Robust Sampling-based Motion Planning for Autonomous Vehicles in Uncertain Environments

by
Brandon Luders
B.S., Aerospace Engineering
Georgia Institute of Technology (2006)
S.M., Aeronautics and Astronautics
Massachusetts Institute of Technology (2008)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June 2014

Author ..... Signature redacted .....
Department of Aeronautics and Astronautics
May 16, 2014

Certified by ..... Signature redacted .....
Jonathan P. How
Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by ..... Signature redacted
Emilio Frazzoli
Professor of Aeronautics and Astronautics

Certified by ..... Signature redacted ....
Karl Iagnemma
Principal Research Scientist, Mechanical Engineering

Certified by ..... Signature redacted ..
Sertac Karaman
Assistant Professor of Aeronautics and Astronautics

Accepted by ..... Signature redacted .....
Paulo C. Lozano
Associate Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Robust Sampling-based Motion Planning for Autonomous Vehicles in Uncertain Environments

by

Brandon Luders

Submitted to the Department of Aeronautics and Astronautics
on May 16, 2014, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

While navigating, autonomous vehicles often must overcome significant uncertainty in their understanding of the world around them. Real-world environments may be cluttered and highly dynamic, with uncertainty in both the current state and future evolution of environmental constraints. The vehicle may also face uncertainty in its own motion. To provide safe navigation under such conditions, motion planning algorithms must be able to rapidly generate smooth, certifiably robust trajectories in real-time.

The primary contribution of this thesis is the development of a real-time motion planning framework capable of generating feasible paths for autonomous vehicles in complex environments, with robustness guarantees under both internal and external uncertainty. By leveraging the trajectory-wise constraint checking of sampling-based algorithms, and in particular rapidly-exploring random trees (RRT), the proposed algorithms can efficiently evaluate and enforce complex robustness conditions.

For linear systems under bounded uncertainty, a sampling-based motion planner is presented which iteratively tightens constraints in order to guarantee safety for all feasible uncertainty realizations. The proposed bounded-uncertainty RRT* (BU-RRT*) algorithm scales favorably with environment complexity. Additionally, by building upon RRT*, BU-RRT* is shown to be asymptotically optimal, enabling it to efficiently generate and optimize robust, dynamically feasible trajectories.

For large and/or unbounded uncertainties, probabilistically feasible planning is provided through the proposed chance-constrained RRT (CC-RRT) algorithm. Paths generated by CC-RRT are guaranteed probabilistically feasible for linear systems under Gaussian uncertainty, with extensions considered for nonlinear dynamics, output models, and/or non-Gaussian uncertainty. Probabilistic constraint satisfaction is represented in terms of chance constraints, extending existing approaches by considering both internal and external uncertainty, subject to time-step-wise and path-wise feasibility constraints. An explicit bound on the total risk of constraint violation is developed which can be efficiently evaluated online for each trajectory. The proposed CC-RRT* algorithm extends this approach to provide asymptotic optimality guaran-

tees; an admissible risk-based objective uses the risk bounds to incentivize risk-averse trajectories.

Applications of this framework are shown for several motion planning domains, including parafoil terminal guidance and urban navigation, where the system is subject to challenging environmental and uncertainty characterizations. Hardware results demonstrate a mobile robot utilizing this framework to safely avoid dynamic obstacles.

Thesis Supervisor: Jonathan P. How
Title: Professor of Aeronautics and Astronautics

Acknowledgments

No thesis exists in a vacuum. While it may list a single author under the title, it cannot come into existence by that one person alone. For me, this thesis is about much more than presenting the contributions and results of my doctoral research. It is a testament to the many people who have played a role in helping me complete this journey. Indeed, it is only through the support of such wonderful colleagues, friends, and family that I have been able to reach this point.

First, I would like to thank my advisor, Prof. Jonathan How, for his guidance, support, and patience throughout my time at MIT. His direction and insight – including an uncanny ability to find new ways to tackle research challenges as they pop up – have been invaluable throughout my graduate career. I am grateful to have learned from an advisor so strongly committed to his students and their growth as researchers.

I would like to thank Prof. Emilio Frazzoli, Prof. Sertac Karaman, and Dr. Karl Iagnemma for their participation in my thesis committee, and the many useful suggestions and ideas they have provided along the way. I would also like to thank Dr. Sameera Ponda and Dr. Ali-akbar Agha-mohammadi, for providing much useful feedback as my thesis readers, and Prof. David Mindell, for serving as department representative during my thesis defense. I must also acknowledge Prof. Robert Freund, who not only serves as my thesis minor advisor, but has been a great source of support during my tenure as Nonlinear Programming TA and beyond. I have also been lucky to work with such a resourceful administrative staff at MIT, including Kathryn Fischer, Beth Marois, Quentin Alexander, and Barbara Lechner.

It has been a privilege to work with my colleagues in the Aerospace Controls Lab. My interactions with my ACL labmates have led to not only useful research collaborations, but also motivation, inspiration, and enduring friendships. In the early years of my doctoral research, I worked with Dan Levine to prototype the initial planning software infrastructure upon which much of my research is built. I am grateful for his participation in these efforts and his thoughts on information-

based motion planning. It was a pleasure to work with Vishnu Desaraju on the Agile Robotics program and planning applications. My collaborations with Ian Sugel and Aaron Ellertson on parafoil terminal guidance have been both fruitful and enjoyable. Luke Johnson provided many useful insights on planning for Dubins vehicles. Georges Aoude developed the many of the prediction models used in this work to demonstrate robust planning with dynamic obstacles, and has been a source of both successful research and motivation throughout my thesis. My joint work with Sameera Ponda doing hardware experiments at Cornell was particularly memorable. Both Sameera and Tuna Toksoz have continued to be a major source of support well past their years at MIT. Finally, I have collaborated frequently with Sarah Ferguson in considering dynamic obstacles with more complex behavior. She has been essential in providing support and hardware/sensor expertise for hardware experiments, and it has been a pleasure to work with her.

I am greatly appreciative to know Ann and Terry Orlando, who served for many years as housemasters of Ashdown House, my residence throughout my doctoral candidacy. Ann and Terry provided essential advice at key junctures in my graduate career, and are some of the best advocates for graduate students I have ever met at MIT.

I am thankful for my friendships with those at MIT who have made my years here fun and memorable, including Josh Joseph, Jaime Mateus, Amanda Zangari, Allie Anderson, and Andy Wright.

Andrea Dubin has been an incredible source of motivation and a wonderful friend. Her friendship has turned what could have been a stressful and daunting final year into a fun and joyful one, for which I am extremely grateful.

Finally, I thank my wonderful and supportive family, whose love and encouragement truly made this possible. Mom and Dad, thank you for your selflessness and unconditional support through all the highs and lows. Chris and Bria, thank you for being such great friends. I am blessed to have you all in my life.

Research funded in part by Ford Motor Company, through the Ford-MIT Alliance; Draper Laboratory, through the University Research and Development Program; the United States Army Logistics Innovation Agency; and The Boeing Company.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Autonomous vehicles (Figure 1-1) face several key challenges when performing real-time path planning in complex, real-world scenarios [1, 2]. Such scenarios typically include a vehicle represented with complex, non-holonomic dynamics, subject to cluttered environments with a large number and variety of constraints. Perhaps most importantly, such environments tend to be dynamic and highly uncertain. This necessitates the consideration of both online planning responses to a changing environment, and uncertainty incorporation within any decision-making for the vehicle.

For motion planning algorithms to be deployable on real-world autonomous systems, it is essential that they be able to quickly identify feasible paths in these highly complex, dynamic, and uncertain environments. Thus, a key research focus has been the development of algorithms which can explicitly model many of the possible sources of uncertainty and incorporate them directly within the planner. This may include both internal uncertainty (*e.g.*, sensing/process noise, localization error, model uncertainty) and external uncertainty (*e.g.*, placement of static obstacles, future behavior of dynamic obstacles) [3].

Significant advances have been made in motion planning technology for various autonomous vehicles in the past decade, much of it spurred by the DARPA Grand Challenge [4] and later DARPA Urban Challenge [1, 5]. However, existing motion

(a) MIT DARPA Urban Challenge Vehicle (b) Agile Robotics autonomous forklift

Figure 1-1: Examples of autonomous vehicles utilizing outdoor navigation

planning technology still often relies on simplifying assumptions, such as limited interaction with other dynamic obstacles; assumed, precise knowledge of the world map; and/or ignorance of significant uncertainty. Thus, the objective of this thesis is to develop a real-time motion planner capable of guiding vehicles through complex and uncertain environments with certifiably robust behavior, enabling real-world applications. In this context, certifiable robustness is defined being able to provide a guaranteed minimum likelihood of constraint satisfaction, needed for safe operation.

Such a motion planner faces several major technical challenges. It must be able to admit problems of arbitrary complexity, including dynamics, complex terrain, incomplete environmental knowledge, logical/timing constraints, and other constraints. It must be able to operate safely in the presence of multiple sources of uncertainty. Finally, it must be able to provide a means of guaranteeing safe operation under such conditions.

1.2 Literature Review

A wide variety of approaches have been developed for motion planning under uncertainty [6, 7]; as such, discussion of existing work can be classified and contrasted along several axes. First, motion planning algorithms can be discussed in terms of which forms of uncertainty are incorporated into the planning problem. Using the frame-

work of LaValle and Sharma [3], algorithms can be categorized by whether they incorporate internal configuration/current-state uncertainty (*e.g.,* sensing noise), internal predictive uncertainty (*e.g.,* process noise), environmental configuration uncertainty (*e.g.,* placement of static obstacles), and/or environmental predictive uncertainty (*e.g.,* future behavior of dynamic obstacles).

Second, algorithms can be partitioned in terms of the assumptions made on the uncertainty model(s), and the corresponding feasibility guarantees. If the uncertainty is large or unbounded, guaranteed feasibility may be unlikely or impossible. Thus, many approaches establish *probabilistic* feasibility: that the likelihood of feasibility for each timestep and/or trajectory must exceed some minimum bounds. The probability distribution of the uncertainty must be known, and is often assumed Gaussian. On the other hand, if all sources of uncertainty are bounded, it may be possible to provide *guaranteed* feasibility against any possible disturbance realization. In this case, it is sufficient to know only the bounds on such uncertainty. The intermediate chapters of this thesis are divided based on which formulation type is being considered.

Finally, motion planning algorithms can be characterized by whether they are *optimization-based, e.g.,* seeking to find the lowest-cost solution over the entire feasible solution space, or *sampling-based, e.g.,* incrementally constructing a set of feasible solutions. The treatment of related work in this section is divided into these two groups, exploring the relative strengths and weaknesses of each approach. Preceding this, however, is a discussion of some of the early techniques developed for motion planning in dynamic and/or uncertain environments.

1.2.1 Motion Planning Algorithms

Motion planning implementations for autonomous vehicles and other mobile robots often face two key challenges. First, they must be able to generate solution paths which are dynamically feasible, *e.g.,* able to be reasonably followed by the autonomous agent. While this kinodynamic motion planning problem can be especially challenging for non-holonomic dynamics, provably accurate approximation algorithms have been developed [8]. A common simplified model for autonomous vehicles assumes a fixed

velocity and bounded curvature, in which case optimal solutions are known for both forward motion [9] and mixed forward/reverse motion [10].

Second, such motion planners must be able to reason about a changing uncertain environment, with dynamic obstacles often being the primary source of uncertainty. A simple approach to predicting the future state of a dynamic obstacle is the velocity obstacle [11], which assumes the agent will maintain its current speed and bearing indefinitely. Other approaches developed include the dynamic window [12] and inevitable collision states (ICS), representing all regions of the configuration space where a collision cannot be avoided [13]. The ICS approach has been more recently extended to reason probabilistically about collisions [14, 15].

1.2.2 Optimization-based Motion Planning

In optimization-based motion planning algorithms, the objective is to identify the solution path or policy that optimizes some specified cost function, over all feasible solutions. A common approach to handling uncertainty in such frameworks is to perform replanning as the problem evolves and uncertainties are realized. However, replanning alone is often insufficient to ensure safe traversal, motivating the incorporation of uncertainty models directly into the formulation.

In model predictive control [16], a model of predicted system behavior is used to optimize an open-loop, finite-horizon input sequence which satisfies a set of explicit constraints, often appended with some cost-to-go function. This optimization is often performed repeatedly and iteratively, also known as receding horizon control (RHC). Because constraints are encoded directly into the optimization, MPC-based motion plans can operate at or near constraint boundaries while maintaining feasibility. However, the choice of optimization engine often restricts what types of constraints may be considered.

A common technique used for motion plan optimization is mixed-integer linear programming (MILP), which provides a very flexible framework for modeling planning problems with both discrete decisions and continuous variables. A linear program is transformed into a MILP by requiring at least one of the decision variables to satisfy

an integer value, typically binary. In the case of non-convex motion planning, binary variables can be used to represent the avoidance of convex obstacles, making MILP-based approaches suitable for this domain [17, 18].

The D* algorithm was also developed as a generalization of A* graph search for dynamic environments [19]. Many variations of D* have been introduced for planning applications, including focussed D*, which focuses repair in specific search space regions [20]; D* Lite, which implements D* using a simpler algorithm [21]; Field D*, which allows state/action interpolation [22]; and anytime repairing A* [23].

An alternative formulation is the Markov decision process, which represents the motion planning problem in terms of discrete states, actions, rewards, and a transition function which probabilistically maps actions to state outcomes [24]. Uncertainty in the outcomes of future actions can be represented in this framework; however, the state is assumed to be perfectly known. The objective of an MDP formulation is to identify a policy mapping states to actions in order to maximize expected reward, typically via value/policy iteration [24].

Motion Planning under Uncertainty

A common approach to guaranteeing absolute safety is robust model predictive control (RMPC) [25], the extension of MPC to incorporate set-based uncertainty models. In RMPC, the optimized input sequence must satisfy all constraints subject to an unknown but bounded disturbance sequence. This disturbance is often represented as a parametric uncertainty or additive uncertainty.

For most problems, some kind of feedback on realized disturbances is necessary to prevent RMPC solutions from becoming overly conservative. While it is possible to enumerate all possible worst-case disturbance sequences, such a formulation grows exponentially with the problem size [26]. Thus, a more common approach is to optimize over affine feedback policies, which combine linear feedback on future states and/or disturbances with a vector of open-loop perturbations. Such techniques include constraint tightening via state feedback [27, 28], constraint tightening via disturbance feedback [29], minimax formulations using linear matrix inequalities [30],

and optimization of the linear feedback policy online [31, 32]. Most of these approaches consider robustness to internal process noise only. Chapter 2 builds on and extends several of these RMPC frameworks to consider multiple, additional forms of uncertainty within sampling-based algorithms.

On the other hand, if the uncertainty is large or unbounded, it is no longer possible to generate meaningful paths which guarantee safety for all possible disturbance realizations. The fundamental question is then how to properly trade off between planner conservatism and the risk of infeasibility. A useful way to capture this trade-off is through chance constraints, which require *probabilistic feasibility:* that the probability of constraint violation not exceed some prescribed value [33]. It is possible to model these probabilistic chance constraints in terms of tightened, deterministic constraints on the conditional mean of the state, representing the amount of conservatism necessary to achieve a desired probability of feasibility [34, 35].

A popular approach for probabilistic motion planning in the recent literature is chance-constrained optimization, which wraps these chance constraints within a MILP, second-order cone program (SOCP), or similar optimization in order to ensure probabilistic feasibility. Blackmore *et al.* [36] guarantee probabilistic feasibility for a linear system subject to Gaussian process noise and localization error, by using Boole's inequality (also known as the union bound) to model a non-convex environment as a disjunctive linear program. Subsequent work considers particle-based approximations of the uncertainty [37, 38] as well as feedback design [39]. Of particular note is the 2009 work of Blackmore and Ono [40], which reduces the chance constraints into a univariate form which can be evaluated more efficiently. While this approach is significantly less conservative, it does assume convexity of the constraint set.

Two major sources of conservatism in this chance-constrained optimization framework are the use of Boole's inequality, which establishes a loose bound on constraint violation, and the need to allocate risk among all constraints *a priori.* While the former is generally not restrictive in practice, the latter can significantly increase conservatism as the number of obstacles and other constraints increases. Alternatively, iterative risk allocation (IRA) [41, 42] can reduce this conservatism. In IRA,

an iterative sequence of two-stage optimizations is performed, with the upper stage optimizing the allocation of risk over all constraints, while the lower stage optimizes the control sequence subject to the tightened constraints. These iterations can be computationally intensive, though the recently-proposed p-Sulu planner [43] can mitigate this by decomposing the planning problem into a sequence of convex (and thus much more efficient) optimizations. The p-Sulu planner also allows the incorporation of temporal constraints [43]. Vitus and Tomlin consider hybrid analytic/sampling formulations which incorporate iterative risk allocation within SOCPs, in addition to considering parametric environmental uncertainty [44, 45].

This thesis builds upon and extends this chance constraint framework for linear Gaussian systems, particularly Blackmore *et al.* [36]. By incorporating chance constraints within a sampling-based algorithm, bounds on the risk of constraint violation can be efficiently computed online, rather than requiring offline pre-allocation or iterative allocation [46]. Coupled with the scalability of sampling-based algorithms, the resulting approach is capable of quickly generating probabilistically feasible solutions. This is accomplished while extending the existing formulation to consider additional forms of uncertainty, including both internal and external uncertainty (Chapter 3).

In work concurrent with the author, Du Toit and Burdick [47] develop a chance-constraint-based formulation for estimating the probability of collision between two uncertain agents, applied to both dynamic programming and RHC frameworks. The resulting covariance estimate is similar in form to the one developed as part of this thesis (Chapter 3) for an uncertain system subject to uncertain obstacle placements. However, Du Toit and Burdick use an integral approximation and assume point/disk obstacles, whereas the approach in this thesis provides precise (though conservative) bounds on avoidance for any polytopic obstacle.

An alternate way to capture other forms of uncertainty is the use partially-observable Markov decision processes (POMDP), which extend the MDP formulation to assume the state can only be perceived indirectly through observations [48]. Under these restrictions, POMDPs operate on a belief state rather than the physical state itself. Given the significantly increased dimensionality of this belief state, POMDP-

based formulations are generally intractable for most optimization-based approaches, though recent success has been found computing locally optimal solutions under the assumptions of Gaussian sensing and motion uncertainty [49].

1.2.3 Sampling-based Motion Planning

The optimization-based algorithms described above are capable of generating paths that minimize a cost function over the feasible solution space. However, for motion planning problems involving complex dynamics and constraints and/or high-dimensional configuration spaces, the computational complexity of such optimizations may scale poorly, to the point of becoming intractable. An alternative is to use sampling-based motion planning algorithms, which instead sample a set of feasible paths from the solution space [7]. While the lowest-cost feasible path found will typically be selected for execution, there is no guarantee of optimality in finite time. However, sampling-based approaches have demonstrated several key advantages for complex motion planning problems, including efficient exploration of high-dimensional configuration spaces; trajectory-wise (*e.g.,* non-enumerative) checking of possibly complex constraints; incremental construction, facilitating anytime use; and/or scalability with available computational resources [7]. In particular, trajectory-wise constraint checking enables sampling-based algorithms to scale well with problem complexity, while incremental construction enables the quick identification of feasible paths. Though finite-time optimality is lost, these resulting benefits are often worth the trade-off, and in practice sampling-based algorithms can generate high-quality solutions in real-time (Chapters 2, 5).

Two of the most commonly used sampling-based motion planning algorithms are probabilistic roadmap methods (PRM) and rapidly-exploring random trees (RRT). Probabilistic roadmaps are typically used in a two-phase process, enabling their use in multi-query applications [50]. In the map-building phase, feasible configuration vertices are randomly sampled throughout the configuration space, with edges being constructed between nearby vertices if the path between them is feasible. In the query phase, the map is augmented with the system's current state and goal location,

and a graph search algorithm is used to find the optimal safe path. Variants of PRM include dynamic PRM [51], which distinguishes between static and dynamic obstacles, and differential PRM [52], which incorporates differential constraints. While the map can be used repeatedly, the edges between vertices may not be dynamically feasible for systems with complex dynamics, unless a steering law or similar tool is used to construct edges between vertices. Additionally, a pre-processing phase (map construction) is required for future online planning.

Rapidly-exploring random trees (RRT) incrementally construct a tree of feasible trajectories rooted at the system's current state [53, 54]. Because each tree branch corresponds to a forward simulation of the vehicle dynamics for a specific input sequence, all trajectories are dynamically feasible by construction. (A bi-directional version has also been developed, with trees growing from both the system's current state and the goal state [55].) New trajectories are added to the tree by sampling a feasible state, then simulating a trajectory from the nearest existing node to that sample. In doing so, RRT demonstrates a Voronoi bias which allows it to quickly explore high-dimensional state spaces. However, RRT is a single-query algorithm: the tree must be constantly updated and regrown in real-time applications if uncertainty leads to deviations from the tree.

While RRT is probabilistically complete, it has been shown to converge almost surely to non-optimal solutions [56]. In other words, the difference in cost between the optimal solution and the lowest-cost solution identified by RRT does not converge to zero as the number of tree samples approaches infinity. This limits the ability of RRT to refine feasible solutions once identified, potentially leading to low-quality trajectories heavily biased on initial tree growth. Alternately, the RRT* algorithm incrementally "rewires" the tree as it is constructed toward lower-cost paths [56, 57]. As a result, RRT* provides guarantees on *asymptotic optimality*: the lowest-cost solution identified by RRT* converges to the optimal cost as the number of tree samples approaches infinity. As a trade-off, a steering law must be available to exactly or approximately connect pairs of states within the tree, which may be computationally intensive and/or unavailable for more complex dynamics.

RRT* has been extended to consider online execution [58], non-holonomic dynamics [59], and pursuit-evasion problems [60], and has been applied to challenging autonomous vehicle dynamics [61, 62]. Webb and van den Berg [63] guarantee asymptotic optimality for any system with controllable linear dynamics through the use of optimal controllers. Similarly, Perez *et al.* [64] apply local linearizations around RRT* trajectories to generate a basin of attraction/stability for nonlinear dynamics, building on the LQR-Trees approach [65].

Other sampling-based variants have been developed, including randomized potential fields [66] (which build on earlier work with potential fields [67]), and rapidly-exploring roadmaps, a hybridization of PRMs and RRTs [68].

Motion Planning under Uncertainty

Guibas *et al.* [69] propose the bounded uncertainty roadmap, which extends PRM to provide safety against obstacles with set-bounded vertex uncertainty. Pepy *et al.* [70] propose Box-RRT, an RRT-based path planning algorithm for nonlinear dynamics subject to set bounded uncertainty in process noise and initial configuration. 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 thesis (Chapter 2) tightens system constraints individually as a direct function of the disturbance bounds, such that constraints are tightened only as much as needed to achieve robust feasibility for a given control policy.

Several RRT-based approaches have been developed which approximate uncertainty statistically via the use of particles. Particle RRT [71] considers uncertain motion due to uneven terrain by sampling each tree branch multiple times, using clustering to create nodes. The work of Kewlani *et al.* [72] functions similarly, instead identifying a finite-series approximation of the uncertainty propagation, in order to reduce model complexity and the resulting number of simulations needed per node. Unlike Particle RRT [71], the proposed particle-based algorithm in this thesis (Section 4.3) can approximate the risk of both time-step-wise and path-wise constraint violation for multiple forms of uncertainty, while maintaining a separate tree node for each

possible action sequence. However, such an approach still requires the simulation of a potentially large set of particles at each node, which can be computationally intensive. An alternative formulation is also considered for directly sampling an analytic distribution (Section 4.4).

Several probabilistic sampling-based formulations are based on sampling the system's belief state [73], reducing the complexity of POMDP-based formulations. Such approaches often make the assumption of linear/linearized dynamics subject to Gaussian uncertainty. Of these, the most well known is perhaps the belief roadmap (BRM) [74], a belief-space variant of PRM for nonlinear systems subject to process and sensing noise. It is shown that the covariance matrix can be factored in order to plan – and update plans – efficiently in belief space. As the graph is constructed, a forward search is used to track the realizable belief states at each node, effectively generating a belief tree; only the paths which minimize the trace of the uncertainty covariance at each node are kept. However, motion planning is assumed to be kinematic, *i.e.,* straight-line connections between nodes. Additionally, construction of the original graph only checks whether the mean-state path is feasible, rather than providing minimum guarantees of constraint satisfaction. This work has been extended to incorporate unscented Kalman filtering with demonstration on a quadrotor testbed [75].

Another MDP-based variant of PRM is the stochastic motion roadmap [76], which constructs an MDP over a sampled roadmap to attempt to maximize the likelihood of reaching the goal region. However, this approach provides no formal optimality guarantees, and requires discretization of inputs. The incremental Markov decision process (iMDP) [77] removes these restrictions, providing asymptotic optimality guarantees for continuous control policies by incrementally constructing and performing value iteration on MDPs. The iMDP algorithm considers the continuous-time stochastic optimal control problem for nonlinear dynamics subject to additive Brownian process noise. At each iteration, a state is sampled from the environment, and a transition function is derived by attempting to steer from the sample to nearby states, similar to RRT*. Because the algorithm generates policies rather than plans, the input sequence

to be applied is not available *a priori* – it must be identified online, by applying the inputs of nodes nearest to the system state after disturbances are realized. As such, one cannot track the evolution of uncertainty distributions using this approach.

Additionally, iMDP does not consider sensing/localization error or uncertain environments, and cannot provide safety guarantees – possible obstacle collisions are modeled as a penalty term within the MDP reward function, which is shown to yield unsafe paths on some iterations [77]. Subsequent work has mitigated the latter issue by enforcing trajectory expected-value constraints [78], though solutions are somewhat conservative [79]. Here collision risk must be represented in a time-discounted Bellman form, rather than as specific time-step-wise/path-wise likelihood bounds or soft constraints [80]. Alternatively, conservatism can be reduced for particular initial states through the use of path-wise, time-consistent [81] risk constraints [79].

The feedback-based information roadmap (FIRM) [82, 83] is designed to be multi-query, unlike BRM. Because FIRM edges and costs are independent of each other, the motion planning problem is essentially subdivided into decoupled motion planning problems along each edge. In the FIRM graph, each node represents a small subset of the belief space, while each edge consists of a sequence of feedback policies comprising a Markov chain in the belief state. Each feedback policy is a stabilizer bringing the system into the next belief state node [82], though a finite-time tracker can also be used transiently [83]. However, all collision probabilities are calculated offline: the environment is assumed to be known and fixed *a priori*. Very recent work considers planning subject to unforeseen large changes in the obstacle environment or vehicle state [84].

Several other PRM formulations have been developed which maintain probabilistic safety bounds for a 2D vehicle avoiding obstacles represented by uncertain vertices [69, 85]. However, these approaches are not applicable to nonholonomic systems, which generally cannot track the piecewise linear roadmap paths. Burns and Brock [86] use an exploration-based heuristic to traverse a PRM with probabilistic feasibility guarantees under sensing uncertainty.

The LQG-MP algorithm [87], contemporaneous with the development of CC-

RRT [46] (Chapter 3), extends the RRT algorithm for a nonlinear system subject to Gaussian process and sensing noises. A variety of motion planning scenarios are considered, including corridors, cluttered environments, and multi-agent scenarios, with heuristics of varying complexity being used to assess path quality and/or risk. While some scenarios consider an uncertain environment (through the presence of other uncertain agents), its incorporation is limited – all obstacles are assumed to be circular, while the risk is computed using an expensive numerical integral evaluation. Additionally, because of the use of heuristics, no guarantees are available on minimum likelihood of constraint satisfaction.

The rapidly exploring random belief tree (RRBT) [88] develops a formulation of RRT* for the belief space of a system with nonlinear dynamics, subject to state and sensing uncertainty. A stabilizing linear controller is assumed to be applied around each nominal trajectory. As the tree is constructed, multiple belief evolutions are tracked through a tree; the tree is rewired when a new belief "dominates" an old one, *i.e.*, has a lower cost and covariance. Only minimum-cost paths and time-step-wise chance constraint bounds are considered. Here the chance constraints are evaluated by checking the uncertainty ellipse corresponding to the covariance matrix for collision against the obstacles. As a result, the probabilistic feasibility check is relatively expensive and conservative, does not provide a bound on the risk of constraint violation, and thus could not be readily used to assess path-wise chance constraints. Additionally, the environment is assumed to be perfectly known.

While this thesis focuses on single-vehicle motion planning, particularly using CC-RRT, Postlethwaite and Kothari [89] extend CC-RRT to multiple vehicles. Finally, we note the work of Patil *et al.* [90], which truncates Gaussian state distributions by conditioning at each stage along a trajectory on the assumption that previous states are collision-free. Doing so reduces conservatism, but also shifts the conditional means of the trajectory, removing the guarantee that the conditional-mean path is dynamically feasible for the autonomous vehicle.

Table 1.1: Overview of algorithms in this thesis

Algorithm	BU-RRT	BU-RRT*	CC-RRT	CC-RRT*
Chapter(s)	2	2	3–4	5
Uncertainty distribution	Any	Any	Gaussian	Gaussian
Bounded uncertainty?	Yes	Yes	No	No
Feasibility guarantees	Absolute	Absolute	Probabilistic	Probabilistic
Asymptotically optimal?	No	Yes	No	Yes

1.3 Outline and Summary of Contributions

This thesis discusses the development of novel real-time motion planning algorithms for a single vehicle. Unlike previous approaches in the literature, the algorithms developed in this thesis provide robustness guarantees subject to multiple forms of both internal and external uncertainty in real-time, even in complex environments. The algorithms are tested in a variety of simulation environments and demonstrated via hardware experiments with an autonomous rover. These algorithms differ in the assumptions made on the nature of the uncertainty environment, the type of feasibility guarantees available for motion plans, and their optimality. Table 1.1 provides an overview of the algorithms proposed in this thesis and the chapters where they are developed.

A major consideration throughout this thesis is the combination of an accurate estimate of the risk of constraint violation with a motion planner that provides probabilistic feasibility guarantees. Probabilistic feasibility is suitable for many motion planning problems where guarantees on absolute safety may be impossible/intractable, overly conservative, or simply unnecessary. For example, navigation tasks in crowded urban environments almost always impose some risk of constraint violation, such as a vehicle in an opposing lane triggering an unavoidable collision. However, if the risk of such outcomes can be evaluated, it can be identified as an acceptable level of risk, enabling meaningful planning to continue. The key idea is that, if the risk of constraint violation can be reasonably approximated, the operator of the planning algorithm can use that knowledge to decide how to best trade off between the risk of constraint violation and planner conservatism.

The algorithms in this thesis build upon sampling-based algorithms and particularly rapidly-exploring random trees (RRT) [53, 54], a sampling-based motion planner with demonstrated utility for autonomous vehicles [1, 91]. As discussed previously, RRTs present several key advantages for online motion planning of autonomous vehicles, including generation of trees of dynamically feasible paths by construction; quick exploration of high-dimensional configuration spaces; incremental construction; and trajectory-wise constraint checking.

- Chapter 2 proposes the bounded-uncertainty RRT (BU-RRT) and bounded-uncertainty RRT* (BU-RRT*) algorithms for motion planning problems subject to bounded uncertainty [92]. The first contribution of this chapter is the development of BU-RRT, a novel sampling-based motion planner which provides guarantees on absolute constraint feasibility for linear systems subject to both bounded internal and external uncertainty in real-time. In particular, constraint feasibility is guaranteed for linear systems subject to bounded process noise, localization error, and/or uncertain environmental constraints. During planning, state constraints are individually tightened for robustness against future uncertainty, while the input constraints can be tightened in order to apply disturbance feedback policies. In addition to extending the complexity of uncertainty and constraint formulations that can be considered online, this contribution also extends existing work in robust MPC [93, 94] by providing safety guarantees under environmental uncertainty, in addition to incorporating a sampling-based planning framework. It is shown that, given a particular choice of input policy within the planner (including open-loop plans), no conservatism is introduced: any feasible solution for the original, robust problem remains feasible for the tightened, deterministic problem. The second contribution is the development of BU-RRT*, which extends RRT* [56, 57] to efficiently generate and optimize robust, dynamically feasible trajectories. Both algorithms are shown to be *probabilistically complete* – if a feasible solution exists, it will be identified by the planner as the number of samples approaches infinity – while asymptotic optimality is also shown for BU-RRT*. Simulation results demonstrate identi-

fication of smooth, guaranteed-safe trajectories in complex scenarios subject to both internal and external uncertainty, including cases where the uncertainty may be asymmetric and/or non-convex.

- Chapter 3 proposes the chance-constrained RRT (CC-RRT) algorithm for motion planning problems subject to large and/or unbounded uncertainty [46]. The primary contribution of this chapter is the development of CC-RRT, a novel sampling-based motion planner which provides guarantees on probabilistic constraint feasibility for linear systems subject to both Gaussian internal and external uncertainty in real-time. In particular, minimum likelihood of constraint feasibility is guaranteed for linear systems subject to Gaussian process noise, localization error, and/or uncertain environmental constraints. Probabilistic constraint satisfaction is represented in terms of chance constraints. Central to this contribution is the derivation of a novel bound on the risk of constraint violation, which can be efficiently evaluated and bounded online within the CC-RRT tree of state distributions (online CC-RRT). Though some conservatism is introduced by this risk bound, the algorithm is effective in identifying risk-bounded trajectories. In addition to extending the types of constraint formulations that can be considered online, this contribution also extends existing work in chance-constrained optimization [37] to consider chance-constrained environment bounds, environmental uncertainty, and both time-step-wise and path-wise probabilistic feasibility guarantees. Extensions to nonlinear dynamics and/or output feedback models are also considered. Simulation results show that this algorithm can be used for efficient identification and execution of probabilistically safe paths in real time.

- Chapter 4 considers applications of chance-constrained motion planning, and in particular CC-RRT, in several motion planning domains of interest. These applications contribute several extensions to the CC-RRT framework, enabling consideration of more complex uncertainty formulations than considered in the sampling-based planning literature. The first contribution of this chapter is

the consideration of dynamic obstacles with uncertain intentions, represented via Gaussian mixture models with uncertainty in both behaviors and trajectories [95]. Probabilistic constraint satisfaction is shown to be maintained for such obstacles, whose uncertainty models are often learned from trajectory prediction algorithms. In particular, joint work showing integration with the RR-GP [95] and DPGP [96] trajectory prediction algorithms is demonstrated for safe urban navigation. The second contribution of this chapter is the development of particle CC-RRT [97], which enables more versatile consideration of uncertainty types and chance constraints than existing particle-based RRT approaches [71], though both can be computationally intensive. Finally, the third contribution of this chapter is the development of a novel motion planning algorithm for parafoil terminal guidance [98]. The resulting CC-RRT variant, analytic CC-RRT, uses multi-modal wind modeling and covariance sampling for collision checking against mapped terrain, yielding superior landing accuracy over state-of-the-art algorithms [99] in complex terrain.

- Chapter 5 proposes the chance-constrained RRT* (CC-RRT*) algorithm for asymptotically optimal motion planning with probabilistic feasibility guarantees for linear Gaussian systems [80]. The first contribution of this chapter is the extension of CC-RRT to guarantee asymptotic optimality, making CC-RRT* the first asymptotically optimal sampling-based algorithm with robustness to both internal and external uncertainty. This algorithm leverages RRT* [56, 57] to efficiently generate and optimize dynamically and probabilistically feasible trajectories. The second contribution of this chapter is the proposal of a novel, risk-based objective function to provide "soft constraints" on risky behavior alongside the hard constraints derived from CC-RRT. This objective, shown to be admissible within RRT*, yields a more versatile consideration of risk-averse behavior compared to existing approaches in the literature. Extensive simulation results demonstrate that CC-RRT* can efficiently identify smooth, robust trajectories for a variety of scenarios, including complex dynamics, high-

dimensional state spaces, dynamic obstacles, and pursuit-evasion problems.

- Chapter 6 contributes the real-time demonstration of the proposed probabilistic motion planning algorithms on hardware in dynamic environments. Using the CC-RRT planner, an autonomous rover is able to safely navigate around pedestrians, robots, and other dynamic obstacles, the primary source of uncertainty in this chapter. Obstacles are detected both from motion capture cameras and from onboard sensors, demonstrating the ability of this approach to be used within perception-driven planning.

- Chapter 7 offers concluding remarks and suggestions for future work.

Chapter 2

Guaranteed Robustness to Bounded Uncertainty

This section presents a novel sampling-based planner which generates trajectories for linear systems with robustness to bounded process noise, localization/initial state error, and/or uncertain environmental constraints. The proposed planner builds on RRT* [56], which extends RRT to guarantee asymptotic optimality, by iteratively modifying individual state/input constraints during trajectory simulation and rewiring to enforce robust feasibility conditions. All trajectories generated by the proposed algorithm are guaranteed safe for any feasible uncertainty realization. The set-based uncertainty representation allows consideration of asymmetric and/or state-dependent uncertainties (*e.g.*, quadrotor ground effect, uneven terrain).

The proposed approach also incorporates the option for state- and disturbance-feedback policies within planning. During planning, state constraints are individually tightened for robustness against future uncertainty, while the input constraints are tightened as needed for future state/disturbance feedback. The proposed approach extends feedback policies developed in the RMPC literature [31, 94] by providing guarantees against both internal and environmental uncertainty, as well as incorporating a sampling-based planning framework. By building on RRTs, the algorithm can generate dynamically feasible trajectories for complex configuration spaces and constraint characterizations. For any given control policy, no conservatism is intro-

duced: the set of feasible solutions is equivalent between the original (uncertain) and modified formulations. Further, trajectory-wise constraint checking ensures that the algorithm scales well with the problem complexity. This scalability enables real-time robust planning in complex, cluttered environments that are computationally challenging or intractable for optimization-based frameworks [31, 94]. The proposed approach is shown to work even if the localization or obstacle placement uncertainty is non-convex.

2.1 Problem Statement

Consider the linear time-invariant (LTI) discrete-time system dynamics subject to process noise

$$x_{t+1} = Ax_t + Bu_t + Gw_t, \tag{2.1}$$

$$w_t \in S_w, \tag{2.2}$$

where $x_t \in \mathbb{R}^{s_x}$ is the state vector, $u_t \in \mathbb{R}^{s_u}$ is the input vector, $w_t \in \mathbb{R}^{s_w}$ is a process noise uncertainty acting on the system, and A, B, G are matrices of suitable dimension. The disturbance w_t is unknown at current and future timesteps, but must belong to the polytopic set S_w, which contains the origin. The initial/current state x_0 may additionally be uncertain via

$$x_0 = \widehat{x}_0 + \widetilde{x}_0, \tag{2.3}$$

$$\widetilde{x}_0 \in S_x, \tag{2.4}$$

where $\widehat{x}_0$ is the initial state estimate and $\widetilde{x}_0$ is unknown at current and future timesteps but must belong to the polytopic set S_x, which contains the origin.

The system is additionally subject to constraints acting on the system state and

input. These constraints are assumed to take the form

$$x_t \in \mathcal{X}_t \equiv \mathcal{X} \backslash \mathcal{X}_{1t} \backslash \cdots \backslash \mathcal{X}_{n_o t}, \tag{2.5}$$

$$u_t \in \mathcal{U}, \tag{2.6}$$

where $\mathcal{X}, \mathcal{X}_{1t}, \ldots, \mathcal{X}_{n_o t} \subset \mathbb{R}^{s_x}$, $\mathcal{U} \subset \mathbb{R}^{s_u}$ are convex polytopes, and the $\backslash$ operator denotes set subtraction (relative complement). The sets $\mathcal{X}$ and $\mathcal{U}$ define a set of time-invariant convex constraints acting on the state and input, respectively. The sets $\mathcal{X}_{1t}, \ldots, \mathcal{X}_{n_o t}$ represent n_o convex, polytopic obstacles to be avoided. The time dependence of $\mathcal{X}_t$ in (2.5) allows the potential inclusion of dynamic obstacles. For each obstacle, the shape and orientation are assumed to be known, while the placement is uncertain. This is represented as

$$\mathcal{X}_{jt} = \mathcal{X}_j^0 + \widehat{c}_{jt} + c_{jt}, \quad \forall\, j \in \mathbb{Z}_{1,n_o}, \tag{2.7}$$

$$c_{jt} \in S_{jt}, \quad \forall\, j \in \mathbb{Z}_{1,n_o}, \tag{2.8}$$

where the $+$ operator denotes set translation and $\mathbb{Z}_{a,b}$ represents the set of integers between a and b inclusive. In this model for the jth obstacle, $\mathcal{X}_j^0 \subset \mathbb{R}^{s_x}$ is a convex polytope of known, fixed shape which contains the origin, $\widehat{c}_{jt}$ is a fixed translation at timestep t, and $c_{jt} \in \mathbb{R}^{s_x}$ represents an uncertain and possibly time-varying translation which must belong to the polytopic set S_{jt}, which contains the origin.

In summary, the system is subject to three separate types of uncertainty: process noise (2.2), localization/initial state error (2.4), and/or obstacle placement uncertainty (2.8). Each uncertainty, though unknown prior to realization, is bounded within the sets S_w, S_x, and S_{jt} $\forall\, j \in \mathbb{Z}_{1,n_o}$, $\forall\, t$, respectively. Because these sets are assumed polytopic, they can be written as the conjunction of linear inequalities

$$S_w = \{w \mid E^w w \leq f^w\}, \tag{2.9}$$

$$S_x = \{x \mid E^x x \leq f^x\}, \tag{2.10}$$

$$S_{jt} = \{c \mid E^{jt} c \leq f^{jt}\},\ \forall\, j \in \mathbb{Z}_{1,n_o},\ \forall\, t, \tag{2.11}$$

where $f^w \in \mathbb{R}^{m_w}$, $f^x \in \mathbb{R}^{m_x}$, $f^{jt} \in \mathbb{R}^{m_{jt}}$, and E^w, E^x, E^{jt} are matrices of appropriate size. Additionally, since the sets (2.9)-(2.11) are assumed to contain the origin, then $f^w, f^x, f^{jt} \geq 0$.

