Chance Constrained RRT for Probabilistic Robustness to Environmental Uncertainty

Brandon D. Luders*, Mangal Kothari† and Jonathan P. How‡
Aerospace Controls Laboratory
Massachusetts Institute of Technology, Cambridge, MA
luders@mit.edu, mk221@le.ac.uk, jhow@mit.edu

For motion planning problems involving many or unbounded forms of uncertainty, it may not be possible to identify a path guaranteed to be feasible, requiring consideration of the trade-off between planner conservatism and the risk of infeasibility. This paper presents a novel real-time planning algorithm, chance constrained rapidly-exploring random trees (CC-RRT), which uses chance constraints to guarantee probabilistic feasibility for linear systems subject to process noise and/or uncertain, possibly dynamic obstacles. By using RRT, the algorithm enjoys the computational benefits of sampling-based algorithms, such as trajectory-wise constraint checking and incorporation of heuristics, while explicitly incorporating uncertainty within the formulation. Under the assumption of Gaussian noise, probabilistic feasibility at each time step can be established through simple simulation of the state conditional mean and the evaluation of linear constraints. Alternatively, a small amount of additional computation can be used to explicitly compute a less conservative probability bound at each time step. Simulation results show that this algorithm can be used for efficient identification and execution of probabilistically safe paths in real time.

I. Introduction

An important ongoing topic in the motion planning literature is the identification of feasible paths for autonomous systems subject to many forms of uncertainty.[1] This uncertainty may be categorized into four groups:[2] (1) uncertainty in the system configuration; (2) uncertainty in the system model; (3) uncertainty in the environment situational awareness; and (4) uncertainty in the future environment state. To achieve safe and reliable path planning in realistic scenarios, where many or all of these uncertainties may be present, it is expected that knowledge of this uncertainty be incorporated into the planning problem. On the other hand, under such conditions it may not be possible to find a guaranteed-feasible solution. If the uncertainty is unbounded, a feasible yet large disturbance may cause the path to become infeasible. Even if the uncertainty is bounded, the system may lack the control authority to correct for disturbances as they are realized. The fundamental question is then how to properly trade off between planner conservatism and the risk of infeasibility.

A useful way to capture this tradeoff is through chance constraints, which require that the probability of constraint violation not exceed some prescribed value.[3] With the appropriate formu-

*Ph.D. Candidate, Dept of Aeronautics and Astronautics; Member AIAA
†Ph.D. Candidate, Control Research Group, University of Leicester, UK
‡Richard Cockburn Maclaurin Professor of Aeronautics and Astronautics, MIT; Associate Fellow AIAA

lation, it is possible to model a probabilistic state constraint in terms of a tightened, deterministic constraint on the conditional mean of the state, representing the amount of conservatism necessary to achieve a desired probability of feasibility.[4,5] This paper considers the chance constraint formulation developed in Blackmore *et al.*[6] for a linear system in a non-convex environment subject to Gaussian process noise; several extensions have also been developed.[7,8] Concurrent work has extended the chance-constrained optimization framework to consider other kinds of uncertainty, such as collision avoidance between uncertain agents.[9] However, such formulations require use of computationally intensive optimizations, such as mixed-integer linear programs or constrained nonlinear programs. For motion planning problems involving complex dynamics, logical constraints, and/or high-dimensional configuration spaces, the computational complexity of such an optimization may scale poorly, to the point of becoming intractable.

This paper presents a chance constraint formulation using incremental sampling-based methods, and in particular rapidly-exploring random trees (RRT),[10] for linear systems subject to process noise and/or uncertain, possibly dynamic obstacles. Sampling-based approaches have demonstrated several advantages for complex motion planning problems, including efficient exploration of high-dimensional configuration spaces, paths which are dynamically feasible by construction, and trajectory-wise (*e.g.* non-enumerative) checking of possibly complex constraints. The RRT algorithm has been demonstrated as a successful planning algorithm for complex real-world systems, such as autonomous vehicles;[11] however, it does not explicitly incorporate uncertainty, as is proposed here. Under the assumption of Gaussian noise, probabilistic feasibility at each time step can be established through simple simulation of the state conditional mean and the evaluation of linear constraints. Alternatively, a small amount of additional computation can be used to explicitly compute a less conservative probability bound at each time step. The focus of this paper is on uncertainty in the system model and the environment situational awareness; any dynamic obstacles are assumed to follow known, deterministic paths.

After establishing the problem statement, we first present an extension of Blackmore's chance constraint formulation,[6] which allows uncertain, possibly dynamic obstacles in addition to system process noise. We then incorporate this formulation into the RRT algorithm and present the chance constrained RRT (CC-RRT) algorithm, including steps for evaluating probability of feasibility as the tree is expanded. Heuristics are presented which leverage the probabilistic nature of this planning tree. Simulation results show that this algorithm can be used for efficient identification and execution of probabilistically safe paths.

II. Related Work

This paper considers extension of the RRT algorithm to achieve probabilistic robustness to process noise, a form of model uncertainty, and uncertain obstacles, a form of environmental sensing uncertainty. This work falls under the much larger class of problems related to motion planning under uncertainty,[1,12] which considers the various forms of uncertainty summarized above.

A particularly large subfield of approaches considers optimal policies for partially-observable environments. Many approaches have been proposed to achieve this; we call particular attention to the use of partially-observable Markov decision processes (POMDPs).[13,14] Greytak and Hover use the same Gaussian overlap framework employed for chance constraints as part of an A* cost function.[15] However, these optimal-policy approaches tend to scale poorly for problems with high-dimensional configuration spaces, hindering their ability to model complex dynamics and constraints.

Much existing work on probabilistic uncertainty for randomized planners has focused exclusively on environmental sensing uncertainty, typically in the form of uncertain maps, without considering configuration uncertainty. Miralles and Bobi[16] represent the obstacle map as a sum of Gaussians,

and construct a minimum-probability-of-collision roadmap to guide the system via potential fields. Burns and Brock[17] use an exploration-based heuristic to traverse a probabilistic roadmap (PRM) with probabilistic feasibility guarantees. Several other PRM formulations have been developed which maintain probabilistic safety bounds for a 2D vehicle avoiding obstacles represented by uncertain vertices.[18,19] However, these approaches are not applicable to nonholonomic systems, which generally cannot track the piecewise linear roadmap paths, and require a significant preprocessing phase for roadmap construction. The proposed chance constrained RRT algorithm requires little to no preprocessing and generates dynamically feasible paths by construction.

Several statistical RRT approaches have been proposed to model vehicles travelling on uneven terrain. The well-known "particle-based" approach of Melchior and Simmons[20] samples each tree branch multiple times, using clustering to create nodes; however, this approach may require many simulations per node to maintain a sufficient representation of the uncertainty. A recently-proposed algorithm identifies a finite-series approximation of the uncertainty propagation, in order to reduce model complexity and the resulting number of simulations needed per node.[21] In contrast, the proposed approach in this paper requires only one simulation per node, with optional additional calculations to compute collision probability at each time step.

Recent work has focused on the problem of randomized planning subject to configuration uncertainty. Pepy and Lambert[22] addresses the ego-sensing problem through use of an extended Kalman filter and simulated localization feedback, assuming a perfect system model. Pepy *et al.*[23] seeks guaranteed robust feasibility for a nonlinear system subject to bounded state uncertainty. However, the approach uses a conservative box-shaped "wrapper" to approximate and bound the reachable set at each node. By comparison, the approach considered in this paper tightens system constraints as a direct function of the disturbance and the desired probability of feasibility.

Finally, we note the work of Prentice and Roy,[24] which constructs a roadmap in the belief state for a linear system subject to Gaussian noise. The belief roadmap is primarily used to identify paths which maintain strong localization, rather than avoid uncertain obstacles. Though the propagation of the system mean and covariance is very similar to this work, the roadmap dictates the use of kinematic planning.

III. Preliminaries

A. Problem Statement

Consider a discrete-time linear time-invariant (LTI) system with process noise,

$$x_{t+1} = Ax_t + Bu_t + w_t, \tag{1}$$
$$x_0 \sim \mathcal{N}(\hat{x}_0, P_{x_0}), \tag{2}$$
$$w_t \sim \mathcal{N}(0, P_{w_t}), \tag{3}$$

where $x_t \in \mathbb{R}^{n_x}$ is the state vector, $u_t \in \mathbb{R}^{n_u}$ is the input vector, and $w_t \in \mathbb{R}^{n_x}$ is a disturbance vector acting on the system. Here $\mathcal{N}(\hat{a}, P_a)$ represents a random variable whose probability distribution is Gaussian with mean $\hat{a}$ and covariance P_a. The disturbance w_t is unknown at current and future time steps, but has the known probability distribution (3).

The system itself is subject to two forms of uncertainty. Equation (2) represents uncertainty in the initial state x_0, corresponding to uncertain localization. Equation (3) represents a zero-mean process noise, in the form of independent and identically distributed random variables w_t ($P_{w_t} \equiv P_w \ \forall \ t$). This noise may correspond to model uncertainty, external disturbances, and/or other factors.

