic planning problems since they are computationally demanding and in a real-time setting the computational time available could vary widely.

II. Related Work

The general motion planning problem of trying to find a collision-free path from some starting state to some goal region has been extensively studied. In particular, sampling-based techniques have received much attention over the last 15 years. The Rapidly-exploring Random Tree (RRT) operates by growing a tree in state space, iteratively sampling new states and then “steering” the existing node in the tree that is closest to each new sample towards that sample. The RRT has many useful properties including probabilistic completeness and exponential decay of the probability of failure with the number of samples [7].

The Rapidly-exploring Random Graph (RRG) proposed by Karaman and Frazzoli is an extension of the RRT algorithm [6]. In addition to the “nearest” connection, new samples are also connected to every node within some ball. The result is a connected graph that not only rapidly explores the state space, but also is locally refined with each added sample. This continuing refinement ensures that in the limit of infinite samples, the RRG contains all possible paths through the environment that can be generated by the steering function used to connect samples. The RRT* algorithm exploits this property to converge to the optimal path by only keeping the edges in the graph that result in lower cost at the vertices with in the ball. While these algorithms have many powerful properties, they assume fully deterministic dynamics and are thus unsuitable for stochastic problems in their proposed form.

Partial observability issues pose severe challenges from a planning and control perspective. Computing globally optimal policies for partially observable systems is possible only for very narrow classes of systems. In the case of discrete states, actions, and observations, exact Partially Observable Markov Decision Process (POMDP) algorithms exist, but are computationally intractable for realistic problems [8]. Many approximate techniques have been proposed to adapt the discrete POMDP framework to motion planning, but they still scale poorly with the number of states, and the prospect of discretizing high-dimensional continuous dynamics is not promising [9].

For systems with linear dynamics, quadratic cost, and Gaussian noise properties (LQG), the optimal policy is obtained in terms of a Kalman filter to maintain a Gaussian state estimate, and a linear control law that operates on the mean of the state estimate [10]. While global LQG assumptions are not justified for the problems we are interested in, many UAVs operate with locally linear control laws about nominal trajectories, and the Kalman filter with various linearization schemes has proven successful for autonomous systems that localize using on-board sensors [4].

The Belief Road Map (BRM) [11] explicitly addresses observability issues by simulating measurements along candidate paths and then choosing the path with minimal uncertainty at the goal. However, the BRM assumes the mean of the system is fully controllable at each time step, meaning that while the path is being executed, the controller is always capable of driving the state estimate back to the desired path. This assumption is valid only for a vehicle flying slowly and conservatively such that dynamic constraints can be ignored. Platt et al. [12] assume maximum likelihood observations to facilitate trajectory optimization techniques and then replan when the actual path deviates past a threshold.

The notion of chance-constrained motion planning is not new. A method of allocating risk for fully observable systems is discussed by Ono et al. [13]. Evaluating a performance metric over a predicted closed-loop distribution for partially observable systems was described by van den Berg et al. [14] and also derived independently by the authors [15], while He et al. [16] use a similar technique with open-loop action sequences. The algorithm proposed by van den Berg et al., termed LQG-MP, picks the best trajectory in terms of minimum cost with a bounded probability of collision from an RRT. However, Karaman and Frazzoli show that the RRT in essence enumerates a finite number of paths, even in the limit of infinite samples [6]. Thus there is no guarantee that a “good” path will be found either in terms of cost or uncertainty properties. In fact, we provide an example problem where an RRT fails to find a solution that satisfies chance-constraints, even though one exists. In contrast, by searching over an underlying graph our approach is both complete and optimal as the graph is refined in the limit.

Search repair after the underlying graph changes is discussed by Koenig et al. [17], however that algorithm is used for deterministic queries, and our search techniques make direct use of covariance propagation properties. Censi et al. [18] and Gonzalez and Stentz [19] use search with a similar pruning technique, however both algorithms operate on static graphs and do not consider dynamic constraints. We extend the pruning strategy for dynamic systems (i.e. a non deterministic mean) and also use the pruning strategy in the context of an incremental algorithm to terminate search.

III. Problem Formulation

The systems we are interested in are generally nonlinear and partially observable. The robot is given a discrete time description of its dynamics and sensors,

$$x_t = f(x_{t-1}, u_{t-1}, w'_t), \qquad w'_t \sim N(0, Q') \tag{1}$$

$$z_t = h(x_t, v'_t), \qquad v'_t \sim N(0, R'), \tag{2}$$

where $x_t \in \mathcal{X}$ is the state vector, $u_t \in \mathcal{U}$ is the input vector, w'_t is an random process disturbance, z_t is the measurement vector, and v'_t is a random component in the sensor readings. The state space $\mathcal{X}$ can be decomposed into $\mathcal{X}^{\text{free}}$ and $\mathcal{X}^{\text{obs}}$ where $\mathcal{X}^{\text{obs}}$ represents the states where the robot is in collision with obstacles.