Similarly, the polytopic sets $\mathcal{U}, \mathcal{X}, \mathcal{X}_{1t}, \ldots, \mathcal{X}_{n_o t}$ can be written as the conjunction of linear inequalities:

$$\mathcal{U} = \{u \mid A_u u \leq b_u\}, \tag{2.12}$$

$$\mathcal{X} = \{x \mid A_0 x \leq b_0\}, \tag{2.13}$$

$$\mathcal{X}_{jt} = \{x_t \mid A_j x_t \leq b_{jt}\},\ \forall\ j \in \mathbb{Z}_{1,n_o}, \tag{2.14}$$

where $b_u \in \mathbb{R}^{n_u}$, $b_0 \in \mathbb{R}^{n_E}$, $b_{jt} \in \mathbb{R}^{n_j}$, and A_u, A_0, A_j are matrices of appropriate size. The obstacle sets (2.14) can alternatively be written as

$$\mathcal{X}_{jt} = \left\{ x_t \mid \bigwedge_{i=1}^{n_j} a_{ij}^T x_t < a_{ij}^T c_{ijt} \right\},\ \forall\ j \in \mathbb{Z}_{1,n_o}, \tag{2.15}$$

$$c_{ijt} = \widehat{c}_{ijt} + c_{jt}, \tag{2.16}$$

where $\bigwedge$ denotes a conjunction and $\widehat{c}_{ijt}$ is a point nominally (*i.e.,* when $c_{jt} \equiv 0$) on the ith constraint of the jth obstacle at timestep t. The non-convex avoidance constraints corresponding to these obstacles may be written as

$$\neg\,(A_j x_t \leq b_{jt})\ \ \forall j \in \mathbb{Z}_{1,n_o} \tag{2.17}$$

$$\Leftrightarrow \quad \bigvee_{i=1}^{n_j} a_{ij}^T x_t \geq a_{ij}^T c_{ijt}\ \forall j \in \mathbb{Z}_{1,n_o}, \tag{2.18}$$

where $\bigvee$ denotes a disjunction.

The primary objective of the motion planning problem is to reach some goal region $\mathcal{X}_{\text{goal}} \subset \mathbb{R}^{s_x}$ while ensuring constraints (2.5)-(2.6) are feasible for all possible disturbance realizations. Consider the disturbance-free version of the dynamics (2.1),

$$\widehat{x}_{t+1} = A\widehat{x}_t + Bu_t, \tag{2.19}$$

and denote

$$t_f = \inf\{t \in \mathbb{Z}_{0,\infty} \mid \widehat{x}_t \in \mathcal{X}_{\text{goal}}\}. \tag{2.20}$$

The path planner seeks to approximately solve the optimal control problem

$$\begin{aligned}
(2.\text{A}) \quad \min_{u_t} \quad & \sum_{t=0}^{t_f-1} \phi(\widehat{x}_t, \mathcal{X}_{\text{goal}}, u_t) \\
\text{s.t.} \quad & x_{t+1} = Ax_t + Bu_t + Gw_t, \quad w_t \in S_w, \quad \forall\, t, \\
& x_0 = \widehat{x}_0 + \widetilde{x}_0, \quad \widetilde{x}_0 \in S_x, \\
& \widehat{x}_{t+1} = A\widehat{x}_t + Bu_t, \quad \forall\, t, \\
& u_t \in \mathcal{U} \quad \forall\, t, \\
& x_t \in \mathcal{X}, \quad \forall\, w_t \in S_w, \quad \forall\, \widetilde{x}_0 \in S_x, \quad \forall\, t, \\
& x_t \notin \mathcal{X}_{jt}, \quad \forall\, w_t \in S_w, \quad \forall\, \widetilde{x}_0 \in S_x, \quad \forall\, c_{jt} \in S_{jt}, \quad \forall\, j \in \mathbb{Z}_{1,n_o}, \quad \forall\, t, \\
& \mathcal{X}_{jt} = \mathcal{X}_j^0 + \widehat{c}_{jt} + c_{jt}, \quad c_{jt} \in S_{jt}, \quad \forall\, j \in \mathbb{Z}_{1,n_o}, \quad \forall\, t,
\end{aligned} \tag{2.21}$$

where ϕ is some cost function to be optimized. In practice, the optimization (2.21) is solved repeatedly as the system navigates the environment in real-time, with x_0 being set to the current state.

2.2 Robustness to Bounded Uncertainty

This section derives and establishes the constraints which must be satisfied by any candidate trajectory at each timestep in order to satisfy the conditions of the problem statement (Section 2.1). First, an alternate formulation of problem (2.A) is posed which incorporates a disturbance feedback policy, in which problem (2.A) is a special case. It is shown that robust feasibility can be guaranteed by considering a modified version of that formulation which tightens the state and input constraints, corresponding to each source of uncertainty and any disturbance feedback which may be applied. Each inequality constraint is tightened using a series of simple, incre-

mental optimizations, which can be pre-computed as problem data becomes available and/or stored for future use. The modified constraints extend existing formulations, *e.g.,* Kuwata [94], to consider external uncertainty and some forms of non-convex uncertainty, in addition to process noise.

Consider an affine disturbance-feedback policy [94]

$$u_t = v_t + \sum_{k=0}^{t-1} P_{t-1-k} G w_k, \tag{2.22}$$

where the disturbance feedback matrices $P_0, P_1, \ldots, P_{t-1}$ are chosen by the operator. An alternate version of problem (2.A) is now posed which considers this feedback policy as an additional constraint,

$$\begin{aligned} \text{(2.B)} \quad \min_{v_t} \quad & dt \sum_{t=0}^{t_f-1} \phi(\widehat{x}_t, \mathcal{X}_{\text{goal}}, u_t) \\ \text{s.t.} \quad & \text{(2.A)}, \\ & u_t = v_t + \sum_{k=0}^{t-1} P_{t-1-k} G w_k, \end{aligned} \tag{2.23}$$

where the above notation implies that all constraints of problem (2.A) are enforced. By incorporating the feedback policy, the affine input terms v_t replace u_t as the decision variables. Additionally, satisfaction of the input constraints $\mathcal{U}$ is now dependent on possible realizations of the process noise w_t via (2.22). However, the original problem (2.A) can be recovered by setting $P_t \equiv 0 \; \forall \; t$.

The objective is now to impose a set of constraints on the nominal dynamics (2.19) such that the constraints (2.5)-(2.6) are robustly satisfied for all possible disturbances. Consider the robust version of problem (2.B),

$$
\begin{aligned}
\text{(2.C)} \quad \min_{u_t} \quad & \sum_{t=0}^{t_f-1} \phi(\widehat{x}_t, \mathcal{X}_{\text{goal}}, u_t) && (2.24)\\
\text{s.t.} \quad & \widehat{x}_{t+1} = A\widehat{x}_t + Bu_t, \ \forall\ t, \\
& A_u u_t \leq b_u - \delta^w_{ut}, \ \forall\ t, && (2.25)\\
& A_0 \widehat{x}_t \leq b_0 - \delta^w_{0t} - \delta^x_{0t}, \ \forall\ t, && (2.26)\\
& \neg(A_j \widehat{x}_t \leq b_{jt} - \delta^w_{jt} - \delta^x_{jt} + \delta^c_{jt}), \ \forall\ j \in \mathbb{Z}_{1,n_o}, \ \forall\ t, && (2.27)
\end{aligned}
$$

where

$$
\begin{aligned}
\delta^w_{ut} &= \sum_{k=0}^{t-1} \max_{w_k \in S_w} A_u P_{t-1-k} G w_k, && (2.28)\\
\delta^w_{0t} &= \sum_{j=0}^{t-1} \max_{w_k \in S_w} A_0 A^{t-1-k} Q_{t-2-k} G w_k, && (2.29)\\
\delta^w_{jt} &= \sum_{j=0}^{t-1} \min_{w_k \in S_w} A_j A^{t-1-k} Q_{t-2-k} G w_k, && (2.30)\\
\delta^x_{0t} &= \max_{\widetilde{x}_0 \in S_x} A_0 A^t \widetilde{x}_0, && (2.31)\\
\delta^x_{jt} &= \min_{\widetilde{x}_0 \in S_x} A_j A^t \widetilde{x}_0, && (2.32)\\
\delta^c_{jt} &= \max_{c_{jt} \in S_{jt}} A_j c_{jt}, && (2.33)\\
Q_n &= I + B \sum_{k=0}^{n} P_k, && (2.34)
\end{aligned}
$$

where I_n is an $n \times n$ identity matrix. Each maximization and minimization operator in (2.28)-(2.33) is applied element-wise to the specified vector. Because S_w, S_x, and S_{jt} are all sets of linear inequalities, these operators are performing simple linear optimizations. Each term incrementally tightens one set of constraints for one of the possible uncertainty sources. Note that the optimizations used in (2.28)-(2.33) to tighten the constraints are effectively computing supports over the uncertainty sets.

Because the set S_w is time-invariant, tightening terms (2.28)-(2.30) can be written

implicitly as

$$\delta_{ut}^{w} = \delta_{u(t-1)}^{w} + \max_{w \in S_w} A_u P_{t-1} G w, \tag{2.35}$$

$$\delta_{0t}^{w} = \delta_{0(t-1)}^{w} + \max_{w \in S_w} A_0 A^{t\ 1} Q_{t-2} G w, \tag{2.36}$$

$$\delta_{jt}^{w} = \delta_{j(t-1)}^{w} + \min_{w \in S_w} A_j A^{t-1} Q_{t-2} G w. \tag{2.37}$$

Additionally, define the uncertainty levels $\Delta_t = \{\Delta_{ut}, \Delta_{0t}, \Delta_{1t}, \ldots, \Delta_{n_o t}\}$ at timestep t as

$$\Delta_{ut} = \delta_{ut}^{w}, \tag{2.38}$$

$$\Delta_{0t} = \delta_{0t}^{x} + \delta_{0t}^{w}, \tag{2.39}$$

$$\Delta_{jt} = \delta_{jt}^{c} - \delta_{jt}^{x} - \delta_{jt}^{w},\ \forall\ j \in \mathbb{Z}_{1,n_o}. \tag{2.40}$$

Using these definitions, the robust constraints (2.25)-(2.27) can be written in the more compact form

$$A_u u_t \leq b_u - \Delta_{ut},\ \forall\ t, \tag{2.41}$$

$$A_0 \widehat{x}_t \leq b_0 - \Delta_{0t},\ \forall\ t, \tag{2.42}$$

$$\neg(A_j \widehat{x}_t \leq b_{jt} + \Delta_{jt}),\ \forall\ j \in \mathbb{Z}_{1,n_o},\ \forall\ t. \tag{2.43}$$

It is now shown that satisfying this set of constraints guarantees robust feasibility for the true dynamics (2.1) under all possible disturbance realizations.

Theorem 2.1 (Robust Feasibility). *Given feedback policy $P_0, \ldots, P_{t_f-1}$, consider the path specified by the input sequence $v_0, v_1, \ldots, v_{t_f-1}$. This path is feasible for problem (2.C) iff it is feasible for problem (2.B). Additionally, this path is feasible for problem (2.A) if it is feasible for problem (2.B).*

Proof. First, equivalence between the deterministic problem (2.C) and the stochastic problem (2.B) is shown. The objective is to demonstrate that if the sequence of inputs $v_0, v_1, \ldots, v_{t_f-1}$ satisfies the constraints of problem (2.C), the nominal constraints (2.B) will be satisfied for any feasible disturbance realization.

Suppose that $v_0, v_1, \ldots, v_{t_f-1}$ are feasible for problem (2.B). The evolution of the true system state x_t, subject to (2.1), can be written in explicit form

$$x_t = A^t x_0 + \sum_{k=0}^{t-1} A^{t-1-k} B u_k + \sum_{k=0}^{t-1} A^{t-1-k} G w_k. \tag{2.44}$$

Substituting (2.44), (2.3), and (2.16) into (2.12), (2.13), and (2.18) yields the constraints

$$\begin{aligned}
A_u u_t &\leq b_u, && (2.45)\\
A_0 A^t \widehat{x}_0 + \sum_{k=0}^{t-1} A_0 A^{t-1-k} B u_k &\leq b_0 - \sum_{k=0}^{t-1} A_0 A^{t-1-k} G w_k - A_0 A^t \widetilde{x}_0, && (2.46)\\
\bigvee_{i=1}^{n_j} a_{ij}^T A^t \widehat{x}_0 + \sum_{k=0}^{t-1} a_{ij}^T A^{t-1-k} B u_k &\geq a_{ij}^T \widehat{c}_{ijt} - \sum_{k=0}^{t-1} a_{ij}^T A^{t-1-k} G w_k \\
&\quad - a_{ij}^T A^t \widetilde{x}_0 + a_{ij}^T c_{jt}. && (2.47)
\end{aligned}$$

Applying the feedback policy (2.22) leads to the constraints

$$\begin{aligned}
A_u v_t &\leq b_u - \sum_{k=0}^{t-1} A_u P_{t-1-k} G w_k, && (2.48)\\
A_0 A^t \widehat{x}_0 + \sum_{k=0}^{t-1} A_0 A^{t-1-k} B v_k &\leq b_0 - \sum_{k=0}^{t-1} A_0 A^{t-1-k} G w_k \\
&\quad - A_0 A^t \widetilde{x}_0 - \sum_{k=0}^{t-1} A_0 A^{t-1-k} B \sum_{l=0}^{k-1} P_{k-1-l} G w_l, && (2.49)\\
\bigvee_{i=1}^{n_j} a_{ij}^T A^t \widehat{x}_0 + \sum_{k=0}^{t-1} a_{ij}^T A^{t-1-k} B v_k &\geq a_{ij}^T \widehat{c}_{ijt} - \sum_{k=0}^{t-1} a_{ij}^T A^{t-1-k} G w_k \\
&\quad - a_{ij}^T A^t \widetilde{x}_0 + a_{ij}^T c_{jt} - \sum_{k=0}^{t-1} a_{ij}^T A^{t-1-k} B \sum_{l=0}^{k-1} P_{k-1-l} G w_l. && (2.50)
\end{aligned}$$

Re-defining the disturbance-free dynamics (2.19) in terms of v_t rather than u_t, *i.e.*,

$$\widehat{x}_{t+1} = A \widehat{x}_t + B v_t \tag{2.51}$$

implies that the left-hand sides of (2.49) and (2.50) can be written as $A_0 \widehat{x}_t$ and $a_{ij}^T \widehat{x}_t$,

respectively. Additionally, the matrix coefficients for the w_t terms in (2.49) and (2.50) can be rewritten via (2.34) to yield the simplified constraints

$$A_u v_t \leq b_u - \sum_{k=0}^{t-1} A_u P_{t-1-k} G w_k, \tag{2.52}$$

$$A_0 \widehat{x}_t \leq b_0 - \sum_{k=0}^{t-1} A_0 A^{t-1-k} Q_{t-2-k} G w_k - A_0 A^t \widetilde{x}_0, \tag{2.53}$$

$$\bigvee_{i=1}^{n_j} a_{ij}^T \widehat{x}_t \geq a_{ij}^T \widehat{c}_{ijt} - \sum_{k=0}^{t-1} a_{ij}^T A^{t-1-k} Q_{t-2-k} G w_k - a_{ij}^T A^t \widetilde{x}_0 + a_{ij}^T c_{jt}. \tag{2.54}$$

Because problem (2.B) is feasible, the constraints (2.52)-(2.54) must be satisfied for any possible disturbance realization $w_t \in S_w$, $\widetilde{x}_0 \in S_x$, $c_{jt} \in S_{jt}\, \forall j \in \mathbb{Z}_{1,n_o}$, $\forall\, t \in \mathbb{Z}_{0,T}$. Thus, for each inequality constraint, it is sufficient to consider the tightest possible bound on the right-hand side. For each uncertain term on the right-hand side, this is accomplished by maximizing or minimizing that term over all feasible uncertainty realizations:

$$A_u v_t \leq b_u - \sum_{k=0}^{t-1} \max_{w_k \in S_w} A_u P_{t-1-k} G w_k, \tag{2.55}$$

$$A_0 \widehat{x}_t \leq b_0 - \max_{w_k \in S_w} \sum_{k=0}^{t-1} A_0 A^{t-1-k} Q_{t-2-k} G w_k - \max_{\widetilde{x}_0 \in S_x} A_0 A^t \widetilde{x}_0, \tag{2.56}$$

$$\begin{aligned} \bigvee_{i=1}^{n_j} a_{ij}^T \widehat{x}_t \geq\ & a_{ij}^T \widehat{c}_{ijt} - \min_{w_k \in S_w} \sum_{k=0}^{t-1} a_{ij}^T A^{t-1-k} Q_{t-2-k} G w_k \\ & - \min_{\widetilde{x}_0 \in S_x} a_{ij}^T A^t \widetilde{x}_0 + \max_{c_{jt} \in S_{jt}} a_{ij}^T c_{jt}. \end{aligned} \tag{2.57}$$

Substituting in (2.28)-(2.33) yields

$$A_u v_t \leq b_u - \delta_{ut}^w, \tag{2.58}$$

$$A_0 \widehat{x}_t \leq b_0 - \delta_{0t}^w - \delta_{0t}^x, \tag{2.59}$$

$$\bigvee_{i=1}^{n_j} a_{ij}^T \widehat{x}_t \geq a_{ij}^T \widehat{c}_{ijt} - \delta_{jt}^w - \delta_{jt}^x + \delta_{jt}^c. \tag{2.60}$$

By rewriting v_t as u_t and (2.60) in the form of (2.17), the constraints (2.25)-(2.27)

are obtained. This ensures that problem (2.C) is also feasible for this path.

Conversely, suppose that $v_0, v_1, \ldots, v_{t_f-1}$ (when v_t is written as u_t) are feasible for problem (2.C), such that the constraints (2.25)-(2.27) are satisfied. Consider some particular feasible disturbance realization $\{\overline{\widetilde{x}}_0, \overline{w}_t \ \forall \ t \in \mathbb{N}_{0,t_f-1}, \overline{c}_{jt} \ \forall \ j \in \mathbb{N}_{1,n_o} \ \forall \ t \in \mathbb{N}_{0,t_f-1}\}$, and define the quantities

$$\gamma^w_{ut} = \sum_{k=0}^{t-1} A_u P_{t-1-k} G \overline{w}_k \leq \delta^w_{ut}, \tag{2.61}$$

$$\gamma^w_{0t} = \sum_{k=0}^{t\ 1} A_0 A^{t-1-k} Q_{t-2-k} G \overline{w}_k \leq \delta^w_{0t}, \tag{2.62}$$

$$\gamma^w_{jt} = \sum_{k=0}^{t-1} A_j A^{t-1-k} Q_{t-2-k} G \overline{w}_k \geq \delta^w_{jt}, \tag{2.63}$$

$$\gamma^x_{0t} = A_0 A^t \overline{\widetilde{x}}_0 \leq \delta^x_{0t}, \tag{2.64}$$

$$\gamma^x_{jt} = A_j A^t \overline{\widetilde{x}}_0 \geq \delta^x_{jt}, \tag{2.65}$$

$$\gamma^c_{jt} = A_j \overline{c}_{jt} \leq \delta^c_{jt}, \tag{2.66}$$

where the inequalities are defined from (2.28)-(2.33). The constraints of problem (2.B) can still be written as (2.52)-(2.54) when using the same input sequence, $v_0, v_1, \ldots, v_{t_f-1}$. For each disturbance realization, (2.52)-(2.54) can thus be written as

$$A_u v_t \leq b_u - \gamma^w_{ut}, \tag{2.67}$$

$$A_0 \widehat{x}_t \leq b_0 - \gamma^w_{0t} - \gamma^x_{0t}, \tag{2.68}$$

$$\bigvee_{i=1}^{n_j} a_{ij}^T \widehat{x}_t \geq a_{ij}^T \widehat{c}_{ijt} - \gamma^w_{jt} - \gamma^x_{jt} + \gamma^c_{jt}; \tag{2.69}$$

However, knowing that (2.25)-(2.27) are feasible, and utilizing the inequality relation-

ships in (2.61)-(2.66), then

$$A_u v_t \leq b_u - \delta^w_{ut} \leq b_u - \gamma^w_{ut}, \tag{2.70}$$

$$A_0 \widehat{x}_t \leq b_0 - \delta^w_{0t} - \delta^x_{0t} \leq b_0 - \gamma^w_{0t} - \gamma^x_{0t}, \tag{2.71}$$

$$\bigvee_{i=1}^{n_j} a^T_{ij} \widehat{x}_t \geq a^T_{ij}\widehat{c}_{ijt} - \delta^w_{jt} - \delta^x_{jt} + \delta^c_{jt} \geq a^T_{ij}\widehat{c}_{ijt} - \gamma^w_{jt} - \gamma^x_{jt} + \gamma^c_{jt}, \tag{2.72}$$

implying that the constraints (2.67)-(2.69) must also be satisfied for this specific path and uncertainty realization.

This implies that the constraints of problem (2.B) are satisfied for this specific path and uncertainty realization. Thus they are satisfied for any feasible disturbance realization, implying that the path is feasible for problem (2.B).

Finally, if the path specified by the input sequence $v_0, v_1, \ldots, v_{t_f-1}$ is feasible for problem (2.B), then generate the input sequence $u_0, u_1, \ldots, u_{t_f-1}$ via (2.22). Since the constraints of problem (2.A) are subsumed by the set of constraints of problem (2.B), problem (2.A) must also be feasible. ■

Theorem 2.1 implies that no conservatism is introduced by this approach for any specific feedback policy. In other words, if any path can be found to satisfy the constraints of problem (2.B) under some specific feedback policy, it can also be found to satisfy the constraints of problem (2.C) for that same feedback policy. However, this theorem does make any claims to feasibility over *all* feedback policies, as emphasized by the clear lack of equivalence between problems (2.A) and (2.B). The question of choosing appropriate feedback policies, a.k.a. feedback design, is beyond the scope of this work, though some specific cases of interest are discussed in the remarks below.

Remark 2.2 (other feedback policies). By applying no feedback, *i.e.*, $P_k = 0 \; \forall \; k \in \mathbb{Z}_{0,t-1}$ in (2.22), an open-loop control policy is instead obtained. In this case, $Q_n = I_n$, $\delta^w_{ut} \equiv 0$ and the nominal input constraints (2.12) can be applied in lieu of (2.25).

Remark 2.3 (state feedback policies). A state-feedback policy of the form $u_t = v_t + K(x_t - \widehat{x}_t)$ [28, 100] can be applied by using disturbance feedback matrices in

(2.22) of the form

$$P_k = K_k L_k, \ \forall \ k \in \mathbb{Z}_{0,t-1}, \tag{2.73}$$

$$L_{k+1} = (A + BK_k)L_k, \ \ L_0 = I_n, \tag{2.74}$$

where the K_k are state feedback matrices of appropriate size. If this feedback is held constant, *i.e.*, $K_k \equiv K$, then $L_k = (A + BK)^k$. It is shown in Ref. [94] that state-feedback policies are subsumed by the set of disturbance-feedback policies in this manner.

Remark 2.4 (non-convex uncertainty). The above results hold even for certain non-convex representations of the uncertainty in $\widetilde{x}_0$ and c_{jt}. Suppose that S_x and S_{jt} are written as the union of polytopic sets,

$$S_x = \bigcup_{p=1}^{p_x} \overline{S}_x^{(p)} \equiv \bigcup_{p=1}^{p_x} \{x \mid E_p^x x \leq \overline{f}_p^x\}, \tag{2.75}$$

$$S_{jt} = \bigcup_{p=1}^{p_{jt}} \overline{S}_{jt}^{(p)} \equiv \bigcup_{p=1}^{p_{jt}} \{c \mid E_p^{jt} c \leq \overline{f}_p^{jt}\}, \ \forall \ j \in \mathbb{Z}_{1,n_o}, \ \forall \ t, \tag{2.76}$$

where p_x and p_{jt} are the number of polytopes required to define S_x and S_{jt}, respectively.

In order to verify robustness to a non-convex S_x and/or S_{jt}, separate constraints (2.26)-(2.27) are imposed for each convex polytope comprising the uncertainty sets (2.75)-(2.76). In each case, the nominal initial state $\widehat{x}_0$ and nominal obstacle constraint location $\widehat{c}_{ijt}$ are temporarily shifted for the constraint checks such that $0 \in S_x^{(p)} \ \forall p \in \mathbb{Z}_{1,p_x}$, $0 \in S_{jt}^{(p)} \ \forall \ p \in \mathbb{Z}_{1,p_{jt}}$:

$$x_0 = \widehat{x}_0^{(p)} + \widetilde{x}_0^{(p)}, \tag{2.77}$$

$$c_{ijt} = \widehat{c}_{ijt}^{(p)} + c_{jt}^{(p)}. \tag{2.78}$$

After this translation, the constraints for the pth polytopes are written as

$$S_x = \bigcup_{p=1}^{p_x} S_x^{(p)} \equiv \bigcup_{p=1}^{p_x} \{x \mid E_p^x x \leq f_p^x\}, \tag{2.79}$$

$$S_{jt} = \bigcup_{p=1}^{p_{jt}} S_{jt}^{(p)} \equiv \bigcup_{p=1}^{p_{jt}} \{c \mid E_p^{jt} c \leq f_p^{jt}\},$$
$$\forall\ j \in \mathbb{Z}_{\mathbf{1},n_o},\ \forall\ t, \tag{2.80}$$

where $f_p^x \geq 0,\ f_p^{jt} \geq 0$. The robust feasibility constraints (2.25)-(2.27) then take the form

$$A_u u_t \leq b_u - \delta_{ut}^w,\ \forall\ t \tag{2.81}$$

$$A_0 \widehat{x}_t^{(p)} \leq b_0 - \delta_{0t}^w - \delta_{0t}^{x(p)}, \forall\ p \in \mathbb{Z}_{1,p_x},\ \forall\ t \tag{2.82}$$

$$\neg(A_j \widehat{x}_t^{(p)} \leq b_{jt}^{(p)} - \delta_{jt}^w - \delta_{jt}^{x(p)} + \delta_{jt}^{c(q)}), \tag{2.83}$$
$$\forall\ p \in \mathbb{Z}_{1,p_x},\ \forall\ q \in \mathbb{Z}_{1,p_{jt}},\ \forall\ j \in \mathbb{Z}_{1,n_o}, \forall\ t,$$

$$\delta_{0t}^{x(p)} = \max_{\widetilde{x}_0^{(p)} \in S_x^{(p)}} A_0 A^t \widetilde{x}_0^{(p)}, \tag{2.84}$$

$$\delta_{jt}^{x(p)} = \min_{\widetilde{x}_0^{(p)} \in S_x^{(p)}} A_j A^t \widetilde{x}_0^{(p)}, \tag{2.85}$$

$$\delta_{jt}^{c(p)} = \max_{c_{jt}^{(p)} \in S_{jt}^{(p)}} A_j c_{jt}^{(p)}, \tag{2.86}$$

where the ith element of $b_{jt}^{(p)}$ is $a_{ijt}^T \widehat{c}_{ijt}^{(p)}$.

Remark 2.5 (convex obstacle uncertainty). If the obstacle uncertainty sets S_{jt} are convex polytopes, then this environmental uncertainty may be removed from the planning problem by replacing each obstacle $\mathcal{X}_{jt}$ with the Minkowski sum $\mathcal{X}_{jt} \oplus S_{jt}$, which is also a convex polytope. The resulting expanded obstacle with zero uncertainty is then used.

2.3 Bounded-Uncertainty RRT*

This section introduces the bounded-uncertainty RRT* (BU-RRT*) algorithm for quickly identifying and refining feasible trajectories, subject to both internal and external uncertainty (Section 2.1). This algorithm has been designed to fit into the constraints of the RRT* framework [56, 57], such that guarantees on asymptotic optimality are preserved. However, unlike RRT*, the bounded-uncertainty RRT* algorithm enforces the robustness constraints (2.25)-(2.27) on the disturbance-free dynamics (2.19) to ensure that all trajectories generated in the tree are safe for any realizable disturbances. As part of the presentation of BU-RRT* and its subsequent analysis, this section also presents the bounded-uncertainty RRT (BU-RRT) algorithm, building upon the conventional RRT algorithm [54]. The BU-RRT algorithm maintains both dynamic and robust feasibility of its generated trajectories, but does not perform the additional "rewiring" steps of BU-RRT* needed to maintain asymptotic optimality.

Both the BU-RRT and BU-RRT* algorithms grow a tree of dynamically feasible trajectories using the disturbance-free dynamics (2.19) for simulation. This tree is denoted by $\mathcal{T}$, consisting of $|\mathcal{T}|$ nodes. Each node N of the tree $\mathcal{T}$ contains a trajectory segment consisting of a sequence of states, which is checked for robust feasibility. A sequence of states is denoted by σ; let $t[N]$ and $\widehat{x}[N]$ denote the terminal timestep and state for node N, respectively, while $t[\sigma]$ denotes the initial timestep of σ. The cost function (2.24) is implemented by setting $\phi(\widehat{x}_t, \mathcal{X}_{\text{goal}}, u_t) = f(\widehat{x}_t)dt$, where dt is the timestep duration and f is the per-timestep cost objective specified by the user. Thus

$$J[N] = dt \sum_{t=0}^{t[N]} f(\widehat{x}_t) \tag{2.87}$$

represents the entire path cost from the starting state to the terminal state of node N. In this work, $f(\widehat{x}) = 1$, implying minimization of the path duration, though more complex cost functions satisfying certain conditions [57] may be considered. For a

state sequence σ, the notation

$$\Delta J(\sigma) = dt \sum_{\widehat{x} \in \sigma} f(\widehat{x}) \tag{2.88}$$

denotes the cost of that sequence. Eq. (2.87) can be constructed recursively by utilizing (2.88): if σ denotes the trajectory of node N with parent N_{parent}, then

$$J[N] = J[N_{\text{parent}}] + \Delta J(\sigma). \tag{2.89}$$

Both the BU-RRT and BU-RRT* algorithms are comprised of a tree expansion step, used to continuously and incrementally grow the tree by simulating new trajectories; and an execution loop which periodically selects the best feasible path for execution and updates problem data online. This chapter focuses on the tree expansion step; the execution loop is presented in detail in Chapter 3 (its implementation for BU-RRT and BU-RRT* is similar).

The tree expansion step for BU-RRT is given in Algorithm 1. It starts with the current tree $\mathcal{T}$ at timestep t and seeks to add additional, robustly feasible nodes to the tree, subject to the constraints $\mathcal{X}_t$ and $\mathcal{U}$ tightened via the uncertainty sets S_w, S_x, and S_{jt} (Algorithm 1, line 1). First, a state is sampled uniformly from the environment via $x = \text{Sample}()$ (line 2). Next, a node in $\mathcal{T}$ nearest to x_{samp} in terms of some distance metric (here, Euclidean norm [56]) is identified (line 3) via the function

$$N_{\text{nearest}} = \text{Nearest}(\mathcal{T}, x) = \arg\min_{N \in \mathcal{T}} \|x - \widehat{x}[N]\|. \tag{2.90}$$

The robust steering law $\sigma = \text{RobustlySteer}(x, y, S, t)$ is then applied to steer the terminal state $\widehat{x}[N_{\text{nearest}}]$ to x_{samp} (line 4). The robust steering law returns a sequence of states originating at x and terminating at y in the form of a dynamically feasible trajectory, via (2.19). Additionally, the inputs u_t applied by this steering law must satisfy the input constraints S tightened starting from timestep t, such that the robust input constraints (2.25) are satisfied.

The resulting state sequence is then checked for robust feasibility via the boolean

Algorithm 1 Bounded-Uncertainty RRT, Tree Expansion

1: Inputs: tree $\mathcal{T}$; current timestep t; constraints $\mathcal{X}_t, \mathcal{U}$; uncertainty sets S_w, S_x, S_{jt}
2: $x_{\text{samp}} \leftarrow$ Sample()
3: $N_{\text{ncarest}} \leftarrow$ Nearest($\mathcal{T}, x_{\text{samp}}$)
4: $\sigma \leftarrow$ RobustlySteer($\widehat{x}[N_{\text{nearest}}], x_{\text{samp}}, \mathcal{U}, t[N_{\text{ncarest}}]$)
5: **if** RobustlyFeasible($\sigma, \mathcal{X}_t, t[\sigma]$) **then**
6: Create node $N_{\min}\{\sigma\}$
7: Add $N_{\min}$ to $\mathcal{T}$
8: **end if**

function RobustlyFeasible(σ, S, t) (line 4). This function tightens the state constraints S starting from timestep t, then checks whether the resulting robust state constraints (2.26)-(2.27) are satisfied at all timesteps in the sequence. If robustly feasible, a new node with that state sequence is created (line 6), then added to $\mathcal{T}$ (line 7).

The tree expansion step for BU-RRT* is given in Algorithm 2. It begins in the same manner as BU-RRT (lines 1–6), including the check of candidate node $N_{\min}$ for robust feasibility (line 5). However, if the node is robustly feasible in this case, it is created but not yet added to $\mathcal{T}$. Instead, nearby nodes are identified for possible connections via the function $\mathcal{N}$ = Near($\mathcal{T}, x, n$) (line 7), which returns a subset of nodes $\mathcal{N} \subseteq \mathcal{T}$. To enable asymptotic optimality guarantees (Section 2.4), BU-RRT* uses [56]

$$\mathcal{N} \;=\; \text{Near}(\mathcal{T}, x, n) \equiv \{N \in \mathcal{T} \mid \|\widehat{x}[N] - x\| \leq r_n\}, \tag{2.91}$$

where r_n is a radius that decreases with the number of tree nodes n.

Once the nearby nodes $\mathcal{N}$ are identified, BU-RRT* seeks to identify the lowest-cost, robustly feasible connection from those nodes to x_{samp} (lines 8–13). For each possible connection, a state sequence is simulated via the robust steering law (line 9). If the resulting sequence is robustly feasible, and the cost of that node – represented as the sum $J[N_{\text{near}}] + \Delta J(\sigma)$, via (2.89) – is lower than the cost of $N_{\min}$ (line 10), then a new node with this sequence replaces $N_{\min}$ (line 11). The lowest-cost node is ultimately added to $\mathcal{T}$ (line 14). Lines 7 13 are collectively referred to here as the *connect-nearby* step.

Algorithm 2 Bounded-Uncertainty RRT*, Tree Expansion

1: Inputs: tree $\mathcal{T}$; current timestep t; constraints $\mathcal{X}_t$,$\mathcal{U}$; uncertainty sets S_w, S_x, S_{jt}
2: $x_{\text{samp}} \leftarrow$ Sample()
3: $N_{\text{nearest}} \leftarrow \text{Nearest}(\mathcal{T}, x_{\text{samp}})$
4: $\sigma \leftarrow \text{RobustlySteer}(\widehat{x}[N_{\text{nearest}}], x_{\text{samp}}, \mathcal{U}, t[N_{\text{nearest}}])$
5: **if** $\text{RobustlyFeasible}(\sigma, \mathcal{X}_t, t[\sigma])$ **then**
6: Create node $N_{\min}\{\sigma\}$
7: $\mathcal{N}_{\text{near}} \leftarrow \text{Near}(\mathcal{T}, x_{\text{samp}}, |\mathcal{T}|)$
8: **for** $N_{\text{near}} \in \mathcal{N}_{\text{near}} \backslash N_{\text{nearest}}$ **do**
9: $\sigma \leftarrow \text{RobustlySteer}(\widehat{x}[N_{\text{near}}], x_{\text{samp}}, \mathcal{U}, t[N_{\text{near}}])$
10: **if** $\text{RobustlyFeasible}(\sigma, \mathcal{X}_t, t[\sigma])$ and $J[N_{\text{near}}] + \Delta J(\sigma) < J[N_{\min}]$ **then**
11: Replace $N_{\min}$ with new node $N_{\min}\{\sigma\}$
12: **end if**
13: **end for**
14: Add $N_{\min}$ to $\mathcal{T}$
15: **for** $N_{\text{near}} \in \mathcal{N}_{\text{near}} \backslash \text{Ancestors}(N_{\min})$ **do**
16: $\sigma \leftarrow \text{RobustlySteer}(\widehat{x}[N_{\min}], \widehat{x}[N_{\text{near}}], \mathcal{U}, t[N_{\min}])$
17: **if** $\text{RobustlyFeasible}(\sigma, \mathcal{X}_t, t[\sigma])$ and $J[N_{\min}] + \Delta J(\sigma) < J[N_{\text{near}}]$ **then**
18: Delete N_{near} from $\mathcal{T}$
19: Add new node $N_{\text{new}}\{\sigma\}$ to $\mathcal{T}$
20: Update descendants of N_{new} as needed
21: **end if**
22: **end for**
23: **end if**

Finally, a rewiring operation, referred to here as the *rewire-nearby step* (lines 15–22), is performed based on attempting robust connections from the new node $N_{\min}$ to nearby nodes, ancestors excluded (line 15). A state sequence is sampled via the steering law from $N_{\min}$ to the terminal state of each nearby node N_{near} (line 16). If the resulting sequence is robustly feasible, and the cost of that node is lower than the cost of N_{near}, then a new node with this sequence replaces N_{near} within the tree $\mathcal{T}$ (lines 18–19).

Each tightening term (2.28)-(2.33) is independent of the input sequence u_t applied to the system (2.19). Since variations in the input sequence are all that differentiates individual branches of the BU-RRT* tree, the values of these tightening terms for a given state depend only on the time elapsed between the tree root and that state. As such, each optimization and tightening amount for a given timestep should be stored after it is first computed, as it will get used repeatedly by other tree branches. Optimizations may even be pre-computed offline, depending on which details about

the motion planning problem and the environment are available *a priori*.

By using the RRT* rewiring mechanism with an exact steering law, all descendants of a rewired node remain dynamically feasible. However, the amount by which the constraints are tightened/modified may change due to rewiring, implying that the robust feasibility constraints (2.25)-(2.27) may need to be re-checked at descendant nodes. However, because the tightening amounts (2.28)-(2.33) can be stored after their initial computation and re-used, these re-checks can be computed efficiently. Additionally, under certain assumptions, uncertainty can be guaranteed to never increase due to rewiring, such that feasibility re-checks are not necessary. This is used in the proof of BU-RRT* probabilistic completeness in Section 2.4 below.

Finally, we note that this algorithm does not explicitly consider the task of selecting paths to reduce internal or map uncertainty; papers which explore this problem in more depth for sampling-based planning include Refs. [88, 101, 102] among many others. Rather, this algorithm demonstrates that robust and asymptotically optimal planning can be efficiently performed even in the presence of such uncertainty.

2.4 Analysis

This section establishes the theoretical properties of the BU-RRT and BU-RRT* algorithms. In particular, it is shown that both algorithms are probabilistically complete for problem (2.B), and additionally that BU-RRT* is asymptotically optimal. These results establish that, for a given feedback policy, both algorithms will find a robustly feasible path if one exists, while BU-RRT* will further refine that solution to converge asymptotically toward the lowest-cost path feasible for problem (2.B).

The proof of asymptotic feasibility is based on two arguments: (1) that reducing the uncertainty level at a given node cannot make it infeasible; and (2) under the assumption that uncertainty increases at future timesteps, rewiring the BU-RRT* tree reduces or does not change the uncertainty levels. To prove the first argument, some additional notation is needed.

The uncertainty levels (2.38)-(2.40) are collectively referred to below as Δ_{at}, where

$a = \{u, 0, 1, \ldots, n_o\}$, or collectively as Δ_t. Let $\Delta_{at}^{(i)}$ denote the ith element of Δ_{at}. Then Δ_{at_1} is said to be larger than Δ_{at_2}, written as $\Delta_{at_1} \geq \Delta_{at_2}$, if $\Delta_{at_1}^{(i)} \geq \Delta_{at_2}^{(i)}$ $\forall$ i. Additionally, Δ_{at} is said to be non-decreasing in t if $\Delta_{a(t+1)} \geq \Delta_{at}$ $\forall$ t. Conversely, Δ_{at_1} is said to be smaller than Δ_{at_2}, written as $\Delta_{at_1} \leq \Delta_{at_2}$, if $\Delta_{at_1}^{(i)} \leq \Delta_{at_2}^{(i)}$ $\forall$ i. Additionally, Δ_{at} is said to be non-increasing in t if $\Delta_{a(t+1)} \leq \Delta_{at}$ $\forall$ t.

Similarly, Δ_{t_1} is larger than Δ_{t_2}, written as $\Delta_{t_1} \geq \Delta_{t_2}$, if $\Delta_{at_1} \geq \Delta_{at_2}$, $\forall$ $a = \{u, 0, 1, \ldots, n_o\}$. The term Δ_t is said to be non-decreasing in t, written as $\Delta_{t+1} \geq \Delta_t$, if $\Delta_{t+1} \geq \Delta_t$, $\forall$ t. Conversely, Δ_{t_1} is smaller than Δ_{t_2}, written as $\Delta_{t_1} \leq \Delta_{t_2}$, if $\Delta_{at_1} \leq \Delta_{at_2}$, $\forall$ $a = \{u, 0, 1, \ldots, n_o\}$. The term Δ_t is said to be non-increasing in t, written as $\Delta_{t+1} \leq \Delta_t$, if $\Delta_{t+1} \leq \Delta_t$, $\forall$ t.

The following lemma establishes that, if the robustness constraints (2.25)-(2.27) are satisfied for some uncertainty level, they will remain satisfied if that uncertainty level is decreased.