There are also constraints acting on the system state and input. These constraints are assumed

to take the form

$$x_t \in \mathcal{X}_t \equiv \mathcal{X} - \mathcal{X}_{t1} - \cdots - \mathcal{X}_{tB}, \tag{4}$$

$$u_t \in \mathcal{U}, \tag{5}$$

where $\mathcal{X}, \mathcal{X}_{t1}, \ldots, \mathcal{X}_{tB} \subset \mathbb{R}^{n_x}$ are convex polyhedra, $\mathcal{U} \subset \mathbb{R}^{n_u}$, and the $-$ operator denotes set subtraction. The set $\mathcal{X}$ defines a set of time-invariant convex constraints acting on the state, while $\mathcal{X}_{t1}, \ldots, \mathcal{X}_{tB}$ represent B convex obstacles to be avoided. The time dependence of $\mathcal{X}_t$ allows the inclusion of both static and dynamic obstacles.

The location of each obstacle is modeled as translationally uncertain, represented as

$$\mathcal{X}_{tj} = \mathcal{X}_j^0 + \delta_{tj} + c_j \;\forall\; j \in \mathbb{Z}_{1,B},\; \forall\; t \tag{6}$$

$$c_j \sim \mathcal{N}(0, P_{c_j}) \;\forall\; j \in \mathbb{Z}_{1,B}, \tag{7}$$

where the $+$ operator denotes set translation and $\mathbb{Z}_{a,b}$ represents the set of integers between a and b inclusive. In this model, $\mathcal{X}_j^0 \subset \mathbb{R}^{n_x}$ is a convex polyhedron of known, fixed shape; $\delta_{tj} \in \mathbb{R}^{n_x}$ is a known translation at time step t; and $c_j \in \mathbb{R}^{n_x}$ is a fixed, unknown translation represented by a zero-mean Gaussian random variable. (All c_j are assumed to be independent.) Thus (6) represents an obstacle of known shape on a known trajectory, but subject to a fixed, unknown translation. This is a reasonable model of real-world sensing systems, where laser range finders or other sensors may identify an approximate location for each obstacle.

The primary objective of the planning problem is to reach the goal region $\mathcal{X}_{\text{goal}} \subset \mathbb{R}^{n_x}$ in minimum time,

$$t_{\text{goal}} = \inf\{t \in \mathbb{Z}_{0,t_f} \mid x_t \in \mathcal{X}_{\text{goal}}\}, \tag{8}$$

while ensuring the constraints (4)-(5) are satisfied at each time step $t \in \{0, \ldots, t_{\text{goal}}\}$ *with probability of at least* p_{safe}. In practice, since there is uncertainty in the state, we assume it is sufficient for the distribution mean to reach the goal region $\mathcal{X}_{\text{goal}}$. A secondary objective may be to avoid some undesirable behaviors, such as proximity to constraint boundaries, and can be represented through a penalty function $\psi(x_t, \mathcal{X}_t, \mathcal{U})$. With this, the motion planning problem can now be defined.

Problem 1. Given the initial state distribution $(\hat{x}_0, P_{x_0})$ and constraint sets $\mathcal{X}_t$ and $\mathcal{U}$, compute the input control sequence $u_t,\ t \in \mathbb{Z}_{0,t_f},\ t_f \in \mathbb{Z}_{0,\infty}$ that minimizes

$$J(\mathbf{u}) = t_{\text{goal}} + \sum_{t=0}^{t_{\text{goal}}} \psi(x_t, \mathcal{X}_t, \mathcal{U}) \tag{9}$$

while satisfying (1) for all time steps $t \in \{0, \ldots, t_{\text{goal}}\}$, and satisfying (4)-(5) at each time step $t \in \{0, \ldots, t_{\text{goal}}\}$ with probability of at least p_{safe}.

B. Chance Constraints

This section reviews the chance constraint formulation of Blackmore *et al.*,[6] in which all obstacles are assumed to have static, known locations, i.e. $\delta_{tj} \equiv c_j \equiv 0$. The extension to uncertain and possibly dynamic obstacles is considered in Section IV.

Given a sequence of inputs $u_0, \ldots, u_{N-1}$, the distribution of the state x_t, represented as the random variable X_t, can be shown to be Gaussian:[6]

$$\begin{aligned} P(X_t | u_0, \ldots, u_{N-1}) &\sim P(X_t | u_0, \ldots, u_{t-1}) \\ &\sim \mathcal{N}(\hat{x}_t, P_{x_t}) \;\; \forall\; t \in \mathbb{Z}_{0,N}, \end{aligned} \tag{10}$$

where N is some time step horizon. The mean $\hat{x}_t$ and covariance P_{x_t} can be represented either explicitly as

$$\hat{x}_t = A^t\hat{x}_0 + \sum_{k=0}^{t-1} A^{t-k-1}Bu_k \quad \forall\ t \in \mathbb{Z}_{0,N}, \tag{11}$$

$$P_{x_t} = A^t P_{x_0}(A^T)^t + \sum_{k=0}^{t-1} A^{t-k-1}P_w(A^T)^{t-k-1} \quad \forall\ t \in \mathbb{Z}_{0,N}, \tag{12}$$

or implicitly as

$$\hat{x}_{t+1} = A\hat{x}_t + Bu_t \quad \forall\ t \in \mathbb{Z}_{0,N-1}, \tag{13}$$

$$P_{x_{t+1}} = AP_{x_t}A^T + P_w \quad \forall\ t \in \mathbb{Z}_{0,N-1}. \tag{14}$$

Note that (13) updates the distribution mean $\hat{x}_t$ using the disturbance-free dynamics, i.e. (1) with $w_t \equiv 0$, and that (14) is independent of the input sequence and thus can be computed *a priori*.

Suppose the objective is to ensure that the probability of collision with any obstacle for a given time step does not exceed $\Delta \equiv 1 - p_{\text{safe}}$; it is then sufficient to show that the probability of collision with each of the B obstacles at that time step does not exceed Δ/B.[6] The jth obstacle is represented through the conjunction of linear inequalities

$$\bigwedge_{i=1}^{n_j} a_{ij}^T x_t < b_{ij} \quad \forall\ t \in \mathbb{Z}_{0,N}, \tag{15}$$

where n_j is the number of constraints defining the jth obstacle. To avoid all obstacles, the system must satisfy B *disjunctions* of constraints at each time step,

$$\bigvee_{i=1}^{n_j} a_{ij}^T x_t \geq b_{ij} \quad \forall\ j \in \mathbb{Z}_{1,B},\ \forall\ t \in \mathbb{Z}_{0,N}. \tag{16}$$

Consider the problem of avoiding the jth obstacle on the tth time step; to avoid the obstacle, it is sufficient to not satisfy any one of the constraints in the conjunction (15). To collide with the obstacle, all of the constraints in (15) must be satisfied. Thus it is true that

$$\begin{aligned} P(\text{collision}) &= P\left(\bigwedge_{i=1}^{n_j} a_{ij}^T X_t < b_{ij}\right) \\ &\leq P(a_{ij}^T X_t < b_{ij}) \quad \forall\ i \in \mathbb{Z}_{1,n_j}. \end{aligned} \tag{17}$$

To ensure that the probability of collision is less than or equal to Δ/B, it is only necessary to show that one of the constraints for the obstacle is satisfied with probability less than or equal to Δ/B:

$$\bigvee_{i=1}^{n_j} P(a_{ij}^T X_t < b_{ij}) \leq \Delta/B. \tag{18}$$

To render this problem tractable for path planning algorithms, the key step is to convert the probabilistic constraints (18) into tightened, deterministic constraints. For the ith constraint of the jth obstacle at time step t, apply the change of variable

$$V = a_{ij}^T X_t - b_{ij}; \tag{19}$$

since $X_t \sim \mathcal{N}(\hat{x}_t, P_{x_t})$, it must also be the case that $V \sim \mathcal{N}(\hat{v}, P_v)$, where

$$\hat{v} = a_{ij}^T \hat{x}_t - b_{ij}, \tag{20}$$

$$P_v = \sqrt{a_{ij}^T P_{x_t} a_{ij}}. \tag{21}$$

With this change of variable, the probabilistic constraint (18) can be written as

$$P(V < 0) \leq \Delta / B. \tag{22}$$

This probabilistic constraint can be shown to be equivalent to a deterministic constraint,[6]

$$P(V < 0) \leq \Delta/B \quad \Leftrightarrow \quad \hat{v} \geq \sqrt{2} P_v \text{erf}^{-1}\left(1 - 2\frac{\Delta}{B}\right), \tag{23}$$

where $\text{erf}(\cdot)$ denotes the standard error function. Using this, the constraints (16) are probabilistically satisfied for the true state x_t if the conditional mean $\hat{x}_t$ satisfies the modified constraints

$$\bigvee_{i=1}^{n_j} a_{ij}^T \hat{x}_t \geq b_{ij} + \bar{b}_{ijt} \quad \forall\, j \in \mathbb{Z}_{1,B},\ \forall\, t \in \mathbb{Z}_{0,N}, \tag{24}$$

$$\bar{b}_{ijt} = \sqrt{2} P_v \text{erf}^{-1}\left(1 - 2\frac{\Delta}{B}\right). \tag{25}$$

The term $\bar{b}_{ijt}$ represents the amount of *deterministic* constraint tightening necessary to ensure *probabilistic* constraint satisfaction. Note that since P_{x_t} can be computed off-line, the tightened constraints (24)-(25) can be computed off-line, as well, implying that the complexity of the nominal formulation does not increase when chance constraints are incorporated.