Our approach to planning is built on an underlying method for finding dynamically feasible solutions and for stabilizing the system. Thus, we assume the availability of a Connect() function for finding a nominal trajectory and stabilizing controller between two states x^a and x^b such that,

$$(\check{X}^{a,b}, \check{U}^{a,b}, \check{K}^{a,b}) = \text{CONNECT}(x^a, x^b) \tag{3}$$

$$\check{X}^{a,b} = (\check{x}_0, \check{x}_1, \check{x}_2, \ldots, \check{x}_{T_{a,b}}) \tag{4}$$

$$\check{U}^{a,b} = (\check{u}_0, \check{u}_1, \check{u}_2, \ldots, \check{u}_{T_{a,b}}) \tag{5}$$

$$\check{x}_0 = x^a, x^b = f(\check{x}_{T_{a,b}}, \check{u}_{T_{a,b}}, 0) \tag{6}$$

$$\check{x}_t = f(\check{x}_{t-1}, \check{u}_{t-1}, 0) \ \forall t \in [1, T_{a,b}], \tag{7}$$

where the trajectory can be stabilized with an on-line state estimate $\hat{x}_t$ as in

$$\check{K}^{a,b} = (K_0, K_1, K_2, \ldots, K_{T_{a,b}}), \tag{8}$$

$$x_t = f\left(x_t, \check{u}_{t-1} - K_t(\hat{x}_t - \check{x}_t), w_t\right). \tag{9}$$

The problem of computing such trajectories and controllers for various robotic vehicles has received an enormous amount of research attention and is beyond the scope of this paper. For general nonlinear systems, techniques such as shooting methods, or direct collocation [20] may be used. For flying vehicles, maneuver primitive approaches are appealing due to their relative computational efficiency [21]. For “Dubins” vehicle dynamics, the optimal trajectory is easily compute in closed form [22]. Stabilizing controllers may be designed, for example, using classical control theory or LQR design, depending on what the system dynamics demand.

The planning problem is specified with some uncertain knowledge of the robot’s initial state given by the probability distribution

$$x_0 \sim N(\hat{x}_0, \Sigma_0), \tag{10}$$

and a goal region in the environment $x_{\text{goal}} \subset \mathcal{X}^{\text{free}}$ to which the robot wishes to travel.

The optimal path planning problem is then to minimize in expectation a stage cost function,

$$\underset{(\check{X},\check{U},\check{K})}{\operatorname{argmin}} E\left[\sum_{t=1}^{T} J(x_t)\right], \tag{11}$$

subject to

$$\check{x}_0 = \hat{x}_0, P(x_T \notin \mathcal{X}_{\text{goal}}) < \delta, \tag{12}$$
$$P(x_t \in \mathcal{X}_{\text{obs}}) < \delta, \forall t \in [0, T], \tag{13}$$

where $\delta < .5$ is a user specified threshold for how much risk to tolerate that defines the chance-constraint and $J : x \mapsto \mathcal{R}^+$ is the cost function. The expectation is with respect to the process and sensor noise w_t and v_t, and minimization is over concatenated paths returned by the CONNECT() function:

$$(\check{X}, \check{U}, \check{K}) = (\textsc{Connect}(x^0, x^1), \\ \textsc{Connect}(x^1, x^2), \ldots, \textsc{Connect}(x^{l-1}, x^l)).$$

This formulation decouples the control design from the path planning optimization. Practically this makes sense for robots where the stabilizing controller is designed with dynamic considerations rather than the specific configuration of an operating environment.

IV. Uncertainty Prediction

In order to evaluate a cost function and check the chance-constraint, we need a distribution over states that may be realized if we execute a given nominal trajectory. Taking appropriate partial derivatives of equations 1 and 2 we obtain the following time varying linear system

$$\tilde{x}_t = A_t\tilde{x}_{t-1} + B_t\tilde{u}_{t-1} + w_t, \qquad w_t \sim N(0, Q_t) \tag{14}$$
$$\tilde{z}_t = C_t\tilde{x}_t + v_t, \qquad v_t \sim N(0, R_t), \tag{15}$$

where $\tilde{x}_t$, $\tilde{u}_t$, $\tilde{z}_t$ are now error quantities, representing the deviation from the nominal path such that $x_t = \check{x}_t + \tilde{x}_t$, $u_t = \check{u}_t + \tilde{u}_t$, and $z_t = \check{z}_t + \tilde{z}_t$.

During execution x_t will not be available to compute the control input. Instead we must use an estimate of x_t which we denote as $\hat{x}_t$. The covariance associated with the state estimate is given by Σ_t. To the extent that f and h are locally linear functions, the Kalman filter is the optimal estimator in the sense of minimum expected estimation error. The filter maintains a Gaussian state estimate, $x_t \sim N(\hat{x}_t, \Sigma_t)$ and operates recursively.[1] A process step first predicts the next state and associated covariance,

$$\bar{\tilde{x}}_t = A_t\hat{\tilde{x}}_{t-1} + B_t\tilde{u}_{t-1} \tag{16}$$
$$\bar{\Sigma}_t = A_t\Sigma_{t-1}A_t^T + Q_t, \tag{17}$$

[1]We can easily convert between estimates of $\hat{x}_t$ and $\hat{\tilde{x}}_t$ by subtracting or adding the deterministic nominal value of $\check{x}_t$ appropriately.

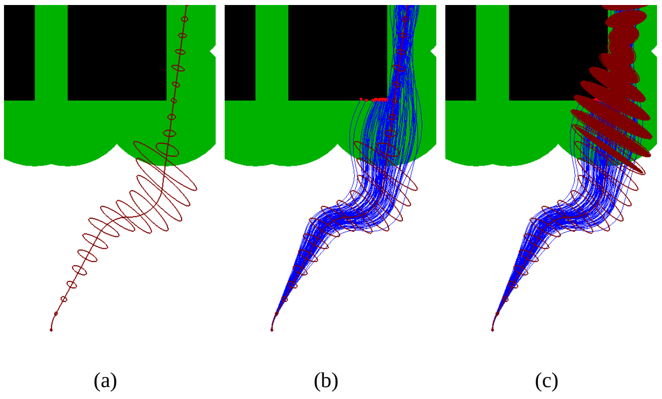


Fig. 1. In this example a vehicle with Dubins dynamics (see section VII) is trying to traverse past the obstacles. The green areas show the regions where the vehicle can obtain range and bearing measurements from the corners of obstacles. (a) shows predicted covariance ellipses using only the Kalman filter which collapse as soon as measurements are received and then appear to pass safely past the obstacles. However, in (b) we see an ensemble of closed-loop trajectories for this system in blue that clearly do not match the expected Kalman filter distribution after measurements are received since it takes time for the controller to pull the robot back onto the nominal path. In (c) the solid ellipses show the closed-loop distribution which correctly predicts the collision.