Lemma 2.6. *Suppose $(\widehat{x}_t, u_t)$ is feasible for constraints (2.25)-(2.27) of problem (2.C) with uncertainty level Δ_t, and Δ is some uncertainty level such that $\Delta_t \geq \Delta$. Then $(\widehat{x}_t, u_t)$ is feasible for constraints (2.25)-(2.27) of problem (2.C) with uncertainty level Δ.*

Proof. Consider (2.61)-(2.66), associating Δ_t with the δ terms and Δ with the γ terms. This lemma is then evident from inspection of (2.70)-(2.72) ■.

The following assumptions are necessary to establish probabilistic completeness:

Assumption 2.7. *All of the following conditions are satisfied:*

1. *The set $\mathcal{U}$ consists of a finite number of inputs. Whenever inputs are simulated, they are chosen randomly from the set $\mathcal{U}$.*

2. *All tree vertices are separated by a distance of at least $\epsilon > 0$.*

3. *There exists a sequence of feasible inputs $u_0, u_1, \ldots, u_{t_f-1}$ such that the constraints of problem (2.C) are satisfied by the state sequence $\widehat{x}_0, \widehat{x}_1, \ldots, \widehat{x}_{t_f}$ and $\widehat{x}_{t_f} \in \mathcal{X}_{goal}$. All states in this sequence lie in the same bounded, open, and connected s_x-dimensional subset of $\mathbb{R}^{s_x}$.*

Assumption 2.7.3 ensures that at least one feasible solution exists. Assumptions 2.7.1-2.7.2 may not necessarily apply directly to many practical applications; however, as long as the input selection used in the steering law effectively expands the frontier of the tree, feasible solutions are likely to be identified in practice.

Theorem 2.8 (Probabilistic Completeness of BU-RRT). *Under Assumption 2.7, BU-RRT is probabilistically complete for problem (2.B).*

Proof. The probabilistic completeness of BU-RRT can be established using the same approach as the proof of probabilistic completeness for RRT [54]. Through the use of the discrete LTI dynamics, all timestep durations are constant, and all motions are locally constrained. Coupled with Assumptions 2.7.1-2.7.3, all assumptions used to establish probabilistic completeness for RRT are also satisfied by BU-RRT.

In particular, the argument proceeds by induction. Suppose that the BU-RRT tree currently contains the state $\widehat{x}_k$ generated by applying the inputs $u_0, u_1, \ldots, u_{k-1}$ from Assumption 2.7.3, for $k < t_f$. By Assumption 2.7.2, all tree vertices have a finite Voronoi region (whose volume is a function of ϵ), including $\widehat{x}_k$. As such, because all states have a non-zero likelihood of being sampled, $\widehat{x}_k$ has a non-zero likelihood of being selected as a nearest node for expansion. From Assumption 2.7.1, input u_k has a non-zero likelihood of being applied in order to generate state $\widehat{x}_{k+1}$. As the number of samples approaches infinity, the likelihood of $\widehat{x}_{k+1}$ thus being added as a descendant of $\widehat{x}_k$ approaches 1 [54]. Since $\widehat{x}_0$ is initialized as the tree root, the proof by induction is complete. Given the equivalence between (2.B) and (2.C) established in Theorem 2.1, BU-RRT is probabilistically complete for problem (2.B). ■

Additional assumptions are utilized for BU-RRT* probabilistic completeness, due to the potential of its connect-nearby and rewire-nearby steps (Section 2.3) to modify the uncertainty levels at tree nodes.

Assumption 2.9. *All obstacles are static, such that $\widehat{c}_{jt} \equiv \widehat{c}_j \; \forall \; t$ and $\mathcal{X}_t$ is the same for all t.*

Assumption 2.9 does *not* necessarily imply that $S_{jt} \equiv S_j \; \forall \; t$.

Assumption 2.10. *At least one of the following conditions is satisfied:*

1. *The uncertainty levels* Δ_{0t} *and* $\Delta_{jt}\ \forall\ j \in \mathbb{Z}_{1,n_o}$ *are non-decreasing in* t.
2. *No localization error is present, and the obstacle uncertainty levels are non-decreasing in* t.

Assumption 2.10 is in place to ensure that the uncertainty level of a node cannot increase due to either the connect-nearby or rewire-nearby step in BU-RRT*. This assumption is typically not restrictive in practice; further, the algorithm typically performs well even if several of the above theoretical assumptions are not satisfied.

Theorem 2.11 (Probabilistic Completeness of BU-RRT*). *Suppose Assumptions 2.7, 2.9, and 2.10 are satisfied. Then BU-RRT* is probabilistically complete for problem (2.B).*

Proof. BU-RRT* performs two additional steps in its tree expansion routine (Algorithm 2), relative to BU-RRT (Algorithm 1): the connect-nearby step (lines 7–13) and the rewire-nearby step (lines 15–22). Since BU-RRT has been shown to be probabilistically complete, it must be shown that these new steps do not break that completeness under the provided assumptions.

By Assumption 2.7.3 and Theorem 2.8, the state sequence $\{\widehat{x}_0, \widehat{x}_1, \ldots, \widehat{x}_{t_f}\}$ has a non-zero likelihood of being contained within a BU-RRT tree as the number of samples approaches infinity. If the same sample sequence is applied in BU-RRT*, then its tree will contain the same set of states $(\widehat{x}_0, \widehat{x}_1, \ldots, \widehat{x}_{t_f})$ as the number of samples approaches infinity.

For completeness, it is sufficient to show that neither the connect-nearby nor rewire-nearby steps in BU-RRT* can *increase* the uncertainty levels of any tree nodes. Indeed, suppose that some vertex $\widehat{x}_j$ of the path specified in Assumption 2.7.3 is modified by one of these steps, such that the (feasible) path from $\widehat{x}_0$ to $\widehat{x}_j$ may differ from $\{\widehat{x}_0, \widehat{x}_1, \ldots, \widehat{x}_j\}$. If these steps cannot increase uncertainty levels, then the uncertainty levels at descendants of $\widehat{x}_j$ either remain the same or are reduced. By Lemma 2.6, each state/input pair of the remaining path segment $\{\widehat{x}_j, \widehat{x}_{j+1}, \ldots, \widehat{x}_{t_f}\}$ remains feasible, such that probabilistic completeness is maintained.

Because $f(\cdot) \equiv 1$, *i.e.,* path cost equals path duration, either connect-nearby or rewire-nearby will strictly decrease the terminal timestep $t[N]$ of any affected node N and its descendants. Thus, showing that the uncertainty levels Δ_t are non-decreasing in t is sufficient to prove probabilistic completeness.

Consider the implicit forms (2.35)-(2.37) of the tightening terms on the process noise. Since $0 \in S_w$, the terms δ^w_{ut}, δ^w_{0t}, and $-\delta^w_{jt}$ are (element-wise) non-decreasing in t. Thus Δ_{ut} is non-decreasing in t.

If Assumption 2.10.1 is satisfied, then Δ_t is non-decreasing in t, as needed for completeness. Otherwise, suppose Assumption 2.10.2 is satisfied. The assumption on localization error implies that $\delta^x_{0t} = \delta^x_{jt} = 0 \; \forall \; j \in \mathbb{Z}_{1,n_o}, \; \forall \; t$, while the assumption on obstacle uncertainty implies δ_{jt} is non-decreasing in $t \;\; \forall \; j \in \mathbb{Z}_{1,n_o}$. The uncertainty levels can then be written as

$$\begin{aligned} \Delta_{ut} &= \delta^w_{ut}, \\ \Delta_{0t} &= \delta^w_{0t}, \\ \Delta_{jt} &= \delta^c_{jt} - \delta^w_{jt}, \; \forall \; j \in \mathbb{Z}_{1,n_o}. \end{aligned}$$

Each remaining term has been shown to be non-decreasing in t, implying that Δ_t is non-decreasing in t. Given the equivalence between problems (2.B) and (2.C) of Theorem 2.1, BU-RRT* is probabilistically complete for problem (2.B), completing the proof. ■

The proof of asymptotic optimality follows a similar path to the proof in Ref. [56]. However, because the feasible state space is a function of the planning timestep, the definitions must be modified accordingly.

Denote $\mathbb{X}_t$ as the set of all states $\widehat{x}_t$ satisfying constraints (2.25)-(2.27) at planning timestep t. A state $x \in \mathbb{X}_t$ is a δ-interior state of $\mathbb{X}_t$ if the closed ball of radius δ centered at x lies entirely inside $\mathbb{X}_t$. The δ-interior of $\mathbb{X}_t$ is defined as $\text{int}_\delta(\mathbb{X}_t) \equiv \{x \in \mathbb{X}_t \mid \mathbb{B}(x;\delta) \subseteq \mathbb{X}_t\}$, where $\mathbb{B}(x;r)$ denotes a ball of radius r centered at x. A robustly feasible path $\{\widehat{x}_0, \ldots, \widehat{x}_{t_f}\}$ is said to have strong δ-clearance if $\widehat{x}_t \in \text{int}_\delta(\mathbb{X}_t) \; \forall \; t \in \mathbb{Z}_{0,t_f}$. A robustly feasible path $p = \{\widehat{x}_0, \ldots, \widehat{x}_{t_f}\}$ is said to have weak δ-clearance if there

exists a path $p' = \{\widehat{x}'_0, \ldots, \widehat{x}'_{t_f}\}$ with strong δ-clearance, and a homotopy ψ with $\psi(0) = p$, $\psi(1) = p'$, and for all $\alpha \in (0, 1]$ there exists $\delta_\alpha > 0$ such that $\psi(\alpha)$ has strong δ_α-clearance [56].

Denote the cost of path p via objective (2.24) of problem (2.C) as $J(p)$. A path p^* that solves problem (2.C) is considered a robustly optimal solution if p^* has weak δ-clearance and, for any sequence of robustly feasible paths $\{p_n\}_{n \in \mathbb{Z}_{0,\infty}}$ with $\lim_{n\to\infty} p_n = p^*$, then $\lim_{n\to\infty} c(p_n) = c(p^*)$ [56].

Assumption 2.12. *All of the following conditions are satisfied:*

1. *The set of all points traversed by an optimal trajectory has measure zero [56].*

2. *The nearby node radius r used in line 7 of Algorithm 2 is a function of the number of tree nodes, N, and is chosen to be*

$$\begin{aligned} r_n &= \min\left\{\gamma\left(\frac{\log n}{n}\right)^{1/s_x}, \eta\right\}, \\ \gamma &> \gamma^* \equiv \left(2\left(1+\frac{1}{s_x}\right)\frac{\mu(\mathbb{X}_0)}{\zeta_{s_x}}\right)^{1/s_x}, \end{aligned}$$

where s_x is the dimension of the state space, $\mu(S)$ is the volume of set S, ζ_d is the volume of the d-dimensional unit sphere, and $\eta > 0$.

The definition of $\gamma^\star$ used by Karaman and Frazzoli [56] includes $\mu(\mathcal{X}_{\text{free}})$, where $\mathcal{X}_{\text{free}}$ is the volume of the collision-free space, rather than $\mu(\mathbb{X}_0)$. However, since $\mathbb{X}_0$ implies the state constraints at $t = 0$, where no tightening is yet needed for robustness, the formulations are equivalent.

Theorem 2.13 (Asymptotic Optimality of BU-RRT*). *Suppose Assumptions 2.7, 2.9, 2.10, and 2.12 are satisfied. Then BU-RRT* is asymptotically optimal for problem (2.B). In other words, for any realization of problem (2.B) that admits a robustly optimal solution with finite cost J^*, then*

$$\mathbb{P}\left(\left\{\lim_{n\to\infty} \sup J_n = J^*\right\}\right) = 1,$$

where J_n denotes the cost of the lowest-cost path in the BU-RRT tree after n tree expansion steps.*

Proof. The proof of asymptotic optimality established in Theorem 38 of Ref. [56] can also be utilized here, with minor modification. As is the case there, it is assumed that η is chosen to be sufficiently large. Additionally, Lemma 2.6 and the resulting Theorem 2.11 establish that the graph construction process in the proof of Theorem 38 [56] maintains probabilistic completeness.

The primary remaining task in this modified proof is to show that the choice of nearby node radius in Assumption 2.12.2 is valid. To fit into the framework of Theorem 38 [56], the choice of γ at each planning timestep t must satisfy

$$\gamma > \gamma_t \equiv \left(2\left(1+\frac{1}{s_x}\right)\frac{\mu(\mathbb{X}_t)}{\zeta_{s_x}}\right)^{1/s_x}.$$

Under Assumptions 2.9 and 2.10, it is clear that $\mu(\mathbb{X}_t)$ is non-decreasing in t, as each subsequent planning timestep further constrains the state space. As such, choosing $\gamma > \gamma^*$ for $r(N)$ implies that $\gamma > \gamma_t$, validating this approach to proving asymptotic optimality for problem (2.C). Given the equivalence between problems (2.B) and (2.C) established in Theorem 2.1, BU-RRT* is asymptotically optimal for problem (2.B), completing the proof. ■

2.5 Simulation Results

This section demonstrates the ability of the proposed algorithm to identify high-quality, robustly feasible trajectories for complex motion planning scenarios. First, detailed simulation trials are performed for a single integrator operating in a simple environment, subject to asymmetric process noise and obstacle placement uncertainty. Both closed-loop and open-loop control policies are considered for both RRT* and BU-RRT*, with an emphasis on solution quality and feasibility. BU-RRT* is then demonstrated for a cluttered environment, with all three types of uncertainty – process noise, localization error, and environment uncertainty – present. An additional

example considers the case of a non-convex localization error. Finally, a scaled-up scenario is presented in which a double integrator UAV must navigate through a cluttered environment. The algorithm has been implemented in Java, utilizing GLPK [103] and Java ILP [104] to perform the optimizations needed for (2.28)-(2.33).

2.5.1 Simple Scenario

Consider the 2D single integrator dynamics

$$x_{t+1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} dt & 0 \\ 0 & dt \end{bmatrix} u_t + \begin{bmatrix} dt & 0 \\ 0 & dt \end{bmatrix} w_t,$$

where $dt = 0.1$ s. The position variables x_t are constrained within a bounded, two-dimensional 10m $\times$ 10m environment, containing three obstacles with uncertain placement (Figure 2-1). The velocity inputs $u_t = (u_t^x, u_t^y)$ are modeled as a linear approximation of a 2-norm constraint $\|u_t\|_2 \leq \bar{v} = 0.5$ m/s, taking the form

$$\cos\left(\frac{2\pi d}{D_U}\right) u_t^x + \sin\left(\frac{2\pi d}{D_U}\right) u_t^y \leq \bar{v}, \ \forall\, d \in \mathbb{Z}_{1,D_U}, \ \forall\, t,$$

where $D_U = 36$ is the discretization level. The steering law executes the feasible input of largest magnitude at each timestep which moves the system in the direction from the old state to the new state. The system is modeled as a circular object with a 10-cm diameter, considered as a point mass during planning by expanding all obstacles by its radius.

The system is subject to both obstacle placement uncertainty and process noise which are *asymmetric*, and thus difficult to model via, *e.g.*, , zero-mean Gaussian process noise (Chapter 3). The process noise w_t at each timestep t is bounded within

the set (2.9) with

$$E^w = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix}, \quad f^w = \begin{bmatrix} 0.1 \\ 0.02 \\ 0.02 \\ 0.01 \end{bmatrix};$$

thus, in terms of Figure 2-1, the process noise is largest to the right and smallest from above. The three obstacles (which are static, such that $S_{jt} \equiv S_j$) have the placement uncertainties (2.11) defined by $E^1 = E^2 = E^3 = E^w$ and

$$f^1 = \begin{bmatrix} 0.2 \\ 0.2 \\ 0.2 \\ 0.2 \end{bmatrix}; \quad f^2 = \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \end{bmatrix}; \quad f^3 = \begin{bmatrix} 0.2 \\ 0.6 \\ 0.6 \\ 0.2 \end{bmatrix}.$$

The largest uncertainty is on the two concealed sides (from the perspective of the agent's initial state) of the largest obstacle, emulating a scenario where the agent may not be able to observe the full size of the obstacle. In this scenario, the external uncertainty is time-invariant, while the internal uncertainty is monotonically increasing with time. Thus, via Theorem 2.11, any nodes re-wired to a new path are guaranteed to not become more uncertain, ensuring that all descendant nodes remain robustly feasible.

The 2D position is sampled uniformly within the bounds of the feasible 2D environment. A path is considered to reach the goal if the final position is within 0.25m of the goal location. Finally, the nearby node function uses a maximum radius $\mu = 1$ m.

Four algorithms are compared throughout these results:

1. **Open-loop BU-RRT***, in which $P_k \equiv 0$ (Remark 2.2);
2. **Closed-loop BU-RRT***, using an affine state-feedback closed-loop policy with $K = -0.05I_2$ (Remark 2.3);

Table 2.1: Properties of solutions returned in Figure 2-1

Algorithm	Path Duration (s)	% Paths Feasible	Time per Node (ms)	Optim. Time (s)
Open-loop RRT*	17.9	18	14.6	0
Closed-loop RRT*		22		
Open-loop BU-RRT*	27.1	100	24.2	1.41
Closed-loop BU-RRT*	21.2	100	15.8	3.09

3. **Open-loop RRT*** [56]; and

4. **Closed-loop RRT*** [105, 106], which uses the same state-feedback closed-loop policy $K = -0.05I_2$, but for the RRT* algorithm.

First, each algorithm is used to grow a 5000-node tree from the initial state, then identify the lowest-cost (*e.g.*, shortest) available path. Figures 2-1 shows representative 5000-node trees (green) and minimum-cost solution paths (orange) generated by RRT* (Figure 2-1(a))[1], open-loop BU-RRT* (Figure 2-1(b)), and closed-loop BU-RRT* (Figure 2-1(c)). In these figures, the agent (brown, bottom-center) seeks to find a robustly feasible path to the goal (green circle, top-left), subject to environmental and obstacle constraints (black), process noise, and obstacle placement uncertainty (hull of possible obstacle locations in red). Table 2.1 lists several properties of the algorithms used and solution paths identified in Figure 2-1. Here path duration refers to time the system arrives at the goal when executing its lowest cost path, while the latter two columns denote the amount of computation used when running the algorithms.

The tree found by RRT* (Figure 2-1(a)) spans the nominally feasible space, yielding the shortest path of the three algorithms, but does not incorporate uncertainty during planning. For the BU-RRT* algorithms (Figures 2-1(b) and 2-1(c)), the evolution of the system uncertainty can be observed via the standoff distances maintained by tree paths relative to the environment and obstacle boundaries. The system maintains the largest distance from the rightmost environmental boundary due to the

[1]The trees generated by open-loop RRT* and closed-loop RRT* both consider the nominal dynamics, and thus are identical. Instead, the algorithms differ in their online control policies, as seen in Figure 2-2.

(a) RRT*

(b) BU-RRT*, open-loop (OL)

(c) BU-RRT*, closed-loop (CL)

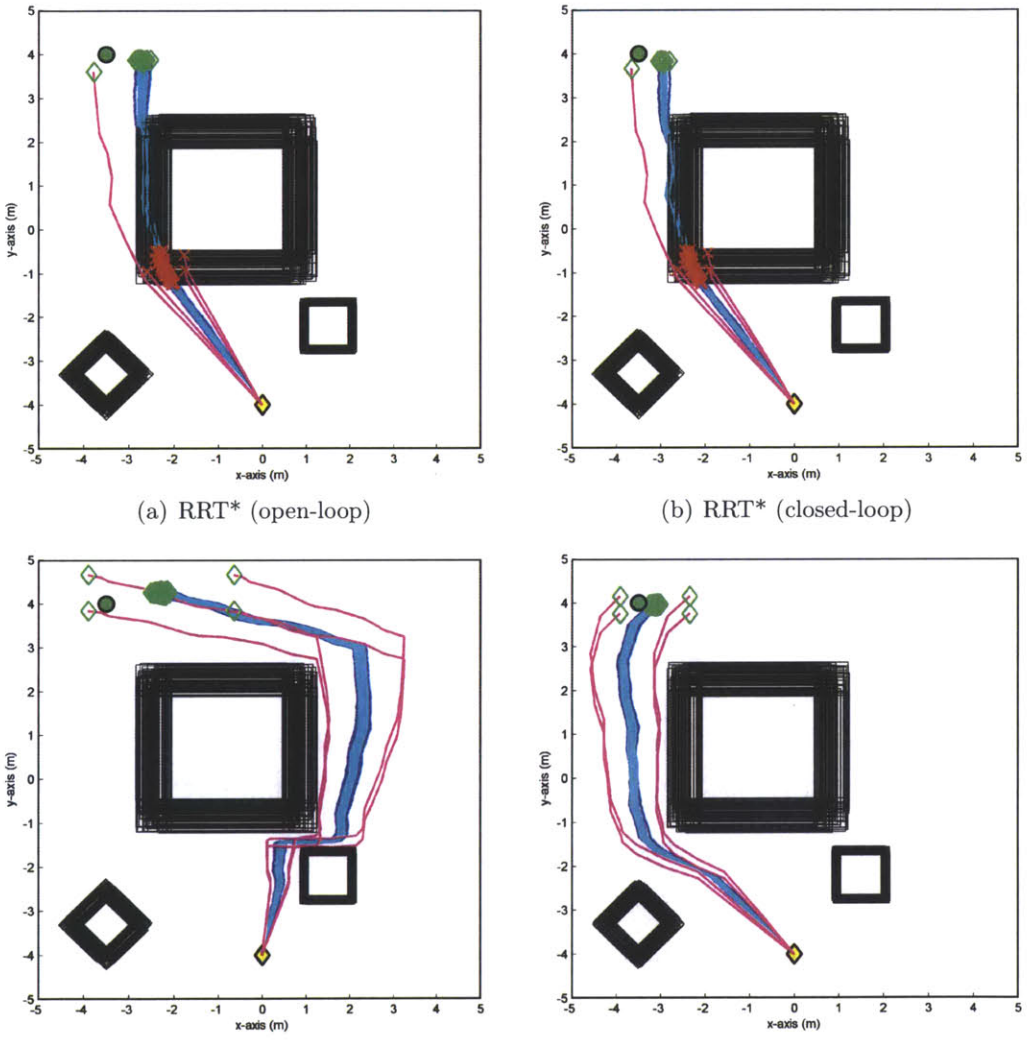
Figure 2-1: Representative trees and minimum-cost paths for RRT* and BU-RRT*, 2D single integrator, simple scenario

accumulation of a large amount of process noise toward that direction. The system maintains a particularly large standoff distance relative to obstacle 3 from the left, due to the combined effect of a large placement uncertainty and process noise in that direction.

Without feedback, the BU-RRT* planner is not able to identify any feasible paths reaching the goal on the left of obstacle 3 (Figure 2-1(b)). Thus, the planner ultimately selects a longer path taking it between obstacles 2 and 3. By tightening the input constraints in order to apply future state feedback in closed-loop BU-RRT*, the feasible space spanned by the tree is increased (Figure 2-1(c)). As a result, closed-loop BU-RRT* is able to identify a shorter, robustly feasible path passing obstacle 3 on the left.

Relative to nominal RRT*, BU-RRT* requires at most only 66% more computation per node in these results (Table 2.1), demonstrating its real-time suitability. Most of the optimization time (which is longer for closed-loop BU-RRT* due to input constraint tightening) is front-loaded in the first hundred nodes, as the LP optimizations (2.28)-(2.33) are performed up to the tree's current timestep horizon. These optimizations require only a few seconds of additional computation, which can be reduced if results are pre-computed.

A key metric for the solution paths generated in Figure 2-1 is how often they would actually be safe if executed for feasible uncertainty realizations. Figure 2-2 shows 100 simulations of the paths chosen by each algorithm in Figure 2-1, subject to various applied realizations of the uncertainty. As described in Theorem 2.1 (Section 2.2), the bounded-uncertainty RRT* approach provides robustness subject to any probability distribution within the uncertainty bounds. The obstacle placement uncertainty is assumed to be uniformly distributed within its bounds, with each realization shown in black. In 48 simulations, the process noise is uniformly distributed within S_w (cyan). In 48 simulations, the process noise randomly samples the vertices of S_w at each timestep (blue). Finally, in 4 simulations, each vertex of S_w is realized identically at *all* timesteps (magenta). Paths safely reaching the goal terminate in a green diamond; paths which collide with an obstacle terminate with a red "X."



(a) RRT* (open-loop)

(b) RRT* (closed-loop)

(c) BU-RRT* (open-loop)

(d) BU-RRT* (closed-loop)

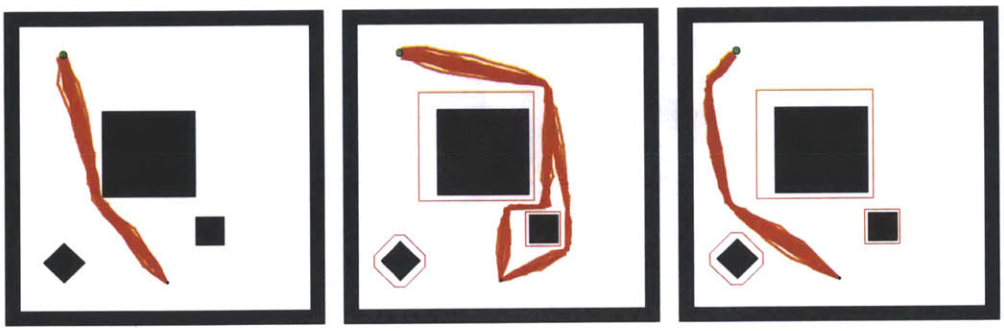
Figure 2-2: Simulated uncertainty realizations of paths chosen by each algorithm in Figure 2-1

When applying the input sequence identified by RRT* in open-loop (Figure 2-2(a)), only 18 of 100 paths safely reach the goal (Table 2.1); the other 82 paths collide with the large obstacle. When the closed-loop feedback is applied to RRT*, the number of safe paths increases only slightly, though the variation between safe paths decreases (Figure 2-2(b)). However, a majority of paths still do not reach the goal safely, due to not maintaining a sufficient distance from obstacles. In contrast, all 200 BU-RRT* simulations remain feasible; however, closed-loop BU-RRT* (Figure 2-2(d)) clearly reduces the variation in the simulated paths relative to open-loop BU-RRT* (Figure 2-2(c)).

Each algorithm is further tested for the simple scenario by analyzing the resulting final paths, and evolution of the path duration, over many trials. Figure 2-3 shows a composite image of all 25 final solution paths returned for each algorithm after 5000 nodes. Due to the optimizing nature of these algorithms, there is little variation in the solution paths across trials. The only qualitative difference is in the case of open-loop BU-RRT* (Figure 2-3(b)); 15 trials identify the shorter path between the two rightmost obstacles, while 10 trials ultimately pass around all obstacles to the right.

Figure 2-4 shows the evolution in the solution path duration, as a function of the number of nodes in each algorithm tree, across the set of 25 trials. The median value is indicated as a solid line; the shaded region denotes the 5th-to-95th percentiles. For each number of nodes, only trials which have identified a path to the goal are considered.

As expected, given the optimizing nature of all algorithms, path duration tends to decrease with additional nodes as those nodes are used to rewire the tree and refine solution paths. RRT* identifies the lowest-cost paths since it does not consider uncertainty within planning, though this leads to a high likelihood of infeasibility (Figure 2-2). Initially, paths identified by closed-loop BU-RRT* (red) tend to have higher path duration than those found by open-loop BU-RRT* (yellow). However, between 500 and 2500 nodes, all closed-loop BU-RRT* trials identify a robustly feasible path between the two leftmost obstacles, dramatically reducing the solution path duration.



(a) RRT* (b) BU-RRT* (OL) (c) BU-RRT* (CL)

Figure 2-3: Overlay of final solution paths, 2D single integrator, simple scenario

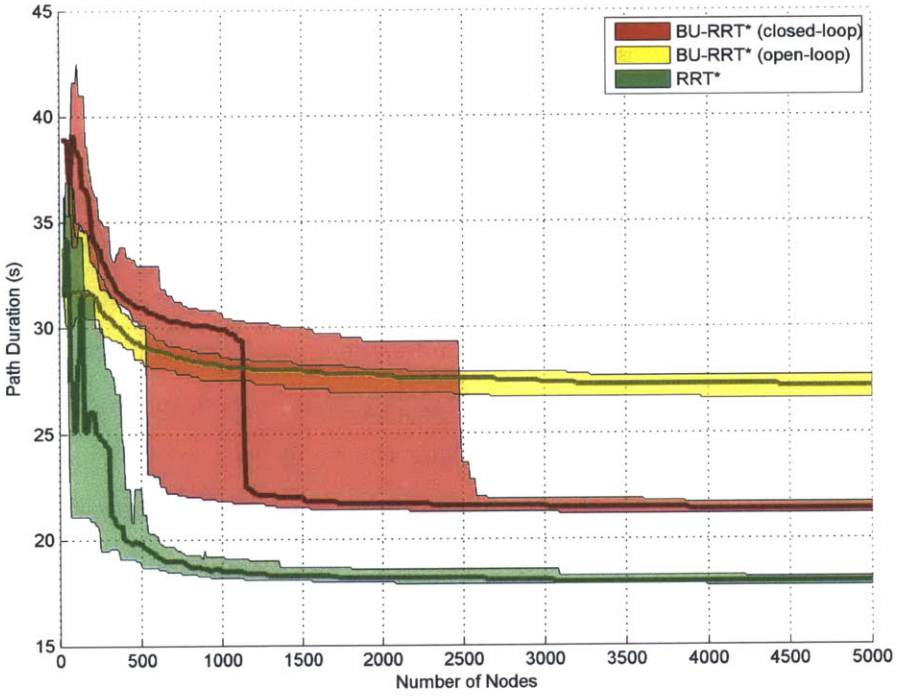
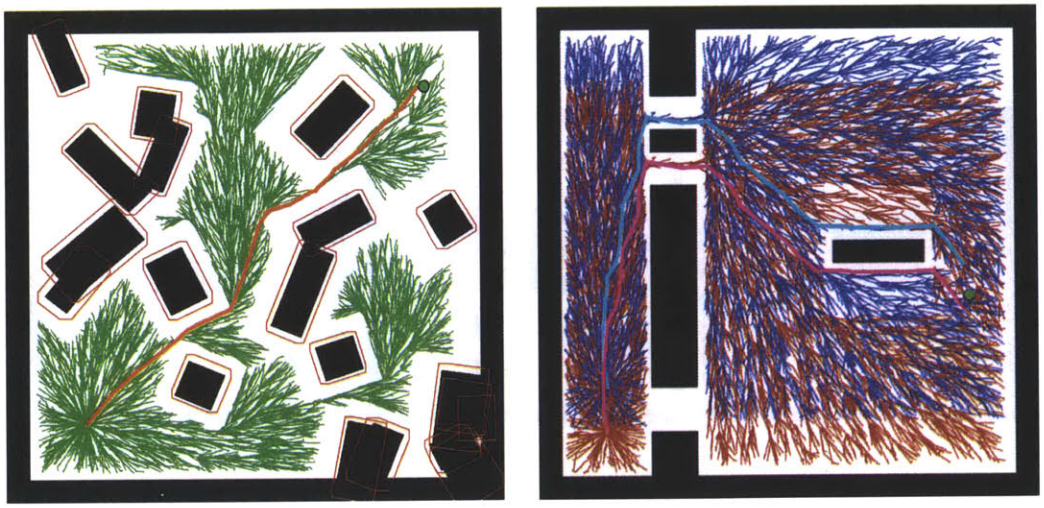


Figure 2-4: Evolution of path duration as a function of number of nodes, 2D single integrator, simple scenario



(a) Large, cluttered environment (20 obstacles)

(b) Environment for system with non-convex localization error

Figure 2-5: Representative trees and minimum-cost paths for open-loop BU-RRT*, 2D single integrator, more complex scenarios

2.5.2 Cluttered Scenario

Figure 2-5(a) demonstrates open-loop BU-RRT* operating in a more complex environment, containing 20 obstacles which each have an asymmetric placement uncertainty randomly chosen between 0.05m and 0.15m in each direction. The system is additionally subject to an initial state error of 0.1m in each direction, as well as a process noise one-fourth the magnitude of Section 2.5.1. Thus, the tree and minimum-cost path found in Figure 2-5(a) demonstrate robustness to asymmetric process noise, localization error, and randomized asymmetric placement uncertainty for each of the 20 obstacles: all three forms of uncertainty that can be incorporated within BU-RRT*.

The planner is automatically able to determine which pathways between obstacles are robustly feasible, then identify an asymptotically optimal path. The algorithm only uses 43.9 ms of computation per node, approximately double the computation used by the same algorithm in the simple example, despite having 20 obstacles instead of 3 obstacles. This demonstrates the scalability of this RRT-based approach relative to optimization/MPC-based frameworks, which may struggle in cluttered environments with many constraints.

2.5.3 Non-Convex Uncertainty Scenario

Consider a system subject to process noise

$$E^w = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix}, \quad f^w = \begin{bmatrix} 0.005 \\ 0.005 \\ 0.005 \\ 0.005 \end{bmatrix}$$

and a non-convex localization error. The localization error consists of two small convex sets spaced 1m apart vertically, where

$$E_1^x = E_2^x = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix}, \quad f_1^x = f_x^2 = \begin{bmatrix} 0.01 \\ 0.01 \\ 0.01 \\ 0.01 \end{bmatrix}$$

This corresponds to two possible realizations of the initial state, spaced 1m apart vertically. Localization error of this nature may occur, for example, due to symmetric environmental features found during localization.

Unlike previous figures, Figure 2-5(b) decomposes a representative BU-RRT* tree into linked blue and red versions, corresponding to the upper (blue) and lower (red) possible realizations of the initial state. Indeed, there are several regions of the state space which can be reached by only one, or none, of the two realizations safely. The cost-minimizing path selected has the two initial state realizations (cyan for upper, magenta for lower) passing on both sides of two obstacles to safely reach the goal. The planner correctly identifies that the passage near its initial state (bottom-left) is too small for both realizations to safely pass through.

2.5.4 Double Integrator Scenario

Consider the more complex 2D double integrator dynamics, representing a hovering vehicle,

$$x_{t+1} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} \frac{1}{2}dt^2 & 0 \\ 0 & \frac{1}{2}dt^2 \\ dt & 0 \\ 0 & dt \end{bmatrix} u_t + \begin{bmatrix} \frac{1}{2}dt^2 & 0 \\ 0 & \frac{1}{2}dt^2 \\ dt & 0 \\ 0 & dt \end{bmatrix} w_t,$$

where, again, $dt = 0.1$ s. The system operates in a randomized cluttered 10m $\times$ 10m environment containing 20 obstacles, with initial state $x_0 = (3.5, 3.5)$ m and goal region center $x_{\text{goal}} = (-3.5, -3.5)$ m. The system is also subject to velocity state constraints $|v_t^x| \leq v_{\max}$, $|v_t^y| \leq v_{\max}$, where $v_{\max} = 2.5$ m/s, as well as the input $u_t = (a_t^x, a_t^y)$ constraints $|a_t^x| \leq \bar{a}$, $|a_t^y| \leq \bar{a}$, where $\bar{a} = 5.0$ m/s^2.

The system is subject to a triangular process noise w_t at each timestep t, bounded within the set (2.9) with

$$E^w = \begin{bmatrix} \sqrt{3} & -1 \\ -\sqrt{3} & 1 \\ 0 & 1 \end{bmatrix}, \quad f^w = \begin{bmatrix} 0 \\ 0 \\ 0.025 \end{bmatrix},$$

representing, for example, a large wind disturbance in a northerly direction. Each of the 20 obstacles has a placement uncertainty randomly chosen between 0.1m and 0.2m in each direction; unlike Figure 2-5(a) the uncertainty is now oriented with the obstacle, rather than the environment.

The sampling strategy augments the position sampling for the single integrator with velocity sampling. With some small probability (5%), the 2D velocity is sampled uniformly within the full velocity constraints $|v_t^x| \leq v_{\max}$, $|v_t^y| \leq v_{\max}$, ensuring that every feasible state has a non-zero probability of being sampled. Otherwise, the 2D

velocity is sampled as

$$v_x = \tilde{v}\cos\tilde{\psi}, \quad v_y = \tilde{v}\sin\tilde{\psi},$$

where $\tilde{\psi}$ is sampled uniformly between 0 and 2π, and $\tilde{v}$ is sampled uniformly between $V_{\min} = 1.0$ m/s and $V_{\max} = 2.0$ m/s. The robust steering law fits a cubic spline to each coordinate in order to maintain an approximate traversal speed $\bar{v} = 1.0$ m/s; any splines that violate the (possibly tightened) acceleration input or velocity state bounds are considered infeasible (see Appendix). A path is considered to reach the goal if the final position is within 0.25m of the goal location, while the nearby node function uses a maximum radius $\mu = 5$ m.

Figure 2-6 demonstrates a representative 2500-node tree (green) and final solution path (orange) generated by open-loop bounded-uncertainty RRT* for this scenario, subject to the aforementioned asymmetric process noise and obstacle placement uncertainty. A robustly feasible path to the goal is quickly found, though the degree of suboptimality in the solution path is higher due to both the reduced number of nodes and the increased state dimension.

2.6 Conclusions

This chapter has presented a novel sampling-based algorithm for guaranteed feasibility of linear systems subject to bounded process noise, localization error, and/or obstacle placement uncertainty. The algorithm can efficiently tighten constraints to maintain robust feasibility for any given feedback policy while also maintaining asymptotic optimality. As will be seen in Chapter 5, this algorithm acts as the bounded-uncertainty counterpart to CC-RRT* and its probabilistic feasibility guarantees [80]. The algorithm also extends previous RMPC formulations to consider additional forms of uncertainty. Simulation results demonstrate the ability of this planner to identify high-quality safe paths in complex scenarios.

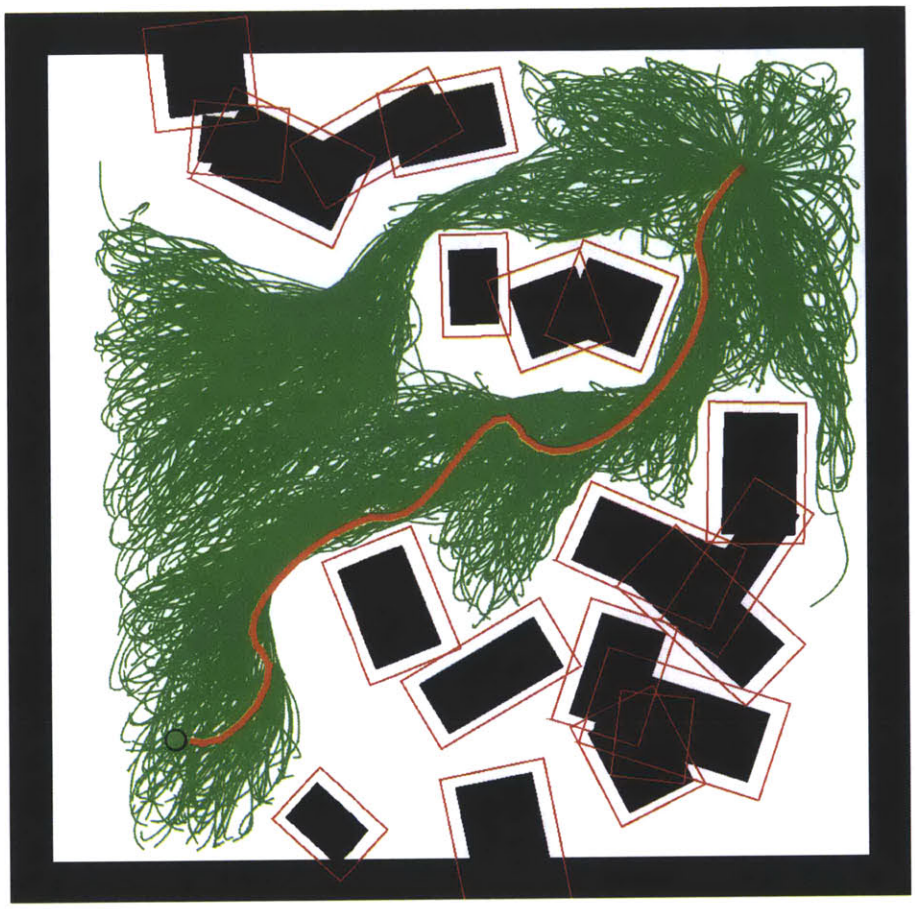


Figure 2-6: Representative trees and minimum-cost paths for BU-RRT*, 2D double integrator, cluttered scenario

Appendix: Double Integrator Steering Law

The double integrator steering law propagates the system forward T seconds, using a piecewise linear acceleration. This steering law can be decoupled for each position coordinate, so consider the general position coordinate x. The acceleration, or $\ddot{x}$ (a_x for the x-coordinate, a_y for the y-coordinate, and a_z for the z-coordinate if applicable), is parameterized as

$$\ddot{x}(t) = at + b,$$

where a and b are scalars. The objective of the steering law is to generate a trajectory that starts ($t = 0$) at initial position p_1 and velocity v_1, and ends ($t = T$) at terminal position p_2 and velocity v_2. Integrating the above equation and inserting the initial conditions yields

$$\begin{aligned} \dot{x}(t) &= \frac{1}{2}at^2 + bt + v_1, \\ x(t) &= \frac{1}{6}at^3 + \frac{1}{2}bt^2 + v_1 t + p_1. \end{aligned}$$

By inserting the terminal conditions and rearranging, this can be written as the system of equations

$$\begin{bmatrix} \frac{1}{2}T^2 & T \\ \frac{1}{6}T^3 & \frac{1}{2}T^2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} v_2 - v_1 \\ p_2 - p_1 - v_1 T \end{bmatrix}.$$

Solving yields

$$\begin{aligned} \begin{bmatrix} a \\ b \end{bmatrix} &= \frac{12}{T^4} \begin{bmatrix} Tp_1 - Tp_2 + \frac{1}{2}T^2 v_1 + \frac{1}{2}T^2 v_2 \\ -\frac{1}{2}T^2 p_1 + \frac{1}{2}T^2 p_2 - \frac{1}{3}T^3 v_1 - \frac{1}{6}T^3 v_2 \end{bmatrix} \\ &= \begin{bmatrix} 12T & -12T & 6T^2 & 6T^2 \\ -6T^2 & 6T^2 & -4T^3 & -2T^3 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ v_1 \\ v_2 \end{bmatrix}. \end{aligned}$$

The input acceleration components a_x, a_y, and a_z are assumed to be bounded in magnitude by $\bar{a}$. Since the acceleration parameterization is piecewise linear, only the starting and ending acceleration terms need to be checked for each coordinate, for each trajectory. Thus, if $|b| \leq \bar{a}$ and $|aT + b| \leq \bar{a}$, the parameterization is admissible; if not, the trajectory is inadmissible. The trajectory is also inadmissible if either velocity component ever exceeds $v_{\max}$ in magnitude.