IV. Chance Constraints for Environmental Uncertainty

In this section, the chance constraint formulation of Section III-B is extended to allow for uncertain and/or dynamic obstacles. In doing so, the chance constraint formulation can incorporate several other types of uncertainty found in common path planning scenarios (Section I).

Consider adding the obstacle uncertainty (6)-(7) to the original chance constraint formulation. The constraints for the jth obstacle (15) can be equivalently written as

$$\bigwedge_{i=1}^{n_j} a_{ij}^T x_t < a_{ij}^T c_{ijt} \quad \forall\, t \in \mathbb{Z}_{0,t_f}, \tag{26}$$

where c_{ijt} is a point nominally (i.e. $c_j = 0$) on the ith constraint at time step t; note that a_{ij} is not dependent on t, since the obstacle shape and orientation are fixed. The corresponding disjunctive constraints (16) are then

$$\bigvee_{i=1}^{n_j} a_{ij}^T x_t \geq a_{ij}^T c_{ijt} \quad \forall\, j \in \mathbb{Z}_{1,B},\ \forall\, t \in \mathbb{Z}_{0,t_f}. \tag{27}$$

As before, to ensure that the probability of collision is less than or equal to Δ/B, it is only necessary to show that one of the constraints is satisfied with probability less than or equal to Δ/B:

$$\bigvee_{i=1}^{n_j} P(a_{ij}^T X_t < a_{ij}^T C_{ijt}) \leq \Delta/B, \tag{28}$$

where $C_{ijt} = c_{ijt} + c_j$ is a random variable due to (6)-(7).

For the ith constraint of the jth obstacle at time step t apply the change of variable

$$V = a_{ij}^T X_t - a_{ij}^T C_{ijt}; \tag{29}$$

the probabilistic constraint is again (22). The mean and covariance for V are computed as follows:

$$\begin{aligned} \hat{v} &= \mathbb{E}[V] = a_{ij}^T \hat{x}_t - \mathbb{E}\left[a_{ij}^T (c_{ijt} + c_j)\right] \\ &= a_{ij}^T \hat{x}_t - a_{ij}^T c_{ijt}, \end{aligned} \tag{30}$$

$$\begin{aligned} P_v &= \sqrt{\mathbb{E}\left[(V - \hat{v})(V - \hat{v})^T\right]} \\ &= \sqrt{\mathbb{E}\left[\left(a_{ij}^T \left(X_t - \hat{x}_t\right) - a_{ij}^T c_j\right)\left(a_{ij}^T \left(X_t - \hat{x}_t\right) - a_{ij}^T c_j\right)^T\right]} \\ &= \sqrt{a_{ij}^T (P_{x_t} + P_{c_j}) a_{ij}}. \end{aligned} \tag{31}$$

Using this, the constraints (27) can then be shown to be probabilistically satisfied through the modification

$$\bigvee_{i=1}^{n_j} a_{ij}^T \hat{x}_t \quad \geq \quad a_{ij}^T c_{ijt} + \bar{b}_{ijt} \quad \forall\, j \in \mathbb{Z}_{1,n_{obs}},\ \forall\, t \in \mathbb{Z}_{0,N}, \tag{32}$$

where $\bar{b}_{ijt}$ is given as in (25) but uses the new definition of P_v, (31).

In summary, to extend the original chance constraint approach to include uncertain and/or dynamic obstacles as modeled, it is sufficient to:

- Replace the right-hand side of the constraint inequality (26) with $a_{ij}^T c_{ijt}$, where c_{ijt} tracks the deterministic trajectory of the obstacle; and
- Replace (20) with (30), and (21) with (31).

This amounts to placing the mean along the possibly dynamic trajectory of the constraint, and adding the translational uncertainty covariance to P_v.

V. Chance Constrained RRT

This section introduces the chance constrained RRT (CC-RRT) algorithm, an extension of the traditional RRT algorithm which allows for probabilistic constraints. Whereas the traditional RRT algorithm grows a tree of states which are known to be feasible, the chance constrained RRT algorithm grows a tree of state distributions which are known to satisfy an upper bound on probability of collision (Figure 1).

The fundamental operation in the standard RRT algorithm is the incremental growth of a tree of dynamically feasible trajectories, rooted at the system's current state x_t.[10] A node's likelihood of being selected to grow the tree is proportional to its Voronoi region for a uniform sampling distribution. As a result, the RRT algorithm is naturally biased toward rapid exploration of the state space. The RRT algorithm allows the user a great deal of freedom in designing problem-specific heuristics and extensions. Most importantly, the RRT algorithm employs trajectory-wise constraint checking, allowing for the incorporation of possibly complex constraints. The CC-RRT algorithm can leverage this by explicitly computing a bound on the probability of collision at each node, rather than simply satisfying tightened constraints for a fixed bound (Section V-B).

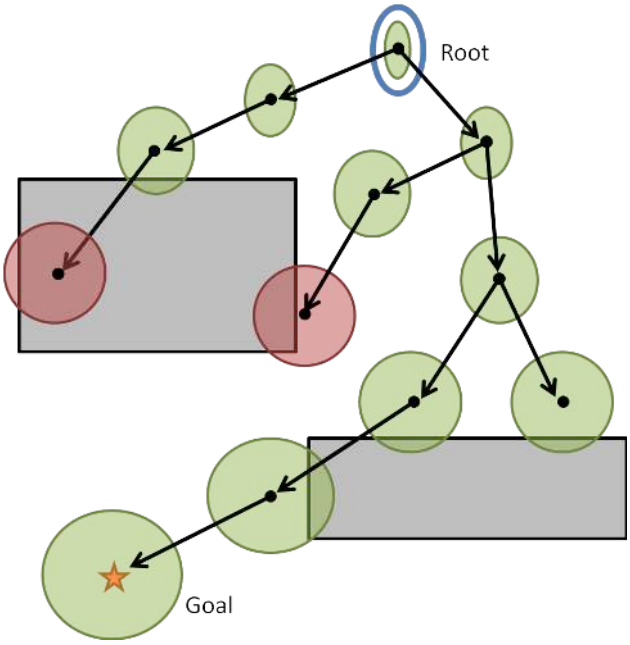


Figure 1. Diagram of the chance constrained RRT algorithm. Given an initial state distribution at the tree root (blue), the algorithm grows a tree of state distributions in order to find a probabilistically feasible path to the goal (yellow star). The uncertainty in the state at each node is represented as an uncertainty ellipse. Each state distribution is checked probabilistically against the constraints (gray). If the probability of collision is too high, the node is discarded (red); otherwise the node is kept (green) and may be used to grow future trajectories.

To grow a tree of dynamically feasible trajectories, it is necessary for the RRT to have an accurate model of the vehicle dynamics (1) for simulation. Since the CC-RRT algorithm grows a tree of state distributions, in this case the model is assumed to be the propagation of the state conditional mean (13) and covariance (14), rewritten here as

$$\hat{x}_{t+k+1|t} = A\hat{x}_{t+k|t} + Bu_{t+k|t}, \tag{33}$$

$$P_{t+k+1|t} = AP_{t+k|t}A^T + P_w, \tag{34}$$

where t is the current system time step and $(\cdot)_{t+k|t}$ denotes the predicted value of the variable at time step $t+k$.

Section V-A introduces the baseline RRT algorithm; this paper applies the real-time RRT algorithm proposed by Frazzoli[25] and later extended by Kuwata *et al.*[26] Section V-B introduces the extensions to this baseline algorithm which allow for satisfaction of probabilistic constraints in the presence of uncertainty. Finally, Section V-C briefly discusses operation of the chance constrained RRT algorithm in real-time, necessitating an execution loop.

A. Tree Expansion

The CC-RRT tree expansion step, used to incrementally grow the tree, is given in Algorithm 1. In this section we discuss the baseline RRT elements as presented in this algorithm.

Each time Algorithm 1 is called, first a sample state is taken from the environment (line 2), and the nodes nearest to this sample, in terms of some heuristic(s), are identified as candidates for tree expansion (line 3). An attempt is made to form a connection from the nearest node to the sample by generating a probabilistically feasible trajectory between them (lines 5–13). This trajectory is

Algorithm 1 CC-RRT, Tree Expansion

1: Inputs: tree $\mathcal{T}$, current time step t
2: Take a sample x_{samp} from the environment
3: Identify the M nearest nodes using heuristics
4: **for** $m \leq M$ nearest nodes, in the sorted order **do**
5: $N_{\text{near}} \leftarrow$ current nodes
6: $(\hat{x}_{t+k|t}, P_{t+k|t}) \leftarrow$ final state distribution of N_{near}
7: **continue** with probability $\Delta_{\max}(N_{\text{near}})$
8: **while** $\Delta_{t+k}(\hat{x}_{t+k|t}, P_{t+k|t}) \leq 1 - p_{\text{safe}}$ **and** $\hat{x}_{t+k|t}$ has not reached x_{samp} **do**
9: Select input $u_{t+k|t} \in \mathcal{U}$
10: Simulate $(\hat{x}_{t+k+1|t}, P_{t+k+1|t})$ using (33)-(34)
11: Create intermediate nodes as appropriate
12: $k \leftarrow k + 1$
13: **end while**
14: **for** each probabilistically feasible node N **do**
15: Update cost estimates for N
16: Add N to $\mathcal{T}$
17: Try connecting N to $\mathcal{X}_{\text{goal}}$ (lines 5-13)
18: **if** connection to $\mathcal{X}_{\text{goal}}$ probabilistically feasible **then**
19: Update upper-bound cost-to-go of N and ancestors
20: **end if**
21: **end for**
22: **end for**

incrementally simulated by selecting some feasible input (line 9), then applying (33)-(34) to yield the state distribution at the next time step. This input may be selected at the user's discretion, such as through random sampling or a closed-loop controller, but should guide the state distribution toward the sample. Probabilistic feasibility is then checked, as discussed in Section V-B; trajectory simulation continues until either the state is no longer probabilistically feasible, or the distribution mean has reached the sample (line 8).