and a measurement update then adjusts the prediction and incorporates the new information into the covariance,

$$S_t = C_t\bar{\Sigma}_tC_t^T + R_t \tag{18}$$
$$L_t = \bar{\Sigma}_tC_t^TS_t^{-1} \tag{19}$$
$$\hat{\tilde{x}}_t = \bar{\tilde{x}}_t + L_t(\tilde{z}_t - C_t\bar{\tilde{x}}_t) \tag{20}$$
$$\Sigma_t = \bar{\Sigma}_t - L_tC_t\bar{\Sigma}_t. \tag{21}$$

where L_t is the Kalman gain. While Σ_t captures the uncertainty that will be present on-line during path execution, it does not represent the full uncertainty from a planning perspective since the mean of the state estimate, $\hat{x}_t$, will not lie on the nominal trajectory as assumed in [11], [12], [18], [19] and others. Figure 1 illustrates visually why this is important for closed loop systems with nontrivial dynamics.

From a planning perspective we need to consider all possible $\hat{x}_t$ that could be realized during path execution. To do this we can use the Kalman filter equations, but treat the observations as random variables (since they have yet to be observed). We will track the distribution over possible state estimates as $P(\hat{\tilde{x}}) \sim N(\mu, \Lambda)$. Taking the appropriate expectations through the prediction step of the Kalman filter (equation 17) for the first moment of the distribution follows as:

$$\bar{\mu}_t = E[\bar{\tilde{x}}_t] = E[A_t\hat{\tilde{x}}_{t-1} + B_t\tilde{u}_{t-1}] \tag{22}$$
$$= E[A_t\hat{\tilde{x}}_{t-1} - B_tK_t\hat{\tilde{x}}_{t-1}] \tag{23}$$
$$= (A_t - B_tK_t)\mu_{t-1}. \tag{24}$$

For the update step (equations 20, 21) we have

$$\mu_t = E(\hat{\tilde{x}}_t) = E(\bar{\tilde{x}}_t + L_t(\tilde{z}_t - C_t\bar{\tilde{x}}_t)) = \bar{\mu}_t \tag{25}$$
$$\mu_t = (A_t - B_tK_t)\mu_{t-1}. \tag{26}$$

However, in general $\hat{\tilde{x}}_0 = \mu_0 = 0$ because the planning problem is specified with an initial state estimate from which the noise free trajectory is built. From equation 26 it is obvious that if $\mu_0 = 0$ then $\mu_t = 0$ $\forall t$. Exploiting the fact that $\mu_t = 0$ and substituting A_K for $A_t - B_t K_t$, the expectations for the second moment follow as

$$\bar{\Lambda}_t = E[(A_K \hat{\check{x}}_{t-1})(A_K \hat{\check{x}}_{t-1})^T] \tag{27}$$
$$= A_K \Lambda_{t-1} A_K^T, \tag{28}$$

with the update step as

$$\Lambda_t = E[(\hat{\check{x}}_t \hat{\check{x}}_t^T)] \tag{29}$$
$$= E[(\bar{\hat{\check{x}}}_t + L_t(\tilde{z}_t - C_t \bar{\hat{\check{x}}}_t))(\bar{\hat{\check{x}}}_t + L_t(\tilde{z}_t - C_t \bar{\hat{\check{x}}}_t))^T] \tag{30}$$
$$= \bar{\Lambda}_t + L_t S_t L_t^T \tag{31}$$
$$= \bar{\Lambda}_t + L_t C_t \bar{\Sigma}_t \tag{32}$$
$$= (A_t - B_t K_t)\Lambda_{t-1}(A_t - B_t K_t)^T + L_t C_t \bar{\Sigma}_t. \tag{33}$$

Equation 33 gives an expression for propagating the distribution of possible state estimates that will be realized during execution. The $(A_t - B_t K_t)\Lambda_{t-1}(A_t - B_t K_t)^T$ term is contractive if K_t is a stabilizing controller for our system. The additive second term is equivalent to the uncertainty that is subtracted from the Kalman filter covariance during the measurement update. Intuitively, the uncertainty subtracted from the on-line state estimate when measurements are received, is added to uncertainty of the expected mean of the state estimate.

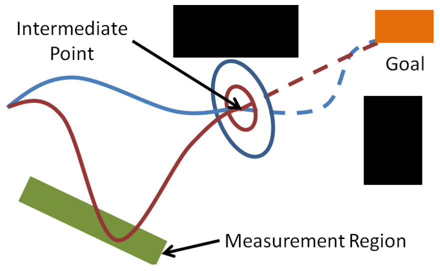
The distribution returned by the Kalman filter, $P(x_t|z_0, z_1, \ldots, z_t) = N(\hat{x}_t, \Sigma_t)$ can be viewed as $P(x|\hat{x})$ since $\hat{x}_t$ encodes the information from the measurements. Equation 33 gives an expression for updating a distribution $P(\hat{x}_t) = N(\check{x}_t, \Lambda_t)$. This provides a natural way to represent the joint belief as $P(x, \hat{x}) = P(x|\hat{x})P(\hat{x})$. Using common Gaussian manipulations we get

$$P(x_t, \hat{x}_t) = N\left(\begin{bmatrix} \check{x}_t \\ \check{x}_t \end{bmatrix}, \begin{bmatrix} \Lambda_t + \Sigma_t & \Lambda_t \\ \Lambda_t & \Lambda_t \end{bmatrix}\right). \tag{34}$$

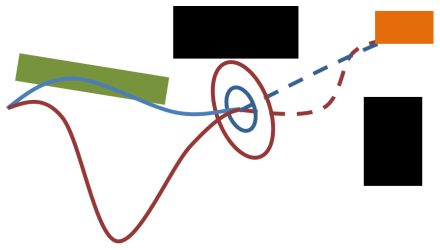
The primary significance of this distribution is the marginal, $P(x_t) = N(\check{x}, \Lambda_t + \Sigma_t)$. For planning purposes this is what we care about. It describes the distribution over trajectories as the sum of the on-line state estimation error, Σ_t, and the uncertainty that arises from not having yet taken observations, Λ_t. This distribution is used to check the chance-constraint (13) and evaluate cost (11) for candidate paths.