The only remaining question is how to set T. This implementation seeks to approximate a traversal speed of $\bar{v}$, subject to a minimum propagation time $\bar{T} =$

1 s. The distance between the two states is $d = \sqrt{(p_2^x - p_1^x)^2 + (p_2^y - p_1^y)^2}$ (where the lettered superscripts denote the type of coordinate) in 2 dimensions and $d = \sqrt{(p_2^x - p_1^x)^2 + (p_2^y - p_1^y)^2 + (p_2^z - p_1^z)^2}$ in 3 dimensions. The propagation time is then chosen to be

$$T = \min\left\{\bar{T},\ \left\lfloor \frac{d}{\bar{v} \cdot dt} \right\rfloor dt\right\},$$

where $\lfloor \cdot \rfloor$ is the floor function; for convenience, the original propagation time $(d/\bar{v})$ is rounded down to the nearest multiple of dt.

Chapter 3

Probabilistic Robustness via Chance Constraints

This chapter presents a novel sampling-based planner, CC-RRT, which generates robust trajectories in real-time subject to process noise, localization/initial state error, and dynamic and/or uncertain environmental constraints. Probabilistic feasibility is guaranteed for linear Gaussian systems by using chance constraints to ensure that the probability of constraint violation does not exceed some user-specified threshold [36]. Additionally, CC-RRT leverages the trajectory-wise constraint checking of RRT [53] to efficiently bound the risk of constraint violation online [46]. As a result, CC-RRT can quickly identify trajectories subject to both internal and environmental uncertainty, with guaranteed minimum bounds on constraint satisfaction probability at each timestep and along each path.

3.1 Problem Statement

Consider the linear time-invariant (LTI) discrete-time system dynamics (2.1) subject to process noise

$$w_t \sim \mathcal{N}(0, P_w), \tag{3.1}$$

where $\mathcal{N}(\widehat{a}, P_a)$ represents a Gaussian random variable with mean $\widehat{a}$ and covariance P_a. The disturbance w_t is unknown at current and future timesteps, but has a known unbounded probability distribution (3.1). The disturbances w_t are independent and identically distributed across all timesteps. The initial/current state x_0 may be assumed to be either perfectly known or uncertain with known probability distribution

$$x_0 \sim \mathcal{N}(\widehat{x}_0, P_{x_0}). \tag{3.2}$$

As defined in Chapter 2, the system is additionally subject to constraints (2.5)-(2.6) acting on the system state and input. For each obstacle, the shape and orientation are assumed to be known, while the placement is uncertain. This is represented as

$$\mathcal{X}_{jt} = \mathcal{X}_j^0 + c_{jt}, \quad \forall\, j \in \mathbb{Z}_{1,n_o}, \tag{3.3}$$

$$c_{jt} \sim \mathcal{N}(\widehat{c}_{jt}, P_{c_{jt}}), \quad \forall\, j \in \mathbb{Z}_{1,n_o}. \tag{3.4}$$

In this model, for the jth obstacle, $\mathcal{X}_j^0 \subset \mathbb{R}^{n_x}$ is a convex polytope of known, fixed shape, while $c_{jt} \in \mathbb{R}^{n_x}$ represents an uncertain and/or time-varying translation, represented as a Gaussian uncertainty distribution.

In summary, the system is subject to three separate types of uncertainty: process noise (3.1), localization/initial state error (3.2), and/or obstacle placement uncertainty (3.4). Each uncertainty, though unknown prior to realization, is assumed to be accurately modeled as Gaussian uncertainty.

The primary objective of the motion planning problem is to reach some goal region $\mathcal{X}_{\text{goal}} \subset \mathbb{R}^{n_x}$ while ensuring the input constraints (2.6) are satisfied, while the state constraints (2.5) are *probabilistically* satisfied. This is represented via two types of chance constraints,

$$\mathbb{P}\left(x_t \in \mathcal{X}_t\right) \geq \delta_s, \quad \forall\, t, \tag{3.5}$$

$$\mathbb{P}\left(\bigwedge_t x_t \in \mathcal{X}_t\right) \geq \delta_p, \tag{3.6}$$

where $\mathbb{P}(\cdot)$ denotes probability and $\delta_s, \delta_p \in [0.5, 1]$ for the chance constraints[1]. The constraint (3.5) dictates that the state constraints be satisfied at each timestep with a probability of at least δ_s, while the constraint (3.6) dictates that the state constraints be satisfied over *all* timesteps with a probability of at least δ_p.

Consider the expected dynamics (2.19); since there is unbounded uncertainty in the state, it is assumed sufficient for the expected mean $\widehat{x}_t$ to reach the goal region $\mathcal{X}_{\text{goal}} \subset \mathbb{R}^{n_x}$. Denote

$$t_f = \inf\{t \in \mathbb{Z}_{0,\infty} \mid \widehat{x}_t \in \mathcal{X}_{\text{goal}}\}. \tag{3.7}$$

The path planner seeks to approximately solve the optimal control problem

$$\begin{aligned}
(3.\text{A}) \quad \min_{u_t} \quad & \phi_f(\widehat{x}_{t_f}, \mathcal{X}_{\text{goal}}) + \sum_{t=0}^{t_f-1} \phi(\widehat{x}_t, \mathcal{X}_{\text{goal}}, u_t) && (3.8)\\
\text{s.t.} \quad & x_{t+1} = Ax_t + Bu_t + Gw_t, \quad w_t \sim \mathcal{N}(0, P_w), \quad x_0 \sim \mathcal{N}(\widehat{x}_0, P_{x_0}), \quad \forall\, t,\\
& \widehat{x}_{t+1} = A\widehat{x}_t + Bu_t, \quad \forall\, t,\\
& u_t \in \mathcal{U} \quad \forall\, t,\\
& \mathbb{P}\left(x_t \in \mathcal{X}_t\right) \geq \delta_s, \quad \forall\, t,\\
& \mathbb{P}\left(\bigwedge_t x_t \in \mathcal{X}_t\right) \geq \delta_p,\\
& \mathcal{X}_t = \mathcal{X} \backslash \mathcal{X}_{1t} \backslash \cdots \backslash \mathcal{X}_{n_o t}, \quad \forall\, t,\\
& \mathcal{X}_{jt} = \mathcal{X}_j^0 + c_{jt}, \quad c_{jt} \sim \mathcal{N}(\widehat{c}_{jt}, P_{c_{jt}}), \quad \forall\, j \in \mathbb{Z}_{1,n_o}, \quad \forall\, t,
\end{aligned}$$

where ϕ, ϕ_f are cost functions to be optimized. These functions may be generalized to more complex forms, such as incorporating the state and input constraints or representing undesirable behaviors in "soft constraint" form, *e.g.,* proximity to constraint boundaries, fuel usage, etc. By using $\widehat{x}_t$ rather than x_t within the objective (3.8), the stochastic elements of this optimization manifest themselves only in the chance constraints (3.5)-(3.6). Section 3.2 details how these chance constraints can

[1]The bound of 0.5 is chosen such that the conditional mean of state distributions must be nominally feasible; see Lemma 5.2.

be implemented as a deterministic optimization.

In practice, the optimization (3.8) is solved repeatedly as the system navigates the environment in real-time, with x_0 being set to the current uncertainty distribution (or state, if perfectly known).

3.2 Chance Constraints

This section reviews the chance constraint formulation that is incorporated and extended in the CC-RRT framework to provide probabilistic guarantees on safety for sampled trajectories. This formulation builds upon and extends the framework considered in Blackmore *et al.* [36], by considering both time-step-wise and path-wise chance constraints as well as environmental uncertainty.

Given a sequence of inputs $u_0, \ldots, u_{t_f-1}$, the distribution of the state x_t, represented as the random variable $\mathbf{X}_t$, can be shown to be Gaussian [36]:

$$\begin{aligned} P(\mathbf{X}_t|u_0,\ldots,u_{t_f-1}) &\sim P(\mathbf{X}_t|u_0,\ldots,u_{t-1}) \\ &\sim \mathcal{N}(\widehat{x}_t, P_{x_t}) \ \ \forall\, t \in \mathbb{Z}_{0,t_f}. \end{aligned} \tag{3.9}$$

The mean $\widehat{x}_t$ and covariance P_{x_t} can be represented either explicitly as

$$\widehat{x}_t = A^t\widehat{x}_0 + \sum_{k=0}^{t-1} A^{t-k-1}Bu_k \ \ \forall\, t \in \mathbb{Z}_{0,t_f}, \tag{3.10}$$

$$P_{x_t} = A^t P_{x_0}(A^T)^t + \sum_{k=0}^{t-1} A^{t-k-1}GP_wG^T(A^T)^{t-k-1} \ \ \forall\, t \in \mathbb{Z}_{0,t_f}, \tag{3.11}$$

or implicitly as

$$\widehat{x}_{t+1} = A\widehat{x}_t + Bu_t \ \ \forall\, t \in \mathbb{Z}_{0,t_f-1}, \tag{3.12}$$

$$P_{x_{t+1}} = AP_{x_t}A^T + GP_wG^T \ \ \forall\, t \in \mathbb{Z}_{0,t_f-1}. \tag{3.13}$$

Note that (3.12) effectively updates the distribution mean $\widehat{x}_t$ using the disturbance-free dynamics (2.19), and that (3.13) is independent of the input sequence and thus

can be computed *a priori* off-line. The latter property does not necessarily apply if the dynamics are nonlinear (Section 3.5).

To render the chance constraints (3.5)-(3.6) tractable, it is desirable to decompose them first by timestep, then by obstacle. First, (3.5)-(3.6) can be equivalently stated in terms of constraint violation, rather than constraint satisfaction:

$$\mathbb{P}\left(x_t \notin \mathcal{X}_t\right) \leq 1 - \delta_s, \quad \forall\, t \in \mathbb{Z}_{0,t_f}, \tag{3.14}$$

$$\mathbb{P}\left(\bigvee_{t=0}^{t_f} x_t \notin \mathcal{X}_t\right) \leq 1 - \delta_p. \tag{3.15}$$

In particular, the latter chance constraint is now considering the possibility of the state constraints being violated at *at least* one timestep. Temporal independence generally cannot be assumed; in other words,

$$\mathbb{P}\left(\bigvee_{t=0}^{t_f} x_t \notin \mathcal{X}_t\right) \neq \sum_{t=0}^{t_f} \mathbb{P}\left(x_t \notin \mathcal{X}_t\right). \tag{3.16}$$

As an example of why independence generally does not hold, consider the example of a system subject to initial state uncertainty but zero process noise. In this situation, whether the system can safely satisfy the constraints at one timestep is highly correlated with whether it was able to satisfy the constraints at previous timesteps. For example, if this system is safely traversing parallel to a linear constraint, it can be assured of continued safety while it continues to do so, due to the lack of process noise.

However, Boole's inequality, also known as the union bound [36], can be applied to upper-bound this probability:

$$\mathbb{P}\left(\bigvee_{t=0}^{t_f} x_t \notin \mathcal{X}_t\right) \leq \sum_{t=0}^{t_f} \mathbb{P}\left(x_t \notin \mathcal{X}_t\right). \tag{3.17}$$

It is similarly the case that the probabilities of violating each component of the state constraints (2.5) are not independent. However, Boole's inequality can again be

applied, decomposing the chance constraints further:

$$\mathbb{P}\left(x_t \notin \mathcal{X}_t\right) \leq \mathbb{P}\left(x_t \notin \mathcal{X}\right) + \sum_{j=1}^{n_o} \mathbb{P}\left(x_t \in \mathcal{X}_{jt}\right), \quad \forall\, t \in \mathbb{Z}_{0,t_f}. \tag{3.18}$$

Consider the jth obstacle at the tth timestep. Because the obstacle is polyhedral, it can be represented through the conjunction of linear inequalities

$$\bigwedge_{i=1}^{n_j} a_{ij}^T(x_t - c_{ijt}) < 0 \quad \forall\, t \in \mathbb{Z}_{0,t_f}, \tag{3.19}$$

where n_j is the number of constraints defining the jth obstacle, and c_{ijt} is a point nominally (*i.e.*, $c_{jt} = \widehat{c}_{jt}$) on the ith constraint at timestep t. Here a_{ij} is not dependent on t, since the obstacle shape and orientation are fixed. To avoid the jth obstacle at the tth timestep, the system must satisfy the disjunction of constraints

$$\bigvee_{i=1}^{n_j} a_{ij}^T(x_t - c_{ijt}) \geq 0. \tag{3.20}$$

To avoid the obstacle, it is sufficient to not satisfy any one of the constraints in the conjunction (3.19). To collide with the obstacle, all of the constraints in (3.19) must be satisfied. Thus it is true that

$$\begin{aligned} \mathbb{P}(\text{collision with } j\text{th obstacle at timestep } t) &= \mathbb{P}\left(\bigwedge_{i=1}^{n_j} a_{ij}^T(\mathbf{X}_t - \mathbf{C}_{ijt}) < 0\right) \\ &\leq \mathbb{P}\left(a_{ij}^T(\mathbf{X}_t - \mathbf{C}_{ijt}) < 0\right) \quad \forall\, i \in \mathbb{Z}_{1,n_j}, \end{aligned} \tag{3.21}$$

where $\mathbf{C}_{ijt} = c_{ijt} + (\mathbf{C}_{jt} - \widehat{c}_{jt})$ is a random variable due to (3.3)-(3.4). As (3.21) imposes n_j upper bounds on the probability of collision for a given obstacle and timestep, only the tightest of those bounds need be considered:

$$\mathbb{P}(\text{collision with } j\text{th obstacle at } t) \leq \min_{j \in \mathbb{Z}_{1,n_j}} \mathbb{P}\left(a_{ij}^T(\mathbf{X}_t - \mathbf{C}_{ijt}) < 0\right). \tag{3.22}$$

Suppose it is desired that the probability of collision with the jth obstacle at

timestep t be less than or equal to Δ. To ensure this is the case, (3.22) implies that it is only necessary to show that one of the constraints for the obstacle is satisfied with probability less than or equal to Δ:

$$\bigvee_{i=1}^{n_j} \mathbb{P}\left(a_{ij}^T(\mathbf{X}_t - \mathbf{C}_{ijt}) < 0\right) \leq \Delta. \tag{3.23}$$

Consider the change of variable

$$\mathbf{V} = a_{ij}^T(\mathbf{X}_t - \mathbf{C}_{ijt}), \tag{3.24}$$

such that (3.23) can be written as

$$\bigvee_{i=1}^{n_j} \mathbb{P}\left(\mathbf{V} < 0\right) \leq \Delta; \tag{3.25}$$

this random variable is also Gaussian. The mean and covariance are computed as follows:

$$\begin{aligned} \widehat{v} &= \mathbb{E}[\mathbf{V}] = a_{ij}^T\widehat{x}_t - \mathbb{E}[a_{ij}^T(c_{ijt} + (\mathbf{C}_{jt} - \widehat{c}_{jt}))] \\ &= a_{ij}^T\widehat{x}_t - a_{ij}^T c_{ijt} \\ &= a_{ij}^T(\widehat{x}_t - c_{ijt}), \end{aligned} \tag{3.26}$$

$$\begin{aligned} P_v &= \sqrt{\mathbb{E}[(\mathbf{V} - \widehat{v})(\mathbf{V} - \widehat{v})^T]} \\ &= \sqrt{\mathbb{E}[(a_{ij}^T\mathbf{X}_t - a_{ij}^T\mathbf{C}_{ijt} - a_{ij}^T\widehat{x}_t + a_{ij}^T c_{ijt})(a_{ij}^T\mathbf{X}_t - a_{ij}^T\mathbf{C}_{ijt} - a_{ij}^T\widehat{x}_t + a_{ij}^T c_{ijt})^T]} \\ &= \sqrt{\mathbb{E}[(a_{ij}^T(\mathbf{X}_t - \widehat{x}_t) - a_{ij}^T(\mathbf{C}_{ijt} - c_{ijt}))(a_{ij}^T(\mathbf{X}_t - \widehat{x}_t) - a_{ij}^T(\mathbf{C}_{ijt} - c_{ijt}))^T]} \\ &= \sqrt{\mathbb{E}[(a_{ij}^T(\mathbf{X}_t - \widehat{x}_t) - a_{ij}^T(\mathbf{C}_{jt} - \widehat{c}_{jt}))(a_{ij}^T(\mathbf{X}_t - \widehat{x}_t) - a_{ij}^T(\mathbf{C}_{jt} - \widehat{c}_{jt}))^T]} \\ &= \sqrt{a_{ij}^T(P_{x_t} + P_{c_{jt}})a_{ij}}. \end{aligned} \tag{3.27}$$

As seen in (3.27), the combined uncertainty of this planning problem for obstacle j at timestep t is conveniently expressed as a sum of covariances.

Each probabilistic constraint in the disjunction (3.25) can be shown to be equivalent to a deterministic constraint [36]

$$P(\mathbf{V} < 0) \leq \Delta \quad \Leftrightarrow \quad \widehat{v} \geq \sqrt{2} P_v \text{erf}^{-1}(1 - 2\Delta), \tag{3.28}$$

where $\text{erf}(\cdot)$ denotes the standard error function.

Using this, the constraints (3.20) are probabilistically satisfied for the true state x_t if the conditional mean $\widehat{x}_t$ satisfies the modified constraints

$$\bigvee_{i=1}^{n_j} a_{ij}^T (\widehat{x}_t - c_{ijt}) \geq \bar{b}_{ijt} \equiv \sqrt{2} P_v \text{erf}^{-1}(1 - 2\Delta). \tag{3.29}$$

The term $\bar{b}_{ijt}$ represents the amount of *deterministic* constraint tightening necessary to ensure *probabilistic* constraint satisfaction.

Next, consider the polyhedral state constraints $\mathcal{X}$, represented as the conjunction of linear inequalities

$$\bigwedge_{i=1}^{n_E} a_{i0}^T (x_t - c_{i0}) < 0 \quad \forall\, t \in \mathbb{Z}_{0,t_f}, \tag{3.30}$$

where n_E is the number of environmental constraints defining $\mathcal{X}$, and c_{i0} is a point on the ith constraint. Because $\mathcal{X}$ is deterministic and time-invariant, c_{i0} is also deterministic and time-invariant. Violation of the constraints $\mathcal{X}$ at the tth timestep is equivalent to the disjunction of constraints

$$\bigvee_{i=1}^{n_E} a_{i0}^T (x_t - c_{i0}) \geq 0. \tag{3.31}$$

Boole's inequality can be applied one more time, such that

$$\mathbb{P}\left(\bigvee_{i=1}^{n_E} a_{i0}^T (\mathbf{X}_t - c_{i0}) \geq 0 \right) \leq \sum_{i=1}^{n_E} \mathbb{P}\left(a_{i0}^T (\mathbf{X}_t - c_{i0}) \geq 0 \right). \tag{3.32}$$

Suppose it is desired that the probability of violating the ith constraint of $\mathcal{X}$ at the tth timestep be less than or equal to Δ_0. Proceeding in a similar manner as above, it

is straightforward to show that this is the case by satisfying the modified constraint

$$a_{i0}^T(\widehat{x}_t - c_{i0}) < -\bar{b}_{i0} \equiv -\sqrt{2}P_v \text{erf}^{-1}(1 - 2\Delta_0), \tag{3.33}$$

$$P_v = \sqrt{a_{i0}^T P_{x_t} a_{i0}}, \tag{3.34}$$

where $\bar{b}_{i0}$ represents the necessary tightening of the environmental constraints to achieve probabilistic constraint satisfaction.

3.2.1 Offline Risk Allocation

The tightening terms $\bar{b}_{ijt}$ in (3.29) and $\bar{b}_{i0}$ in (3.33) are each a function of two quantities: the imposed risk allocation (Δ and Δ_0, respectively) and the state and obstacle covariances (P_{x_t} and $P_{c_{jt}}$, respectively). Since the covariances can be computed off-line, the tightened constraints (3.29) and (3.33) can be computed off-line if specific values of Δ and Δ_0 are pre-allocated. In so doing, the complexity of the nominal formulation need not increase when incorporating chance constraints. In a similar manner as Ref. [36], Probabilistic feasibility of any state or state sequence can be checked via these tightened, deterministic constraints. The consideration of risk in this framework is analogous to Blackmore *et al.*, but now using sampling-based planning (Section 3.3).

In order to so do, the risk of constraint violation must be allocated *a priori* to each timestep, obstacle, and constraint, in accordance with each usage of Boole's inequality above. For the jth obstacle at timestep t, denote the risk allocation Δ of (3.29) as $\overline{\Delta}_{jt}$. Similarly, for the ith constraint of the environmental bounds $\mathcal{X}$ at timestep t, denote the risk allocation Δ_0 of (3.33) as $\overline{\Delta}_{it0}$. For a given Δ and Δ_0, at timestep t the state distribution conditional mean $\widehat{x}_t$ must satisfy (3.29) for all obstacles and (3.33) for all constraints of $\mathcal{X}$. The stochastic motion planning problem (3.A) of (3.8) then takes the form

$$
\begin{aligned}
\text{(3.B)} \quad \min_{u_t} \quad & \phi_f(\widehat{x}_{t_f}, \mathcal{X}_{\text{goal}}) + \sum_{t=0}^{t_f-1} \phi(\widehat{x}_t, \mathcal{X}_{\text{goal}}, u_t) && (3.35)\\
\text{s.t.} \quad & \widehat{x}_{t+1} = A\widehat{x}_t + Bu_t, \quad \forall\, t,\\
& P_{x_{t+1}} = AP_{x_t}A^T + GP_wG^T, \quad \forall\, t,\\
& u_t \in \mathcal{U} \quad \forall\, t,\\
& \bigvee_{i=1}^{n_j} a_{ij}^T(\widehat{x}_t - c_{ijt}) \geq \sqrt{2}P_v\text{erf}^{-1}\left(1 - 2\overline{\Delta}_{jt}\right) \quad \forall\, j \in \mathbb{Z}_{1,n_o},\ \forall\, t, && (3.36)\\
& a_{i0}^T(\widehat{x}_t - c_{i0}) < -\sqrt{2}P_v\text{erf}^{-1}\left(1 - 2\overline{\Delta}_{it0}\right) \quad \forall\, i \in \mathbb{Z}_{1,n_E},\ \forall\, t. && (3.37)
\end{aligned}
$$

If the risk allocations are chosen to fall within the desired thresholds δ_s and δ_p, then solutions to problem (3.B) will also be feasible for problem (3.A).

Theorem 3.1 (Robust Feasibility of CC-RRT Offline Risk Allocation). *Consider problem (3.B) with $\overline{\Delta}_{jt} > 0\ \forall\ j \in \mathbb{Z}_{1,n_o}\ \forall\ t$ and $\overline{\Delta}_{it0} > 0\ \forall\ i \in \mathbb{Z}_{1,n_E}\ \forall\ t$ chosen such that*

$$
\begin{aligned}
\sum_{i=1}^{n_E} \overline{\Delta}_{it0} + \sum_{j=1}^{n_o} \overline{\Delta}_{jt} &= 1 - \delta_s, \ \forall\, t, && (3.38)\\
\sum_{t=0}^{t_f} \left(\sum_{i=1}^{n_E} \overline{\Delta}_{it0} + \sum_{j=1}^{n_o} \overline{\Delta}_{jt} \right) &= 1 - \delta_p. && (3.39)
\end{aligned}
$$

If the path of state distributions $(\widehat{x}_0, P_{x_0}), (\widehat{x}_1, P_{x_1}), \ldots, (\widehat{x}_{t_f}, P_{x_{t_f}})$ specified by the input sequence $u_0, u_1, \ldots, u_{t_f-1}$ is feasible for problem (3.B), then it is also feasible for problem (3.A).

Proof. Applying (3.22)-(3.23) (with $\Delta = \overline{\Delta}_{jt}$) to (3.18) yields

$$
\begin{aligned}
\mathbb{P}(x_t \notin \mathcal{X}_t) &\leq \mathbb{P}(x_t \notin \mathcal{X}) + \sum_{j=1}^{n_o} \mathbb{P}(x_t \in \mathcal{X}_{jt}) && (3.40)\\
&\leq \mathbb{P}(x_t \notin \mathcal{X}) + \sum_{j=1}^{n_o} \overline{\Delta}_{jt}. && (3.41)
\end{aligned}
$$

Additionally applying (3.32)-(3.33) (with $\Delta_0 = \overline{\Delta}_{it0}$) to (3.41) yields

$$\begin{aligned} \mathbb{P}\left(x_t \notin \mathcal{X}_t\right) &\leq \overline{\Delta}_{it0} + \sum_{j=1}^{n_o} \overline{\Delta}_{jt} && (3.42) \\ &= 1 - \delta_s, && (3.43) \end{aligned}$$

satisfying (3.5) of problem (3.A).

From (3.17) and the above discussion,

$$\begin{aligned} \mathbb{P}\left(\bigvee_{t=0}^{t_f} x_t \notin \mathcal{X}_t\right) &\leq \sum_{t=0}^{t_f} \mathbb{P}\left(x_t \notin \mathcal{X}_t\right) && (3.44) \\ &\leq \left(\sum_{i=1}^{n_E} \overline{\Delta}_{it0} + \sum_{j=1}^{n_o} \overline{\Delta}_{jt}\right) && (3.45) \\ &= 1 - \delta_p, && (3.46) \end{aligned}$$

satisfying (3.6) of problem (3.A). All other constraints are common between the two formulations, implying feasibility of problem (3.A). ■

The most straightforward way to allocate risk is to simply allocate equally to each component of the state constraints (2.5) [36], *i.e.*, each obstacle and the environmental constraints. Doing so for the time-step-wise chance constraint (3.14) yields

$$\begin{aligned} \overline{\Delta}_{jt}^{(s)} &= \frac{1-\delta_s}{n_o+1} \; \forall \; j,t, && (3.47) \\ \overline{\Delta}_{it0}^{(s)} &= \frac{1-\delta_s}{(n_o+1)n_E} \; \forall \; i,t, && (3.48) \end{aligned}$$

where the $(\cdot)^{(s)}$ superscript denotes time-step-wise allocations. Using an equal allocation for the path-wise chance constraint (3.15) yields

$$\begin{aligned} \overline{\Delta}_{jt}^{(p)} &= \frac{1-\delta_p}{(t_f+1)(n_o+1)} \; \forall \; j,t, && (3.49) \\ \overline{\Delta}_{it0}^{(p)} &= \frac{1-\delta_p}{(t_f+1)(n_o+1)n_E} \; \forall \; i,t, && (3.50) \end{aligned}$$

where the $(\cdot)^{(p)}$ superscript denotes path-wise allocations. If both types of constraints

are imposed, the more restrictive values are used, *i.e.*, $\overline{\Delta}_{jt} = \min\{\overline{\Delta}_{jt}^{(s)}, \overline{\Delta}_{jt}^{(p)}\} \ \forall \ j,t$ and $\overline{\Delta}_{it0} = \min\{\overline{\Delta}_{it0}^{(s)}, \overline{\Delta}_{it0}^{(p)}\} \ \forall \ i,t$. However, other types of risk allocations may be considered.

Through this offline risk allocation approach, the nominal complexity of the original problem is maintained, simply with modified/tightened constraint boundaries. However, such an approach is often quite conservative in practice, as it requires *a priori* allocation of risk to problem constraints. In most cases, it will not be known in advance which constraints are likely to be active at each timestep of a potentially long path through the environment. If a constraint is allocated more risk than it typically incurs online, the margin goes unused. If a constraint is allocated less risk than it typically incurs online, solutions paths will have to maintain a larger standoff from these constraints than desired in order to stay within the bounds. Both outcomes lead to more conservative planning. Additionally, this approach must bound the number of path timesteps ahead of time in order to ensure the probabilistic guarantees are met, limiting the available planning horizon. Approaches such as Iterative Risk Allocation (IRA) [41] address this issue by adjusting risk allocations based on their proclivity to become active. However, this approach iterates on the risk allocation through successive optimizations, requiring additional computation.

3.2.2 Online Risk Evaluation

Alternately, it is possible to identify a more precise bound on the probability of collision, by leveraging the relationship in (3.28) to compute the exact probability of satisfying each individual constraint for a given distribution $\mathcal{N}(\widehat{x}, P_x)$. This novel operation is only possible via trajectory-wise constraint checking, as in a sampling-based algorithm such as RRT; it cannot be performed within a single optimization. Like IRA, risk is dynamically assigned to each constraint set; however, whereas IRA requires a series of optimizations to do so, a sampling-based algorithm such as CC-RRT (Section 3.3) can compute suitable risk bounds immediately for each trajectory that is simulated. Though the resulting approach is still conservative due to the use

of Boole's inequality (Section 3.2), it provides much tighter bounds in practice that anything achievable *a priori* via offline risk evaluation.

Through repeated use of this operation, a bound can be computed for each timestep of a trajectory, as well as the trajectory itself, on the probability of collision. This bound can be of great use for heuristics within the RRT algorithm (Section 5.2.1, in addition to checking probabilistic feasibility. There is some tradeoff in the computational effort needed to compute this bound at each timestep, but the increase is sufficiently small to maintain the approach's suitability for on-line implementation.

First, a lemma establishes evaluation of a scalar chance constraint, of the form used in (3.28).

Lemma 2.2. *If* $\mathbf{V} \sim \mathcal{N}(\widehat{v}, P_v)$ *is a Gaussian random variable, then*

$$P(\mathbf{V} < 0) = \frac{1}{2}\left(1 - \text{erf}\left[\frac{\widehat{v}}{\sqrt{2}P_v}\right]\right). \tag{3.51}$$

Proof. Again consider (3.28), rewritten here as

$$P(\mathbf{V} < 0) \leq \gamma \quad \Leftrightarrow \quad \widehat{v} \geq f(\gamma), \tag{3.52}$$

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 $\widehat{v} = f(\bar{\gamma})$. Exploiting the equivalence in (3.52), for some $\epsilon > 0$ (where $|\bar{\gamma} \pm \epsilon| < 1$) it is the case that

$$\bar{\gamma} - \epsilon < P(\mathbf{V} < 0) \leq \bar{\gamma} + \epsilon. \tag{3.53}$$

As $\epsilon \to 0$, (3.52) becomes the equivalence

$$P(\mathbf{V} < 0) = \bar{\gamma} \quad \Leftrightarrow \quad \widehat{v} = f(\bar{\gamma}). \tag{3.54}$$

Solving for $\bar{\gamma}$ yields the desired relationship. ■

Consider the ith constraint of the jth obstacle at timestep t, as specified in (3.21), with associated change of variable (3.24). Let $\Delta_{ijt}(\widehat{x}, P_x)$ denote the probability that this constraint is satisfied for a Gaussian distribution with mean $\widehat{x}$ and covariance P_x; from (3.51),

$$\begin{aligned}\Delta_{ijt}(\widehat{x}_t, P_{x_t}) &= \mathbb{P}\left(a_{ij}^T(\mathbf{X}_t - \mathbf{C}_{ijt}) < 0\right) \\ &= \frac{1}{2}\left(1 - \operatorname{erf}\left[\frac{a_{ij}^T(\widehat{x}_t - c_{ijt})}{\sqrt{2a_{ij}^T(P_{x_t} + P_{c_{jt}})a_{ij}}}\right]\right). \end{aligned} \tag{3.55}$$

Similarly, let Δ_{i0t} denote the probability that the ith constraint of $\mathcal{X}$ is violated at timestep t for a Gaussian distribution with mean $\widehat{x}$ and covariance P_x; then

$$\begin{aligned}\Delta_{i0t}(\widehat{x}_t, P_{x_t}) &= \mathbb{P}\left(a_{i0}^T(\mathbf{X}_t - c_{i0}) \geq 0\right) \\ &= \frac{1}{2}\left(1 - \operatorname{erf}\left[\frac{a_{i0}^T(c_{i0} - \widehat{x}_t)}{\sqrt{2a_{i0}^T(P_{x_t})a_{i0}}}\right]\right). \end{aligned} \tag{3.56}$$

As shown in the proof of robust feasibility below, these components can then be inserted into each usage of Boole's inequality (3.17), (3.18), (3.32) to directly bound the probabilities of constraint violation. Define the terms

$$\Delta_{jt}(\widehat{x}_t, P_{x_t}) = \min_{i=1,\ldots,n_j} \Delta_{ijt}(\widehat{x}_t, P_{x_t}), \tag{3.57}$$

$$\Delta_{0t}(\widehat{x}_t, P_{x_t}) = \sum_{i=1}^{n_0} \Delta_{i0t}(\widehat{x}_t, P_{x_t}), \tag{3.58}$$

$$\Delta_t(\widehat{x}_t, P_{x_t}) = \Delta_{0t}(\widehat{x}_t, P_{x_t}) + \sum_{j=1}^{n_o} \Delta_{jt}(\widehat{x}_t, P_{x_t}), \tag{3.59}$$

$$\Delta(\widehat{x}_t, P_{x_t}) = \sum_{t=0}^{t_f} \Delta_t(\widehat{x}_t, P_{x_t}). \tag{3.60}$$

Finally, the original problem (3.A) is posed using these risk bounds:

$$\text{(3.C)} \quad \min_{u_t} \quad \phi_f(\widehat{x}_{t_f}, \mathcal{X}_{\text{goal}}) + \sum_{t=0}^{t_f-1} \phi(\widehat{x}_t, \mathcal{X}_{\text{goal}}, u_t) \tag{3.61}$$

$$\text{s.t.} \quad \widehat{x}_{t+1} = A\widehat{x}_t + Bu_t, \quad \forall\, t,$$

$$P_{x_{t+1}} = AP_{x_t}A^T + GP_wG^T, \quad \forall\, t,$$

$$u_t \in \mathcal{U} \quad \forall\, t, \tag{3.62}$$

$$\Delta_t(\widehat{x}_t, P_{x_t}) \leq 1 - \delta_s, \quad \forall\, t, \tag{3.63}$$

$$\Delta(\widehat{x}_t, P_{x_t}) \leq 1 - \delta_p. \tag{3.64}$$

where Δ_t and Δ are as defined in (3.59)-(3.60). It is now shown that any feasible path for this problem is also feasible for problem (3.A).

Theorem 3.3 (Robust Feasibility of CC-RRT Online Risk Evaluation). *If the path of state distributions* $(\widehat{x}_0, P_{x_0}), (\widehat{x}_1, P_{x_1}), \ldots, (\widehat{x}_{t_f}, P_{x_{t_f}})$ *specified by the input sequence* $u_0, u_1, \ldots, u_{t_f-1}$ *is feasible for problem (3.C), then it is also feasible for problem (3.A).*

Proof. By applying (3.17)-(3.18) and (3.32), (3.17) can be represented as

$$\begin{aligned}
\mathbb{P}\left(\bigvee_{t=0}^{t_f} x_t \notin \mathcal{X}_t\right) &\leq \sum_{t=0}^{t_f}\left[\mathbb{P}\left(x_t \notin \mathcal{X}\right) + \sum_{j=1}^{n_o}\mathbb{P}\left(x_t \in \mathcal{X}_{jt}\right)\right] \\
&\leq \sum_{t=0}^{t_f}\left[\sum_{i=1}^{n_0}\mathbb{P}\left(a_{i0}^T(\mathbf{X}_t - c_{i0}) \geq 0\right)\right. \\
&\qquad \left. + \sum_{j=1}^{n_o}\min_{i=1,\ldots,n_j}\mathbb{P}\left(a_{ij}^T(\mathbf{X}_t - \mathbf{C}_{ijt}) < 0\right)\right].
\end{aligned}$$

By using (3.55)-(3.60), then

$$\begin{aligned}
\mathbb{P}\left(\bigvee_{t=0}^{t_f} x_t \notin \mathcal{X}_t\right) &\leq \sum_{t=0}^{t_f}\left[\sum_{i=1}^{n_0}\Delta_{i0t}(\widehat{x}_t, P_{x_t}) + \sum_{j=1}^{n_o}\min_{i=1,\ldots,n_j}\Delta_{ijt}(\widehat{x}_t, P_{x_t})\right] \\
&= \sum_{t=0}^{t_f}\left[\Delta_{0t}(\widehat{x}_t, P_{x_t}) + \sum_{j=1}^{n_o}\Delta_{jt}(\widehat{x}_t, P_{x_t})\right] \\
&= \sum_{t=0}^{t_f}\Delta_t(\widehat{x}_t, P_{x_t}) \qquad (3.65) \\
&= \Delta(\widehat{x}_t, P_{x_t}), \qquad (3.66)
\end{aligned}$$

From (3.65), to satisfy the time-step-wise chance constraint (3.14), it is thus sufficient to show that (3.63) is satisfied. To satisfy the path-wise chance constraint (3.15), it is sufficient to show that (3.64) is satisfied. ■

While online risk evaluation is much less conservative than offline risk allocation, there is still conservatism introduced by the use of Boole's inequality and the relationship (3.22) [36]. In particular, the latter relationship ultimately bounds risk based on overlap between the uncertainty distribution and the most active obstacle constraint. This overlap is computed assuming that constraint is an infinite hyperplane, rather than part of a finite, bounded object. As a result, there may be feasible paths to problem (3.A) that cannot be identified through the deterministic bounds used in (3.B) or (3.C). However, in the case of online risk evaluation (3.61), the bounds are still effective in practice in finding robustly feasible paths without too much conservatism. 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 still effectively remove risk, and could not be easily formulated in an optimization-based framework.

Finally, suppose that the input u_t applying to the dynamics (2.1) is generated via the closed-loop controller

$$u_t = r_t - Lx_t, \qquad (3.67)$$

where $r_t \in \mathbb{R}^{n_u}$ is some reference state and L is a gain matrix of suitable dimension. Under this modification, the system dynamics (2.1) can then be written as

$$x_{t+1} = (A - BL)x_t + Br_t + Gw_t. \tag{3.68}$$

Thus, the closed-loop dynamics can be treated identically to the open-loop dynamics, by replacing A with $A - BL$ and treating r_t as the input sequence rather than u_t. In particular, the conditional mean and covariance of the state distribution (3.12)-(3.13) are written as

$$\widehat{x}_{t+1} = (A - BL)\widehat{x}_t + Br_t \quad \forall\, t \in \mathbb{Z}_{0,t_f-1}, \tag{3.69}$$

$$P_{x_{t+1}} = (A - BL)P_{x_t}(A - BL)^T + GP_wG^T \quad \forall\, t \in \mathbb{Z}_{0,t_f-1}. \tag{3.70}$$

3.3 CC-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 3-1 provides a diagram of the CC-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

Two versions of the CC-RRT algorithm are presented. In offline CC-RRT, all constraints are tightened offline, such that the nominal planning problem is modified to ensure probabilistic feasibility without additional online computation (Section 3.2.1),

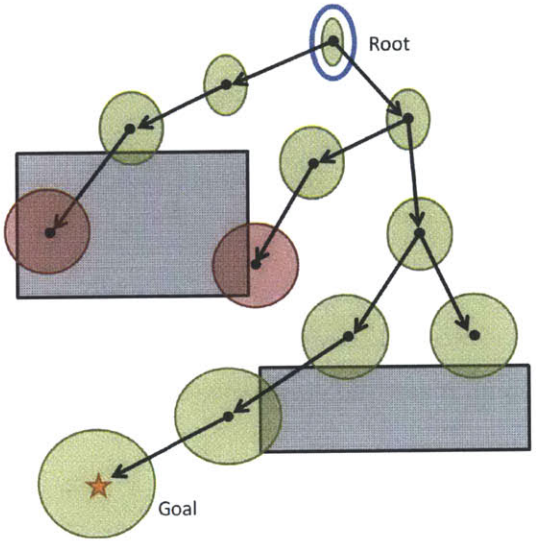


Figure 3-1: Chance-constrained RRT

Online CC-RRT, on the other hand, leverages the trajectory-wise constraint checking of the RRT algorithm to explicitly compute a bound on the probability of collision at each node (Section 3.2.2). While this latter approach does require online computation of risk bounds, results show that these computations are highly efficient, and in return significantly reduce the conservatism in trajectory simulation and execution.

To grow a tree of dynamically feasible trajectories, it is necessary for the RRT algorithm, and thus CC-RRT algorithm, to have an accurate model of the vehicle dynamics (2.1) for simulation. For each simulated trajectory, the CC-RRT algorithm propagates the predicted state distribution, which under the assumption of Gaussian uncertainty is itself Gaussian, via (3.9). Thus, at each timestep of each simulated trajectory, it is only necessary to propagate the state conditional mean (3.12) and covariance (3.13), rewritten here in model form as

$$\widehat{x}_{t+k+1|t} = A\widehat{x}_{t+k|t} + Bu_{t+k|t}, \tag{3.71}$$

$$P_{t+k+1|t} = AP_{t+k|t}A^T + GP_wG^T, \tag{3.72}$$

where t is the current system timestep and $(\cdot)_{t+k|t}$ denotes the predicted value of the variable at timestep $t+k$. When using the $(\cdot)_{t+k|t}$ notation, the subscript $(\cdot)_x$ in the

covariance P_x is often suppressed in order to simplify presentation.