Even if the trajectory does not safely reach the sample, it is useful and efficient to keep probabilistically feasible portions of this trajectory for future expansion. For this reason, intermediate nodes may be occasionally inserted during the trajectory generation process (line 11). Each node contains a trajectory segment, simulated over possibly many time steps; future connections from this node must begin at the end of this trajectory. As a result, one or more probabilistically feasible nodes may be generated from trajectory simulation, each of which is added to the tree (line 16).

A number of heuristics are also utilized to facilitate tree growth, identify probabilistically feasible trajectories to the goal, and identify "better" paths (in terms of (9)) once at least one probabilistically feasible path has been found. Samples are identified (line 2) by probabilistically choosing between a variety of global and local sampling strategies, some of which may be used to efficiently generate complex maneuvers.[26] The nearest node selection scheme (lines 4-5) strategically alternates between several distance metrics for sorting the nodes, including an exploration metric based on cost-to-go and a path optimization metric based on estimated total path length.[25] Each time a sample is generated, $m \geq 1$ attempts are made to connect a node to this sample before being discarded.[26] Since the primary objective is to quickly find a path to the goal, an attempt is made to connect newly-added nodes directly to $\mathcal{X}_{\text{goal}}$ (line 17). Finally, both lower and upper bounds on the cost-to-go are maintained at each node. A branch-and-bound cost scheme is used (line 19) to prune portions of the tree whose lower-bound cost-to-go is larger than the upper-bound cost-to-go of an ancestor, since those portions have no chance of achieving a lower-cost path.[25]

B. Chance Constraint Extensions

The CC-RRT algorithm extends the baseline RRT algorithm in two fundamental ways, such that it can be used to identify a tree of trajectories which probabilistically satisfy the constraints subject to process noise and/or uncertainty in possibly dynamic obstacles. These extensions utilize the chance constraint formulation reviewed and developed in Sections III-B and IV. First, probabilistic feasibility is checked for each state distribution, either by using the tightened bounds of the original formulation (referred to as "Offline CC-RRT") or by explicitly computing a bound on the collision probability at each time step (referred to as "Online CC-RRT"); the distinction is discussed in further detail below. Second, a measure of the collision probability over an entire *path* can be used to bias tree growth toward low-risk portions of the tree.

1. *Offline CC-RRT*

The formulation presented by Blackmore *et al.*[6] is formulated for a non-convex optimization and applies a fixed probability bound Δ/B across all obstacles, resulting in tightened deterministic constraints which can be computed off-line for each time step. The same approach can be used for checking probabilistic feasibility of simulated trajectories in the CC-RRT algorithm (Algorithm 1). Using the extensions in Section IV, one can compute whether the deterministic chance constraints (32) are satisfied for all obstacles for a given conditional mean $\hat{x}$ and covariance P_x. Growth of the simulated trajectory continues only if these deterministic chance constraints are satisfied.

2. *Online CC-RRT*

It is also possible to identify a more precise bound on the probability of collision specific to each time step. This approach leverages the relationship in (23) to compute the exact probability of satisfying each individual constraint for a given distribution $\mathcal{N}(\hat{x}, P_x)$ - an operation which is possible due to iterative constraint checking in the RRT algorithm. This dynamic assignment of risk to each constraint uses similar logic as Iterative Risk Allocation (IRA),[27] which considers convex chance constraints. However, whereas IRA iterates on the risk allocation for successive optimizations, online CC-RRT can directly compute an appropriate risk allocation for each constraint. Through repeated use of this operation, a bound can be computed for each time step on the probability of collision. This bound can be of great use for heuristics within the RRT algorithm, in addition to checking probabilistic feasibility. The tradeoff is the computational complexity introduced by computing this bound at each time step, though the increase is sufficiently small to maintain the approach's suitability for on-line implementation (Section VI).

We first show that the equivalence relationship in (23) can be rewritten with both inequalities written as equalities. Consider (23) for some probability $\gamma \equiv \Delta/B,\ \gamma \in (0,1)$ and a given distribution mean $\hat{v}$ and covariance P_v, rewritten here as

$$P(V < 0) \leq \gamma \quad \Leftrightarrow \quad \hat{v} \geq f(\gamma), \tag{35}$$

where $f(\gamma) = \sqrt{2}P_v \text{erf}^{-1}(1-2\gamma)$. The inverse error function increases monotonically and continuously from $-\infty$ to $+\infty$ over its domain $(-1,+1)$. Since $P_v \geq 0$, this implies that $f(\gamma)$ decreases monotonically and continuously from $+\infty$ to $-\infty$ over its domain $(0,1)$. As a result, there must exist some value $\bar{\gamma}$ such that $\hat{v} = f(\bar{\gamma})$. Exploiting the equivalence in (35), we have for some $\epsilon > 0$ (where $|\bar{\gamma} \pm \epsilon| < 1$) such that

$$\bar{\gamma} - \epsilon < P(V < 0) \leq \bar{\gamma} + \epsilon. \tag{36}$$

It is then clear that as $\epsilon \to 0$, (35) becomes the desired equivalence

$$P(V < 0) = \bar{\gamma} \quad \Leftrightarrow \quad \hat{v} = f(\bar{\gamma}). \tag{37}$$

Solving for $\bar{\gamma}$ yields

$$P(V<0) = \frac{1}{2}\left(1 - \text{erf}\left[\frac{\hat{v}}{\sqrt{2}P_v}\right]\right). \tag{38}$$

Returning to the formulation of Section IV, consider the ith constraint of the jth obstacle at time step t, with associated change of variable (29)-(31). Let $\Delta_{ijt}(\hat{x}, P_x)$ denote the probability that this constraint is satisfied for a Gaussian distribution with mean $\hat{x}$ and covariance P_x; using (38), we have that

$$\Delta_{ijt}(\hat{x}, P_x) = \frac{1}{2}\left(1 - \text{erf}\left[\frac{a_{ij}^T\hat{x}_t - a_{ij}^T c_{ijt}}{\sqrt{2a_{ij}^T(P_{x_t} + P_{c_j})a_{ij}}}\right]\right). \tag{39}$$

Now define

$$\Delta_t(\hat{x}_t, P_{x_t}) \equiv \sum_{j=1}^{B} \min_{i=1,\ldots,n_j} \Delta_{ijt}(\hat{x}_t, P_{x_t}), \tag{40}$$

used in line 8 of Algorithm 1; it can be shown that this term provides an upper bound on the probability of a collision with any obstacle at time step t. Indeed, for time step t,

$$\begin{aligned} P(\text{collision}) &\leq \sum_{j=1}^{B} P(\text{collision with obstacle } j) \\ &\leq \sum_{j=1}^{B} \min_{i=1,\ldots,n_j} P(a_{ij}^T X_t < a_{ij}^T C_{ijt}) \\ &= \sum_{j=1}^{B} \min_{i=1,\ldots,n_j} \Delta_{ijt}(\hat{x}_t, P_{x_t}) = \Delta_t(\hat{x}_t, P_{x_t}). \end{aligned}$$

Here the first inequality uses the addition law of probability, the second inequality uses (17), the first equality uses (38), and the second equality uses (40).

3. Nearest Node Bias

In many cases, a user may be more interested in probabilistic feasibility over an entire *path* rather than at a single *time step*. However, the computation of the probabilistic feasibility over this path is intractable for all but the simplest configurations, mainly due to the fact that the random variables $X_i \; \forall \; t \in \mathbb{Z}_{0,t_f}$ are *not* independent. Recent results have shown that the computation can be simplified for a linear system with Gaussian noise by "stacking" the states and constraints,[8] but this is still computationally intensive for real-time operations of considerable duration.

In this work, we use a simple heuristic to represent the "risk" of a tree path. Let $\Delta_{\max}(N)$ denote the risk of the tree path from the root to node N; it is set equal to the maximum value of Δ_t for *any* time step on that trajectory. With this heuristic, a path which has a moderate probability of collision at every time step is considered to be less risky than a path which has a single time step with a high probability of collision, and a low probability of collision at all other times.