V. Rapidly-exploring Random Belief Tree

The Rapidly-exploring Random Belief Tree (RRBT) interleaves graph construction and search over the graph to project a tree into belief space. The algorithm operates on a set of vertices, V, and edges, E, that define a graph in state space. Each vertex $v \in V$ has a state, $v.x$, and a set of associated beliefs nodes $v.N$. Each belief node $n \in v.N$ has a state estimate covariance $n.\Sigma$, a distribution over state estimates $n.\Lambda$, a cost $n.c$, and a parent belief n.parent. Belief nodes correspond to a unique path through the graph that could be followed to reach the vertex v, and the member variables of belief nodes ($n.\Sigma$, $n.\Lambda$, and $n.c$) are the properties that result from following that path. Each edge $e \in E$ contains the trajectory and control law to traverse between the associated vertices and is defined by the Connect() function. A search queue, Q, of belief nodes keeps track of paths that need updating at each iteration of the algorithm.



(a)



(b)

Fig. 2. This figure shows two different paths that reach an intermediate point with different covariance. In (a) the red path dips down and gets measurements and thus has lower uncertainty at the intermediate point. This lower uncertainty gives it a lower cost to go, shown by the dotted line, since it can traverse closer to obstacles. Thus, at the intermediate point, neither the red or blue paths can be pruned. In contrast, in (b) the blue path receives measurements and thus has lower cost and uncertainty at the intermediate point. In this scenario it dominates the red path, and the red path would be pruned.

The covariance prediction (equations 21, 33), cost expectation evaluation (equation 11), and chance-constraint checking (equation 13) is implemented by a Propagate(e, n_{start}) function that takes as arguments an edge and a belief node at the starting vertex for that edge, and returns a belief node at the ending vertex for that edge. If the chance-constraint is violated by the uncertainty obtained in propagating the covariances the function returns no belief.

Additionally, we require the following functions: Sample() returns i.i.d. uniform samples from $\mathcal{X}^{\text{free}}$, Nearest($V, v_{\text{new}}$) takes the current set of vertices as an argument and returns v_{nearest}, the vertex in V that minimizes some distance function to v_{new}, and Near(V, v_{new}) returns every vertex within some ball centered at v_{new} of radius $\rho \propto (\log(n)/n)^{(1/d)}$ where n is the number of state vertices and d is the state dimension. For a thorough discussion on the importance of the ball size see [6].

A. Comparing Partial Paths

At this point we observe that any graph of nominal trajectories through state space implies an infinite set of possible paths through that graph. Search algorithms like Dijkstra's algorithm and A* impose a total ordering on paths to each vertex in the graph based on cost, thus finding a single optimal path to each node. This total ordering is also the property that the RRT* algorithm exploits to "rewire" and maintain a tree in the state space. This works because the optimal cost to the goal from any vertex is not a function of the path taken to that vertex. However, as illustrated in figure 2, for our chance-constrained framework, this is generally not the case. As we demonstrate in section VI, we can impose a partial ordering of the form:

$$n_a < n_b \Leftrightarrow (n_a.\Sigma < n_b.\Sigma) \wedge (n_a.\Lambda < n_b.\Lambda) \wedge (n_a.c < n_b.c) \quad (35)$$

where n_a and n_b represent different partial paths to the same vertex, while being guaranteed not to prune an optimal path.

However, we still have a problem in that infinite loops in the graph may exist within this partial ordering. The covariance update equations make it possible for uncertainty to monotonically decrease while cost must monotonically increase. Physically, this may correspond to a robot circling in an information-rich part of the environment to improve the state estimate. We can rule out this infinite looping by introducing a parameter, ϵ, into the comparison which allows n_a to dominate n_b if the covariances associated with n_a are larger than those associated with n_b by a tolerance factor:

$$n_a \lesssim n_b \Leftrightarrow (n_a.\Sigma < (n_b.\Sigma + \epsilon I)) \wedge (n_a.\Lambda < (n_b.\Lambda + \epsilon I)) \wedge (n_a.c < n_b.c). \quad (36)$$

In practice ϵ can be set quite small, and provides a remarkably simple and efficient method for pruning useless paths.

The partially ordered sets of nodes at each vertex are maintained by an APPENDBELIEF(v, n_{new}) function that takes as arguments a state vertex, and a new belief node. This function first checks to see if the new belief is dominated by any existing beliefs at v using equation 36. If it is dominated, the function returns failure. If it is not, the function then appends the new node and checks to see if it dominates any existing nodes using equation 35, pruning when necessary.

B. Algorithm Description

Algorithm 1 depicts the RRBT algorithm. The graph is initialized with a single vertex and single belief corresponding to the initial state estimate as specified in equation 10 on lines 1-3. This belief will form the root of the belief tree.

At each iteration of the main loop, the state graph is updated by sampling a new state and then adding edges to the nearest and near vertices as in the RRG algorithm. Whenever an existing vertex has an outgoing edge added, all the belief nodes at that vertex are added to the queue. It should be noted that the new vertex is only added to the graph (along with the appropriate edges) if the chance-constraint can be satisfied by propagating an existing belief at the nearest vertex to the new sampled vertex as shown by the check on line 8. This is analogous to "collision-free" checks in a standard RRT.