This algorithm consists of two primary components for real-time planning. The first component, the tree expansion step, is used to incrementally grow the tree by simulating new trajectories; it is generally run continuously, using any available computational resources. The second component, the execution loop, periodically selects the best available path for execution, in addition to updating the current state of the vehicle, environment, and tree. 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 [107]. Given the extensive computations involved to construct the tree, as much of the tree should be retained as possible, especially for real-time applications. As such, the resulting algorithm runs both components simultaneously, executing some portion of the tree while continuing to grow it.

The subsequent sections detail the algorithms for each of these components, including versions of the tree expansion step for offline CC-RRT (Section 3.3.1) and online CC-RRT (Section 3.3.2), as well as the execution loop (Section 3.3.3).

3.3.1 Offline CC-RRT Tree Expansion Step

The tree expansion step used for offline CC-RRT is given in Algorithm 3. Each time Algorithm 3 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 5–12). This trajectory is incrementally simulated by selecting some feasible input (line 8), then applying (3.71) to yield the state distribution mean $\widehat{x}_{t+k+1|t}$ at the next timestep (line 9). 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.

After each forward simulation, probabilistic feasibility of the new state distribution is checked by determining whether $\widehat{x}_{t+k|t}$ satisfies the tightened state constraints (3.36)-(3.37) (line 7), for appropriate choices of $\overline{\Delta}_{jt}$ and $\overline{\Delta}_{it0}$ (*i.e.*, satisfying

Algorithm 3 Offline CC-RRT, Tree Expansion

1: Inputs: tree $\mathcal{T}$, current timestep 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 node
6: $\widehat{x}_{t+k|t} \leftarrow$ final state distribution mean of N_{near}
7: **while** $\widehat{x}_{t+k|t}$ satisfies (3.36)-(3.37) **and** $\widehat{x}_{t+k|t}$ has not reached x_{samp} **do**
8: Select input $u_{t+k|t} \in \mathcal{U}$
9: Simulate $\widehat{x}_{t+k+1|t}$ using (3.71)
10: Create intermediate nodes as appropriate
11: $k \leftarrow k+1$
12: **end while**
13: **for** each feasible node N **do**
14: Update cost estimates for N
15: Add N to $\mathcal{T}$
16: Try connecting N to $\mathcal{X}_{\text{goal}}$ (lines 5–12)
17: **if** connection to $\mathcal{X}_{\text{goal}}$ feasible **then**
18: Update upper-bound cost-to-go of N and ancestors
19: **end if**
20: **end for**
21: **end for**

(3.38)-(3.39) – recall that the planning horizon length must be bounded accordingly). Implicit in (3.36)-(3.37) is the modeling of $P_{t+k|t}$ and $P_{c_{jt}}$; however, as both the covariances and the tightened constraints can be computed offline, they need not be referenced directly within the algorithm. Trajectory simulation continues until either the state is no longer probabilistically feasible, or the distribution mean has reached the sample (line 7). In this manner, this algorithm behaves similarly to nominal RRT, with the distribution mean $\widehat{x}_{t+k|t}$ replacing the true (disturbance- free) state $x_{t+k|t}$, and the state constraints $\mathcal{X}_t$ replaced with the tightened constraints (3.36)-(3.37) to enforce probabilistic feasibility bounds.

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 10). Each node contains a trajectory segment, simulated over possibly many timesteps; future connections from this node must begin at the end of this trajectory (line 6). As a result, one or more probabilistically feasible nodes may

be generated from trajectory simulation, each of which is added to the tree (line 13).

A number of heuristics are also employed to facilitate tree growth, identify probabilistically feasible trajectories to the goal, and identify lower-cost paths (in terms of (3.35)) 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 [105]. The nearest node selection scheme (lines 3–4) 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 [107]. Each time a sample is generated, $m \geq 1$ attempts are made to connect a node to this sample before being discarded [105]. An attempt is also made to connect newly-added nodes directly to $\mathcal{X}_{\text{goal}}$ (line 16). 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 18) 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 [107].

3.3.2 Online CC-RRT Tree Expansion Step

The tree expansion step used for online CC-RRT is given in Algorithm 4. The primary distinction between this algorithm and offline CC-RRT (Algorithm 3) is the manner in which probabilistic feasibility is checked. Given the state distribution mean $\widehat{x}_{t+k|t}$ and covariance $P_{t+k|t}$ (which can be computed offline, or calculated directly as in line 10) at simulation timestep $t+k$, the time-step-wise risk bound $\Delta_t(\widehat{x}_{t+k|t}, P_{t+k|t})$ and path-wise risk bound $\Delta(\widehat{x}_{t+k|t}, P_{t+k|t})$ are computed online using (3.59) and (3.60), respectively (lines 7 and 13). Trajectory simulation continues until either quantity violates (3.64) (for Δ_t) or (3.63) (for Δ), or the distribution mean has reached the sample (line 8).

Algorithm 4 Online CC-RRT, Tree Expansion

1: Inputs: tree $\mathcal{T}$, current timestep 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 node
6: $(\widehat{x}_{t+k|t}, P_{t+k|t}) \leftarrow$ final state distribution of N_{near}
7: Compute $\Delta_t(\widehat{x}_{t+k|t}, P_{t+k|t})$ using (3.59) and $\Delta(\widehat{x}_{t+k|t}, P_{t+k|t})$ using (3.60)
8: **while** $\Delta(\widehat{x}_{t+k|t}, P_{t+k|t}) \leq 1 - \delta_p$ **and** $\Delta_t(\widehat{x}_{t+k|t}, P_{t+k|t}) \leq 1 - \delta_s$ **and** $\widehat{x}_{t+k|t}$ has not reached x_{samp} **do**
9: Select input $u_{t+k|t} \in \mathcal{U}$
10: Simulate $(\widehat{x}_{t+k+1|t}, P_{t+k+1|t})$ using (3.71)-(3.72)
11: Create intermediate nodes as appropriate
12: $k \leftarrow k + 1$
13: Compute $\Delta_t(\widehat{x}_{t+k|t}, P_{t+k|t})$ using (3.59) and $\Delta(\widehat{x}_{t+k|t}, P_{t+k|t})$ using (3.60)
14: **end while**
15: **for** each feasible node N **do**
16: Update cost estimates for N
17: Add N to $\mathcal{T}$
18: Try connecting N to $\mathcal{X}_{\text{goal}}$ (lines 5–14)
19: **if** connection to $\mathcal{X}_{\text{goal}}$ feasible **then**
20: Update upper-bound cost-to-go of N and ancestors
21: **end if**
22: **end for**
23: **end for**

3.3.3 Execution Loop

The execution loop for both offline and online CC-RRT, which is performed at time intervals of Δt, is given by Algorithm 5. During each cycle, the objective of this algorithm is to identify the lowest-cost path in the tree that is still probabilistically feasible, and use the remaining time to grow the tree.

If new observations are available for the vehicle's current state, these may be applied to update and repropagate the tree first (line 3). In particular, if full state information is available for the root at state x_t, then that node can be considered to be perfectly known, *i.e.*, $x_t \sim \mathcal{N}(\widehat{x}_t, 0)$, when propagating to child nodes. It is desirable to perform this update if it is available, as it will reduce the uncertainty at future timesteps.

For the duration of the timestep, the tree is repeatedly expanded using Algorithm 3 or Algorithm 4, depending on the type of CC-RRT being used (lines 4–6). Following

Algorithm 5 CC-RRT, Execution Loop

1: Initialize tree $\mathcal{T}$ with node $(\widehat{x}_0, P_{x_0})$ for $t = 0$
2: **while** $\widehat{x}_t \notin \mathcal{X}_{\text{goal}}$ **do**
3: Use observations, if any, to repropagate state distributions
4: **while** time remaining for this timestep **do**
5: Expand the tree by adding nodes (Algorithm 3 or Algorithm 4)
6: **end while**
7: Identify path $\{N_{\text{root}}, \ldots, N_{\text{target}}\}$ that minimizes (3.35)/(3.61)
8: **if** no paths exist **then**
9: Apply safety action and **goto** line 17
10: **end if**
11: Repropagate path state distributions using (3.71)-(3.72)
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**

this tree growth, the objective (3.35) (for offline CC-RRT) or (3.61) (for online CC-RRT) is used to identify the lowest-cost path in the tree (line 7). Once a path is chosen, a "lazy check" [105] is performed in which the path is repropagated from the current state distribution using the same model dynamics (3.71)-(3.72) (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 portion of the path that is no longer probabilistically feasible 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 (*e.g.*, apply maximum braking to come to a stop) is applied to attempt to keep the vehicle in a safe state (line 9).

3.4 Analysis

This section establishes that, under appropriate assumptions, both versions of the CC-RRT algorithm maintain the probabilistic completeness of the RRT algorithm for problem (3.B)/(3.C). The proof below is provided for online CC-RRT, as it is

the algorithm that will be developed further in subsequent work (Chapters 4-5); the derivation for offline CC-RRT follows very closely.

Assumption 3.4. *All of the following conditions are satisfied:*

1. *The set $\mathcal{U}$ consists of a finite number of inputs. Whenever inputs are simulated, they are chosen randomly from the set $\mathcal{U}$.*

2. *All tree vertices are separated by a distance of at least $\epsilon > 0$.*

3. *There exists a sequence of feasible inputs $u_0, u_1, \ldots, u_{t_f-1}$ such that the constraints of problem (2.C) are satisfied by the state distribution sequence $(\widehat{x}_0, P_{x_0}), (\widehat{x}_1, P_{x_1}), \ldots, (\widehat{x}_{t_f}, P_{x_{t_f}})$ and $\widehat{x}_{t_f} \in \mathcal{X}_{goal}$. All states in this sequence lie in the same bounded, open, and connected s_x-dimensional subset of $\mathbb{R}^{s_x}$.*

Assumption 3.4.3 ensures that at least one robustly feasible solution exists. Assumptions 3.4.1-3.4.2 may not necessarily apply directly to many practical applications; however, as long as the input selection used in the steering law effectively expands the frontier of the tree, feasible solutions are likely to be identified in practice.

Theorem 3.5 (Probabilistic Completeness of CC-RRT). *Under Assumption 3.4, online CC-RRT is probabilistically complete for problem (3.C).*

Proof. The probabilistic completeness of CC-RRT can be established using the same approach as the proof of probabilistic completeness for RRT [54]. Through the use of the discrete LTI dynamics, all timestep durations are constant, and all motions are locally constrained. Coupled with Assumption 3.4, all assumptions used to establish probabilistic completeness for RRT are also satisfied by CC-RRT.

In particular, the argument proceeds by induction. Suppose that the CC-RRT tree currently contains the state distribution $(\widehat{x}_k, P_{x_k})$ generated by applying the input sequence $u_0, u_1, \ldots, u_{k-1}$ from Assumption 3.4.3, for $k < t_f$. By Assumption 3.4.2, all tree vertices have a finite Voronoi region (whose volume is a function of ϵ), including $\widehat{x}_k$. As such, because all states have a non-zero likelihood of being sampled, $\widehat{x}_k$ has a non-zero likelihood of being selected as a nearest node for expansion.

From Assumption 3.4.1, input u_k has a non-zero likelihood of being applied in order to generate state distribution $(\widehat{x}_{k+1}, P_{x_{k+1}})$. As the number of samples approaches infinity, the likelihood of $(\widehat{x}_{k+1}, P_{x_{k+1}})$ thus being added as a descendant node of $(\widehat{x}_k, P_{x_k})$ approaches 1. Since $(\widehat{x}_0, P_{x_0})$ is initialized as the tree root, the proof by induction is complete. ■

Thought CC-RRT is probabilistically complete, it is *not* asymptotically optimal. In fact, it is straightforward to show that, like RRT, CC-RRT will not converge on minimum-cost solutions even if they are feasible [56]. This is addressed by the asymptotically optimal version of this algorithm, CC-RRT* (Section 5).

3.5 Nonlinear Dynamics

If the system dynamics are not linear, as in (2.1), then the probabilistic constraint satisfaction guarantees of Theorem 3.1 (Section 3.2.1) and Theorem 3.3 (Section 3.2.2) are no longer applicable. However, through appropriate linearization of nonlinear dynamics, the resulting bounds can still provide useful estimates on timestep and path safety, especially for small timesteps. This section presents how to operate CC-RRT using nonlinear dynamics, through linearization about a sequence of states and inputs that satisfy the disturbance-free nonlinear dynamics. This sequence of states become the conditional means of the trajectories simulated within the CC-RRT, such that the state distributions simulated by the algorithm remain dynamically feasible.

The covariances are computed based on the linearized dynamics at each state along a trajectory. This results in a predicted state distribution which is still Gaussian. Because of the dependency on the states and inputs, the covariances can no longer be computed offline, as in CC-RRT for linear dynamics. Rather, the covariances are now specific to each tree branch, representing the uncertainty accrued by sending the nonlinear dynamics along that trajectory.

Consider the nonlinear time-invariant (LTI) discrete-time system dynamics

$$x_{t+1} = f(x_t, u_t, w_t), \tag{3.73}$$

where x_t, u_t, and w_t are defined as in Section 3.1, and subject to the same constraints. In particular, the system is still subject to Gaussian process noise (3.1) and initial state error (3.2). The disturbance-free dynamics are defined as those in which the process noise is zero, *i.e.*, $w_t \equiv 0$:

$$\widehat{x}_{t+1} = f(\widehat{x}_t, u_t, 0). \tag{3.74}$$

Consider some state sequence $\{\widehat{x}_0, \widehat{x}_1, \ldots\}$ which is feasible for (3.74) under the corresponding input sequence $\{u_0, u_1, \ldots\}$. The objective is to construct a set of linear dynamics at each timestep which linearizes about this nominal plan, such that suitable covariances can be approximated. Define the error quantity $\widetilde{x}_t$ between the "true state" x_t and the nominal plan state $\widehat{x}_t$,

$$\widetilde{x}_t = x_t - \widehat{x}_t, \tag{3.75}$$

where $\widetilde{x}_0 = 0$. (Because the inputs are perfectly known, no error term is needed for them in this open-loop case.)

The linearized dynamics then take the form

$$\widetilde{x}_{t+1} = A_t \widetilde{x}_t + G_t w_t, \tag{3.76}$$

$$A_t \equiv \frac{\partial f}{\partial x_t}(\widehat{x}_t, u_t, 0), \tag{3.77}$$

$$G_t \equiv \frac{\partial f}{\partial w_t}(\widehat{x}_t, u_t, 0); \tag{3.78}$$

unlike (2.1) and (3.73), these dynamics are time-varying. The system (3.76) is subject to the same zero-mean process noise (3.1) as (3.73). However, in view of (3.75), the initial state error is zero-mean:

$$\widetilde{x}_0 \sim \mathcal{N}(0, P_{x_0}). \tag{3.79}$$

Because $\widetilde{x}_0 = 0$ and all uncertainty is zero-mean, it is clear that the state error

remains zero-mean at all timesteps, *i.e.*,

$$\widetilde{x}_t \sim \mathcal{N}(0, P_{x_t}). \tag{3.80}$$

The evolution of the covariance can be computed in a similar manner as in (3.13), yielding

$$P_{x_{t+1}} = A_t P_{x_t} A_t^T + G_t P_w G_t^T. \tag{3.81}$$

Thus, to apply CC-RRT with nonlinear dynamics, the model simulation dynamics (3.71)-(3.72) should be replaced with the linearized equivalent form,

$$\widehat{x}_{t+k+1|t} = f(\widehat{x}_{t+k|t}, u_{t+k|t}, 0), \tag{3.82}$$

$$P_{t+k+1|t} = A_{t+k|t} P_{t+k|t} A_{t+k|t}^T + G_{t+k|t} P_w G_{t+k|t}^T, \tag{3.83}$$

where the $(\cdot)_{t+k|t}$ notation is used as in (3.71)-(3.72). Because the conditional means still use the full nonlinear dynamics (3.73), paths generated in CC-RRT for nonlinear dynamics remain dynamically feasible. However, due to this linearization, the theoretical guarantees presented in Section 3.2 for the case of linear dynamics under Gaussian noise are no longer valid. Nonetheless, it is straightforward to show that the quality of the risk approximation improves as the timestep duration is decreased. The quality of the risk approximation also improves as the accuracy of the linearization of the nonlinear dynamics increases. Each of these results in a linearization which more accurately approximates the true, nonlinear dynamics.

A common application of such linearized models is to stabilize nonlinear dynamics around a nominal path, through the use of a feedback law on the error term $\widetilde{x}_t$. In this framework, given a nominal input $\widehat{u}_t$ and a state error $\widetilde{x}_t$, the actual input applied is $u_t = \widehat{u}_t + \widetilde{u}_t$, where

$$\widetilde{u}_t = -L_t \widetilde{x}_t \tag{3.84}$$

and L_t is a feedback gain matrix of suitable dimension. In this case, an alternate version of the linearization (3.76) can be constructed which linearizes about the input in addition to the state and disturbance, in order to incorporate feedback.

In this case, consider some state sequence $\{\widehat{x}_0, \widehat{x}_1, \ldots\}$ which is feasible for (3.74) under the corresponding input sequence $\{\widehat{u}_0, \widehat{u}_1, \ldots\}$. Since $\widetilde{u}_t = u_t - \widehat{u}_t$, the linearized dynamics in this case take the form

$$\begin{aligned} \widetilde{x}_{t+1} &= A_t \widetilde{x}_t + B_t \widetilde{u}_t + G_t w_t, & (3.85) \\ B_t &\equiv \frac{\partial f}{\partial u_t}(\widehat{x}_t, \widehat{u}_t, 0), & (3.86) \end{aligned}$$

where A_t and G_t are defined in (3.77) and (3.78), respectively, with $u_t = \widehat{u}_t$. Inserting the feedback law (3.84) into (3.85) yields

$$\begin{aligned} \widetilde{x}_{t+1} &= A_t \widetilde{x}_t + B_t(-L_t \widetilde{x}_t) + G_t w_t \\ &= (A_t - B_t L_t)\widetilde{x}_t + G_t w_t. & (3.87) \end{aligned}$$

Thus, when applying CC-RRT to these feedback-linearized dynamics, the model simulation dynamics (3.71)-(3.72) should be replaced with

$$\begin{aligned} \widehat{x}_{t+k+1|t} &= f(\widehat{x}_{t+k|t}, \widehat{u}_{t+k|t}, 0), & (3.88) \\ P_{t+k+1|t} &= (A_{t+k|t} - B_{t+k|t} L_{t+k|t}) P_{t+k|t} (A_{t+k|t} - B_{t+k|t} L_{t+k|t})^T & (3.89) \\ &\quad + G_{t+k|t} P_w G_{t+k|t}^T. \end{aligned}$$

In both cases, as new state information becomes available, the linearizations and resulting covariances should be updated (Algorithm 5) to propagate this information through the tree.

3.6 Output-model CC-RRT

Throughout this thesis, the focus is on developing motion planning approaches that can be robust to many qualitatively distinct forms of uncertainty, whether internal or external to the system, or based on current sensing or future predictability [3]. Much of the emphasis in this work is on process noise and environmental uncertainty sources, given the large body of literature already focusing (sometimes exclusively) on the internal sensing question. Regardless, the CC-RRT algorithm can be modified to handle the case where full state information is not available, and is instead provided through an output model consisting of noisy sensors. This section presents how to modify the CC-RRT framework to handle output filtering, utilizing a similar approach to that presented in Patil *et al.* [90].

Assume the dynamics remain in the linear form (2.1), but the full state x_t is not available. Instead, there is additional an LTI output model with sensing noise of the form

$$y_t = Cx_t + Hv_t, \tag{3.90}$$

$$v_t \sim \mathcal{N}(0, P_v), \tag{3.91}$$

where $y_t \in \mathbb{R}^{s_y}$ is the output vector and C, H are matrices of suitable dimension. The disturbance v_t is unknown at current and future timesteps, but belongs to the Gaussian zero-mean uncertainty (3.91).

A Kalman filter [108] can be constructed to maintain an optimal state estimate with mean $\overline{x}$ and covariance Q. Suppose the observation at timestep $t+1$ is y_{t+1}. The Kalman update from timestep t (with mean $\overline{x}_t$ and covariance Q_t) to timestep

$t+1$ (with mean $\overline{x}_{t+1}$ and covariance Q_{t+1}) then takes the form

$$\begin{aligned}
\overline{x}_{t+1} &= \overline{x}^+_{t+1} + K_{t+1}(y_{t+1} - C\overline{x}^+_{t+1}), && (3.92)\\
Q_{t+1} &= (I - K_{t+1}C)Q^+_{t+1}, && (3.93)\\
\overline{x}^+_{t+1} &= A\overline{x}_t + Bu_t, && (3.94)\\
Q^+_{t+1} &= AQ_tA^T + GP_wG^T, && (3.95)
\end{aligned}$$

where K_t is the optimal Kalman gain

$$\begin{aligned}
K_t &= Q^+_t C^T (CQ^+_t C^T + HP_vH^T)^{-1} && (3.96)\\
&= (AQ_{t-1}A^T + GP_wG^T)C^T \left[C(AQ_{t-1}A^T + GP_wG^T)C^T + HP_vH^T\right]^{-1}.
\end{aligned}$$

The propagation steps (3.94)-(3.95) and update steps (3.92)-(3.93) can be combined into the single steps

$$\begin{aligned}
\overline{x}_{t+1} &= (I - K_{t+1}C)(A\overline{x}_t + Bu_t) + K_{t+1}y_{t+1}, && (3.97)\\
Q_{t+1} &= (I - K_{t+1}C)(AQ_tA^T + GP_wG^T). && (3.98)
\end{aligned}$$

This filter can be incorporated into the predicted planning of CC-RRT by simultaneously maintaining dynamics for both the true state x_t and state estimate $\overline{x}_t$ along each trajectory. For this predictive planning, apply the true output model (3.90) to the mean update equation (3.97), yielding

$$\overline{x}_{t+1} = (I - K_{t+1}C)(A\overline{x}_t + Bu_t) + K_{t+1}(Cx_{t+1} + Hv_{t+1}). \quad (3.99)$$

Plugging in the true dynamics (2.1) yields

$$\begin{aligned}
\overline{x}_{t+1} &= A\overline{x}_t + Bu_t - K_{t+1}CA\overline{x}_t - K_{t+1}CBu_t && (3.100)\\
&\quad + K_{t+1}C(Ax_t + Bu_t + Gw_t) + K_{t+1}Hv_{t+1}\\
&= (I - K_{t+1}C)A\overline{x}_t + K_{t+1}CAx_t + Bu_t + K_{t+1}CGw_t + K_{t+1}Hv_{t+1}.
\end{aligned}$$

From (2.1) and (3.100), combined dynamics can be formed [90]:

$$\underbrace{\begin{bmatrix} x_{t+1} \\ \overline{x}_{t+1} \end{bmatrix}}_{X_{t+1}} = \underbrace{\begin{bmatrix} A & 0 \\ K_{t+1}CA & (I - K_{t+1}C)A \end{bmatrix}}_{\overline{A}_t} \underbrace{\begin{bmatrix} x_t \\ \overline{x}_t \end{bmatrix}}_{X_t} + \underbrace{\begin{bmatrix} B \\ B \end{bmatrix}}_{\overline{B}_t} u_t \tag{3.101}$$
$$+ \underbrace{\begin{bmatrix} G & 0 \\ K_{t+1}CG & K_{t+1}H \end{bmatrix}}_{\overline{G}_t} \underbrace{\begin{bmatrix} w_t \\ v_{t+1} \end{bmatrix}}_{W_t},$$

which can be written in the more compact form

$$X_{t+1} = \overline{A}_t X_t + \overline{B}_t u_t + \overline{G}_t W_t. \tag{3.102}$$

These combined dynamics are subject to the combined uncertainty

$$W_t \sim \mathcal{N}\left(0, \underbrace{\begin{bmatrix} P_w & 0 \\ 0 & P_v \end{bmatrix}}_{\Pi_w}\right), \tag{3.103}$$

$$X_0 \sim \mathcal{N}\left(\underbrace{\begin{bmatrix} \overline{x}_0 \\ \overline{x}_0 \end{bmatrix}}_{\hat{X}_0}, \underbrace{\begin{bmatrix} P_{x_0} & 0 \\ 0 & 0 \end{bmatrix}}_{\Pi_0}\right); \tag{3.104}$$

here the initial covariance of the state estimate $\overline{x}_t$ is assumed zero.

Remark 3.6 (covariance factoring). To avoid the intensive computation needed for matrix inversion due to (3.96), a factored form of the covariance can be instead applied [74]. In particular, define $Q_t = E_t F_t^{-1}$, where $E_0 = Q_0$ and $F_0 = I$. The resulting efficient covariance update of Ref. [74], written in the framework of this

chapter, then takes the form

$$\begin{bmatrix} E_{t+1} \\ F_{t+1} \end{bmatrix} = \begin{bmatrix} A & GP_wG^TA^{-T} \\ C^T(HP_vH^T)^{-1}CA & A^{-T} + C^T(HP_vH^T)^{-1}CGP_wG^TA^{-T} \end{bmatrix} \begin{bmatrix} E_t \\ F_t \end{bmatrix}.$$

where the notation $(\cdot)^{-T}$ denotes the inverse transpose.

3.6.1 Open-Loop Case

If inputs are applied to the dynamics (2.1), (3.90) in open-loop, the filter (3.96)-(3.98) is only needed in the execution phase, as inputs are applied to the actual dynamics, observations received, and the tree updated (Algorithm 5, line 3). The same simulation model dynamics (3.71)-(3.72) can be applied in this case.

Similarly, if the dynamics are nonlinear (3.73), the linearized model dynamics (3.82)-(3.83) can still be applied. However, if the sensor model (3.90) is nonlinear, then predicted observations $\widehat{y}_t$ should also be computed and maintained along each CC-RRT trajectory. These predictions are needed along with the full linearized dynamics to apply the filter at each execution timestep, as discussed below.

Suppose the sensor model takes the nonlinear form

$$y_t = g(x_t, v_t), \tag{3.105}$$

where it is still subject to the same zero-mean Gaussian sensing noise (3.91). For a given predicted state $\widehat{x}_t$, define the predicted observation as the most likely observation,

$$\widehat{y}_t = g(\widehat{x}_t, 0), \tag{3.106}$$

and define the observation error $\widetilde{y}_t = y_t - \widehat{y}_t$. The linearized output dynamics then

take the form

$$\widetilde{y}_t = C_t \widetilde{x}_t + H_t v_t, \tag{3.107}$$

$$C_t \equiv \frac{\partial g}{\partial y_t}(\widehat{y}_t, 0), \tag{3.108}$$

$$H_t \equiv \frac{\partial g}{\partial v_t}(\widehat{y}_t, 0). \tag{3.109}$$

When applying the filter during execution, suppose that observation y_1 is received after applying input u_0. The filter then updates its mean and covariance estimates on the state error $\widetilde{x}_0$ via (3.97)-(3.98), yielding

$$\widehat{\widetilde{x}}_1 = (I - K_1 C_1) A_0 \widehat{\widetilde{x}}_0 + K_1 \widetilde{y}_1, \tag{3.110}$$

$$Q_1 = (I - K_1 C_1)(A_0 P_0 A_0^T + G_0 P_w G_0)^T. \tag{3.111}$$

When planning from the next timestep, $x_1 \sim \mathcal{N}(\widehat{x}_1 + \widehat{\widetilde{x}}_1, P_1)$.

3.6.2 Closed-Loop Case

If inputs are instead selected via a feedback law on the state estimate $\overline{x}_t$,

$$u_t = r_t - L_t \overline{x}_t, \tag{3.112}$$

then the combined dynamics (3.102) take the alternate form

$$X_{t+1} = \overline{\overline{A}}_t X_t + \overline{B}_t r_t + \overline{G}_t W_t, \tag{3.113}$$

$$\overline{\overline{A}}_t = \begin{bmatrix} A & -BL_t \\ K_{t+1}CA & (I - K_{t+1}C)A - BL_t \end{bmatrix}, \tag{3.114}$$

where the only changes are that $\overline{A}_t$ is replaced with $\overline{\overline{A}}_t$ and u_t with r_t. As a result of this change, however, x_t and $\overline{x}_t$ are now coupled, and must both be propagated along every predicted CC-RRT trajectory. Emulating the form of (3.12)-(3.13), at future

timesteps, the combined state X_t is Gaussian with $X_t \sim \mathcal{N}(\widehat{X}_t, \Pi_t)$ and

$$\widehat{X}_{t+1} = \overline{\overline{A}}_t \widehat{X}_t + \overline{B}_t r_t, \tag{3.115}$$

$$\Pi_{t+1} = \overline{\overline{A}}_t \Pi_t \overline{\overline{A}}_t^T + \overline{G}_t \Pi_w \overline{G}_t^T. \tag{3.116}$$

The true state mean and covariance are then extracted via [90]

$$\widehat{x}_{t+1} = \begin{bmatrix} 1^T & 0^T \end{bmatrix} \widehat{X}_{t+1} \tag{3.117}$$

$$= \begin{bmatrix} 1^T & 0^T \end{bmatrix} (\overline{\overline{A}}_t \widehat{X}_t + \overline{B}_t r_t), \tag{3.118}$$

$$P_{x_{t+1}} = \begin{bmatrix} I & 0 \end{bmatrix} \Pi_{t+1} \begin{bmatrix} I \\ 0 \end{bmatrix} \tag{3.119}$$

$$= \begin{bmatrix} I & 0 \end{bmatrix} (\overline{\overline{A}}_t \Pi_t \overline{\overline{A}}_t^T + \overline{G}_t \Pi_w \overline{G}_t^T) \begin{bmatrix} I \\ 0 \end{bmatrix} \tag{3.120}$$

The model form equivalent of (3.71)-(3.72) used for each CC-RRT trajectory takes a similar form:

$$\widehat{X}_{t+k+1|t} = \overline{\overline{A}}_t \widehat{X}_{t+k|t} + \overline{B}_t r_{t+k|t}, \tag{3.121}$$

$$\Pi_{t+k+1|t} = \overline{\overline{A}}_t \Pi_{t+k|t} \overline{\overline{A}}_t^T + \overline{G}_t \Pi_w \overline{G}_t^T, \tag{3.122}$$

$$\widehat{x}_{t+k+1|t} = \begin{bmatrix} 1^T & 0^T \end{bmatrix} \widehat{X}_{t+k+1|t}, \tag{3.123}$$

$$P_{t+k+1|t} = \begin{bmatrix} I & 0 \end{bmatrix} \Pi_{t+k+1|t} \begin{bmatrix} I \\ 0 \end{bmatrix}. \tag{3.124}$$

If the dynamics take the nonlinear form (3.73), (3.90), the formulation proceeds similarly, but instead using the linearized dynamics (3.76), (3.107).

3.7 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 δ_s 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, and performs well even if the dynamics are nonlinear. Additional, more comprehensive simulation results contrasting this algorithm with CC-RRT can be found in Chapter 5.

3.7.1 Simple Scenario

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

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

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 $\mathcal{X}_t$ consist of speed bounds ($|v_t^x| < 0.5$ and $|v_t^y| < 0.5$) and obstacle avoidance constraints at each timestep. It is assumed without loss of generality that the vehicle is a point mass (if not, the vehicle size can be applied by tightening state constraints accordingly). The environment contains four obstacles in fixed, known locations (Figure 3-2).

The environment boundaries of the room are not treated as chance constraints, *i.e.*, it is sufficient for the distribution mean to remain within the room bounds. In this case (3.37) is not used within offline risk allocation, while $\Delta_{0t} \equiv 0$ in (3.59) within online risk evaluation. The use of environmental and path-wise (via δ_p) chance constraints is explored in more detail in Chapter 5.

The initial state covariance and disturbance covariance are specified as

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

respectively. The input u is selected at each timestep 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 (Algorithms 3/4) and execution (Algorithm 5), $A+BK$ is used in place of A in (3.71)-(3.72) [106]. The cost functions (3.35), (3.61) represent the duration of the path to goal, *i.e.*, $\phi_f \equiv 0$ and $\phi = dt$. Simulations were performed using an implementation of Algorithms 3–5 in Java, run on an Intel 2.53 GHz quad-core desktop with 4GB of RAM, with $\Delta t = 1$s.

Each simulation uses one of three algorithms: nominal RRT, offline CC-RRT, or online CC-RRT. Five cases are considered, with 10 trials performed for each:

- Nominal RRT
- Offline CC-RRT with $\delta_s = 0.5$
- Online CC-RRT with $\delta_s = 0.5$
- Online CC-RRT with $\delta_s = 0.9$
- Online CC-RRT with $\delta_s = 0.99$

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.

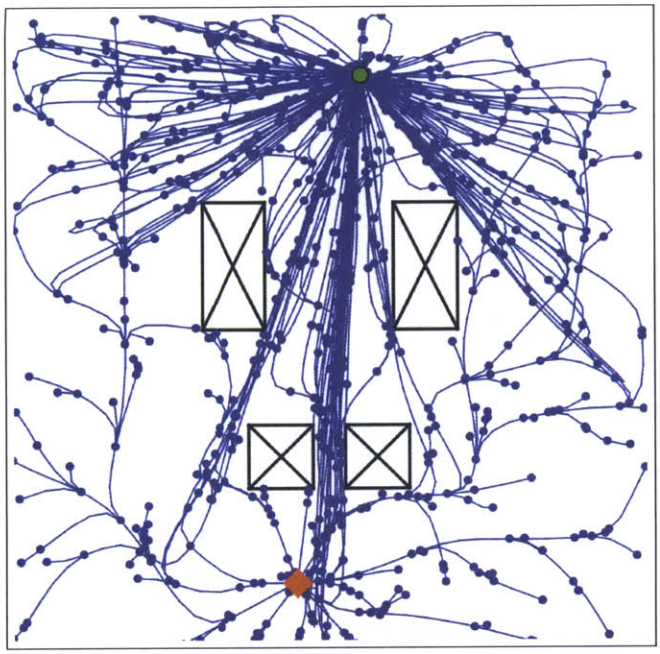
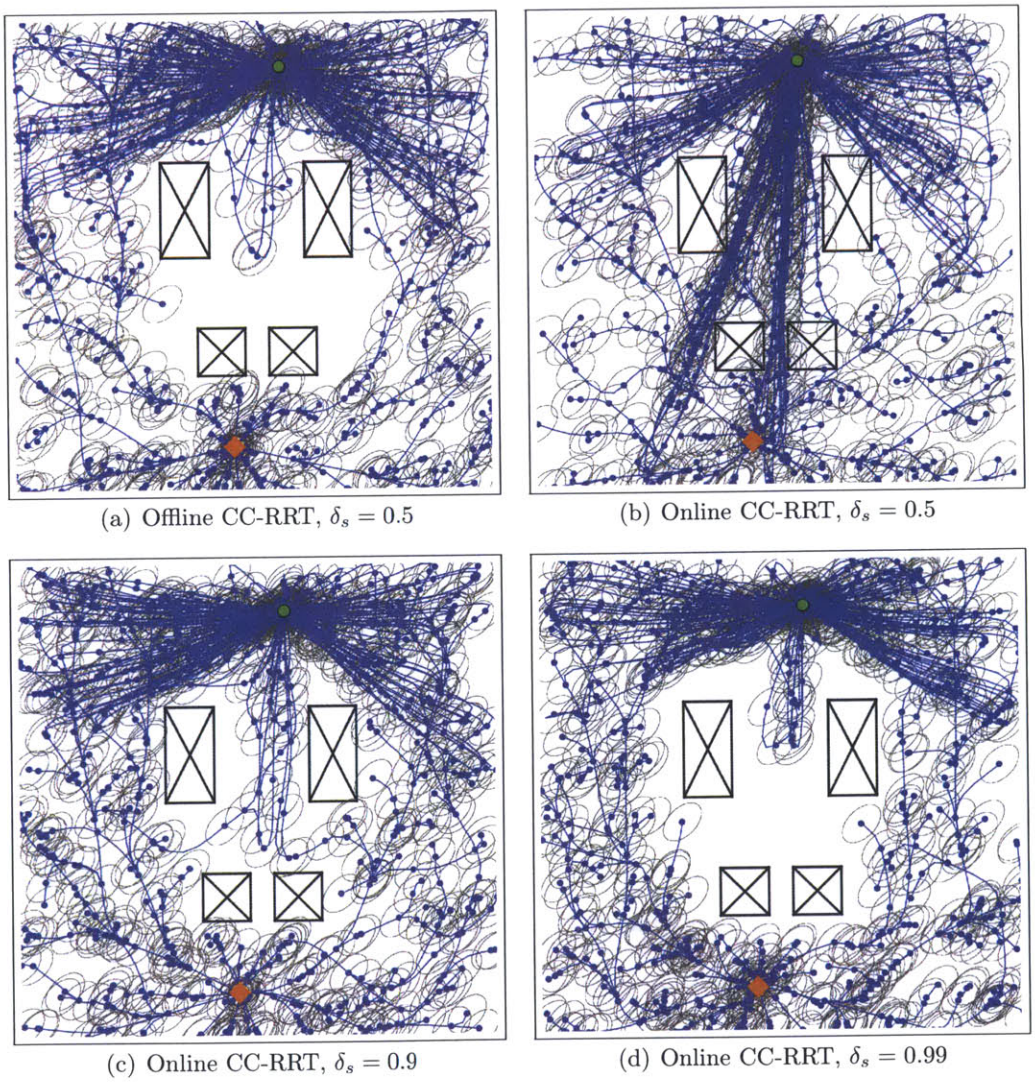


Figure 3-2: Representative tree for RRT, simple scenario

Figures 3-2 and 3-3 show a sample tree for each case after the 20 seconds of computation; in the latter figure, the distribution mean (blue circle) and 2-σ uncertainty ellipse (gray) are displayed for each node. In these and subsequent 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). Due to the system's closed-loop nature, the state distributions quickly converge to a steady-state value [106]. As a result, most uncertainty ellipses appear to be identical in the figures. However, there is indeed an evolution in the uncertainty ellipse size, starting from the initial state error at the tree root.

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 3-2).

For online CC-RRT with $\delta_s = 0.5$ (Figure 3-3(b)), the algorithm identifies every homotopically distinct path between obstacles to the goal (moving from bottom to top). For the case of a Gaussian distribution subject to a single linear inequality



(a) Offline CC-RRT, $\delta_s = 0.5$

(b) Online CC-RRT, $\delta_s = 0.5$

(c) Online CC-RRT, $\delta_s = 0.9$

(d) Online CC-RRT, $\delta_s = 0.99$

Figure 3-3: Representative trees for CC-RRT, simple scenario

constraint, a constraint of $\delta_s = 0.5$ is active if the distribution mean falls exactly on the constraint boundary. Thus, in this case, all distribution means still fall outside the obstacle boundaries (and will continue to do so for any $\delta_s \in [0.5, 1]$ - see Lemma 5.2 in Section 5.2), but may come extremely close. In that sense, online CC-RRT $\delta_s = 0.5$ closely emulates the behavior of nominal RRT.

As δ_s increases for online CC-RRT from 0.5 to 0.9 (Figure 3-3(c)), the probabilistic constraints tighten and restrict the feasible configuration space of the vehicle. Whereas the $\delta_s = 0.5$ case identifies many paths to goal between the obstacles, the $\delta_s = 0.9$ case only identifies two, and cannot traverse the narrowest gap between the bottommost obstacles. Additionally, the uncertainty ellipses in Figure 3-3(b) significantly intersect the obstacles, whereas any intersection is minimal in Figure 3-3(c). When $\delta_s = 0.99$ (Figure 3-3(d)), all trajectories take the wider corridors around the obstacles to reach the goal, at the expense of a longer path duration.

While both Figures 3-3(a) and 3-3(b) use $\delta_s = 0.5$ and provide the same guarantee of probabilistic feasibility, the offline CC-RRT algorithm (Figure 3-3(a)) yields a much more conservative result, more closely resembling online CC-RRT with $\delta_s = 0.99$. In other words, the solutions generated by the offline CC-RRT algorithm are much more conservative then those generated by the online CC-RRT algorithm for the same probabilistic feasibility guarantees.

Table 3.1 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 unaware of the risk posed by uncertainty in the formulation, safely reaching the goal only in a single trial. For the CC-RRT cases, as δ_s 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 $\delta_s = 0.99$, online CC-RRT safely reaches the goal in all ten trials. As expected from the figures, the performance of offline CC-RRT with $\delta_s = 0.5$ is comparable to online CC-RRT with higher values of δ_s.

Finally, while there is a modest runtime increase for offline CC-RRT and a slightly

Table 3.1: Properties of solutions for RRT and CC-RRT, simple scenario

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

Table 3.2: Properties of solutions for RRT and CC-RRT, cluttered scenario

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

[a] Cumulative time spent in Algorithm 3/4 divided by the number of nodes generated.
[b] Duration of initial path to goal, if one exists.
[c] Number of trials where system executed a path to goal without colliding with any obstacles.

larger runtime increase for online CC-RRT, both are competitive with the average runtime for nominal RRT. As Table 3.1 indicates, the time per node relative to RRT increases only about 10% for offline CC-RRT and 40-50% for online CC-RRT, both of which are suitable for real-time operation.

3.7.2 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 the previous scenario are maintained except for the environment itself, which is replaced with a more cluttered environment (Figure 3-4). 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 the previous scenario, for the same five cases. Figures 3-4 and 3-5 show a sample tree for each case after the 20 seconds of computation; very similar conclusions can be drawn as from the simple scenario.