This heuristic can be used in Algorithm 1 to bias tree growth toward lower-risk portions of the tree. For each nearest node N_{near} selected, the algorithm flips a weighted coin to decide whether to attempt to connect to the sample. The attempt is made with probability $1 - \Delta_{\max}(N_{\text{near}})$; otherwise the next nearest node is considered (line 7).

Algorithm 2 CC-RRT, Execution Loop

1: Initialize tree $\mathcal{T}$ with node at $(\hat{x}_0, P_{x_0}), t = 0$
2: **while** $\hat{x}_t \notin \mathcal{X}_{\text{goal}}$ **do**
3: Use measurements, if any, to repropagate state distributions
4: **while** time remaining for this time step **do**
5: Expand the tree by adding nodes (Algorithm 1)
6: **end while**
7: Use cost estimates to identify best path $\{N_{\text{root}}, \ldots, N_{\text{target}}\}$
8: **if** no paths exist **then**
9: Apply safety action and **goto** line 17
10: **end if**
11: Repropagate the path state distributions using (33)-(34)
12: **if** repropagated best path is probabilistically feasible **then**
13: Apply best path
14: **else**
15: Remove infeasible portion of best path and **goto** line 7
16: **end if**
17: $t \leftarrow t + \Delta t$
18: **end while**

Remark 1 (conservativeness). It has been noted that (18) introduces a level of conservativeness which increases linearly in the number of constraints.[6] This can be mitigated by using (40), which computes a separate probability of collision for each obstacle. It is also possible to heuristically reduce conservatism by, for example, only including obstacles near the distribution mean. Though the guarantee of probabilistic feasibility is lost, such heuristic methods cannot be easily formulated in an optimization-based framework.

C. Execution Loop

For environments which are dynamic and uncertain, the RRT tree must keep growing during the execution cycle to account for changes in the situational awareness.[25] Given the extensive computations involved to construct the tree, as much of the tree should be retained as possible, especially for real-time applications. Algorithm 2 shows how the algorithm executes some portion of the tree while continuing to grow it.

The planner updates the current best path to be executed by the system every Δt seconds. During each cycle, for the duration of the time step, the tree is repeatedly expanded using Algorithm 1 (lines 4-6). Following this tree growth, the cost estimates are used to select the "best" path in the tree (line 7). The cost metric may be selected at the user's discretion, but typically involves minimizing length/duration, risk (as captured in the heuristic $\Delta_{\max}$), and/or other factors. Once a path is chosen, a "lazy check"[26] is performed in which the the path is repropagated from the current state distribution using the same model dynamics (33)-(34) (line 11) and tested for probabilistic feasibility. If this path is still probabilistically feasible, it is chosen as the current path to execute (lines 12–13). Otherwise, the infeasible portion of the path is removed and the process is repeated (lines 14-15) until either a probabilistically feasible path is found or the entire tree is pruned. If the latter case occurs, the system has no path to execute, and some "safety" motion primitive is applied to attempt to keep the vehicle in a safe state (e.g. come to a stop).

In real-world scenarios, it is expected that as the system executes a path, measurements are received which allow it to reduce its uncertainty in its own state. Whenever these measurements are received, it is appropriate to update the state distributions for the entire tree, as this will reduce the uncertainty at each future time step (lines 4–5).

Remark 2 (feedback). In the absence of feedback on sensor measurements and/or disturbance realizations, the covariance at each node may grow quite large. In particular, for an unstable system, the dispersion of the state distribution will continue to grow without bound. Recent results have demonstrated the importance of incorporating feedback on such information in maintaining manageable levels of uncertainty.[9] In particular, if bounded error (deviation from nominal path) can be guaranteed, the implication in this work is that the covariance will be bounded, as well.[28]

VI. Simulation Results

Simulation results are now presented which demonstrate the effectiveness of the chance constrained RRT approach in efficiently computing paths for motion planning problems which satisfy probabilistic constraints. Several key points are demonstrated through these results. First, without chance constraints, the RRT algorithm is not incorporating knowledge of the uncertainty environment, and may select paths which are excessively risky. Second, as p_{safe} increases, the algorithm selects more conservative paths, which are less likely to collide with an obstacle but require additional length/time to reach the goal. Finally, it is shown that this approach scales favorably in the number of obstacles considered.

Consider the operation of a double integrator (quadrotor) in a two-dimensional non-convex environment. The system dynamics are

$$x_{t+1} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} \frac{dt^2}{2} & 0 \\ 0 & \frac{dt^2}{2} \\ 1 & 0 \\ 0 & 1 \end{bmatrix} u_t + w_t, \qquad \left(x_t = \begin{bmatrix} x_t \\ y_t \\ v_t^x \\ v_t^y \end{bmatrix} \right)$$

where $dt = 0.1$s, subject to input constraints $\mathcal{U} = \{(u^x, u^y) \mid |u^x| \leq 1, |u^y| \leq 1\}$. The state constraints $\mathcal{X}$ consist of speed bounds ($|v_t^x| < 0.5$ and $|v_t^y| < 0.5$) and obstacle avoidance constraints at each time step; it is tacitly assumed that the vehicle is a point mass. The initial state covariance and disturbance covariance are respectively specified as

$$P_w = \begin{bmatrix} 0.002 & 0.001 & 0 & 0 \\ 0.001 & 0.002 & 0 & 0 \\ 0 & 0 & 0.0001 & 0 \\ 0 & 0 & 0 & 0.0001 \end{bmatrix}, \quad P_{x_0} = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Finally, we require that the distribution mean remain within the room bounds for each environment, though it is not necessary to probabilistically satisfy the wall bounding constraints.

The input u is selected at each time step according to the reference law

$$u = K(x_t - r_t), \quad K = \begin{bmatrix} -0.3 & 0 & -0.6 & 0 \\ 0 & -0.3 & 0 & -0.6 \end{bmatrix},$$

where the reference r_t is moved from the parent node waypoint to the sample waypoint x_{samp} at 0.3 m/s. Since this controller is applied both during trajectory simulation (Algorithm 1) and execution (Algorithm 2),[28] $A + BK$ is used in place of A in (34).

Simulations were performed using an implementation of Algorithms 1-2 in Java, run on an Intel 2.53 GHz quad-core desktop with 4GB of RAM, with $\Delta t = 1$s. In the experiments that follow, we assume that no measurements are taken (line 3 of Algorithm 2); thus the state distribution at each node is fixed for all time steps. Each simulation uses one of three algorithms: nominal RRT, offline CC-RRT, or online CC-RRT. Please refer to Section V-B for further details on the distinctions between these algorithms.

A. Simple Scenario

First, consider a square environment containing four obstacles in fixed, known locations (Figure 2); thus the extensions of Section IV are not needed. Five cases are considered, with 10 trials performed for each:

- Nominal RRT
- Offline CC-RRT with $p_{\text{safe}} = 0.5$
- Online CC-RRT with $p_{\text{safe}} = 0.5$
- Online CC-RRT with $p_{\text{safe}} = 0.9$
- Online CC-RRT with $p_{\text{safe}} = 0.99$

In each trial, the planner is given 20 seconds to begin growing the tree. After this planning time has expired, the vehicle selects the lowest-cost path in the tree to execute, then continues to perform the RRT algorithm in real-time, simultaneously growing the tree while executing paths within it.

Figures 2–3 show a sample tree for each case after the 20 seconds of computation; in the latter figure, a 2-σ uncertainty ellipse is also displayed for each node.[a] Since the nominal RRT algorithm is naive to any disturbances which may be present, the trajectories are allowed to come arbitrarily close to obstacles. However, since the nominal RRT algorithm also requires deterministic, not probabilistic, constraint satisfaction, no trajectory ever intersects any of the obstacles (Figure 2). This may not necessarily be the case for CC-RRT, especially for low values of p_{safe}.

In the case of online CC-RRT with $p_{\text{safe}} = 0.5$ (Figure 3(b)), the algorithm typically finds every homotopically distinct path between the obstacles to the goal. In general, for $p_{\text{safe}} = 0.5$, a state distribution will probabilistically satisfy a single constraint if the distribution mean also satisfies the constraint. As a result, generalizing back to the multi-obstacle, multi-constraint case of Figure 3(b), the observed behavior approximates the behavior of nominal RRT. As p_{safe} increases for online CC-RRT from 0.5 to 0.9 (Figure 3(c)), the probabilistic constraints tighten and restrict the feasible configuration space of the vehicle. Whereas the $p_{\text{safe}} = 0.5$ case identifies many paths to goal between the obstacles, the $p_{\text{safe}} = 0.9$ case only identifies two, and cannot traverse the narrowest gap between the bottommost obstacles. Additionally, observe that the uncertainty ellipses in Figure 3(b) significantly intersect the obstacles, whereas any intersection is minimal in Figure 3(c). In the extreme case of $p_{\text{safe}} = 0.99$ (Figure 3(d)), all trajectories take the wider corridors around the obstacles to reach the goal, at the expense of a longer path duration. Finally, note that while both Figures 3(a) and 3(b) use $p_{\text{safe}} = 0.5$ and provide the same guarantee of probabilistic feasibility, the offline CC-RRT algorithm (Figure 3(a)) yields a much more conservative result, more closely resembling online CC-RRT with $p_{\text{safe}} = 0.99$.