After all the edges have been added, the queue is exhaustively searched using uniform cost search from lines 19-27, using the the pruning criteria discussed above and implemented by APPENDBELIEF(). The choice of uniform cost search is important because it guarantees that within an iteration of the algorithm (adding a new sample), no new belief will be appended at a state vertex and then pruned. This is a direct consequence of the partial ordering in equation 35 including cost, and the fact that with uniform cost search and a positive cost function, the cost of nodes being examined must monotonically increase.

Algorithm 1 RRBT Algorithm

```
1:  $n.\Sigma := \Sigma_0$; $n.\Lambda := 0$; $n.c := 0$; $n$.parent := NULL;
2:  $v.x := x_{\text{init}}$; $v.N := \{n\}$;
3:  $V := \{v\}$; $E := \{\}$
4:  while $i < M$ do
5:      $x_{\text{rand}}$ :=SAMPLE()
6:      $v_{\text{nearest}}$ := NEAREST($V, x_{\text{rand}}$)
7:      $e_{\text{nearest}}$ =CONNECT($v_{\text{nearest}}.x, x_{\text{rand}}$)
8:      if $\exists v_{\text{nearest}}.n$ :PROPAGATE($e_{\text{nearest}}, n$) then
9:          $V := V \cup v(x_{\text{rand}})$
10:         $E := E \cup e_{\text{nearest}}$
11:         $E := E \cup$ CONNECT($x_{\text{rand}}, v_{\text{nearest}}.x$)
12:         $Q := Q \cup v_{\text{nearest}}.N$
13:         $V_{\text{near}}$ := NEAR($V, v_{\text{rand}}$)
14:         for all $v_{\text{near}} \in V_{\text{near}}$ do
15:             $E := E \cup$ CONNECT($v_{\text{near}}.x, x_{\text{rand}}$)
16:             $E := E \cup$ CONNECT($x_{\text{rand}}, v_{\text{near}}.x$)
17:             $Q := Q \cup v_{\text{near}}.N$
18:         end for
19:         while $Q \neq \emptyset$ do
20:             $n$ :=POP($Q$)
21:             for all $v_{\text{neighbor}}$ of $v(n)$ do
22:                 $n_{\text{new}}$ :=PROPAGATE($e_{\text{neighbor}}, n$)
23:                 if APPENDBELIEF($v_{\text{neighbor}}, n_{\text{new}}$) then
24:                     $Q := Q \cup n_{\text{new}}$
25:                 end if
26:             end for
27:         end while
28:     end if
29:     $i := i + 1$
30: end while
```

VI. CONVERGENCE ANALYSIS

In this section we show, given some reasonable assumptions about the environment and the system dynamics, that the RRBT algorithm converges to the optimal path in the limit of infinite samples. We begin by stating necessary assumptions.

Assumption 1: Let e_1 = CONNECT(x^a, x^c), e_2 = CONNECT(x^a, x^b), and e^3 =CONNECT(x^b, x^c). If $x^b \in e_1$, then the concatenation $[e_2, e_3]$ must be equal to e_1 and as a consequence have equal expected cost for any initial belief.

This assumption states that the CONNECT() function must be consistent for intermediate points and that the cost function must also be consistent. It further states that our CONNECT() function must correctly and consistently interpolate the LQG properties. This is necessary since our algorithm relies on refining through infinite sampling which implies that samples will be infinitely close together. Since for most robots the discrete dynamics equations will be derived from a continuous system description, this implies that we must be able to compute a "partial" step by rediscretizing the continuous system with the appropriate time step.

Assumption 2: There exists a ball of radius $\gamma \in \mathbb{R}_+$ at every point $x \in \mathcal{X}$ such that *(i)* for all $x' \in \mathcal{X}^\gamma$, $\int_{\mathcal{X}_{\text{obs}}} P(x')dx' < \delta$, where $P(x')$ is a reachable belief at x', and *(ii)* $x \in \mathcal{X}^\gamma$

This assumption is a stochastic-chance-constrained parallel to assumption 14 in [6]. It states that the obstacles in the environment are spaced such that it is possible to move the mean of a distribution within some ball and not violate the chance-constraint. This is necessary to give the graph a finite sample volume to converge in.

Assumption 3: The structure of $\mathcal{X}_{\text{obs}}$ and is such that if $x^a \sim N(\hat{x}, \Sigma^a)$, $x^b \sim N(\hat{x}, \Sigma^b)$, and $\Sigma_a < \Sigma_b$ then $P(x^a \in \mathcal{X}^{\text{obs}}) \leq P(x^b \in \mathcal{X}^{\text{obs}})$ for all $\hat{x} \in \mathcal{X}_{\text{free}}$

This assumption simply states that decreasing the covariance can't increase the probability of collision at a given state estimate. This may be violated for very sparse environments with small obstacles and large uncertainty, but is practically very reasonable.

Assumption 4: The cost function is convex in the sense that if $x^a \sim N(\hat{x}, \Sigma^a)$, $x^b \sim N(\hat{x}, \Sigma^b)$, and $\Sigma_a < \Sigma_b$ then $E\left[J(x^a)\right] \leq E\left[J(x^b)\right]$ for all $\hat{x} \in \mathcal{X}_{\text{free}}$.

While this is a restrictive assumption, we note that it includes a uniform cost function over the state, resulting in shortest path behavior. Additionally, the environmental obstacles need not be convex since the cost function is decoupled from the obstacle constraints. Further, even if the actual cost function is not globally convex, it may still be locally convex along the optimal path and the above assumption can still be met for Σ^a and Σ^b below a certain threshold.