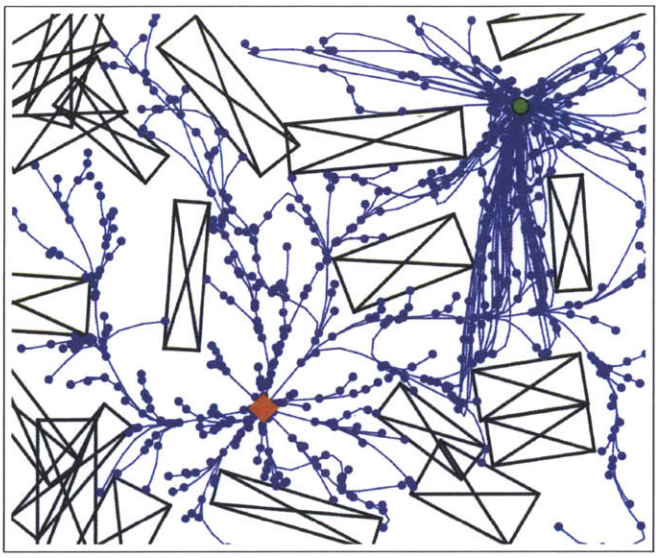
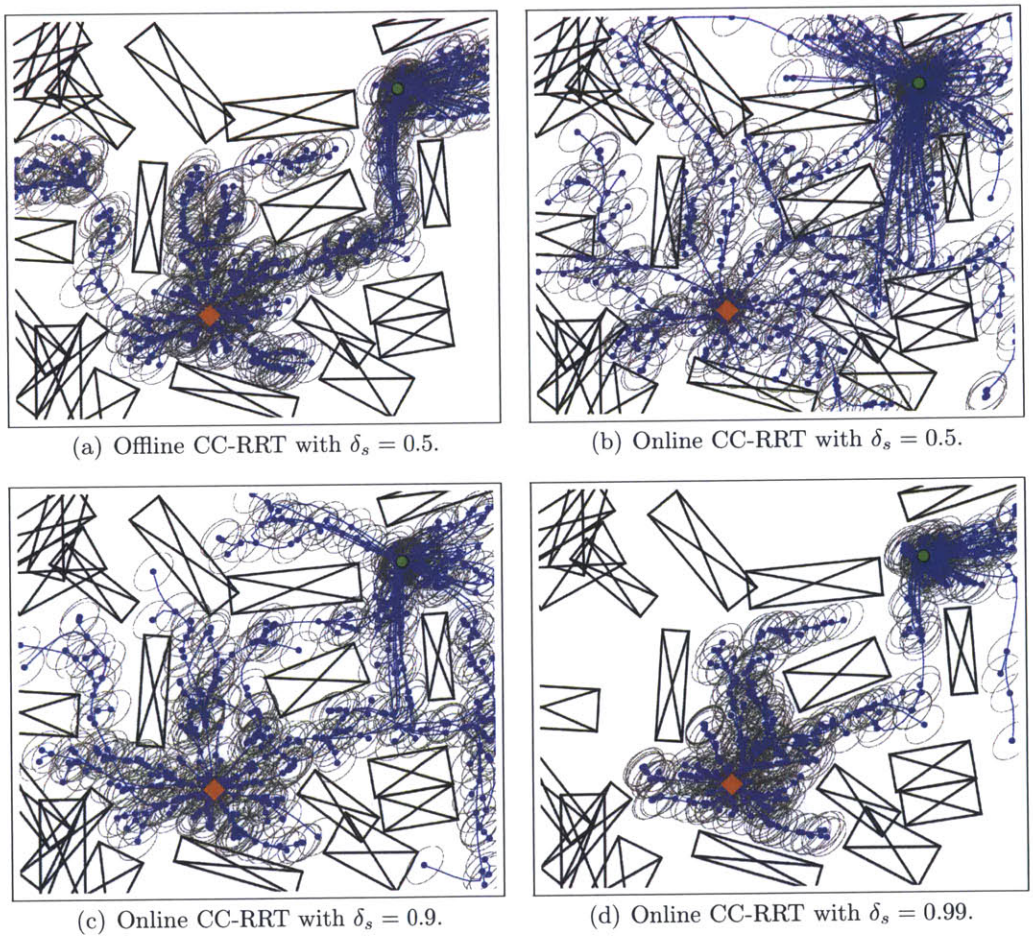


Figure 3-4: Representative tree for RRT, cluttered environment

Of note is the uppermost path to goal found by online CC-RRT with $\delta_s = 0.5$ (Figure 3-5(b)), passing along the very top of the figure. While the probability of collision at each timestep must not exceed 0.5, in practice such a path would be extremely difficult to execute safely.

Table 3.2 presents the averaged results over the 10 trials for each case. Even though the number of obstacles has increased by a factor of 5, the average runtime per node has only increased by a factor of 3 to 4 in each case, with nominal RRT and offline CC-RRT scaling slightly better than the online CC-RRT cases. Nonetheless, this data provides empirical evidence that the CC-RRT algorithms leverage the benefits of sampling-based algorithms to scale favorably with the problem/environment complexity, without requiring significant additional computation.

Otherwise, the results are similar to the simple scenario (Table 3.1), with online CC-RRT again achieving feasibility across all ten trials for $\delta_s = 0.99$. On the other hand, online CC-RRT performs poorly in this scenario for $\delta_s = 0.5$, with only one trial safely reaching the goal.



(a) Offline CC-RRT with $\delta_s = 0.5$.

(b) Online CC-RRT with $\delta_s = 0.5$.

(c) Online CC-RRT with $\delta_s = 0.9$.

(d) Online CC-RRT with $\delta_s = 0.99$.

Figure 3-5: Representative trees for CC-RRT, cluttered scenario

3.7.3 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 Section 3.7.1 (Figure 3-2). 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 upper-left obstacle placement distribution has covariance P_a, while all other obstacle placement distributions have covariance P_b, where

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

The online CC-RRT algorithm is run on this scenario with $\delta_s = 0.99$; 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 timestep does not exceed 1%. Figure 3-6 shows a representative tree generated by this algorithm. In this figure, the 2-σ uncertainty ellipse is shown both for every obstacle and for every node; it is clear that the residual state error near the start and the placement of the upper-left obstacle dominate the process noise within the uncertainty environment. 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 near it to become probabilistically infeasible.

3.7.4 Nonlinear Dynamics Scenario

The following results demonstrate the validity of linearizing dynamics within the CC-RRT framework to approximate probabilistic constraint satisfaction bounds for nonlinear dynamics. Consider the operation of a skid-steered vehicle in a 2D envi-

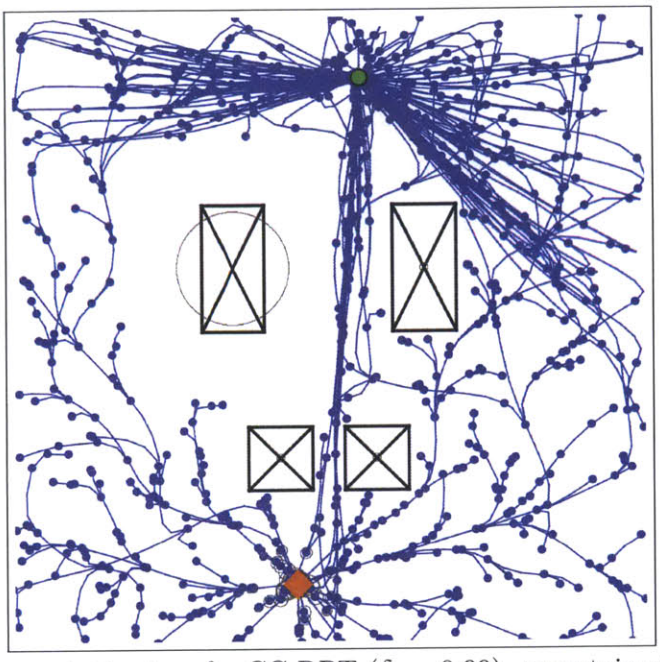
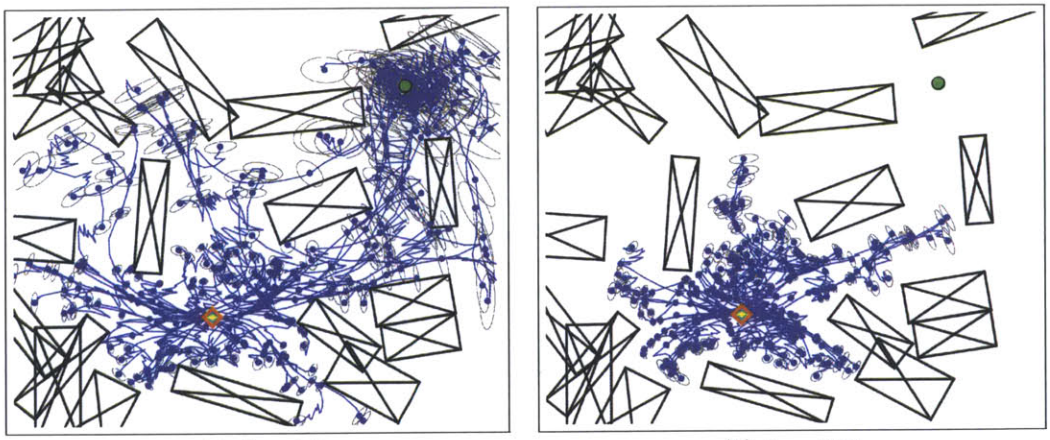


Figure 3-6: Representative tree for CC-RRT ($\delta_s = 0.99$), uncertain-obstacle scenario

ronment, with nonlinear dynamics

$$\begin{aligned}
x_{t+1} &= x_t + dt(1/2)(v_t^L + v_t^R)\cos\theta_t, \\
y_{t+1} &= y_t + dt(1/2)(v_t^L + v_t^R)\sin\theta_t, \\
\theta_{t+1} &= \theta_t + dt(v_t^R - v_t^L), \\
v_t^L &= \text{sat}\left(\bar{v}_t + 0.5\Delta v_t + w_t^L, -0.5, 0.5\right), \\
v_t^R &= \text{sat}\left(\bar{v}_t - 0.5\Delta v_t + w_t^R, -0.5, 0.5\right),
\end{aligned}$$

where $dt = 0.02$ s, (x, y) is the vehicle position, θ is the heading, v^L and v^R are the left and right wheel speeds, respectively, $\text{sat}(a, b, c)$ saturates a between b and c, and (w^L, w^R) is a small process noise. The system inputs are specified in terms of a mean velocity $\bar{v}$ and differential velocity Δv. A variation of the pure pursuit controller [109] is applied, assuming forward direction only. Paths are expanded by randomly sampling many inputs, then selecting those inputs that guide the vehicle closest to the sample (leading to much of the observed "zig-zag" behavior in Figure 3-7).



(a) $\delta_s = 0.5$ (b) $\delta_s = 0.99$

Figure 3-7: Representative trees for CC-RRT, nonlinear dynamics, cluttered scenario

Figure 3-7 demonstrates the use of Algorithm 4 using the linearized, open-loop dynamics (3.88)-(3.89); trees are shown after 20 seconds of tree growth, including distribution means and 1-σ covariance uncertainty ellipses. Unlike previous results, the state distribution uncertainties on these nodes tend to vary with the tree branch, and in particular tend to align in a direction perpendicular to the direction of motion. This reflects the linearization used to estimate the covariances at each tree node, based on an uncertainty that tends to affect lateral motion, rather than longitudinal motion. For small values of δ_s, a feasible path is found to the goal (Figure 3-7(a)); if δ_s is increased high enough, no such path may exist (Figure 3-7(b)).

3.8 Conclusions

This chapter 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 both internal and external uncertainty, via the use of chance constraints. In addition to incorporating a sampling-based framework, this approach builds upon previous chance-constrained formulations by including probabilistic robustness to both process noise and uncertain, possibly dynamic obstacles, subject to time-step-wise and path-wise risk bounds. 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 timestep, providing the user with a metric to directly control the level of conservatism in the planning approach. Furthermore, as demonstrated through the simulation results, the approach is scalable in the number of obstacles, allowing for efficient computation of safe paths even in heavily cluttered environments.

The CC-RRT algorithm is effective in quickly generating robustly feasible solutions through complex environments, making it suitable for many autonomous navigation applications (Chapter 4). However, due to its lack of optimality, the paths resulting from the algorithm may be non-smooth and inconsistent in some cases, even if a large number of tree nodes are generated. This can be addressed by incorporating the use of RRT* [56, 57] to provide guarantees on asymptotic optimality, with the trade-off of requiring use of a steering law to connect states. Chapter 5 presents CC-RRT*, which integrates these optimality guarantees while maintaining the ability to quickly find safe paths under uncertainty.

Chapter 4

Autonomous Navigation with Chance-Constrained RRT

This section considers several extensions to the chance-constrained rapidly exploring random tree (CC-RRT) framework which address some of the challenges faced by autonomous vehicles navigating in dynamic and/or uncertain environments. A unifying theme in this chapter is the demonstrated ability of CC-RRT to incorporate complex, learned models of uncertainty when simulating and executing in its decision-making to select safe, risk-aware trajectories.

First, CC-RRT is extended in Section 4.1 to consider dynamic obstacles with uncertain intentions. The CC-RRT algorithm developed in Chapter 3 admits the inclusion of dynamic obstacles while maintaining probabilistic feasibility guarantees. Here, dynamic obstacles are considered which may have uncertainty in both their behavior, and in their future trajectory given a possible behavior. This allows consideration of obstacles whose placement is better represented as a time-parameterized Gaussian mixture model, rather than one single time-parameterized Gaussian distribution.

One possible application of this extension is the robust avoidance of vehicles and pedestrians, as often encountered during navigation in urban environments. Section 4.2 demonstrates how CC-RRT has been tightly integrated with two dynamic obstacle prediction algorithms, RR-GP (joint work with Georges Aoude [95]) and DPGP (joint work with Sarah Ferguson [96]), to enable an autonomous vehicle to safely avoid

moving threats.

In Section 4.3, an alternate way is presented to approximate the risk of constraint violation, through the use of simulated particles to represent the uncertainty environment. This framework, leveraging ideas from particle filtering, allows for the consideration of non-Gaussian uncertainty models [97]. Unlike previously-developed approaches [71], the proposed particle CC-RRT algorithm can approximate the risk of both time-step-wise and path-wise constraint violation for multiple forms of uncertainty, while maintaining a separate tree node for each possible action sequence. However, such an approach requires the simulation of a potentially large set of particles at each node, which can be computationally intensive.

Alternately, even if a simple analytic model of an uncertainty environment is available, it may be desirable to directly sample that analytic model for approximating the risk of constraint violation. One example of this is in the application of parafoil terminal guidance, in which the constraint being considered is an arbitrary 3D surface not easily represented with polyhedral constraints. The parafoil terminal guidance problem poses several additional challenges, including underactuated dynamics and large/variable wind disturbances. Section 4.4 explores an alternate formulation of CC-RRT, analytic CC-RRT, designed to address these specific domain challenges (joint work with Ian Sugel [98]). Simulation results have demonstrated that this approach can significantly improve landing accuracy relative to state-of-the-art approaches in complex terrain environments [98].

In this and subsequent chapters, online CC-RRT is used exclusively instead of offline CC-RRT. As such, online CC-RRT will be referred simply as CC-RRT in the material that follows.

4.1 Dynamic Obstacles with Uncertain Intentions

This section extends the CC-RRT framework to consider dynamic obstacles with uncertain motion patterns. It is shown that, with appropriate modification, the CC-RRT algorithm can still identify probabilistically feasible paths subject to such

constraints [95].

In Section 3.1, it is assumed that each obstacle $\mathcal{X}_{jt}\ \forall\ j \in \mathbb{Z}_{1,n_o}$, is a convex polytope whose shape and orientation is known, but whose placement is uncertain. This is represented as (3.3)-(3.4), reprinted here as

$$\begin{aligned} \mathcal{X}_{jt} &= \mathcal{X}_j^0 + c_{jt}, \quad \forall\ j \in \mathbb{Z}_{1,n_o}, \\ c_{jt} &\sim \mathcal{N}(\widehat{c}_{jt}, P_{c_{jt}}), \quad \forall\ j \in \mathbb{Z}_{1,n_o}. \end{aligned}$$

In this model, for the jth obstacle, $\mathcal{X}_j^0 \subset \mathbb{R}^{n_x}$ is a convex polytope of known, fixed shape, while $c_{jt} \in \mathbb{R}^{n_x}$ represents a translation of that obstacle subject to a Gaussian uncertainty, with mean $\widehat{c}_{jt}$ and covariance $P_{c_{jt}}$. The dependency on timestep t in the mean and covariance reflects that the nature of the uncertainty may evolve from one timestep to the next, as is the case for dynamic obstacles.

In this section, a dynamic obstacle with uncertain intentions is assumed to follow one of M possible behaviors, indexed via $k \in \mathbb{Z}_{1,M}$. The uncertainty model for the dynamic obstacle with uncertain intentions is based on using observations to assess the obstacle's intent and trajectory, and consists of two components:

- A likelihood $\delta^{(k)}$ that the obstacle is following the kth behavior, where $\delta^{(k)} > 0\ \forall\ k \in \mathbb{Z}_{1,M}$ and $\sum_{k=1}^{M} \delta^{(k)} = 1$;
- A time-parameterized Gaussian uncertainty distribution for the obstacle at timestep t if it is following the kth behavior: $c_t^{(k)} \sim \mathcal{N}(\widehat{c}_t^{(k)}, P_{c_t}^{(k)})$.

In this context, the original uncertainty model (3.3)-(3.4) represents a dynamic obstacle with $M = 1$ and $\delta^{(1)} = 1$.

Now, suppose the jth obstacle is a dynamic obstacle with uncertain intentions. The uncertainty model is represented as a Gaussian mixture model of the time-parameterized Gaussian uncertainty distributions, where the behavior likelihoods provide the mixand weights:

$$c_{jt} \sim \sum_{k=1}^{M} \delta_j^{(k)} \mathcal{N}(\widehat{c}_{jt}^{(k)}, P_{c_{jt}}^{(k)}). \tag{4.1}$$

In problem (3.C), the state constraints are evaluated via the robustness conditions (3.55)-(3.60) and constraints (3.63)-(3.64), reprinted here as

$$\begin{aligned}
\Delta_{ijt}(\widehat{x}_t, P_{x_t}) &= \frac{1}{2}\left(1 - \text{erf}\left[\frac{a_{ij}^T(\widehat{x}_t - c_{ijt})}{\sqrt{2a_{ij}^T(P_{x_t} + P_{c_{jt}})a_{ij}}}\right]\right), \\
\Delta_{i0t}(\widehat{x}_t, P_{x_t}) &= \frac{1}{2}\left(1 - \text{erf}\left[\frac{a_{i0}^T(c_{i0} - \widehat{x}_t)}{\sqrt{2a_{i0}^T(P_{x_t})a_{i0}}}\right]\right), \\
\Delta_{jt}(\widehat{x}_t, P_{x_t}) &= \min_{i=1,\ldots,n_j} \Delta_{ijt}(\widehat{x}_t, P_{x_t}), \\
\Delta_{0t}(\widehat{x}_t, P_{x_t}) &= \sum_{i=1}^{n_0} \Delta_{i0t}(\widehat{x}_t, P_{x_t}), \\
\Delta_t(\widehat{x}_t, P_{x_t}) &= \Delta_{0t}(\widehat{x}_t, P_{x_t}) + \sum_{j=1}^{n_o} \Delta_{jt}(\widehat{x}_t, P_{x_t}), \\
\Delta(\widehat{x}_t, P_{x_t}) &= \sum_{t=0}^{t_f} \Delta_t(\widehat{x}_t, P_{x_t}), \\
\Delta_t(\widehat{x}_t, P_{x_t}) &\leq 1 - \delta_s, \quad \forall\, t, \\
\Delta(\widehat{x}_t, P_{x_t}) &\leq 1 - \delta_p.
\end{aligned}$$

At each timestep, the probability of collision with dynamic obstacle j can be written as a weighted sum of the probabilities of collision for the dynamic obstacle j under each behavior. With this modification, all existing probabilistic guarantees (Chapter 3) are maintained by treating each behavior's state distribution as a separate obstacle with the resulting risk scaled by $\delta_j^{(k)}$ [95]. Represent the risk bound (3.55) on the ith constraint of the jth obstacle following the kth behavior at timestep t as

$$\Delta_{ijt}^{(k)}(\widehat{x}_t, P_{x_t}) = \frac{1}{2}\left(1 - \text{erf}\left[\frac{a_{ij}^T(\widehat{x}_t - c_{ijt}^{(k)})}{\sqrt{2a_{ij}^T(P_{x_t} + P_{c_{jt}}^{(k)})a_{ij}}}\right]\right), \tag{4.2}$$

where $c_{ijt}^{(k)} \in C_{ijt}^{(k)}$ is a point nominally on this constraint if the kth behavior is being

followed. Then

$$
\begin{aligned}
P(\text{collision}) &\leq \sum_{j=1}^{n_o} P(\text{collision with obstacle } j) && (4.3)\\
&= \sum_{j=1}^{n_o}\sum_{k=1}^{M} \delta_j^{(k)} P(\text{collision with obstacle } j, \text{behavior } k) && (4.4)\\
&\leq \sum_{j=1}^{n_o}\sum_{k=1}^{M} \delta_j^{(k)} \min_{i=1,\ldots,n_j} P(a_{ij}^T X_t < a_{ij}^T C_{ijt}^{(k)}) && (4.5)\\
&= \sum_{j=1}^{n_o}\sum_{k=1}^{M} \delta_j^{(k)} \min_{i=1,\ldots,n_j} \Delta_{ijt}^{(k)}(\widehat{x}_t, P_{x_t}). && (4.6)
\end{aligned}
$$

The desired result is then obtained by redefining Δ_{jt}, from (3.57), as

$$\Delta_{jt}(\widehat{x}_t, P_{x_t}) = \sum_{k=1}^{M} \delta^{(k)} \min_{i=1,\ldots,n_j} \Delta_{ijt}^{(k)}(\widehat{x}_t, P_{x_t}). \quad (4.7)$$

4.2 Application: Urban Navigation

The modification (4.7) is particularly useful for CC-RRT when it is tightly integrated with a trajectory prediction algorithm which generates predictions of the form (4.1). This has been demonstrated for two prediction algorithms for dynamic obstacles in urban environments, RR-GP [95] and DPGP [96]. Both algorithms provide a likelihood and time-varying Gaussian state distribution for each possible behavior of a dynamic obstacle at each future timestep. As such, both algorithms are well-suited for the CC-RRT framework. This section briefly reviews the uncertainty models generated by each prediction algorithm, and provides simulation examples demonstrating the ability of an autonomous vehicle to safely navigate in urban environments with dynamic obstacles.

4.2.1 RR-GP

The RR-GP algorithm developed by Aoude *et al.* [95] learns motion pattern models by combining Gaussian process (GP) predictions with a sampling-based reachability

refinement, which conditions the GP predictions to enforce dynamic and environmental constraints [95]. By doing so, the accuracy of the behavior and trajectory predictions is significantly increased without having to increase the GP resolution, which typically requires extensive computation. Fulgenzi *et al.* [110] use Gaussian processes to model moving obstacles within an RRT path planner. However, this approach relies solely on Gaussian processes for its modeling, which can lead to less accurate prediction, and uses heuristics to assess path safety rather than providing guarantees.

The RR-GP algorithm was primarily designed to predict motion of moving vehicles in urban environments, which often demonstrate complex motion patterns with underlying intentions (*e.g.*, which way to turn when entering an intersection). Even with perfect sensing, two major sources of uncertainty remain: the future intentions of such a vehicle, and even if known, the exact trajectory to be followed for that behavior. Such agents are typically subject to significant dynamic constraints, such as minimum turning rates and bounded acceleration, which should be incorporated in the prediction. Additionally, such agents are operating within relatively structured environments such as narrow roads, which may limit the feasible options available to such an agent.

Figure 4-1 provides a visual illustration of the RR-GP approach shown in Aoude *et al.* [95]. Gaussian processes are learned for each behavior from observations of agents demonstrating that behavior, which are used as labeled training trajectories. Samples from those GPs (orange dots) are taken at fixed-timestep intervals for each motion. A tree of trajectories (brown) is then generated using those samples, taking into account the actual size of the dynamic obstacle (green circle) and environmental obstacles it is expected to avoid (gray). Since all trajectories remaining in the tree must be dynamically feasible and satisfy all environmental constraints, the remaining samples provide a conditioned estimate of the dynamic obstacle predictions at each timestep.

Suppose a dynamic model is following one of M motion patterns, which are assumed to be known *a priori* in this framework. The learned GPs take the form of

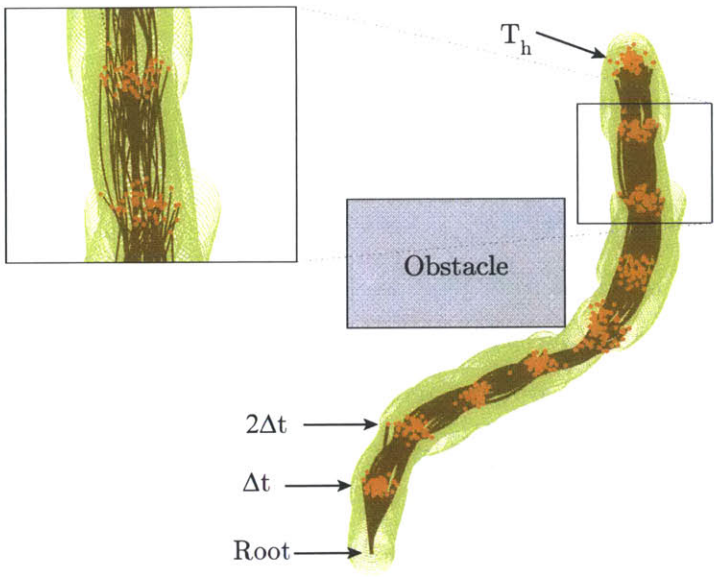


Figure 4-1: Illustration of RR-GP algorithm (Source: Aoude *et al.*)

a 2D velocity flow field, such that, if a dynamic obstacle's current position and behavior is known, its predicted next position is provided via Euler integration on the corresponding flow field [95]. The mixture model used for these predictions takes the form

$$\mathbb{P}(\mathbf{x}) = \sum_{k=1}^{M} \mathbb{P}(b_k)\mathbb{P}(\mathbf{x}|b_k), \tag{4.8}$$

where $\mathbf{x}$ is the observed trajectory, b_k is the kth behavior, and $\mathbb{P}(b_k)$ is its prior probability. After observing $\mathbf{x}$, the posterior probability of the kth behavior is shown to be [95]

$$\mathbb{P}(b_k|\mathbf{x}) \propto \mathbb{P}(b_k)\mathbb{P}(\mathbf{x}|b_k). \tag{4.9}$$

This form closely maps to (4.1), with $\delta_j^{(k)} \mapsto \mathbb{P}(b_k)$ representing the behavior uncertainty and $\mathcal{N}(\widehat{c}_{jt}^{(k)}, P_{c_{jt}}^{(k)}) \mapsto \mathbb{P}(\mathbf{x}|b_k)$ representing the trajectory uncertainty. The prediction of the dynamic obstacle's 2D position at timestep $t+K$, given observation

history $(x_{0:t}, y_{0:t})$, ultimately takes the form [95]

$$\mathbb{P}(x_{t+K}, y_{t+K}|x_{0:t}, y_{0:t}) = \sum_{k=1}^{M} \mathbb{P}(b_k|x_{0:t}, y_{0:t})\mathbb{P}(x_{t+K}, y_{t+K}|x_t, y_t, b_k). \tag{4.10}$$

4.2.2 RR-GP Example

An example is now provided from joint work with Aoude [95] which demonstrates the CC-RRT planner designing paths enabling an autonomous vehicle (referred to as the "host vehicle") to safely avoid an unknown, dynamic vehicle (referred to as the "target vehicle") using RR-GP predictions. All simulations were run on a 2.53GHz quad-core laptop with 3.48GB of RAM.

To ensure the RR-GP algorithm is tested on realistic driving behavior, the target vehicle's motion is selected randomly from among a set of simulated trajectories, pre-generated for each behavior by having a human operator manually drive a vehicle in simulation. The manual driving was performed via a wireless steering apparatus, tuned to emulate traditional, nonlinear control of an automobile.

Consider a ground vehicle operating in a constrained, two-dimensional environment 11.2 m $\times$ 5.5 m in size (Figure 4-2). In this scenario, the objective of the host vehicle is to go straight through the intersection at bottom-center of Figure 4-2(a), reaching a goal location on the opposite side. However, to get there, the host vehicle must successfully avoid an errant (rule-violating) driver which is traveling through the intersection in the perpendicular direction, and is likely to cross the intersection at the same time as the host vehicle. There are three possible behaviors for the target vehicle as it enters the intersection: (a) left turn, (b) right turn, and (c) straight. The host vehicle is assumed to have a radius of 20 cm, while the target vehicle has a radius of 14 cm; both start at zero velocity.

To ensure theoretical guarantees can be met exactly, the host vehicle is modeled

as a double integrator,

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ v^x_{t+1} \\ v^y_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ v^x_t \\ v^y_t \end{bmatrix} + \begin{bmatrix} \frac{dt^2}{2} & 0 \\ 0 & \frac{dt^2}{2} \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u^x_t \\ u^y_t \end{bmatrix},$$

where $dt = 0.1$s, subject to avoidance constraints $\mathcal{X}$ (including velocity bounds) and input constraints

$$\mathcal{U} = \{(u^x, u^y) \mid |u^x| \leq 4, |u^y| \leq 4\}.$$

To emphasize the impact of the dynamic obstacle's uncertainty, the host vehicle's own dynamics are assumed deterministic. Trajectories are simulated and executed in closed-loop via the controller

$$\begin{aligned} u^x_t &= -1.5(x_t - r^x_t) - 3(v^x_t - r^{v_x}_t), \\ u^y_t &= -1.5(y_t - r^y_t) - 3(v^y_t - r^{v_y}_t), \end{aligned}$$

where (r^x_t, r^y_t) is the reference position and $(r^{v_x}_t, r^{v_y}_t)$ is the reference velocity; the reference r_t is moved continuously between waypoints at a fixed speed of 0.35 m/s. The speed of the target vehicle is bounded at 0.4 m/s.

Several algorithms are tested in this scenario:

- Nominal RRT, which runs a standard real-time, closed-loop RRT [105] algorithm treating the target vehicle as a static obstacle at its most recent location;
- Naive RRT, which uses the same algorithm as Nominal RRT, but ignores the target vehicle entirely in order to establish a baseline minimum likelihood of safety; and
- CC-RRT (Algorithms 4-5), for $\delta_s \in \{0.5, 0.8, 0.9, 0.99, 0.999\}$.

Fifty trials are performed for each case; each trial differs only in the path followed

Table 4.1: Simulation results for RRT vs. CC-RRT with RR-GP

Algorithm	δ_s	**% to Goal**[a]	**Path Duration, s**[b]	**Time per Node, ms**
Naive RRT	–	38%	10.01 (0.11%)	0.611
Nominal RRT	–	46%	10.90 (8.96%)	0.662
CC-RRT	0.5	92%	11.52 (15.2%)	1.610
CC-RRT	0.8	88%	11.65 (16.5%)	1.620
CC-RRT	0.9	92%	11.69 (16.9%)	1.620
CC-RRT	0.99	96%	12.51 (25.1%)	1.537
CC-RRT	0.999	100%	12.84 (28.4%)	1.492

[a] Percentage of trials where system executed a path to goal without colliding with any obstacles.
[b] Percentage is average increase in path duration relative to minimal-time (obstacle-free) path, 10.0s. Only paths which reach goal without collision are included

by the target vehicle and the random sampling used in the RR-GP and CC-RRT algorithms. In particular, the sequence of selected target vehicle paths is consistent across all sets of 50 trials.

Table 4.1 presents averaged results over the 50 trials for each case on several figures of merit: the percentage of trials in which the vehicle safely reaches the goal; the average duration of such paths; and the average time to generate an RRT/CC-RRT tree node. In all five cases using CC-RRT, the host vehicle safely navigates the intersection with a much higher likelihood than any of the RRT instances. The CC-RRT results demonstrate the clear trade-off between overall path safety (in terms of percentage of trials which reach the goal) and average path duration when using CC-RRT. As δ_s is increased from 0.5 to 0.999, the percentage of safe trajectories generally increases, culminating with the host vehicle using CC-RRT with $\delta_s = 0.999$ reaching the goal safely in all fifty trials. On the other hand, as δ_s is increased and the planner becomes more conservative, the average time duration of the safe trajectories increases.

Figure 4-2 provides representative screenshots of the RR-GP and CC-RRT algorithms during trial #25 of the intersection scenario, for two different values of δ_s. The host vehicle's path history and current path are in orange; the objective of the host vehicle (large orange circle) is to reach the goal position (green circle) while avoiding

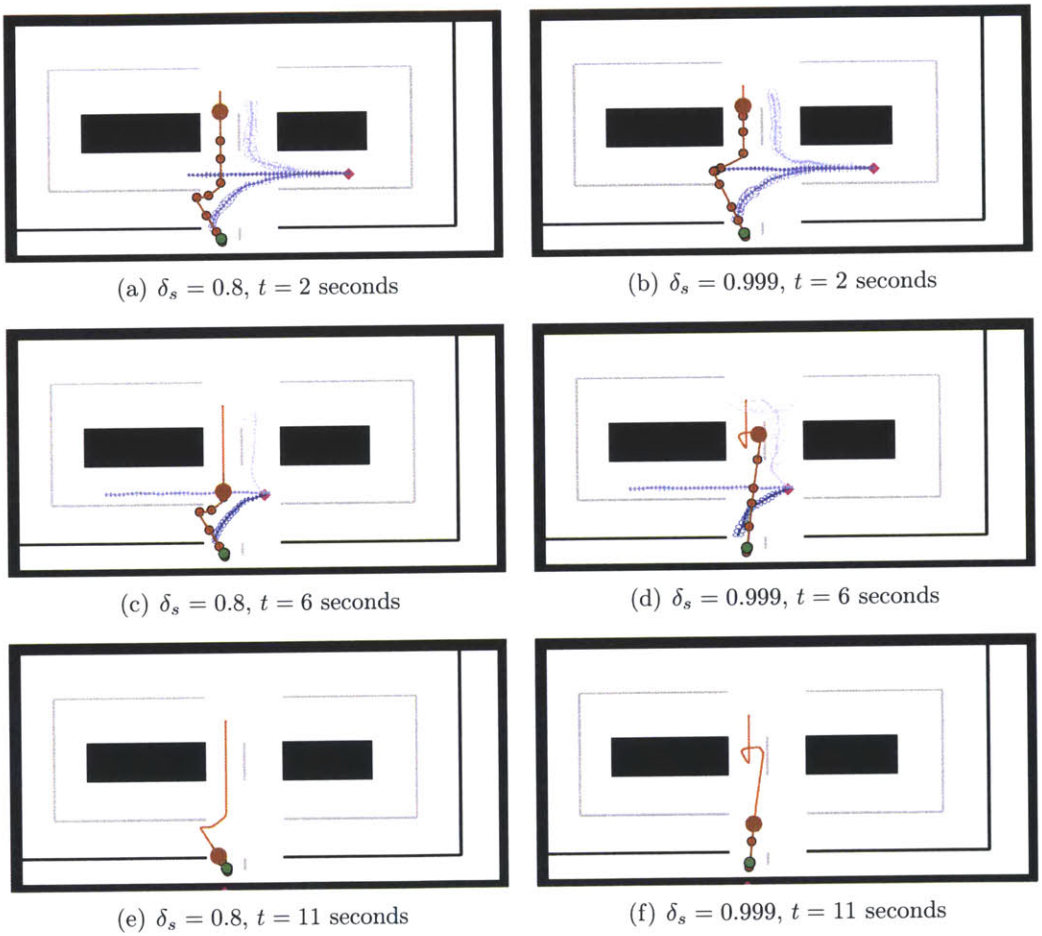
all static obstacles (black) and the dynamic target vehicle (magenta diamond). The blue paths indicate the paths predicted by the RR-GP algorithm for each possible behavior, including $2-\sigma$ uncertainty ellipses; more likely paths are indicated with brighter shades of blue. All objects are shown at true size; the gray lines are lane markings, which do not serve as constraints in this scenario.

Figure 4-2 sheds some light on how different values of δ_s affect the types of paths chosen by the planner. In this particular trial, the target vehicle ultimately makes a left turn through the intersection, and would collide with the host vehicle if it did not deviate from an initial straight-path trajectory. The RR-GP algorithm is initially undecided whether the target vehicle is going straight or turning left (as indicated by the shading on the predicted trajectories in Figures 4-2(a) and 4-2(b)); by $t = 6$ seconds RR-GP is very confident that the vehicle is turning left (Figures 4-2(c) and 4-2(d)).

When $\delta_s = 0.8$, the planner selects a path with the minimum perturbation needed to avoid the target vehicle's most likely trajectories (Figure 4-2(a)). As the target vehicle closes in on the intersection (Figure 4-2(c)), the host vehicle continues to hedge that it can cross the intersection safely and avoid the target vehicle's approach in either direction, and thus does not modify its plan.

In contrast, when $\delta_s = 0.999$, the planner selects a larger initial perturbation to maintain a large distance between the host and target vehicles (Figure 4-2(b)). The host vehicle then demonstrates much more risk-averse behavior by loitering outside the intersection (Figure 4-2(d)) for several seconds before making its approach. Ultimately, the host vehicle using $\delta_s = 0.8$ (Figure 4-2(e)) reaches the goal before the host vehicle using $\delta_s = 0.999$ (Figure 4-2(f)). In real driving scenarios, the most desirable behavior is likely somewhere between these two extremes.

Table 4.1 shows that with Naive RRT, by ignoring the target vehicle, the time-optimal path is almost always achieved, but a collision takes place in a majority of trials, with collisions occurring in most instances of the target vehicle going straight or left. In some instances, the Nominal RRT algorithm maintains safety by selecting an alternative trajectory when the target vehicle's current position renders the host



(a) $\delta_s = 0.8$, $t = 2$ seconds

(b) $\delta_s = 0.999$, $t = 2$ seconds

(c) $\delta_s = 0.8$, $t = 6$ seconds

(d) $\delta_s = 0.999$, $t = 6$ seconds

(e) $\delta_s = 0.8$, $t = 11$ seconds

(f) $\delta_s = 0.999$, $t = 11$ seconds

Figure 4-2: Representative snapshots of integrated RR-GP and CC-RRT algorithms, trial #25

vehicle's current trajectory infeasible. However, the overall likelihood of safety is still low. In many cases, the target vehicle collides with the host vehicle from the side, which cannot be prevented through replanning.

Finally, note that the average time to either generate an RRT node or call RR-GP is largely independent of δ_s for CC-RRT. There is a modest increase in average time per node when moving from naive or nominal RRT to CC-RRT – approximately scales by a factor of 2.5 – which is consistent with previous results on CC-RRT scalability (Chapter 3).

4.2.3 DPGP

The DPGP algorithm is a Bayesian nonparametric clustering algorithm which extends the GP modeling framework of RR-GP by providing offline, unsupervised learning of the mixture model (4.8) [111]. Provided with a set of unlabeled training trajectories, the DPGP prediction algorithm automatically determines the most likely number of clusters. The mixture model (4.8) is represented by a Dirichlet process, whose concentration parameter determines whether a new cluster should be formed [111].

In contrast to RR-GP, DPGP is also useful for modeling the behavior of pedestrians in urban environments. Pedestrians pose several particularly challenging tasks for behavior and trajectory prediction, such as unique and previously unobserved behaviors and instantaneous changes in motion [96]. Consideration of results for pedestrians are explored further in the hardware experiments (Chapter 6).

Depending on the implementation, the reachability-based conditioning of RR-GP may also be used in tandem with DPGP. In this case, DPGP replaces the original GP prediction component of RR-GP.

4.2.4 DPGP Example

Several examples are now provided demonstrating the ability of the CC-RRT planner to safely avoid a dynamic obstacle via DPGP predictions. In these examples, the DPGP predictions are enhanced with the reachability-based refinement of RR-

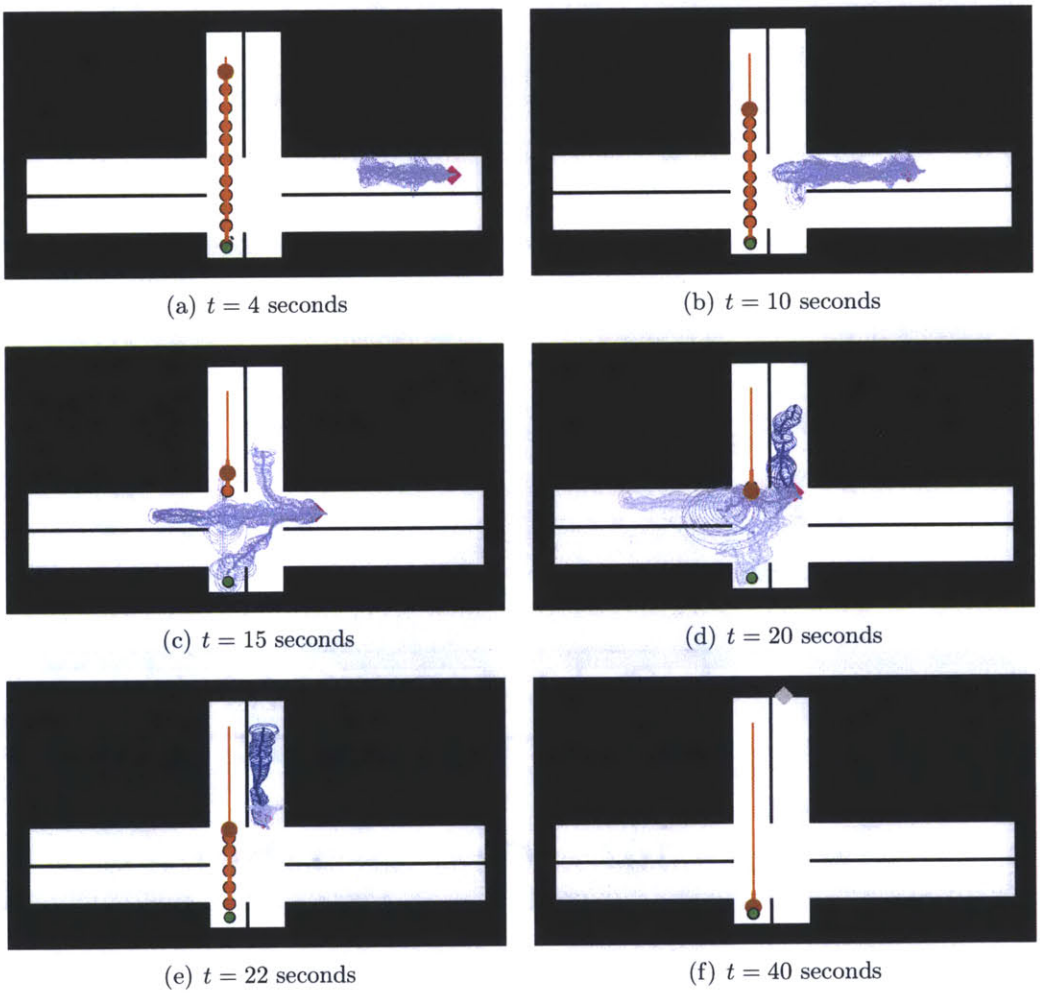
GP [95]. Similar to Section 4.2.2, the host vehicle is modeled as a double integrator.