Table 1 presents the averaged results over the 10 trials for each case. The nominal RRT algorithm achieves the shortest average path to goal, but is also oblivious to the risk posed by uncertainty in the formulation, safely reaching the goal only in a single trial. For the CC-RRT cases, as p_{safe} increases, the average path length increases, as does the likelihood of safely reaching the goal. This is the expected behavior, since the CC-RRT algorithm either explicitly or implicitly tightens the configuration space, increasing the length of the best path to goal, in order to ensure a higher likelihood of feasibility. In particular, note that for $p_{\text{safe}} = 0.99$, online CC-RRT safely reached the goal in all trials. As seen in the figures, the performance of offline CC-RRT with $p_{\text{safe}} = 0.5$ is comparable to online CC-RRT with higher values of p_{safe}. Finally, we observe that while there is a modest runtime increase for offline CC-RRT and a slightly larger runtime increase for online CC-RRT, both are competitive with the average runtime for nominal RRT.

[a] Due to the system's closed-loop nature, the state distributions quickly converge to a steady-state value. As a result, most uncertainty ellipses appear to be identical in the figures. However, there is indeed an evolution in the

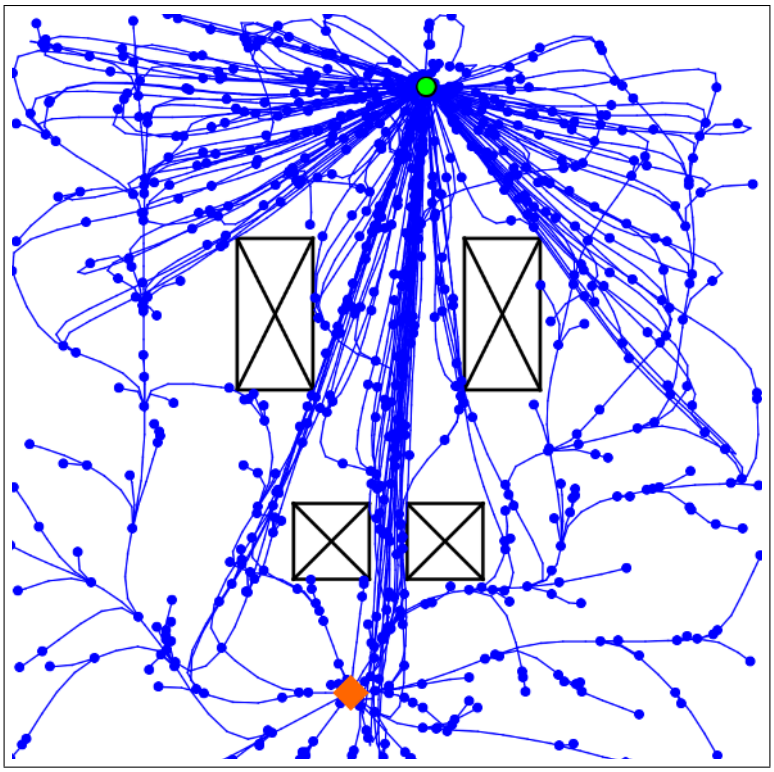


Figure 2. Sample tree (blue) generated by the nominal RRT algorithm for the simple scenario environment. In simulation figures, the vehicle seeks to find a (probabilistically) feasible path from its starting location (orange diamond) to the goal (green circle) while avoiding all obstacles (black).

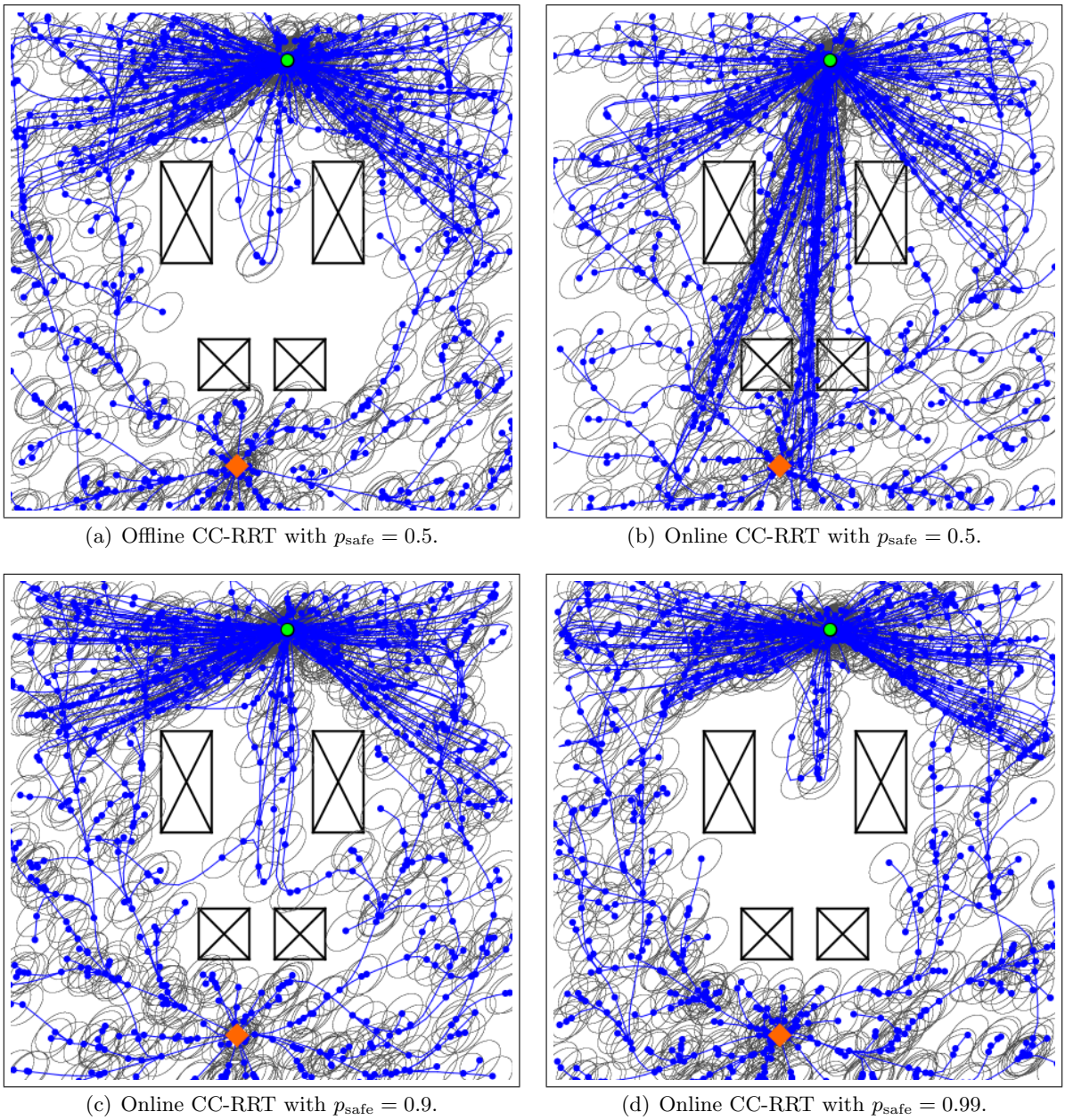
B. Cluttered Scenario

The main driver of the computational complexity of the CC-RRT algorithm is the number of obstacles; the purpose of this scenario is to demonstrate how the runtime scales with the number of obstacles. In this scenario, all parameters from Section VI-A are maintained except for the environment itself, which is replaced with a more cluttered environment (Figure 4). Whereas the simple scenario has only 4 obstacles, the cluttered scenario has 20, an increase by a factor of 5.

An identical set of trials was performed as in Section VI-B for the same five cases. Figures 4–5 show a sample tree for each case after the 20 seconds of computation; very similar conclusions can be drawn as from the simple scenario. Of note is the uppermost path to goal found by online CC-RRT with $p_{\text{safe}} = 0.5$ (Figure 5(b)), passing along the top of the figure. While the probability of collision at each time step must not exceed 0.5, in practice such a path would be extremely difficult to execute safely.

Table 2 presents the averaged results over the 10 trials for each case. The average runtime per node has increased by a factor between 3 and 4 in each case, with the nominal RRT and offline CC-RRT cases scaling slightly less than the online CC-RRT cases. Nonetheless, this data provides empirical evidence that the additional computation needed for the CC-RRT algorithm scales approximately as a fixed percentage of the nominal RRT algorithm. Otherwise, the results are similar to the simple scenario (Table 1), with online CC-RRT again achieving feasibility across all trials for $p_{\text{safe}} = 0.99$. On the other hand, online CC-RRT performs poorly in this scenario for $p_{\text{safe}} = 0.5$, with only one trial safely reaching the goal.

uncertainty ellipse size, starting from the localization error at the root.



(a) Offline CC-RRT with $p_{\text{safe}} = 0.5$.

(b) Online CC-RRT with $p_{\text{safe}} = 0.5$.

(c) Online CC-RRT with $p_{\text{safe}} = 0.9$.

(d) Online CC-RRT with $p_{\text{safe}} = 0.99$.