Assumption 5: The partial derivatives that lead to equations 14 and 15 are exact.

This is the most restrictive assumption. It states that our system must be perfectly locally linear and further, that the LQG properties (R, Q, A, B, and C) must be the same during the planning phase and execution phase. While this is certainly not generally true, for many systems this is a reasonable approximation, and it is justified since we are using a feedback control law to stay close to the nominal trajectory. This is also more realistic than assuming maximum likelihood observations. Instead we are assuming that we can predict the properties of the measurements, without assuming we know the actual values of the measurements.

Lemma 1: Let $\mathcal{P}^{\text{cc}}$ denote the set of all finite length paths through $\mathcal{X}$ such that for every $x_t \in p$ for every $p \in \mathcal{P}^{\text{cc}}$ $P(x_t \in \mathcal{X}^{\text{obs}}) < \delta$. Let $\mathcal{P}^{\text{cc}}_{V_i,E_i}$ denote a similar set contained in the graph of the RRBT algorithm at iteration i. $\lim_{i\to\infty} \mathcal{P}^{\text{cc}}_{V_i,E_i} = \mathcal{P}^{\text{cc}}$.

Proof: (Sketch) This follows from assumptions 1 and 2 along with results presented in [6]. The idea is that if the obstacles in the environment are spaced such that there is some reachable belief that will permit a distribution to be shifted within a ball, then in the limit of infinite samples, there will be an infinitely dense connected graph in the ball. Since this property is assumed to hold for all x, the environment will be covered by an infinitely dense connected graph. ■

Lemma 1 states that in the limit of infinite samples, the underlying graph built by the RRBT algorithm contains all finite length paths that respect the chance-constraint. We must therefore show that the search tree of beliefs that we maintain on top of this graph contains all possible paths in the graph that *could* be optimal. Our pruning strategy exploits the fact that LQG belief propagation is invariant with respect to inequality in initial beliefs. To demonstrate this, we use the general Binomial Matrix Inversion Lemma,

$$(A+B)^{-1} = A^{-1} - A^{-1}B(B+BA^{-1}B)^{-1}BA^{-1}. \quad (37)$$

We make use of the general Lemma as it relates to positive definite covariance manipulations with the following Lemma.

Lemma 2: For two covariance matrices A and B, there exists another symmetric positive definite matrix C such that $A^{-1} = (A+B)^{-1} + C$.

Proof: This follows immediately from equation 37 and the observation that if A and B are symmetric-positive-definite, then the quantity $A^{-1}B(B+BA^{-1}B)^{-1}BA^{-1}$ must also be symmetric positive-definite. ■

This property extends to the covariance of the Kalman filter with the following Theorem.

Theorem 1: For two covariance matrices, Σ^1_0 and Σ^2_0, where there exists some positive-definite matrix D_0 such that $\Sigma^1_{x_0} + D_0 = \Sigma^2_{x_0}$, there will always be another positive definite matrix D_t such that $\Sigma^1_{x_t} + D_t = \Sigma^2_{x_t} \forall t \in [0, \infty)$.

Proof: We begin by noting that the Kalman filter relies upon a two step recursion. Thus if the property holds through each step of the recursion it holds for all $t < \infty$.

For the process step we have

$$\begin{aligned}
\bar{\Sigma}^2_t &= A\Sigma^2_{t-1}A^T + Q = A(\Sigma^1_{t-1} + D)A^T + Q \\
&= A\Sigma^1_{t-1}A^T + ADA^T + Q \\
\bar{\Sigma}^1_t &= A\Sigma^1_{t-1}A^T + Q \\
\bar{\Sigma}^2_t - \bar{\Sigma}^1_t &= ADA^T = D'.
\end{aligned}$$

For the measurement update we turn to the information form of the Kalman update,

$$\Sigma^2_t = (\bar{\Sigma}^{2\,-1}_t + R^1)^{-1} = ((\bar{\Sigma}^1_t + D')^{-1} + R^1)^{-1}.$$

by Lemma 2 we can write

$$\Sigma^2_{x_t} = (\bar{\Sigma}^{1\,-1}_{x_t} - D'' + R^1)^{-1},$$

and again

$$\Sigma^2_{x_t} = (\bar{\Sigma}^{1\,-1}_{x_t} + R^1)^{-1} + D'''.$$

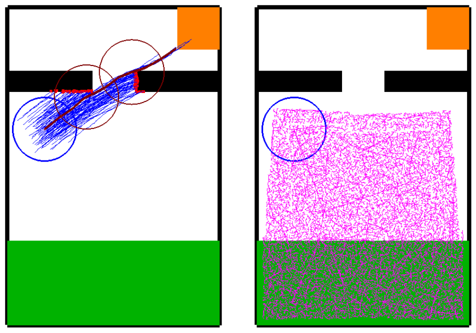
Thus we have $\Sigma^2_{x_t} = \Sigma^1_{x_t} + D'''$ where D''' is positive-definite. ■

The following Theorem states that a similar property holds for the mean uncertainty.

Theorem 2: For two state covariance matrices, Σ^1_0 and Σ^2_0, where there exists some positive-definite matrix D_0 such that $\Sigma^1_t + D_0 = \Sigma^2_t$, and two corresponding mean covariance matrices, Λ^1_0 and Λ^2_0, with the same property $\Lambda^1_0 + E_0 = \Lambda^2_0$, there will be some positive-definite matrix E_t such that $\Lambda^1_t + E_t = \Lambda^2_t$ always holds.