Section 4-3 considers a modified version of the intersection scenario in Figure 4-2, with the host vehicle constrained to move in a straight line as it crosses the intersection. The host vehicle's path history and current path are in orange; the objective of the host vehicle (large orange circle) is to reach the goal position (green circle) while avoiding all static obstacles (black) and the dynamic target vehicle (magenta diamond). Unlike Section 4.2.2, the lane boundaries also serve as obstacles/constraints in this example. The blue paths indicate the paths predicted by the RR-GP algorithm for each possible behavior, including $2-\sigma$ uncertainty ellipses; more likely paths are indicated with brighter shades of blue. As before, there are three possible behaviors for the target vehicle as it enters the intersection: (a) left turn, (b) right turn, and (c) straight. The target vehicle is assumed to have right-of-way, and thus is not obligated to stop once it reaches the intersection.

Initially, the planner gives the host vehicle a path to have it cross the intersection (Figure 4-3(a)). However, as the target vehicle approaches the intersection (Figure 4-3(b)) and its planning horizon overlaps with the host vehicle's path, the planner curtails the host vehicle's path to stop at the intersection entry, in case the target vehicle decides to go straight through the intersection (Figure 4-3(c)). Eventually, the host vehicle must come to a full stop: though DPGP predicts that the vehicle is most likely turning right, it is not yet sufficiently confident to enable the planner to resume crossing the intersection (Figure 4-3(d)). Once it is clear that the target vehicle is turning right, the planner resumes an intersection crossing (Figure 4-3(e)), eventually reaching the goal (Figure 4-3(f)).

Figure 4-4 and 4-5 give two examples for an obstacle field scenario, in which the host vehicle is moving left-to-right while avoiding a dynamic obstacle moving in the opposite direction. The dynamic obstacle has six possible behaviors, corresponding to which of the three corridors it traverses in the central passage, and which of two different speeds is used (slow and fast). In Figure 4-4, the CC-RRT planner tree (green) is also visualized.

Initially, with the dynamic obstacle not yet proceeding forward, the CC-RRT



(a) $t = 4$ seconds

(b) $t = 10$ seconds

(c) $t = 15$ seconds

(d) $t = 20$ seconds

(e) $t = 22$ seconds

(f) $t = 40$ seconds

Figure 4-3: Representative snapshots of integrated DPGP and CC-RRT algorithms, modified intersection scenario

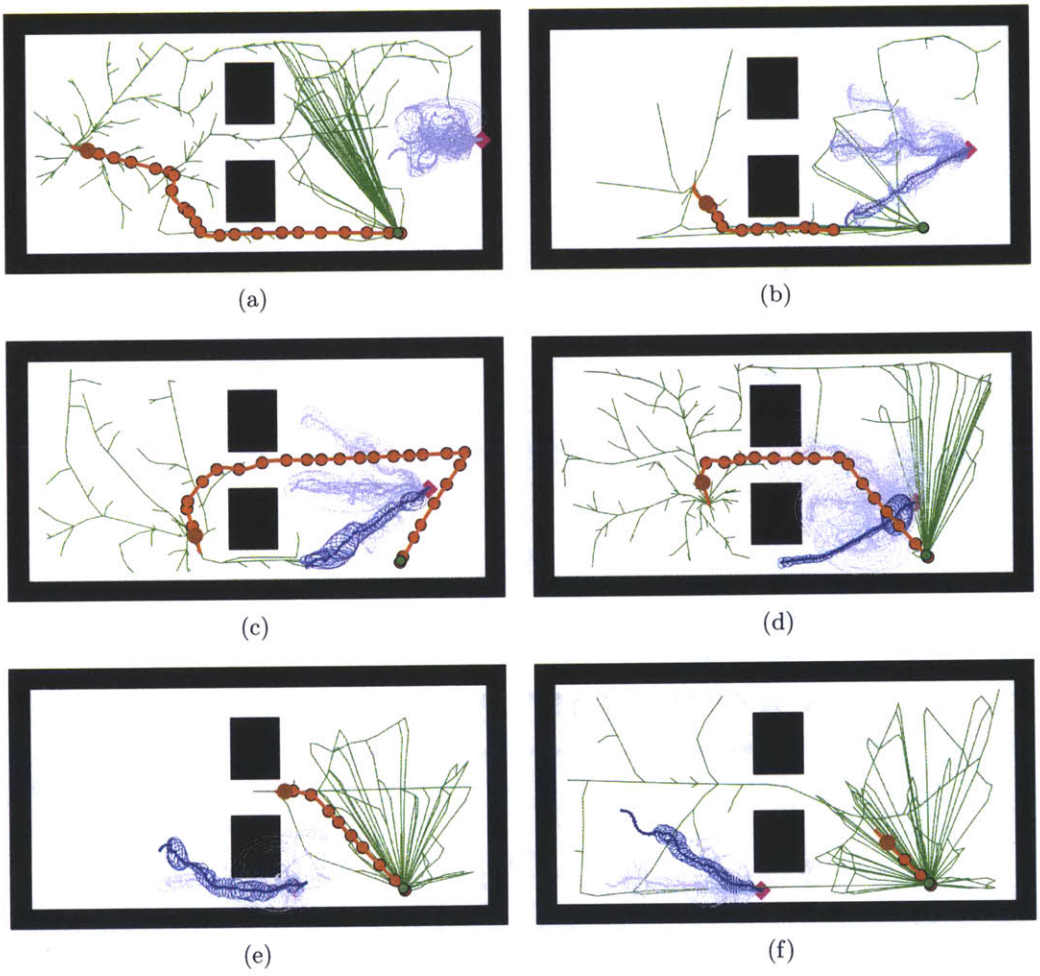


Figure 4-4: Representative snapshots of integrated DPGP and CC-RRT algorithms, obstacle field example #1

planner quickly finds a path all the way to the goal (Figure 4-4(a)). However, as the dynamic obstacle begins moving, the DPGP predictions quickly detect that the obstacle is planning to traverse the bottom corridor, curtailing the host vehicle's planned safe path (Figure 4-4(b)). The planner quickly finds an alternate route through the central corridor (Figure 4-4(c)); though this path overlaps with several possible behaviors, the low likelihood of those behaviors (indicated by the very light blue shading in Figure 4-4(c) results in a risk of collision below the necessary threshold. As the host vehicle begins executing this path and the DPGP predictions become more confident, the path is refined to reach the goal more quickly (Figure 4-4(d)). In this case, the planner path for the host vehicle overlaps with the target vehicle's most likely path, and will eventually come close to the target vehicle's current location. However, the prediction model anticipates that the target vehicle will have continued moving left to the central corridor by the time the host vehicle arrives, ensuring that the path is safe. Indeed, the host vehicle is able to continue executing this path to reach the goal safely (Figures 4-4(e) and 4-4(f)). The second example proceeds in a similar manner (Figure 4-5).

4.3 Particle CC-RRT

When applying CC-RRT to nonlinear dynamics (3.73), linearization can be applied to maintain a Gaussian state uncertainty representation along each tree trajectory (Section 3.5). However, if the dynamics are themselves subject to a non-Gaussian uncertainty, a Gaussian state representation is no longer appropriate for accurately representing the predicted uncertainty at future timesteps. For example, a Gaussian approximation may underestimate the true likelihood of colliding with an obstacle (Figure 4-6). This section introduces a particle-based chance constraint framework for CC-RRT, known as particle CC-RRT, which can be used to statistically approximate the uncertainty at a resolution which can be dictated by the user [97].

In particle CC-RRT, a set of particles are maintained at each node which statistically represent the true uncertainty distribution, which cannot be assumed to be Gaus-

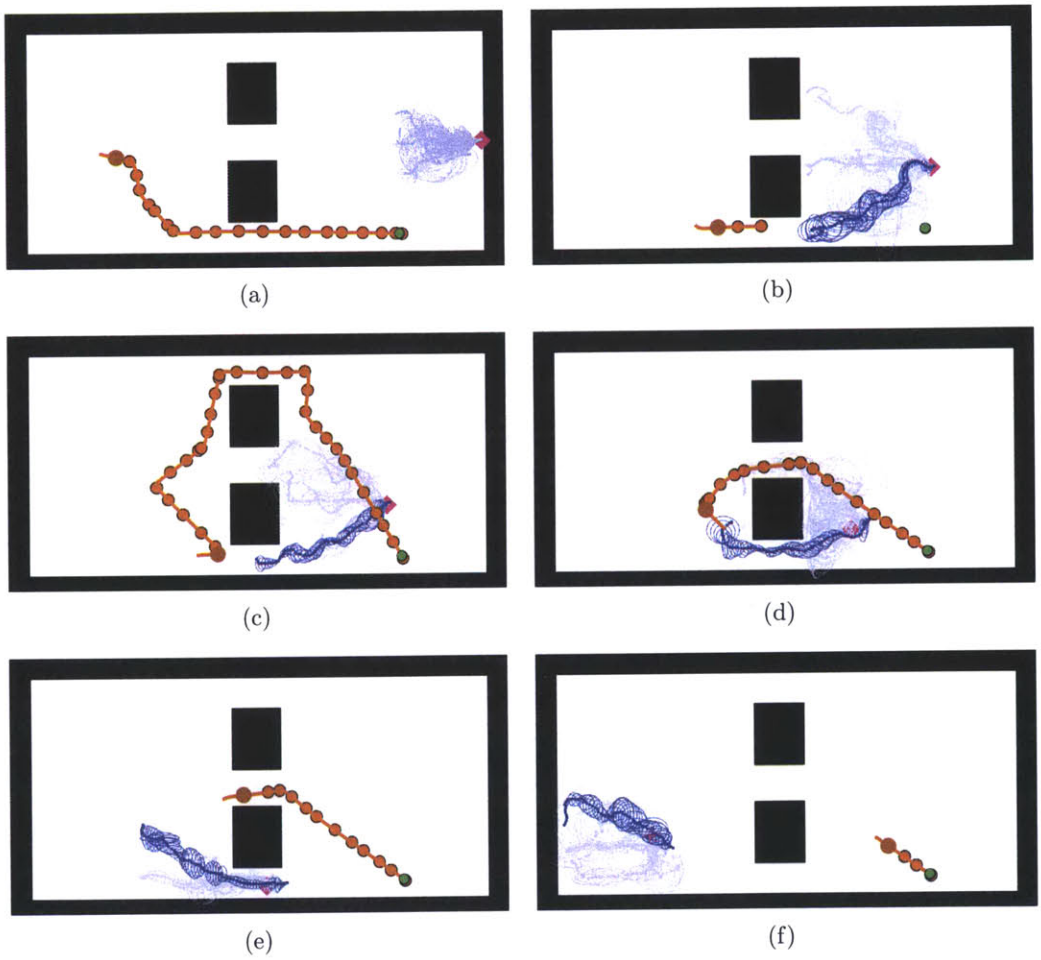


Figure 4-5: Representative snapshots of integrated DPGP and CC-RRT algorithms, obstacle field example #2

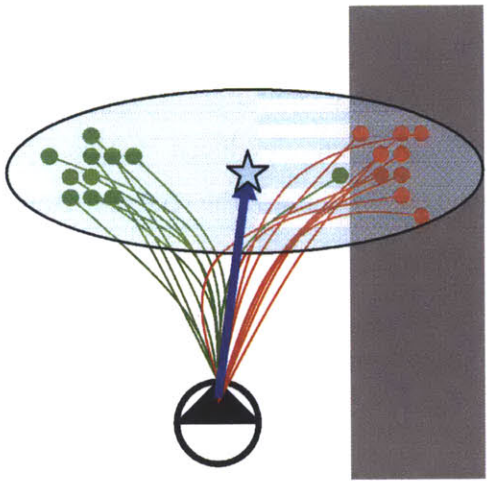


Figure 4-6: Modeling non-Gaussian uncertainties using Gaussian model

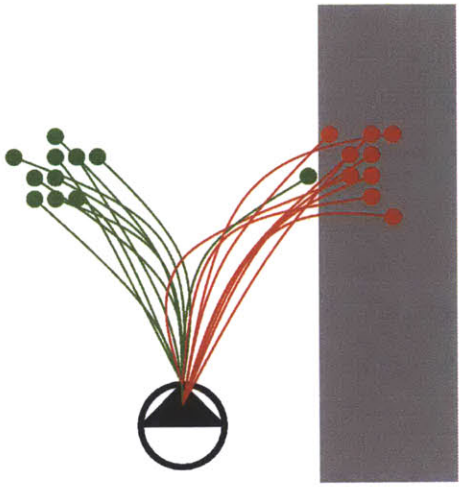


Figure 4-7: Modeling non-Gaussian uncertainties using particle CC-RRT

sian. Each particle is simulated through the true dynamics (3.73), randomly sampling each uncertainty source that is present. The chance constraints (3.5),(3.6) are then tested by assessing the fraction of particles which violate the problem constraints at each timestep and along each trajectory, respectively. Figure 4-7 demonstrates the propagation of a single particle CC-RRT node consisting of 20 particles, each sampling the modeled uncertainty. Each particle is checked for feasibility against obstacles (gray); infeasible particles (red) are discarded, while feasible particles (green) are maintained for future propagation.

The particle CC-RRT framework is generalizable both in the types of probabilistic feasibility which are assessed (time-step-wise and/or path-wise) and in the types of uncertainty that are modeled using particles. Though this section considers uncertainty in the state distribution only, this framework can be readily extended to consider hybrid combinations of multiple uncertainty types. For example, an agent's process noise may be represented using particles, while interactions with dynamic obstacles are modeled using traditional Gaussian distributions.

While probabilistic feasibility bounds can only be statistically approximated using particle CC-RRT, the algorithm can approximate chance constraints with modeling errors and levels of conservatism which both approach zero as the number of samples

approaches infinity. Indeed, this approach overcomes the conservatism introduced via the use of Boole's inequality and other bounds in the development of CC-RRT (Section 3.2). However, maintaining a large set of particles is often computationally intensive, and challenges the desired real-time nature of algorithms developed in this work. Identifying a number of particles which accurately approximates the true uncertainty distribution, without requiring excessive computation, is a key consideration in this approach.

Consider the nonlinear dynamics (3.73) subject to process noise and localization/initial state error uncertainty

$$w_t \sim \mathbf{W}(w), \tag{4.11}$$

$$x_0 \sim \mathbf{X}_0(x), \tag{4.12}$$

respectively, where $\mathbf{W}$ and $\mathbf{X}_0$ are random variables representing their true uncertainty distributions. In particle CC-RRT, as each trajectory is simulated, the state distribution at each simulation timestep is approximated as a weighted sum of n_P particles. At simulation timestep $t+k$ and execution timestep t, this can be represented as

$$\left\{ x^{(p)}_{t+k|t}, \omega^{(p)}_{t+k|t} \mid p \in \mathbb{Z}_{1,n_P} \right\} \in \mathbf{X}_{t+k|t}(x_{t+k|t}), \tag{4.13}$$

where $x^{(p)}_{t+k|t}$ is the state of the pth particle and $\omega^{(p)}_{t+k|t} > 0$ is its weight; the weights are generally normalized such that $\sum_{i=1}^{n_P} \omega^{(p)}_{t+k|t} = 1$. The particle-based mean of $\mathbf{X}_{t+k|t}(x_{t+k|t})$ can thus be computed as

$$\widehat{x}_{t+k|t} = \sum_{p=1}^{n_P} \omega^{(p)}_{t+k|t} x^{(p)}_{t+k|t}. \tag{4.14}$$

At each simulation timestep, the particle states $x^{(p)}_{t+k|t}$ are propagated through the

true dynamics (3.73), including uncertainties sampled from their true distributions,

$$x^{(p)}_{t+k+1|t} = f(x^{(p)}_{t+k|t}, u_{t+k|t}, w^{(p)}_{t+k|t}), \quad \forall\, p \in \mathbb{Z}_{1,n_P}, \tag{4.15}$$

$$w^{(p)}_{t+k|t} \sim \mathbf{W}(w), \quad \forall\, p \in \mathbb{Z}_{1,n_P}, \tag{4.16}$$

$$x^{(p)}_{0|0} \sim \mathbf{X}_0(x), \quad \forall\, p \in \mathbb{Z}_{1,n_P}. \tag{4.17}$$

While the state $x^{(p)}_{t+k|t}$ and sampled disturbance $w^{(p)}_{t+k|t}$ are unique to each particle on a given trajectory, the same input $u_{t+k|t}$ is applied to all particles. Each weight is also updated based on the likelihood of the uncertainty sampled, via

$$\omega^{(p)}_{t+k+1|t} \propto \omega_{t+k|t} \cdot \mathbb{P}(\mathbf{W}(w) = w^{(p)}_{t+k|t}) \quad \forall\, p \in \mathbb{Z}_{1,n_p}. \tag{4.18}$$

Weights should be re-normalized at each timestep such that they continue to sum to 1.

Each particle represents a fraction of the likelihood of feasibility at each timestep, corresponding to its weight $\omega^{(p)}_{t+k|t}$. At each simulation timestep, a binary collision check is performed on each particle state $x^{(p)}_{t+k|t}$ against the state constraints $\mathcal{X}_{t+k}$ of (2.5). Since the weights are normalized, the time-step-wise chance constraint (3.5) can then be statistically approximated as

$$\gamma_{t+k|t} \equiv \sum_{p=1}^{n_P} \omega_{t+k|t} \mathbb{I}\left(x^{(p)}_{t+k|t} \in \mathcal{X}_{t+k}\right) \geq \delta_s, \tag{4.19}$$

where $\mathbb{I}$ is the indicator function, *i.e.,* 1 if the contained statement is true and 0 otherwise.

Particles are discarded as soon as they become infeasible, as they no longer represent a realizable trajectory. However, to maintain the resolution of the particle-based uncertainty approximation, new particles are resampled from existing ones such that n_P particles are always propagated from each timestep. The likelihood of a particle being re-sampled is proportional to its weight. In this manner, the path-wise chance

constraint (3.6) can be statistically approximated as

$$\delta_{t+k|t} \geq \delta_p, \tag{4.20}$$

where

$$\delta_{t+k+1|t} = \delta_{t+k|t}\gamma_{t+k|t}, \quad \delta_{0|0} = 1. \tag{4.21}$$

Algorithm 6 details the tree expansion step for PCC-RRT. During each simulation timestep, after any infeasible particles are resampled (line 9) and an input selected (line 10), each particle is individually simulated (line 12) and assigned an appropriate weight (line 13). After weights have been re-normalized (line 15) and the new mean state $\widehat{x}_{t+k+1|t}$ computed (line 16), any infeasible particles are removed (line 17). Propagation of the particles continues until either the approximate chance constraints (4.19), (4.20) are violated, or the mean state reaches the sample x_{samp}.

The execution loop is a straightforward application of Algorithm 5. At timestep $t = 0$, the initial set of particles is initialized with $x_{0|0}^{(p)}$ sampled from (4.17).

Figure 4-8 considers a simple example of particle CC-RRT, using the skid-steered dynamics of Section 3.7.4. Figure 4-8 shows representative trees (blue edges) generated by the particle CC-RRT algorithm after 40 seconds of growth for the vehicle (orange) attempting to reach the goal (green); 100 particles are generated for each node. The vehicle is controlled in closed-loop; at each timestep there is a 50% chance that either its left or right skid will slip (but not both), causing a uniformly-distributed disturbance to that skid's speed. As in Chapter 3, δ_s denotes the required probability of feasibility at each timestep; a node's size decreases as it comes closer to violating this constraint. On the other hand, δ_p denotes the required probability of feasibility along an entire path; a node's color changes from green to red as it comes closer to violating this constraint. In comparing Figure 4-8(a) to Figure 4-8(b), it is seen that increasing δ_p causes nodes to become infeasible more quickly due to accumulation of particles in collision along tree trajectories.

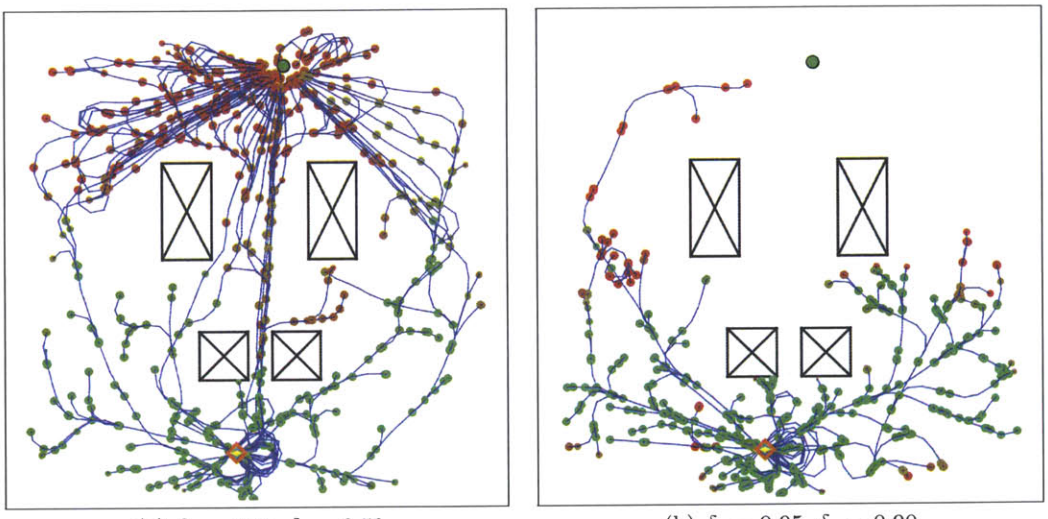
Algorithm 6 Particle CC-RRT, Tree Expansion

1: Inputs: tree $\mathcal{T}$, current timestep 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 node
6: $(\widehat{x}^{(p)}_{t+k|t}, \omega^{(p)}_{t+k|t}) \leftarrow$ final set of particle states and weights for N_{near}
7: Compute $\widehat{x}_{t+k|t}$ using (4.14)
8: **while** (4.19),(4.20) satisfied **and** $\widehat{x}_{t+k|t}$ has not reached x_{samp} **do**
9: Resample particles up to n_P
10: Select input $u_{t+k|t} \in \mathcal{U}$
11: **for** each particle p **do**
12: Simulate $x^{(p)}_{t+k+1|t}$ using (4.15),(4.16)
13: Assign weight $\omega^{(p)}_{t+k+1|t}$ using (4.18)
14: **end for**
15: Normalize weights
16: Compute $\widehat{x}_{t+k+1|t}$ using (4.14)
17: Remove infeasible particles
18: $k \leftarrow k+1$
19: **end while**
20: **for** each feasible node N **do**
21: Add N to $\mathcal{T}$
22: Try connecting N to $\mathcal{X}_{\text{goal}}$ (lines 5–19)
23: **end for**
24: **end for**

4.4 Application: Parafoil Terminal Guidance

One application requiring particularly careful consideration of uncertainty within motion planning is parafoil terminal guidance: guiding a parafoil from a potentially high altitude to land precisely with a desired position. This motion planning problem presents several key challenges. Parafoil dynamics are highly nonlinear and underactuated, with potentially large turning radii and severely limited altitude control. Landing may require interacting with arbitrary, non-convex terrain maps, which may not easily be represented in a polyhedral form as in the baseline CC-RRT algorithm (Chapter 3). Perhaps most significantly, parafoils are subject to uncertain and variable wind environments which can lead to significant errors between predicted and actual trajectories, often resulting in unacceptable landing accuracy.

The CC-RRT framework has been adapted in joint work with Ian Sugel [98, 112]



(a) $\delta_s = 0.95$, $\delta_p = 0.50$ (b) $\delta_s = 0.95$, $\delta_p = 0.90$

Figure 4-8: Particle CC-RRT for skid-steering vehicle

to address the special concerns of the parafoil terminal guidance problem, enabling a large, autonomous parafoil to robustly execute collision avoidance and precision landing on mapped terrain, even with significant wind uncertainties [98, 112]. The proposed algorithm, analytic CC-RRT, offers two key extensions of the CC-RRT framework. First, unlike RR-GP (Section 4.2), a learned uncertainty model for the wind dynamics is incorporated directly into the simulated dynamics. This uncertainty model is multi-modal, with each mode corresponding to different classes of wind behavior which have been learned offline, and can be used to classify wind observations online.

Second, even though the augmented vehicle/wind dynamics yield an analytic state distribution for the vehicle at future timesteps, these uncertainty distributions are sampled for collision checking. These "covariance samples" are evenly spaced within the state distribution at each timestep at distances proportional to the covariance and thus are assigned equal weight. Sampling the uncertainty distributions allows for consideration of the terrain map as a 3D surface function, rather than forcing it into polyhedral obstacle boundaries to fit the baseline CC-RRT formulation. Additionally, unlike the particle-based formulation (Section 4.3), dynamic state propagation

of individual samples is not required: only the mean state needs to be simulated. By having an analytic state distribution available along each trajectory, covariance samples can be quickly generated, yielding a robustness formulation with stronger computational efficiency than particle CC-RRT. This section reviews these extensions to the CC-RRT framework [98].

The overall parafoil dynamics take the form [98]

$$\begin{aligned} p_{x,t+1} &= p_{x,t} + dt\left(v(p_{z,t})\cos\psi_t + w_{x,t}\right), \\ p_{y,t+1} &= p_{y,t} + dt\left(v(p_{z,t})\sin\psi_t + w_{y,t}\right), \\ p_{z,t+1} &= p_{z,t} + dt\left(\frac{-v(p_{z,t})}{L_D} + w_{z,t}\right), \\ \mathbf{s}_{t+1} &= \mathbf{s}_t + dt\left(A\mathbf{s}_t + Bu_t\right), \\ \psi_{t+1} &= \text{sat}\left(\psi_t + dt\left(C\mathbf{s}_t + Du_t\right), -\omega_{\max}, \omega_{\max}\right), \end{aligned} \tag{4.22}$$

where $\mathbf{x}_t = (p_{x,t}, p_{y,t}, p_{z,t}, \mathbf{s}_t, \psi_t)$ is the state vector, $\mathbf{p}_t = (p_{x,t}, p_{y,t}, p_{z,t})$ is the position, ψ_t is the heading, L_D is the lift-to-drag ratio, and $\omega_{\max}$ is the maximum turning rate. The parafoil velocity v is a simple function of the parafoil altitude, while $\mathbf{s}_t$ is a fifth-order state vector with dynamics (A, B, C, D) corresponding to the lag dynamics of the parafoil [98].

The wind disturbance $\mathbf{w}_t = (w_{x,t}, w_{y,t}, w_{z,t})$ is modeled as the multi-modal linear dynamics [112]

$$\mathbf{w}_{t+1} = \overline{\mathbf{w}} + (I + dtA_c)(\mathbf{w}_t - \overline{w}) + dtB_c\mathbf{v}_t, \ \forall\ c \in \mathbb{Z}_{1,N_C}, \tag{4.23}$$

where $\overline{\mathbf{w}}$ is a 3-D mean wind estimate computed via a finite impulse response filter [112], $\mathbf{v}_t \sim \mathcal{N}(0, I)$ is zero-mean, unit-variance Gaussian noise, and (A_c, B_c) are dynamics specific to each of the N_C wind classes.

The wind classes are identified by performing DP-means clustering on a set of features extracted from live airdrop data, partitioning the airdrop data [112]. For wind class c, the (A_c, B_c) are then chosen to fit the empirical data for wind profiles in that class. For the 2D wind models used in this work, $A_c = \alpha_c I$ and $B_c = \beta_c I$. Online

wind classification is performed via SVM classification, identifying which version of (4.23) is used to augment the parafoil model (4.22).

Even though the combined parafoil/wind dynamics are nonlinear, the effect of the wind disturbances on the dynamics is linear, such that a simple analytical state distribution can be derived. By defining the 2D variation state

$$\delta \mathbf{x} = \begin{bmatrix} \delta p_{x,t} & \delta p_{y,t} & \delta w_{x,t} & \delta w_{y,t} \end{bmatrix}^T, \tag{4.24}$$

the variation dynamics take the form [112]

$$\delta \mathbf{x}_{t+1} = \underbrace{\begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1+dt\alpha_c & 0 \\ 0 & 0 & 0 & 1+dt\alpha_c \end{bmatrix}}_{\overline{A}} \delta \mathbf{x}_t + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ dt\beta_c & 0 \\ 0 & dt\beta_c \end{bmatrix}}_{\overline{B}} \mathbf{v}_t. \tag{4.25}$$

Because this is a linear Gaussian system, all future state distributions take the form $\mathbf{x}_t \in \mathcal{N}(\widehat{\mathbf{x}}_t, Q_{x_t})$. As in the original formulation of CC-RRT with nonlinear dynamics (Section 3.5), $\widehat{\mathbf{x}}_t$ is computed through direct simulation of (4.22)-(4.23), while the covariance Q_{x_t} can be written in implicit form

$$Q_{x_{t+1}} = \overline{A} Q_{x_{t+1}} \overline{A}^T + \overline{B}\overline{B}^T, \tag{4.26}$$

with $\overline{A}$ and $\overline{B}$ defined appropriately [112].

The covariance samples used for analytic CC-RRT are then equi-spaced samples of the position covariance

$$P_{x_t} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \times Q_{x_t} \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{4.27}$$

In practice, the N_s covariance samples are placed on one or more uncertainty ellipses for fixed σ-values. Figure 4-9 shows an example of the covariance samples (black) generated for an example parafoil trajectory (orange).

In analytic CC-RRT, each simulated trajectory within the tree uses the parafoil dynamics (4.22) and wind model (4.23), written in shorthand model form as

$$\widehat{\mathbf{x}}_{t+k+1|t} = f(\widehat{\mathbf{x}}_{t+k|t}, \mathbf{u}_{t+k|t}, \widehat{\mathbf{w}}_{t+k|t}), \tag{4.28}$$

$$\widehat{\mathbf{w}}_{t+k+1|t} = f_w(\widehat{\mathbf{w}}_{t+k|t}, \overline{\mathbf{w}}_t, 0). \tag{4.29}$$

At simulation timestep $t+k$ and execution timestep t, denote the mean of the position distribution as $(\widehat{p}_{x,t+k|t}, \widehat{p}_{y,t+k|t}, \widehat{p}_{z,t+k|t})$, and the location of the jth covariance sample as $(\widehat{p}_{x,t+k|t} + \delta p^{(j)}_{x,t+k|t}, \widehat{p}_{y,t+k|t} + \delta p^{(j)}_{y,t+k|t}, \widehat{p}_{z,t+k|t})$. A path-wise chance constraint (3.6) is imposed, and approximated as $p_{\text{collide}} \leq 1 - \delta_p$, where [98]

$$p_{\text{collide}} = \frac{1}{N_s} \sum_{j=1}^{N_s} \mathbb{I}\left[\bigwedge_{i=0}^{k} \widehat{p}_{z,t+i|t} \leq T\left(\widehat{p}_{x,t+i|t} + \delta p^{(j)}_{x,t+i|t}, \widehat{p}_{y,t+i|t} + \delta p^{(j)}_{y,t+i|t} \right) \right], \tag{4.30}$$

$\mathbb{I}[\cdot]$ is the indicator function, and $T(p_x, p_y)$ is the terrain map being used. A node is considered to have landed on the terrain if $p_{\text{collide}} > 1 - \delta_p$ or the nominal trajectory intersects the terrain, *i.e.*,

$$\widehat{p}_{z,t+k|t} > T\left(\widehat{p}_{x,t+k|t}, \widehat{p}_{y,t+k|t} \right). \tag{4.31}$$

The tree expansion step and execution loop for Analytic CC-RRT are presented in Algorithms 7 and 8, respectively [98]. Due to its sampling-based nature, the analytic CC-RRT algorithm is designed to handle arbitrary initial altitudes, approach geometries, and terrain surfaces. Additionally, through the use of wind modeling and covariance sample collision-checking, analytic CC-RRT is robust to significant wind disturbances which may be highly dynamic throughout terminal approach, including updrafts and downdrafts.

For the cost function, analytic CC-RRT uses (3.61) with $\phi \equiv 0$ and ϕ_f equal to a

Algorithm 7 Analytic CC-RRT: Tree Growth

1: Take a sample $\mathbf{x}_{\text{samp}}$ from the environment
2: Identify the m nearest nodes using heuristics
3: **for** $m \leq M$ sorted nearest nodes **do**
4: $N_{near} \leftarrow$ current node, $(\widehat{\mathbf{x}}_{t+k|t}, \widehat{\mathbf{w}}_{t+k|t}) \leftarrow$ final vehicle and wind state of N_{near}
5: $p_{\text{collide}} \leftarrow$ fraction of feasible covariance samples after N_{near}
6: **while** $p_{\text{collide}} < 1 - \delta_p$ **and** (4.31) true **and** $\widehat{\mathbf{x}}_{t+k|t}$ has not reached $\mathbf{x}_{\text{samp}}$ **do**
7: Select input $\mathbf{u}_{t+k|t} \in \mathcal{U}$
8: Simulate $(\widehat{\mathbf{x}}_{t+k+1|t}, \widehat{\mathbf{w}}_{t+k+1|t})$ using (4.28),(4.29)
9: Create intermediate nodes as appropriate
10: Compute/retrieve $P_{t+k+1|t}$ using (4.26)
11: Compute p_{collide} using (4.30)
12: $k \leftarrow k + 1$
13: **end while**
14: **for** each identified node N **do**
15: Add N to tree
16: Try connecting N to $\mathbf{x}_G$
17: **end for**
18: **end for**

reachability-based cost-to-go estimate [98]. Inputs are selected in line 7 of Algorithm 7 based on a reference model which generates circular arcs to connect existing nodes to new samples [98].

Figure 4-9 shows a screenshot of an analytic CC-RRT simulation in progress. The planner constructs a tree of feasible trajectories (teal), and executes a path (orange) which guides the parafoil (blue) to land on the terrain (background; changes from green to red with increasing altitude) at the goal (green circle).

4.4.1 Analytic CC-RRT Example

This section reviews simulation results from Luders *et al.* [98] demonstrating the effectiveness of analytic CC-RRT in improving average and worst-case accuracy relative to the state-of-the-art on complex terrain. Additional simulation results can be found in that work which demonstrate the algorithm's invariance to terrain and initial altitude, as well as the superiority of the composite wind model and cost-to-go over their individual components alone.

Three algorithms are compared throughout this section:

Algorithm 8 Analytic CC-RRT: Execution

Require: Initial vehicle state $\mathbf{x}_I$, initial wind measurement $\mathbf{w}_I$, goal state $\mathbf{x}_G$

$t \leftarrow 0$, $\mathbf{x}_t \leftarrow \mathbf{x}_I$, $\mathbf{w}_t \leftarrow \mathbf{w}_I$
Initialize tree with node at $\mathbf{x}_t$
while $\mathbf{x}_t \notin \mathcal{X}$ **do**
    Update current vehicle state $\mathbf{x}_t$, wind measurement $\mathbf{w}_t$, and mean wind estimate $\overline{\mathbf{w}}_t$
    Propagate mean state $\mathbf{x}_t$ by computation time $\rightarrow \mathbf{x}_{t+\Delta t}$ using (4.28),(4.29)
    Update tree feasibility and costs
    while time remaining for this timestep **do**
        Expand the tree by adding nodes (Algorithm 7)
    end while
    Use cost to identify lowest-cost path $\{N_{\text{root}}, \ldots, N_{\text{target}}\}$
    if at least one path exists **then**
        Apply best path
    else
        Apply "safe" action
    end if
    $t \leftarrow t + \Delta t$
end while
Mark vehicle as landed at $\mathbf{x}_t$

- **RRT with mean wind**, which represents a nominal RRT planner in which uses the mean wind estimate $\overline{\mathbf{w}}$, but assumes no future wind variation (*i.e.*, $\delta\mathbf{w} \equiv 0$, $\mathbf{w}_t \equiv \overline{\mathbf{w}}$).

- **Analytic CC-RRT**, *i.e.*, Algorithms 7-8, with $\delta_p = 0.9$.

- **BLG**, or band-limited guidance, which utilizes band-limited control to ensure accurate tracking and prediction, as well as knowledge of the mean wind estimate $\overline{\mathbf{w}}$ and replanning to account for system disturbances [99].

To simplify comparisons, a fixed number of iterations are performed for each algorithm during each 1-Hz planning cycle, with the number of iterations chosen for BLG and the RRT-based algorithms such that both use a comparable amount of computation on each planning cycle. The figure of merit in these simulations is the miss distance, *i.e.*, the 2D distance between the location where the parafoil lands and the desired/target location; the mean and worst-case values of miss distance are of particular interest.

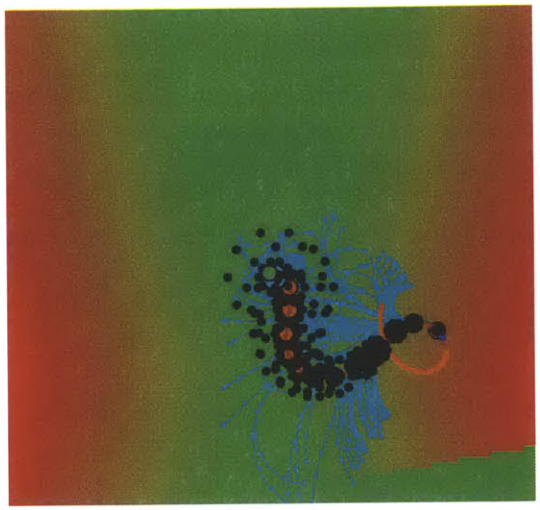
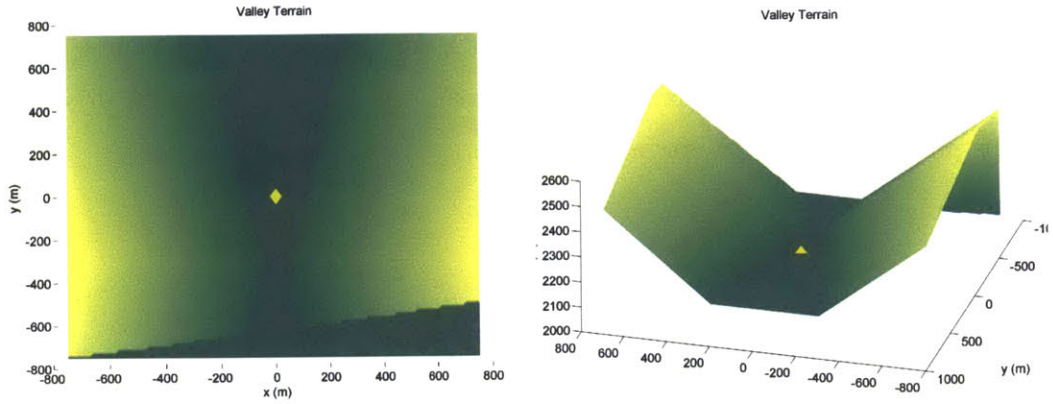


Figure 4-9: Example of analytic CC-RRT

For each simulation trial, each algorithm is tested on a random wind profile and initial condition. The parafoil state is initialized 500 meters above the goal, with a random heading and lateral distance randomly chosen between 100 and 400 meters from the goal. The terrain used in the simulations is the 1.5 km $\times$ 1.5 km valley terrain pictured in Figures 4-9 and 4-10. This represents a particularly challenging terrain for the parafoil terminal guidance problem, for several reasons. First, the slope of the valley is greater than the glide-slope of the parafoil, limiting planning options at lower altitudes by making approach from either side impossible. Second, the large low-altitude regions away from the goal (bottom-right and top in Figure 4-10(a)), where terrain collisions can be avoided for longer path durations, are likely to lead to terrain interactions as the parafoil's path crosses in and out of those regions. Finally, by placing the goal near a terrain "bottleneck," planning becomes more difficult near the goal compared to planning away from the goal.