Figure 3. Sample tree generated by the CC-RRT algorithm for the simple scenario environment. Each node corresponds to the state distribution mean; a 2-σ uncertainty ellipse is centered at each node.

Table 1. Simulation Results, Simple Scenario

Algorithm	p_{safe}	Time per Node, ms[a]	Path Found?[b]	Path Duration, s[c]	Safe to Goal?[d]
Nominal RRT	N/A	1.38	10/10	27.2	1/10
Offline CC-RRT	0.5	1.51	10/10	34.3	8/10
Online CC-RRT	0.5	1.94	10/10	27.2	3/10
Online CC-RRT	0.9	2.04	10/10	34.9	7/10
Online CC-RRT	0.99	2.06	10/10	39.8	10/10

Table 2. Simulation Results, Cluttered Scenario

Algorithm	p_{safe}	Time per Node, ms[a]	Path Found?[b]	Path Duration, s[c]	Safe to Goal?[d]
Nominal RRT	N/A	4.45 (×3.2)	10/10	22.6	1/10
Offline CC-RRT	0.5	4.98 (×3.3)	10/10	25.6	7/10
Online CC-RRT	0.5	7.56 (×3.9)	10/10	22.7	1/10
Online CC-RRT	0.9	7.38 (×3.6)	10/10	24.3	9/10
Online CC-RRT	0.99	7.75 (×3.8)	10/10	25.9	10/10

[a] Cumulative time spent in Algorithm 1 divided by the number of nodes generated.
[b] Number of trials where path to goal was found after initial 20 seconds of tree growth.
[c] Duration of initial path to goal, if one exists.
[d] Number of trials where system executed a path to goal without colliding with any obstacles. (Recall that p_{safe} refers to feasibility for a single *time step*, whereas this entry corresponds to feasibility across the entire *path*.)

C. Uncertain Obstacle Scenario

This scenario demonstrates the capability of the CC-RRT algorithm to incorporate uncertain obstacles in its probabilistic feasibility computations. The environment is the same as in the first scenario (Figure 2). The disturbance covariance is significantly reduced (from P_w to $\bar{P}_w \equiv P_w/100$), while the obstacles are now uncertain and thus have their own probability distributions. (The case of an uncertain *and* dynamic obstacle is omitted for brevity.) The upper-left obstacle distribution has covariance P_a, while all other obstacle distributions have covariance P_b, where

$$P_a = \begin{bmatrix} 0.2 & 0 & 0 & 0 \\ 0 & 0.2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad P_b = \frac{P_a}{200}.$$

Consider running the online CC-RRT algorithm on this scenario with $p_{\text{safe}} = 0.99$; even though the process noise has been significantly reduced, the tree trajectories must be very conservative in path selection to ensure the probability of collision at any time step does not exceed 1%. Figure 6 shows a representative tree generated by this algorithm after one minute of tree expansion. In this figure, the 2-σ uncertainty ellipse is shown both for every obstacle and for every node; it is clear that the only major sources of uncertainty are the residual localization error near the start and the upper-left obstacle. Whereas trajectories can come very close to the three rightmost obstacles and maintain probabilistic feasibility, the high uncertainty of the upper-left obstacle causes any trajectory which comes within a box's width of it (approximately 1m) to become probabilistically infeasible.

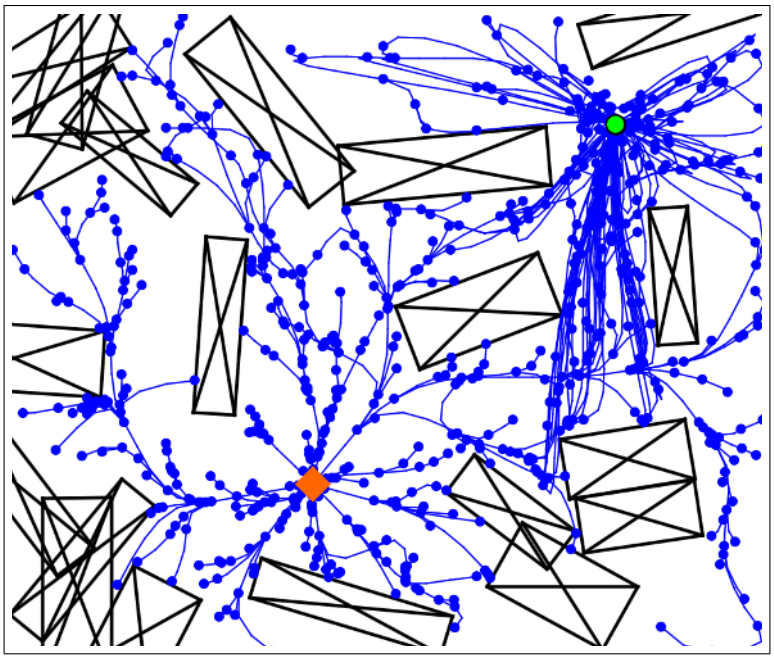


Figure 4. Sample tree generated by the nominal RRT algorithm for the cluttered scenario environment.

VII. Conclusions

This paper has presented a novel sampling-based algorithm, CC-RRT, which allows for efficient computation of probabilistically feasible paths through a non-convex environment subject to uncertainty, including process noise and uncertain, possibly dynamic obstacles. If the tightened constraints are computed off-line for a fixed probability bound, the complexity of the CC-RRT algorithm is essentially unchanged relative to the nominal RRT algorithm. Alternatively, a small amount of additional computation can be used to explicitly compute a tight probability bound at each time step, providing the user with a metric to directly control the level of conservatism in the planning approach. Furthermore, as demonstrated through simulation results, the approach is scalable in the number of obstacles, allowing for efficient computation of safe paths even in heavily cluttered environments. Future work will consider alternative representations of obstacle uncertainty, such as probability distributions on the obstacle vertices or obstacle process noise,[9] as well as incorporation of better approximations of path feasibility.

Acknowledgments

Research funded in part by AFOSR grant FA9550-08-1-0086. The authors would like to recognize Daniel Levine and Aditya Undurti for their contributions to this research effort.

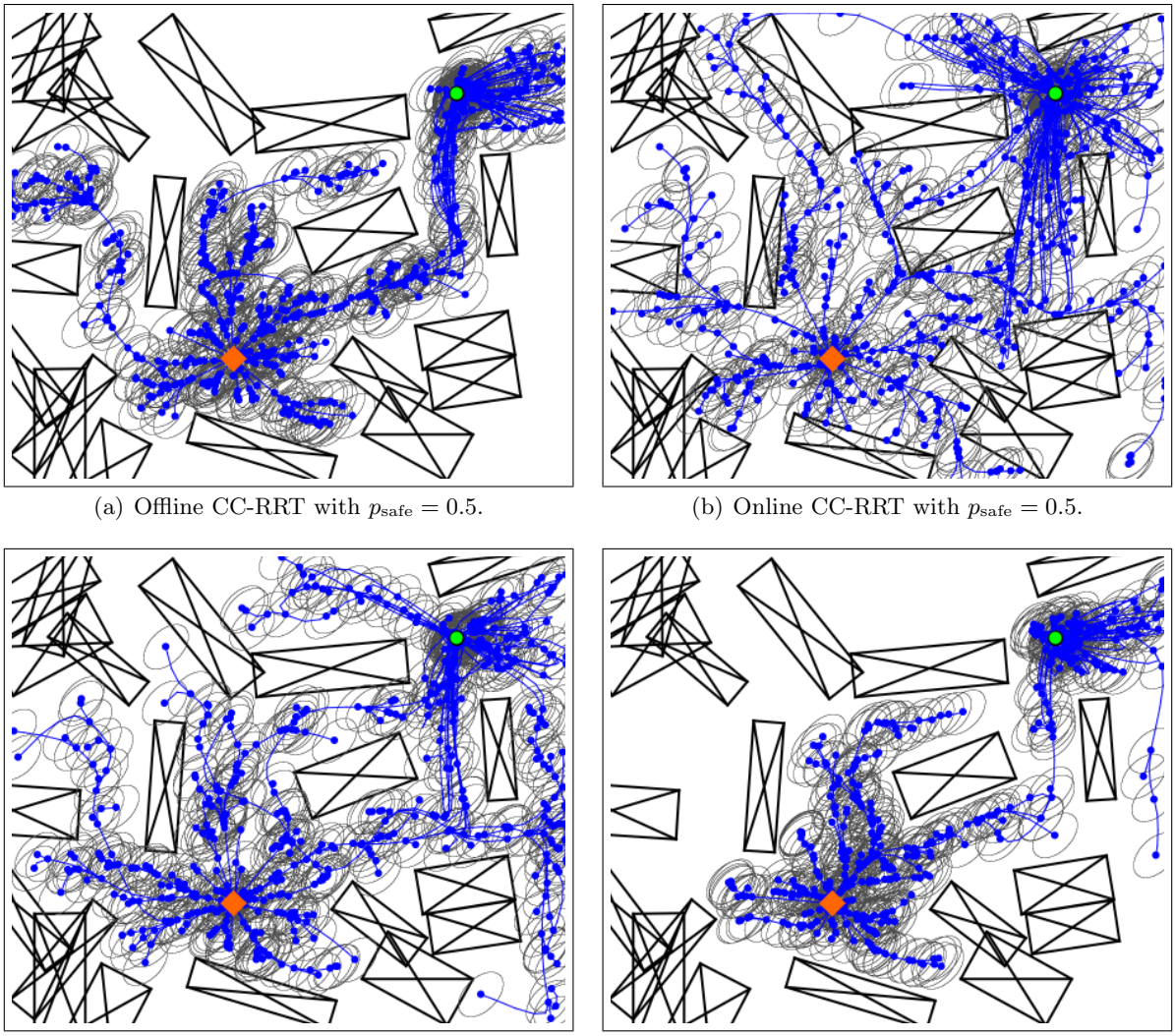
References

[1]LaValle, S. M., *Planning Algorithms*, Cambridge University Press, Cambridge, U.K., 2006.