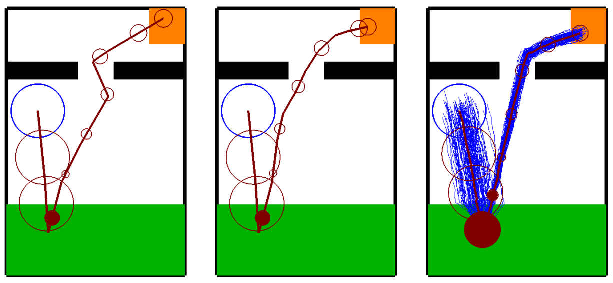
The proof follows in a similar manner to Theorem 1, by plugging into the belief propagation equations and applying Lemma 2.

Theorem 3: For two beliefs n_a and n_b at the same state, let p_a and p_b be the nominal trajectories to the beliefs, and let p^g_a and p^g_a be the optimal nominal trajectories from n_a and n_b to the goal. If $n_a \lesssim n_b$ then $E\left[\sum_{p^g_a} J(x_t)\right] + n_a.c \leq E\left[\sum_{p^g_b} J(x_t)\right] + n_b.c + c_\epsilon$, and $\lim_{\epsilon\to 0} c_\epsilon = 0$.



(a) (b)

Fig. 3. In this environment the robot can only receive measurements in the bottom green region. Ignoring uncertainty and moving straight to the orange goal region (a) results in a high probability of collision, while growing an RRT while checking the chance-constraint (b) will not find a solution. The purple lines in (b) depict nominal paths through the environment none of which reach the upper region since any sample drawn from the that region will be connected to a path that hasn't visited the green region, and thus cannot safely pass the obstacles.



(a) (b) (c)

Fig. 4. (a) shows the RRBT algorithm after 100 iterations, (b) after 500 iterations, and (c) after 10000 iterations. The algorithm quickly finds a feasible solution that goes down to the information region to localize and then pass safely between the obstacles. As more samples are added, this path is refined. Note that the solution is slightly conservative in that the path goes far enough into the green region to ensure the probability mass within the chance-constraint actually receives measurements.

Proof: This result follows directly from assumptions 3 and 4 combined with Theorems 1 and 2. If $n_a \lesssim n_b$, then the optimal cost to go for n_a must be within some constant factor of that for n_b. Since the limit yields $n_a < n_b$ the constant factor has to approach 0. The accumulated cost is strictly less than that of n_b and thus the total cost is also strictly less than that for n_b. ■

Theorems 1 through 3 prove that the pruning strategy is conservative in that it only removes suboptimal paths; since by Theorem 1 the original graph contains all paths achievable with the CONNECT() function, in the limit of infinite samples and $\epsilon \longrightarrow 0$, the belief tree will contain the optimal path.

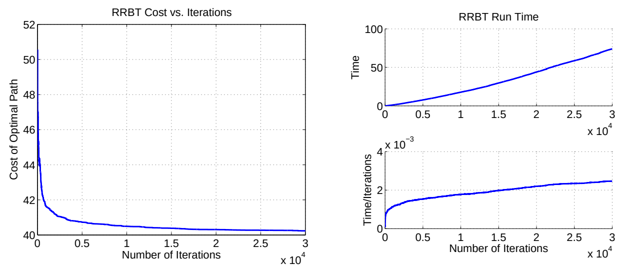
VII. Experimental Results

We first implemented the algorithm on a 2D system with dynamics:

$$x_t = x_{t-1} + u_{t-1} + w_t), \qquad w_t \sim N(0, 0.01I) \tag{38}$$

$$z_t = x_t + v_t', \qquad v_t \sim N(0, R), \tag{39}$$

where $R = \infty I$ or $R = 0.01I$ depending on the location in the environment. Figure 3 shows a specific configuration



(a) (b)

Fig. 5. This figure shows the cost of the best path in the tree (a), the run time (b - top), and the run time per iteration (b - bottom) as a function of the number of samples for the environment in figure 4, averaged over 20 runs.

of an environment that forces trade-offs between information gathering and finding short paths, where the robot can only receive measurements in a small region away from the goal. In this scenario, moving straight to the goal will not satisfy the chance-constraint.

To evaluate the probability of collision on each step, we used the conservative approximation of checking the ellipse defined by the covariance matrix and a desired chance bound for collisions with obstacles. This is computationally faster than integrating the distribution over $\mathcal{X}^{\text{obs}}$, and since the problem formulation states that the specification is an upper bound, this is a reasonable approximation to make. For more aggressive (but still conservative) approaches see [13].

For the example in figure 3, simply growing an RRT and checking the chance-constraint along the paths, as proposed in [14], fails to find a feasible solution since the Voronoi-bias will prevent expansion of the paths that have passed through the measurement region. In contrast, the RRBT algorithm, not only find a feasible path, it refines towards the optimal path as shown in figure 4. The cost and runtime statistics averaged over 20 runs in this environment are shown in figure 5. As our theoretical predict, we can see the cost converging as a function of the number of samples. Additionally, the computational complexity per iteration is sub-linear.

It is important to note that the algorithm is dependent on being able to predict the properties of the measurements that will be received. Since, for the problems we are interested in, the measurements are a function of state, the actual measurement properties may vary from the predicted covariance. In our implementation we handle this by only predicting a measurement if only probability mass below the chance-constrained level is outside of a measurement region. This can be seen in figure 4 where the solution goes far enough into the measurement region to ensure that every state element inside the covariance ellipse receives measurements.