[2]LaValle, S. M. and Sharma, R., "On Motion Planning in Changing, Partially-Predictable Environments," *International Journal of Robotics Research*, Vol. 16, No. 6, 1995, pp. 775–824.

[3]Li, P., Wendt, M., and Wozny, G., "A probabilistically constrained model predictive controller," *Automatica*, Vol. 38, 2002, pp. 1171–1176.

[4]van Hessem, D. H., *Stochastic inequality constrained closed-loop model predictive control with application to chemcial process operation*, Ph.D. thesis, Technische Universiteit Delft, June 2004.



(a) Offline CC-RRT with $p_{\text{safe}} = 0.5$.

(b) Online CC-RRT with $p_{\text{safe}} = 0.5$.

(c) Online CC-RRT with $p_{\text{safe}} = 0.9$.

(d) Online CC-RRT with $p_{\text{safe}} = 0.99$.

Figure 5. Sample tree generated by the CC-RRT algorithm for the cluttered scenario environment. Each node corresponds to the state distribution mean; a 2-σ uncertainty ellipse is centered at each node.

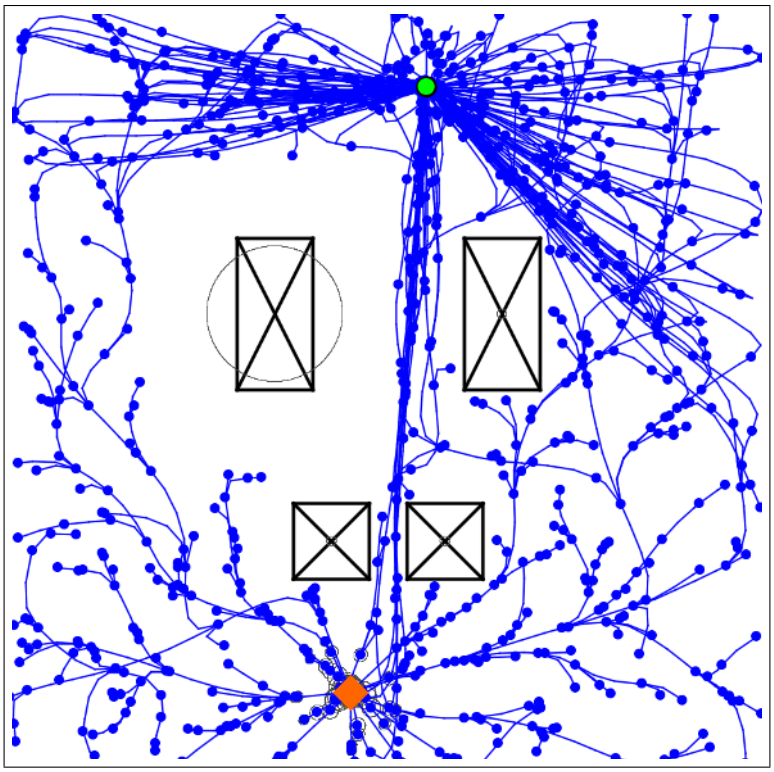


Figure 6. Sample tree generated by the by the online CC-RRT algorithm with $p_{\mathbf{safe}} = 0.99$ for the uncertain obstacle scenario environment. Each node corresponds to the state distribution mean; a 2-σ uncertainty ellipse is centered at each node (most of which are too small to be visible).

[5]Yan, Y. and Bitmead, R. R., "Incorporating state estimation into model predictive control and its application to network traffic control," *Automatica*, Vol. 41, 2005, pp. 595–604.

[6]Blackmore, L., Li, H., and Williams, B., "A Probabilistic Approach to Optimal Robust Path Planning with Obstacles," *Proceedings of the IEEE American Control Conference*, 2006, pp. 2831–2837.

[7]Blackmore, L., "Robust Path Planning and Feedback Design under Stochastic Uncertainty," *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Honolulu, HI, August 2008.

[8]Blackmore, L. and Ono, M., "Convex Chance Constrained Predictive Control without Sampling," *Proceedings of the AIAA Guidance, Navigation and Control Conference*, 2009.

[9]Toit, N. E. D., *Robot Motion Planning in Dynamic, Cluttered, and Uncertain Environments: the Partially Closed-Loop Receding Horizon Control Approach*, Ph.D. thesis, California Institute of Technology, November 2005.

[10]LaValle, S. M., "Rapidly-Exploring Random Trees: A New Tool for Path Planning," Tech. Rep. 98-11, Iowa State University, October 1998.

[11]Leonard, J., How, J., Teller, S., Berger, M., Campbell, S., Fiore, G., Fletcher, L., Frazzoli, E., Huang, A., Karaman, S., Koch, O., Kuwata, Y., Moore, D., Olson, E., Peters, S., Teo, J., Truax, R., Walter, M., Barrett, D., Epstein, A., Maheloni, K., Moyer, K., Jones, T., Buckley, R., Antone, M., Galejs, R., Krishnamurthy, S., and Williams, J., "A Perception-Driven Autonomous Urban Vehicle," *Journal of Field Robotics*, Vol. 25, No. 10, 2008, pp. 727–774.

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

[13]Roy, N., Gordon, G., and Thrun, S., "Planning under Uncertainty for Reliable Health Care Robotics," *Field and Service Robotics*, Vol. STAR 24, 2006, pp. 417–426.

[14]Foka, A. and Trahanias, P., "Real-time hierarchical POMDPs for autonomous robot navigation," *Robotics and Autonomous Systems*, Vol. 55, 2007, pp. 561–571.

[15]Greytak, M. and Hover, F., "Motion Planning with an Analytic Risk Cost for Holonomic Vehicles," *Proceedings of the IEEE Conference on Decision and Control*, 2009.

[16]Miralles, A. S. and Bobi, M. A. S., "Global Path Planning in Gaussian Probabilistic Maps," *Journal of Intelligent and Robotic Systems*, Vol. 40, 2004, pp. 89–102.

[17]Burns, B. and Brock, O., "Sampling-Based Motion Planning With Sensing Uncertainty," *Proceedings of the IEEE International Conference on Robotics and Automation*, Roma, Italy, April 2007, pp. 3313–3318.

[18]Missiuro, P. E. and Roy, N., "Adapting Probabilistic Roadmaps to Handle Uncertain Maps," *Proceedings of the IEEE International Conference on Robotics and Automation*, Orlando, FL, May 2006, pp. 1261–1267.

[19]Guibas, L. J., Hsu, D., Kurniawati, H., and Rehman, E., "Bounded Uncertainty Roadmaps for Path Planning," *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics*, 2008.

[20]Melchior, N. A. and Simmons, R., "Particle RRT for Path Planning with Uncertainty," *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007.

[21]Kewlani, G., Ishigami, G., and Iagnemma, K., "Stochastic Mobility-based Path Planning in Uncertain Environments," *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, St. Louis, MO, USA, October 2009, pp. 1183–1189.

[22]Pepy, R. and Lambert, A., "Safe Path Planning in an Uncertain-Configuration Space using RRT," *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, October 2006, pp. 5376–5381.

[23]Pepy, R., Kieffer, M., and Walter, E., "Reliable Robust Path Planning," *International Journal of Applied Math and Computer Science*, Vol. 1, 2009, pp. 1–11.

[24]Prentice, S. and Roy, N., "The Belief Roadmap: Efficient planning in lienar POMDPs by factoring the covariance," *Proceedings of the International Symposium of Robotics Research*, Hiroshima, Japan, November 2007.

[25]Frazzoli, E., Dahleh, M. A., and Feron, E., "Real-Time Motion Planning for Agile Autonomous Vehicles," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 1, January-February 2002, pp. 116–129.

[26]Kuwata, Y., Teo, J., Fiore, G., Karaman, S., Frazzoli, E., and How, J. P., "Real-time Motion Planning with Applications to Autonomous Urban Driving," *IEEE Transactions on Control Systems Technology*, Vol. 17, No. 5, September 2009, pp. 1105–1118.

[27]Ono, M. and Williams, B. C., "Iterative Risk Allocation: A New Approach to Robust Model Predictive Control with a Joint Chance Constraint," *Proceedings of the IEEE Conference on Decision and Control*, 2008.

[28]Luders, B. D., Karaman, S., Frazzoli, E., and How, J. P., "Bounds on Tracking Error using Closed-Loop Rapidly-Exploring Random Trees," *Proceedings of the IEEE American Control Conference*, June-July 2010.