In addition to the 2D system we also tested the algorithm in a domain with Dubins vehicle dynamics and range and bearing beacon measurements from the corners of obstacles which serves as an approximate model for a fixed-wing vehicle that uses a corner detector with a LIDAR. The

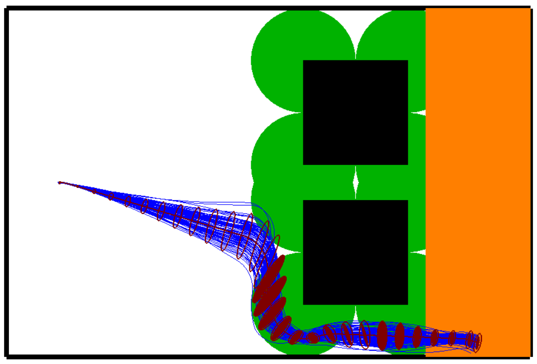


Fig. 6. This figure shows the algorithm running in a more complicated environment with Dubins dynamics and beacon measurements. We see that the algorithm finds a path that turns parallel to the obstacles so as to localize and stabilize onto the nominal trajectory before going past the obstacles to the goal.

continuous Dubins vehicle dynamics are described by:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} V\cos(\theta) \\ V\sin(\theta) \\ \omega \end{bmatrix}, \tag{40}$$

where,

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \theta \end{bmatrix}, u = \begin{bmatrix} V \\ \omega \end{bmatrix}. \tag{41}$$

Due to space constraints, we refer the reader to [23] for details on discretizing and implementing this model, and for the specifics of the measurement model. A path returned by the RRBT algorithm for a sample environment is shown in figure 6. This is a challenging environment where uncertainty accumulates as the vehicle heads towards obstacles. When the vehicle reaches the obstacles it receives measurements and uncertainty collapses. However, as shown in figure 1, proceeding directly past the obstacles is not possible since it takes time for the actual path to stabilize down onto the nominal path. Instead, the algorithm returns a solution that turns parallel to the obstacles, gets measurements, stabilizes onto the nominal path, and then safely goes to the goal.

VIII. Conclusion

For robots with continuous dynamics that operate in partially observable, stochastic domains, motion planning presents significant challenges. In this paper we present an algorithm, the Rapidly-exploring Random Belief Tree, that leverages a local LQG control solution to predict a distribution over trajectories for candidate nominal paths, and then uses incremental sampling refinement to optimize over the space of nominal trajectories. While we have demonstrated the utility of the algorithm for simulation examples, significant future work remains.

Further theoretical work is necessary in investigating the computational complexity. Our experimental results suggest that the complexity is sub-linear per iteration, but more analysis is necessary to confirm this. A key question is how the number of belief nodes scales relative to the number of state vertices.

We also plan to implement the algorithm on an actual fixed-wing platform. In making the algorithm feasible for use in real-time, there are a number of possibilities for introducing heuristics for speed. An A* heuristic could be used in the search to focus towards the goal. Once the goal is found, the same heuristic could be used to bound the tree and graph growth and eliminate suboptimal regions. Further, by introducing an “expected value of information” it would be possible to reduce the number of belief nodes that must be maintained at each state vertex.

IX. Acknowledgements

This work was supported by ONR under MURI N00014-09-1-1052.

References

[1] A. Coates, P. Abbeel, and A. Ng. Learning for control from multiple demonstrations. In *Proc. ICML*, 2008.
[2] R. Cory and R. Tedrake. Experiments in fixedwing UAV perching. In *Proc AIAA Guidance, Navigation, and Control Conference*, 2008.
[3] D. Mellinger, N. Michael, and V. Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. In *Int. Symposium on Experimental Robotics*, 2010.
[4] A. Bachrach, R. He, and N. Roy. Autonomous flight in unknown indoor environments. In *Inter. Jour. Micro Air Vehicles*, 2009.
[5] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy. Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments. In *SPIE Unmanned Systems Tech. XI*, 2009.
[6] S. Karaman and E. Frazzoli. Incremental sampling-based optimal motion planning. In *Robotics: Science and Systems*, 2010.
[7] S. LaValle and J. Kuffner. Randomized kinodynamic planning. *IJRR*, 20(3), 2001.
[8] L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.
[9] N. Roy and S. Thrun. Coastal navigation with mobile robots. In *Proc NIPS*, 1999.
[10] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2nd edition, 2000.
[11] S. Prentice and N. Roy. The belief roadmap: Efficient planning in linear POMDPs by factoring the covariance. In *Proc. ISRR*, 2007.
[12] R. Platt, R. Tedrake, L. Kaelbling, and T. Lozano-Perez. Belief space planning assuming maximum likelihood observations. In *Proceedings of Robotics: Science and Systems*, 2010.
[13] M. Ono and B. Williams. An efficient motion planning algorithm for stochastic dynamic systems with constraints on probability of failure. In *Proc. AAAI*, 2008.
[14] J. V. D. Berg, P. Abbeel, and K. Goldberg. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. In *Proc. RSS*, 2010.
[15] A. Bry and N. Roy. Exact belief state computation for piecewise LQG planning. Technical report, Massachusetts Institute of Technology, 2010.
[16] R. He and N. Roy. Efficient POMDP forward search by predicting the posterior belief distribution. Technical report, Massachusetts Institute of Technology, September 2009.
[17] S. Koenig, M. Likhachev, and D. Furcy. Lifelong planning A*. *Artificial Intelligence*, 155(1-2), 2004.
[18] Andrea Censi, Daniele Calisi, Alessandro De Luca, and Giuseppe Oriolo. A Bayesian framework for optimal motion planning with uncertainty. In *ICRA*, Pasadena, CA, May 2008.
[19] Juan P. Gonzalez and Anthony Stentz. Using linear landmarks for path planning with uncertainty in outdoor environments. In *Proc IROS*, pages 1203–1210, Piscataway, NJ, USA, 2009. IEEE Press.
[20] O. von Stryk. Numerical solution of optimal control problems by direct collocation. In *Optimal Control*, pages 129–143, 1993.
[21] E. Frazzoli. *Robust Hybrid Control for Autonomous Vehicle Motion Planning*. Department of aeronautics and astronautics, Massachusetts Institute of Technology, Cambridge, MA, June 2001.
[22] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.
[23] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.