Table 4.2 shows the statistics for each algorithm over 500 trials on the valley terrain Analytic CC-RRT demonstrates matching or improved landing accuracy, relative to RRT with mean wind and BLG, at nearly all percentiles. Both RRT-based algorithms show significant improvement over BLG for all but the worst-case trials. The mean landing accuracy for both is lower than BLG by a factor of 2. This ratio continues



(a) top view

(b) skew-side view

Figure 4-10: Valley terrain used in Analytic CC-RRT simulations

Table 4.2: Miss distance data for valley terrain comparison, over 500 trials (in meters)

Algorithm	**Mean**	**StDev**	**50%**	**80%**	**90%**	**95%**	**98%**	**Max**
RRT with mean wind	33.9	49.3	19.3	56.6	79.7	101	151	548
Analytic CC-RRT	30.8	32.8	18.7	52.5	75.8	103	126	218
BLG	63.5	89.0	37.9	66.1	153.2	226.9	430.5	581

approximately up to the 95th percentile, and increases to a factor of 3–4 by the 98th percentile (Table 4.2). In particular, about 12% of BLG trials have an miss distance exceeding 100m, whereas only about 5% of the RRT-based trials have a miss distance exceeding 100m. The BLG algorithm also demonstrates a "long tail": 4% of trials have a miss distance of 300m or worse, while the worst-case trial misses by 581m.

Analytic CC-RRT demonstrates superior performance over both RRT and BLG over the worst 5% of trials. Up to the 95th percentile, performance is similar between the two algorithms, though analytic CC-RRT shows slight improvement at most percentiles (and a better mean accuracy). This suggests that for those trials in which terrain interaction is unlikely, the robustness-based enhancements in analytic CC-RRT do not significantly influence performance relative to RRT with mean wind prediction alone. However, the two CDF curves diverge beyond the 95th percentile. At the 98th percentile, analytic CC-RRT miss distance is 17% lower than RRT. By the worst-case trial (*i.e.*, 99.8%), analytic CC-RRT miss distance is 60% lower. All trials of analytic CC-RRT have an accuracy of 218m or less, whereas RRT demonstrates

some trials exceeding 500m (Table 4.2). This "shorter tail" for analytic CC-RRT, relative to both RRT with mean wind and BLG, demonstrates the robustness of the algorithm to pathological uncertainty conditions, which might otherwise drive the vehicle prematurely into the terrain.

Chapter 5

Asymptotically Optimal Planning with Probabilistic Robustness

This chapter presents a novel sampling-based planner, CC-RRT*, which generates robust, asymptotically optimal trajectories in real-time, subject to process noise, localization/initial state error, and dynamic and/or uncertain environmental constraints. As with CC-RRT (Chapter 3), probabilistic feasibility is guaranteed for linear Gaussian systems by using trajectory-wise constraint checking of RRT [53] to efficiently bound the risk of constraint violation online [46]. However, CC-RRT* builds upon the RRT* framework [56, 57] to provide guarantees on asymptotic optimality of the lowest-cost probabilistically feasible path found by "rewiring" the tree toward lower-cost paths, with the tradeoff of requiring a steering law. The resulting real-time algorithm asymptotically converges toward minimum-length, dynamically feasible trajectories which satisfy all time-step-wise and path-wise probabilistic feasibility constraints specified, even in complex environments. Alternatively, a novel, risk-based objective function is posed which allows the user to trade-off between minimizing path duration and risk-averse behavior. This objective uses the same risk bounds computed to check the probabilistic feasibility constraints (Chapter 3), such that no additional computation is required, and is shown to be admissible as an RRT* objective. Unlike RRT*, CC-RRT* can model soft probabilistic feasibility constraints via the risk-based objective, in addition to hard probabilistic feasibility constraints.

5.1 The CC-RRT* Algorithm

The CC-RRT* algorithm builds upon the CC-RRT algorithm (Chapter 3), which enables the use of probabilistic constraints [46]. As is the case with CC-RRT, CC-RRT* grows a tree of state distributions known to satisfy an upper bound on probability of collision. At the same time, the CC-RRT* algorithm has been designed to fit into the constraints of the RRT* framework [56], such that guarantees on asymptotic optimality are maintained. The RRT* framework expands on RRT by "rewiring" connections within the tree, in two ways. First, anytime a feasible connection to a new sample is identified, rather than adding that connection directly to the tree as a new node (as in RRT), all "nearby" nodes (in terms of a metric defined below) are also checked. The lowest-cost, feasible connection among all such nodes is added to the tree. Second, a connection is attempted from this new node to all nearby nodes. If such a connection is feasible and achieves a lower cost than the existing path to that node, the existing path is replaced. The CC-RRT* algorithm expands this framework by checking *probabilistic* feasibility, using the chance constraints (3.5) and/or (3.6) via the form (3.63)-(3.64).

For each simulated trajectory, the CC-RRT* algorithm propagates the predicted state distribution, which under the assumption of Gaussian uncertainty is itself Gaussian, via (3.9). Thus, at each timestep of each simulated trajectory, it is only necessary to propagate the state conditional mean (3.12) and covariance (3.13), using the model form (3.71)-(3.72). In this manner, the distribution mean $\widehat{x}_t$ replaces the role of the true state x_t in the nominal RRT algorithm.

The CC-RRT* tree is denoted by $\mathcal{T}$, consisting of $|\mathcal{T}|$ nodes. Each node N of the tree $\mathcal{T}$ consists of a sequence of state distributions, checked for probabilistic feasibility. As described above, each state distribution can be characterized by a distribution mean $\widehat{x}$ and covariance P. The terminal mean and covariance of a node (*i.e.,* at the end of its sequence) are denoted by $\widehat{x}[N]$ and $P[N]$, respectively. A sequence of means and covariances will typically be denoted by $\bar{\sigma}$ and $\bar{\Pi}$, respectively.

The cost function takes the form (3.61), with $\phi_f = 0$, and is utilized in two forms

within the CC-RRT* algorithm. Let $t[N]$ denote the terminal timestep for node N. The notation

$$J[N] = dt \sum_{t=0}^{t[N]} f(\widehat{x}_t, P_t), \tag{5.1}$$

denotes the entire path cost from the starting state to the terminal state of node N, where dt is the timestep duration and f is the per-timestep cost objective specified by the user. Alternatively, for the state distribution sequence $(\bar{\sigma}, \bar{\Pi})$, the notation

$$\Delta J(\bar{\sigma}, \bar{\Pi}) = dt \sum_{(\widehat{x},P)\in(\bar{\sigma},\bar{\Pi})} f(\widehat{x}, P) \tag{5.2}$$

denotes the cost of that sequence. Eq. (5.1) can be constructed recursively by utilizing (5.2): if $(\bar{\sigma}, \bar{\Pi})$ denotes the trajectory of node N with parent N_{parent}, then

$$J[N] = J[N_{\text{parent}}] + \Delta J(\bar{\sigma}, \bar{\Pi}). \tag{5.3}$$

As is the case for algorithms proposed in previous chapters, the CC-RRT* algorithm consists of a tree expansion step and execution loop. The tree expansion step for CC-RRT* is given by Algorithm 9. It starts with the current tree $\mathcal{T}$ at timestep t (line 1 of Algorithm 9) and seeks to add additional nodes to the tree, possibly removing some in the process via rewiring. First, a state is sampled from the environment via the "Sample" function (line 2). The function x = Sample() must return independent and identically distributed samples from X_{free}, the obstacle-free portion of the state space [57]. It is assumed in the problem statement (Section 3.1) that all state elements are bounded. Thus, the approach utilized here is to sample each state element independently, with each realization having a non-zero probability, then filter out any infeasible samples.

Next, a node in $\mathcal{T}$ nearest to x_{samp}, N_{nearest}, in terms of some distance metric is identified via the "Nearest" function (line 3). The function N = Nearest($\mathcal{T}, x$) uses

Algorithm 9 CC-RRT*, Tree Expansion

1: Inputs: tree $\mathcal{T}$, current timestep t
2: $x_{\text{samp}} \leftarrow$ Sample()
3: $N_{\text{nearest}} \leftarrow \text{Nearest}(\mathcal{T}, x_{\text{samp}})$
4: $(\bar{\sigma}, \bar{\Pi}) \leftarrow \text{Steer}(\widehat{x}[N_{\text{nearest}}], P[N_{\text{nearest}}], x_{\text{samp}})$
5: **if** ProbFeas$(\bar{\sigma}, \bar{\Pi})$ **then**
6: Create node $N_{\text{min}}\{\bar{\sigma}, \bar{\Pi}\}$
7: $\mathcal{N}_{\text{near}} \leftarrow \text{Near}(\mathcal{T}, x_{\text{samp}}, |\mathcal{T}|)$
8: **for** $N_{\text{near}} \in \mathcal{N}_{\text{near}} \backslash N_{\text{nearest}}$ **do**
9: $(\bar{\sigma}, \bar{\Pi}) \leftarrow \text{Steer}(\widehat{x}[N_{\text{near}}], P[N_{\text{near}}], x_{\text{samp}})$
10: **if** ProbFeas$(\bar{\sigma}, \bar{\Pi})$ and $J[N_{\text{near}}] + \Delta J(\bar{\sigma}, \bar{\Pi}) < J[N_{\text{min}}]$ **then**
11: Replace N_{min} with new node $N_{\text{min}}\{\bar{\sigma}, \bar{\Pi}\}$
12: **end if**
13: **end for**
14: Add N_{min} to $\mathcal{T}$
15: **for** $N_{\text{near}} \in \mathcal{N}_{\text{near}} \backslash \text{Ancestors}(N_{\text{min}})$ **do**
16: $(\bar{\sigma}, \bar{\Pi}) \leftarrow \text{Steer}(\widehat{x}[N_{\text{min}}], P[N_{\text{min}}], \widehat{x}[N_{\text{near}}])$
17: **if** ProbFeas$(\bar{\sigma}, \bar{\Pi})$ and $J[N_{\text{min}}] + \Delta J(\bar{\sigma}, \bar{\Pi}) < J[N_{\text{near}}]$ **then**
18: Delete N_{near} from $\mathcal{T}$
19: Add new node $N_{\text{new}}\{\bar{\sigma}, \bar{\Pi}\}$ to $\mathcal{T}$
20: **end if**
21: **end for**
22: **end if**

the Euclidean norm metric via [56]

$$N_{\text{nearest}} = \text{Nearest}(\mathcal{T}, x) = \arg\min_{N \in \mathcal{T}} \|x - \widehat{x}[N]\|. \tag{5.4}$$

The Steering law "Steer" is then applied to steer the terminal state mean $\widehat{x}[N_{\text{nearest}}]$ to x_{samp} (line 4). The function $(\bar{\sigma}, \bar{\Pi}) = \text{Steer}(x, P, y)$ returns a sequence of state distributions, characterized by their means $(\bar{\sigma})$ and covariances $(\bar{\Pi})$, which originates at state x and terminates at state y. The state distributions must be dynamically feasible: the mean sequence $\bar{\sigma}$ must satisfy (3.71) with initial mean x, while the covariance sequence $\bar{\Pi}$ must satisfy (3.72) with initial covariance P. Previous work [56] dictates that a function be used of the form

$$\begin{aligned} z = \text{Steer}(x, y) \equiv \arg\min_{z} \quad & \|z - y\| \\ \text{s.t.} \quad & \|z - x\| \leq \mu, \end{aligned} \tag{5.5}$$

Algorithm 10 ProbFeas

1: Inputs: dynamically feasible K-time-step sequence of means $\bar{\sigma}$, covariances $\bar{\Pi}$
2: **for** $k = 1$ to K **do**
3: $(\widehat{x}, P) \leftarrow k$th element of $(\bar{\sigma}, \bar{\Pi})$
4: Compute $\Delta_t(\widehat{x}, P)$ using (3.59) and $\Delta(\widehat{x}, P)$ using (3.60)
5: **if** $\Delta(\widehat{x}, P) > 1 - \delta_p$ **or** $\Delta_t(\widehat{x}, P) > 1 - \delta_s$ **then**
6: **return** `false`
7: **end if**
8: **end for**
9: **return** `true`

for some prespecified $\mu > 0$. The formulation used here requires a full path to be generated from x to y when one exists, such that $z = y$ and this condition is automatically satisfied.

The resulting distribution sequence is then checked for probabilistic feasibility via the "ProbFeas" function (line 5). The Boolean function ProbFeas$(\bar{\sigma}, \bar{\Pi})$, detailed in Algorithm 10, returns `true` if the distribution sequence $(\bar{\sigma}, \bar{\Pi})$ satisfies the probabilistic feasibility conditions (3.63)-(3.64), and `false` otherwise. In particular, the subroutine iterates over all elements of $(\bar{\sigma}, \bar{\Pi})$ (Algorithm 10, line 2). For the kth element with mean $\widehat{x}$ and covariance P (line 3), $\Delta_t(\widehat{x}, P)$ and/or $\Delta(\widehat{x}, P)$ (depending on which chance constraints are being enforced) are computed (line 4). It is assumed that the values of Δ_t at previous timesteps are available for the computation of $\Delta(\widehat{x}, P)$, typically achieved by storing incremental values along each tree trajectory. If either value exceeds the maximum allowable risk (line 5), the subroutine returns `false` (line 6); if this does not occur for any state distribution, the subroutine returns `true` (line 9).

Returning to Algorithm 9, if $(\bar{\sigma}, \bar{\Pi})$ is probabilistically feasible (Algorithm 9, line 5), a new node with that distribution sequence is created (line 6), but not yet added to the tree $\mathcal{T}$. Instead, nearby nodes are identified for possible connections via the "Near" function (line 7). The function $\mathcal{N} = \text{Near}(\mathcal{T}, x, n)$ returns a subset of nodes $\mathcal{N} \subseteq \mathcal{T}$, and is considered a generalization of Nearest to return multiple nodes, not just one. To enable probabilistic asymptotic optimality guarantees, CC-RRT* uses

the implementation [57]

$$\mathcal{N} \;=\; \text{Near}(\mathcal{T}, x, n) \equiv \{N \in \mathcal{T} \mid \|\widehat{x}[N] - x\| \leq r_n\}, \tag{5.6}$$

where r_n is a radius that decreases with the number of tree nodes n, chosen to enforce asymptotic optimality conditions (Section 5.2).

Once the nearby nodes $\mathcal{N}$ are identified, CC-RRT* seeks to identify the lowest-cost, probabilistically feasible connection from those nodes to x_{samp} (Algorithm 9, lines 8–13). For each possible connection, a distribution sequence is simulated (line 9). If the resulting sequence is probabilistically feasible, and the cost of that node – represented as the sum $J[N_{\text{near}}] + \Delta J(\bar{\sigma})$, via (2.89) – is lower than the cost of N_{min} (line 10), then a new node with this sequence replaces N_{min} (line 11). The lowest-cost node is ultimately added to $\mathcal{T}$ (line 14). Lines 7–13 of Algorithm 9 are collectively referred to as the *connect-nearby* step.

Finally, a rewiring operation, known as the *rewire-nearby step* (Algorithm 9, lines 15–22) is performed based on trying connections from N_{min} to nearby nodes. A distribution sequence is sampled via the steering law from N_{min} to each nearby node N_{near} (line 16). If the resulting sequence is probabilistically feasible, and the cost of that node is lower than the cost of N_{near} (line 17), then the new distribution sequence is chosen to replace N_{near}. This rewiring is performed by first removing N_{near} from the tree $\mathcal{T}$ (line 18), then adding the new node to $\mathcal{T}$ in its stead (line 19).

The CC-RRT* algorithm's execution loop, which is performed at time intervals of Δt, is given by Algorithm 11. During each cycle, the objective of this algorithm is to identify the lowest-cost path in the tree that is still probabilistically feasible, and use the remaining time to grow the tree.

If new observations are available for the vehicle's current state and/or environment (such as movement of dynamic obstacles), these may be applied to the tree first, resulting in the update of the current root node N_{root} (Algorithm 11, line 3). If the terminal covariance $P[N_{\text{root}}]$ changes, it should be propagated through the rest of the tree $\mathcal{T}$. For the duration of the timestep, the tree is repeatedly expanded

Algorithm 11 CC-RRT*, Execution Loop

1: Initialize tree $\mathcal{T}$ with node $(\hat{x}_0, P_{x_0})$ for $t = 0$
2: **while** $\hat{x}_t \notin \mathcal{X}_{\text{goal}}$ **do**
3: Use observations, if any, to repropagate state distributions
4: **while** time remaining for this timestep **do**
5: Expand the tree by adding nodes (Algorithm 9)
6: **end while**
7: Identify path $\{N_{\text{root}}, \ldots, N_{\text{target}}\}$ that minimizes (5.1)
8: **if** no paths exist **then**
9: Apply safety action and **goto** line 17
10: **end if**
11: **for** each node $N\{\bar{\sigma}, \bar{\Pi}\}$ in path **do**
12: **if** ProbFeas$(\bar{\sigma}, \bar{\Pi})$ `false` **then**
13: Remove infeasible portion of path and **goto** line 7
14: **end if**
15: **end for**
16: Execute path
17: $t \leftarrow t + \Delta t$
18: **end while**

using Algorithm 9 (lines 4–6). Following this tree growth, the objective (5.1) is used to identify the lowest-cost path in the tree (line 7). In practice, only paths which terminate in the goal region $\mathcal{X}_{\text{goal}}$ are considered; if no such path exists, the path which terminates closest to the goal region (in terms of Euclidean distance) is selected.

Once a path is chosen, the path is re-checked for probabilistic feasibility [105] against the current constraints (lines 11–15) If this path is still probabilistically feasible, it is chosen as the current path to execute (lines 16). Otherwise (line 12), the portion of the path that is no longer probabilistically feasible is removed (line 13), and the process is repeated until either a probabilistically feasible path is found or the entire tree is pruned. If the latter case occurs (line 8), the system has no path to execute, and some "safety" motion primitive (*e.g.*, come to a stop) is applied to attempt to keep the vehicle in a safe state (line 9).

By using the RRT* rewiring mechanism with an exact steering law, all descendants of a rewired node remain dynamically feasible. As in that algorithm, the reduced path cost at the rewired node N_{new} should be propagated downward through all descendants. On the other hand, the terminal covariance/risk may change due to

rewiring, implying that it should also be propagated to – and probabilistic feasibility re-checked at – descendant nodes. Even if all descendant nodes remain feasible, it may still be desirable to update the descendant portion of the tree to ensure the risk bounds are accurate.

Because all uncertainty covariances can be computed and time-indexed off-line, the probabilistic feasibility re-checks can be computed efficiently. Additionally, under certain assumptions, uncertainty can be guaranteed to never increase due to rewiring, such that feasibility re-checks may not not necessary. This is used in the proof of CC-RRT* probabilistic completeness in Section 5.2 below. In practice, however, the algorithm remains effective even if full tree updates are not performed at every timestep.

5.2 Analysis

This section establishes the probabilistic completeness and asymptotic optimality of CC-RRT* under appropriate assumptions. Before proceeding further, however, several theoretical properties of solutions to problem (3.C), used in the context of CC-RRT*, must be established. First, the robust feasibility of CC-RRT* – already established via Theorem 3.3, since both algorithms use the same constraint set (3.61) – is re-stated.

Theorem 5.1 (Robust Feasibility of CC-RRT* Risk Evaluation). *If the path of state distributions* $(\widehat{x}_0, P_{x_0}), (\widehat{x}_1, P_{x_1}), \ldots, (\widehat{x}_{t_f}, P_{x_{t_f}})$ *specified by the input sequence* $u_0, u_1, \ldots, u_{t_f-1}$ *is feasible for problem (3.C) as sampled by CC-RRT*, then it is also feasible for problem (3.A).*

Proof. See proof of Theorem 3.3. ■

Lemma 5.2 (Feasibility of Distribution Mean). *Suppose* $(\widehat{x}_t,\ P_{x_t}, P_{c_{jt}})$ *is feasible for constraints (3.62)-(3.64) of problem (3.C), where* $\delta_s, \delta_p \in [1/2, 1]$. *Then* $\widehat{x}_t \in \mathcal{X}_t$.

Proof. Because the error function returns values in the open interval $(-1, +1)$, then from (3.55)-(3.56) $\Delta_{ijt}, \Delta_{i0t} \in (0, +1)$. Thus all terms in the left-hand sides of (3.57)-

(3.60) are positive.

From (3.63)-(3.64), if $\delta_s, \delta_p \in [1/2, 1]$, then $\Delta_t, \Delta \in (0, 1/2]$. Because all terms of (3.59) are positive, this implies that $\Delta_{0t} \leq 1/2$ and $\Delta_{jt} \leq 1/2 \; \forall \; j \in \mathbb{Z}_{1,n_o}$.

Because all terms of (3.58) are positive, then $\Delta_{0t} \leq 1/2$ implies that $\Delta_{i0t}(\widehat{x}_t, P_{x_t}) \leq 1/2$. Incorporating this inequality into (3.56) and rearranging yields

$$\begin{aligned} \text{erf}\left[\frac{a_{i0}^T(c_{i0} - \widehat{x}_t)}{\sqrt{2a_{i0}^T(P_{x_t})a_{i0}}}\right] \geq 0 \quad &\Rightarrow \quad \gamma_{i0t} \equiv \frac{a_{i0}^T(c_{i0} - \widehat{x}_t)}{\sqrt{2a_{i0}^T(P_{x_t})a_{i0}}} \geq 0 \qquad (5.7) \\ &\Rightarrow \quad a_{i0}^T(c_{i0} - \widehat{x}_t) \geq 0 \\ &\Rightarrow \quad a_{i0}^T(\widehat{x}_t - c_{i0}) \leq 0 \quad \forall \; i \in \mathbb{Z}_{1,n_E} \\ &\Rightarrow \quad \widehat{x}_t \in \mathcal{X}. \qquad (5.8) \end{aligned}$$

Consider (3.57) for each $j \in \mathbb{N}_{1,n_o}$. Since $\Delta_{jt} \leq 1/2$, there exists at least one $i^* \in 1, \ldots, n_j$ such that $\Delta_{i^*jt}(\widehat{x}_t, P_{x_t}, P_{c_{jt}}) \leq 1/2$. Incorporating this inequality into (3.55) and rearranging yields

$$\begin{aligned} \text{erf}\left[\frac{a_{i^*j}^T(\widehat{x}_t - c_{i^*jt})}{\sqrt{2a_{i^*j}^T(P_{x_t} + P_{c_{jt}})a_{i^*j}}}\right] \geq 0 \quad &\Rightarrow \quad \gamma_{i^*jt} \equiv \frac{a_{i^*j}^T(\widehat{x}_t - c_{i^*jt})}{\sqrt{2a_{i^*j}^T(P_{x_t} + P_{c_{jt}})a_{i^*j}}} \geq 0 \qquad (5.9) \\ &\Rightarrow \quad a_{i^*j}^T(\widehat{x}_t - c_{i^*jt}) \geq 0 \\ &\Rightarrow \quad \widehat{x}_t \notin \mathcal{X}_{jt}, \qquad (5.10) \end{aligned}$$

where the last relationship is implied by the inequality satisfying the jth obstacle's disjunction of constraints. Together, (5.8) and (5.10) imply (2.5), i.e. $\widehat{x}_t \in \mathcal{X}_t$. ■

Lemma 5.3 (Feasibility under Relaxed Constraints). *Suppose* $(\widehat{x}_t, P_{x_t}, P_{c_{jt}})$ *is feasible for constraints (3.62)-(3.64) of problem (3.C), where* $\delta_s, \delta_p \in [1/2, 1]$. *Consider* $(\widehat{x}_t, P'_{x_t}, P'_{c_{jt}})$ *where* $P'_{x_t} \leq P_{x_t}$ *and* $P'_{c_{jt}} \leq P_{c_{jt}} \; \forall \; j \in \mathbb{Z}_{1,n_o}$. *Then* $(\widehat{x}_t, P'_{x_t}, P'_{c_{jt}})$ *is feasible for constraints (3.62)-(3.64) of problem (3.C).*

Proof. From Lemma 5.2, it is known that $\widehat{x}_t \in \mathcal{X}_t$. For obstacle j, denote the indices of the obstacle inequality constraints that are *not* satisfied by $\widehat{x}_t$ as $\mathcal{I}_j = \left\{i \in \mathbb{Z}_{1,n_j} \mid a_{ij}^T(\widehat{x}_t - c_{ijt}) \geq 0\right\}$; since $\widehat{x}_t \notin \mathcal{X}_j$, $\mathcal{I}_j$ must be non-empty. This implies

that γ_{i0t} (as defined in (5.7)) is always positive, and γ_{ijt} (as defined in (5.9)) is always positive for $i \in \mathcal{I}_j$.

In moving from P_{x_t}, P_{c_t} to P'_{x_t}, P'_{c_t}, since $P'_{x_t} \leq P_{x_t}$ and $P'_{c_{jt}} \leq P_{c_{jt}}$, the denominators of γ_{i0t} and γ_{ijt} (for $i \in \mathbb{I}_j$) are non-increasing. Since the numerators are positive, this implies that γ_{i0t} and γ_{ijt} (for $i \in \mathbb{I}_j$) are positive and non-decreasing. Rewrite (3.55) and (3.56) as

$$\Delta_{ijt}(\widehat{x}_t, P_{x_t}, P_{c_{jt}}) = \frac{1}{2}\left(1 - \operatorname{erf}[\gamma_{ijt}]\right), \tag{5.11}$$

$$\Delta_{i0t}(\widehat{x}_t, P_{x_t}) = \frac{1}{2}\left(1 - \operatorname{erf}[\gamma_{i0t}]\right). \tag{5.12}$$

Recall that the error function is monotonically increasing, taking values in $(-1, 0)$ for negative inputs and $(0, 1)$ for positive inputs. If γ_{i0t} and γ_{ijt} (for $i \in \mathcal{I}_j$) are positive and non-decreasing, this implies that Δ_{ijt} (for $i \in \mathcal{I}_j$) and Δ_{i0t} are non-increasing and within the range $(0, 1/2)$.

For $i \notin \mathcal{I}_j$, the numerator of γ_{ijt} is negative, implying that γ_{ijt} is negative. Thus, in view of (5.11) and the discussion above, $\Delta_{ijt} \in (1/2, 1)\ \forall\ i \notin \mathcal{I}_j$. As a result, when considering the minimization in (3.57), values of Δ_{ijt} with $i \notin \mathcal{I}_j$ will always be dominated by values of Δ_{ijt} with $i \in \mathcal{I}_j$, for which $\Delta_{ijt} \in (0, 1/2)$. Thus (3.57) can be rewritten as

$$\Delta_{jt}(\widehat{x}_t, P_{x_t}, P_{c_{jt}}) = \min_{i \in \mathcal{I}_j, i=1,\ldots,n_j} \Delta_{ijt}(\widehat{x}_t, P_{x_t}, P_{c_{jt}}). \tag{5.13}$$

If Δ_{ijt} (for $i \in \mathcal{I}_j$) and Δ_{i0t} are non-increasing, then Δ_{jt} and Δ_{0t} are non-increasing by (3.58) and (5.13). This, in turn, implies that $\Delta_t(\widehat{x}_t, P_{x_t}, P_{c_{jt}})$ is non-increasing by (3.59), and $\Delta(\widehat{x}_t, P_{x_t}, P_{c_{jt}})$ is non-increasing by (3.60).

Finally, it is given that the values of Δ_t and Δ for $(\widehat{x}_t, P_{x_t}, p_{c_{jt}})$ satisfy constraints (3.62)-(3.64) of problem (3.C). Since Δ_t and Δ have been shown to be non-increasing, the values of Δ_t and Δ for $(\widehat{x}_t, P'_{x_t}, P'_{c_{jt}})$ must be the same or smaller $\forall\ t$ – implying that constraints (3.62)-(3.64) of problem (3.C) are still satisfied. ■

Additional assumptions are utilized for CC-RRT* probabilistic completeness, due

to the potential of its connect-nearby and rewire-nearby steps (Section 5.1) to modify the uncertainty levels at tree nodes.

Assumption 5.4. *All of the following conditions are satisfied:*

1. *All obstacles are static, such that* $\widehat{c}_{jt} \equiv \widehat{c}_j \ \forall \ t$ *and* $\mathcal{X}_t$ *is the same for all* t.
2. *In the objective function (3.61) of problem (3.C),* $\phi_f \equiv 0$ *and* $\phi \equiv dt$.
3. *For* $t_1 \leq t_2$, $P_{x_{t_1}} \leq P_{x_{t_2}}$ *and* $P_{c_{jt_1}} \leq P_{c_{jt_2}}$.

Assumption 5.4.1 does *not* necessarily imply that $P_{c_{jt}} \equiv P_{c_j} \ \forall \ t$. Assumption 5.4.2 implies that the objective function seeks to minimize path duration, subject to the input constraints (3.62) and probabilistic state constraints (3.63)-(3.64).

Finally, Assumption 5.4.3 is in place to ensure that the covariance of a node cannot increase due to either the connect-nearby or rewire-nearby step in CC-RRT*. This assumption is typically not restrictive in practice; further, the algorithm typically performs well even if several of the above theoretical assumptions are not satisfied.

Theorem 5.5 (Probabilistic Completeness of CC-RRT*). *Suppose Assumptions 3.4 and 5.4 are satisfied. Then CC-RRT* is probabilistically complete for problem (3.C).*

Proof. CC-RRT* performs two additional steps in its tree expansion routine relative to CC-RRT (Section 3.3.2): the connect-nearby step (Algorithm 9, lines 7–13) and the rewire-nearby step (Algorithm 9, lines 15-22).

In view of both Assumption 3.4.3 and Theorem 3.5, the state distribution sequence $\{(\widehat{x}_0, P_{x_0}), (\widehat{x}_1, P_{x_1}), \ldots, (\widehat{x}_{t_f}, P_{x_{t_f}})\}$ has a non-zero likelihood of being contained within a CC-RRT tree as the number of samples approaches infinity. If the same sample sequence is applied in CC-RRT*, then its tree will contain the same set of state distributions $((\widehat{x}_0, P_{x_0}), (\widehat{x}_1, P_{x_1}), \ldots, (\widehat{x}_{t_f}, P_{x_{t_f}}))$ as the number of samples approaches infinity.

For completeness, it is sufficient to show that neither the connect-nearby nor rewire-nearby steps in CC-RRT* can *increase* the covariances of any tree nodes.

Indeed, suppose that some vertex $(\widehat{x}_j, P_{x_j})$ of the path specified in Assumption 3.4.3 is modified by one of these steps, such that the (feasible) path from $\widehat{x}_0$ to $\widehat{x}_j$ may differ from $\{(\widehat{x}_0, P_{x_0}), (\widehat{x}_1, P_{x_1}), \ldots, (\widehat{x}_j, P_{x_j})\}$. If these steps cannot increase covariances, then the covariances at descendants of $\widehat{x}_j$ either remain the same or are reduced. By Lemma 5.3, the remaining path segment $\{(\widehat{x}_j, P_{x_j}), (\widehat{x}_{j+1}, P_{x_{j+1}}), \ldots, (\widehat{x}_{t_f}, P_{x_{t_f}})\}$ remains feasible, such that probabilistic completeness is maintained.

Via Assumption 5.4.2, *i.e.*, $f(\cdot) \equiv 1$ in (5.1), i.e. path cost equals path duration. As a result, either connect-nearby or rewire-nearby will strictly decrease the terminal timestep t of any affected node and its descendants. By Assumption 5.4.3, the covariances will also decrease; coupled with Lemma 5.3, this completes the proof. ■

The proof of asymptotic optimality follows a similar path to the proof in Karaman and Frazzoli [56]. However, because the feasible state space is a function of the planning timestep, the definitions must be modified accordingly.

Denote $\mathbb{X}_t$ as the set of all states $\widehat{x}_t$ satisfying constraints (3.63)-(3.64) at planning timestep t, with corresponding state covariance P_{x_t}. A state $x \in \mathbb{X}_t$ is a δ-interior state of $\mathbb{X}_t$ if the closed ball of radius δ centered at x lies entirely inside $\mathbb{X}_t$. The δ-interior of $\mathbb{X}_t$ is defined as $\text{int}_\delta(\mathbb{X}_t) \equiv \{x \in \mathbb{X}_t \mid \mathbb{B}(x;\delta) \subseteq \mathbb{X}_t\}$, where $\mathbb{B}(x;r)$ denotes a ball of radius r centered at x. A robustly feasible path $\{(\widehat{x}_0, P_{x_0}), \ldots, (\widehat{x}_{t_f}, P_{x_{t_f}})\}$ is said to have strong δ-clearance if $\widehat{x}_t \in \text{int}_\delta(\mathbb{X}_t)\ \forall\ t \in \mathbb{Z}_{0,t_f}$. A robustly feasible path $p = \{(\widehat{x}_0, P_{x_0}), \ldots, (\widehat{x}_{t_f}, P_{x_{t_f}})\}$ is said to have weak δ-clearance if there exists a path $p' = \{(\widehat{x}'_0, P_{x_0}), \ldots, (\widehat{x}'_{t_f}, P_{x_{t_f}})\}$ with strong δ-clearance, and a homotopy ψ with $\psi(0) = p$, $\psi(1) = p'$, and for all $\alpha \in (0, 1]$ there exists $\delta_\alpha > 0$ such that $\psi(\alpha)$ has strong δ_α-clearance [56].

Denote the cost of path p via objective (3.61) of problem (3.C), with Assumption 5.4.2, as $J(p)$. The cost function is assumed to be admissible as established in Ref. [57] and discussed further below (Section 5.2.1) – this is clearly the case for $\phi \equiv dt$ as assumed here. A path p^* that solves problem (3.C) is considered a robustly optimal solution if p^* has weak δ-clearance and, for any sequence of robustly feasible paths $\{p_n\}_{n \in \mathbb{Z}_{0,\infty}}$ with $\lim_{n\to\infty} p_n = p^*$, then $\lim_{n\to\infty} c(p_n) = c(p^*)$ [56].

Assumption 5.6. *All of the following conditions are satisfied:*

1. *The set of all points traversed by an optimal trajectory has measure zero [56].*
2. *The nearby node radius r used in line 7 of Algorithm 9 is a function of the number of tree nodes, N, and is chosen to be*

$$\begin{aligned} r_n &= \min\left\{\gamma\left(\frac{\log n}{n}\right)^{1/s_x}, \eta\right\}, \\ \gamma &> \gamma^* \equiv \left(2\left(1+\frac{1}{s_x}\right)\frac{\mu(\mathbb{X}_0)}{\zeta_{s_x}}\right)^{1/s_x}, \end{aligned}$$

where s_x is the dimension of the state space, $\mu(S)$ is the volume of set S, ζ_d is the volume of the d-dimensional unit sphere, and $\eta > 0$.

The definition of γ^* used by Karaman and Frazzoli [56] includes $\mu(\mathcal{X}_{\text{free}})$, where $\mathcal{X}_{\text{free}}$ is the volume of the collision-free space, rather than $\mu(\mathbb{X}_0)$. However, since $\mathbb{X}_0$ implies the state constraints at $t = 0$, where no tightening is yet needed for robustness, the formulations are equivalent.

Theorem 5.7 (Asymptotic Optimality of CC-RRT*). *Suppose Assumptions 3.4, 5.4, and 5.6 are satisfied. Then CC-RRT* is asymptotically optimal for problem (3.C). In other words, for any realization of problem (3.C) that admits a robustly optimal solution with finite cost J^*, then*

$$\mathbb{P}\left(\left\{\lim_{n\to\infty}\sup J_n = J^*\right\}\right) = 1,$$

where J_n denotes the cost of the lowest-cost path in the CC-RRT tree after n tree expansion steps.*

Proof. The proof of asymptotic optimality established in Theorem 38 of Karaman and Frazzoli [56] can also be utilized here, with minor modification. As is the case there, it is assumed that η is chosen to be sufficiently large. Additionally, Lemma 5.3 and the resulting Theorem 5.5 establish that the graph construction process in the proof of Theorem 38 [56] maintains probabilistic completeness.

The primary remaining task in this modified proof is to show that the choice of nearby node radius in Assumption 5.6.2 is valid. To fit into the framework of Theorem 38 [56], the choice of γ at each planning timestep t must satisfy

$$\gamma > \gamma_t \equiv \left(2\left(1+\frac{1}{s_x}\right)\frac{\mu(\mathbb{X}_t)}{\zeta_{s_x}}\right)^{1/s_x}.$$

However, under the assumptions of Theorem 5.5, it is straightforward to show that $\mu(\mathbb{X}_t)$ is non-decreasing in t, as each subsequent planning timestep further constrains the space of feasible conditional means. As such, choosing $\gamma > \gamma^*$ for $r(N)$ implies that $\gamma > \gamma_t$, validating this approach to proving asymptotic optimality for problem (3.C). ■

5.2.1 Risk-based Objective

The alternative cost function presented in this section is a novel feature of this work, as it explicitly incorporates the risk of constraint violation inherent to chance constraints. Using (5.1), the cost function takes the form

$$f(\widehat{x}_t, P_t) = C_T + C_R\Delta_t(\widehat{x}_t, P_t) + C_M \max_{i=\{0,\ldots,t\}}\left(\Delta_t(\widehat{x}_t, P_t)\right), \tag{5.14}$$

where $C_T \geq 0$, $C_R \geq 0$, $C_M \geq 0$, $C_T C_R C_M > 0$. The cost component with coefficient C_T seeks to minimize time/path duration. The cost component with coefficient C_R represents the accumulated risk across all timesteps, as measured by the risk bound $\Delta_t(\widehat{x}_t, P_t)$. Finally, the cost component with coefficient C_M penalizes the maximum risk bound encountered at any timestep along the path.

It is now shown that, in the context of Ref. [57], the cost function (5.14) is an admissible cost heuristic for RRT*, and thus CC-RRT*.

Theorem 5.8 (Risk-based Objective Admissibility). *The function (5.14) is an admissible cost heuristic for CC-RRT*.*

Proof. Using the framework established in the work of Karaman and Frazzoli [57],

there are three additional conditions that must be satisfied by (5.14) to establish admissibility within RRT*, and thus CC-RRT*. To simplify notation, this proof uses simply $\bar{\sigma}$ to denote a path segment and $\Delta J(\bar{\sigma})$ its cost, rather than $(\bar{\sigma}, \bar{\Pi})$ and $\Delta J(\bar{\sigma}, \bar{\Pi})$ from (5.2).

First, the cost must be monotonic; that is, if $\bar{\sigma}_1|\bar{\sigma}_2$ denotes the concatenation of path segments $\bar{\sigma}_1$ and $\bar{\sigma}_2$, then $\Delta J(\bar{\sigma}_1) \leq \Delta J(\bar{\sigma}_1|\bar{\sigma}_2)$. The cost function (5.1) with (5.14) is a time integral of a quantity which is always non-negative; thus the cost is monotonic.

Second, the cost must be additive; that is, $\Delta J(\bar{\sigma}_1|\bar{\sigma}_2) = \Delta J(\bar{\sigma}_1) + \Delta J(\bar{\sigma}_2)$. The cost function is also clearly additive, as the cost function over any path segment can always be decomposed into the sum of cost functions over partitions of the path segment, by splitting the timesteps. (Finding a higher-risk state further down a path does not retroactively increase the value of the maximum risk at previous timesteps.)

Third, the cost must be Lipschitz continuous: there exists some κ such that

$$|\Delta J(\bar{\sigma}_1) - \Delta J(\bar{\sigma}_2)| \leq \kappa \sup_{\tau \subset [0,1]} \|\sigma_1(\tau) - \sigma_2(\tau)\|_2$$

for all paths σ_1 and σ_2, parameterized by τ. The C_T-component of the cost function is clearly Lipschitz continuous. For the C_R-component, smooth shifts in the state uncertainty distribution yield smooth variations on the risk of collision with a bounded slope via (3.55)-(3.56) and thus are also Lipschitz continuous. The C_M-component is essentially a maximum operator on the C_R-component, so it too is Lipschitz continuous.

As all three conditions are met, (5.14) is an admissible heuristic. ■

Theorem 5.8 implies that the risk-based objective (5.14) provides an alternate means of converging toward optimal paths within the space of feasible solutions identified by CC-RRT*, utilizing knowledge of the risk environment.

In the results that follow, two distinct variations of CC-RRT* are considered, depending on whether or not (5.14) is applied (with positive values of C_R and/or C_M). The variant of CC-RRT* in which the risk-based objective is used is referred to

as CC-RRT*-Risk. As demonstrated below, each variant has its own strengths and weaknesses which should be considered carefully.

5.3 Simulation Results

This section presents simulation results which demonstrate the effectiveness of the CC-RRT* algorithm in efficiently identifying smooth, robust trajectories subject to both internal and external uncertainties. Applying CC-RRT* with a time-based objective (*e.g.*, $C_R, C_M = 0$) is shown to generate trees of trajectories satisfying all robustness chance constraints, yielding an asymptotically optimal trajectory that is both probabilistically and dynamically feasible. As the likelihood of constraint violation tends to increase with proximity to obstacles, the chance constraints will typically be active in the final trajectory, with solutions often approaching the maximum allowable risk. Alternatively, by incorporating measures of risk within the objective (*e.g.*, $C_R, C_M > 0$), the resulting trajectories are shown to demonstrate more risk-averse behavior, avoiding riskier actions unless deemed necessary to reduce traversal time (as determined by the relative weights of the cost coefficients). The hard probabilistic constraints are still imposed in this case; however, as shown the subsequent results, the tree shape is significantly reconfigured to minimize time in risky regions as risk-based penalties become more prominent.

Six variants of the RRT algorithm are compared throughout this section:

1. Nominal **RRT**, with a min-time objective;
2. **RRT***, with a min-time objective;
3. **CC-RRT**, with a min-time objective;
4. **CC-RRT-Risk**, which utilizes the objective (5.14) with $C_T, C_R, C_M > 0$ but is otherwise identical to CC-RRT;
5. **CC-RRT***, with a min-time objective (*i.e.*, $C_T > 0$, $C_R = C_M = 0$ in (5.14)); and

6. **CC-RRT*-Risk**, which utilizes the objective (5.14) with $C_T, C_R, C_M > 0$.

Simulation results are presented for a variety of dynamics, environments, and uncertainty conditions. First, results are provided for the case of a 2D single integrator subject to uncertain localization, process noise, and obstacles with placement uncertainty. Examples of the typical trees and solutions resulting from each algorithm are given, illustrating key differences in their operation, as well as how the tree structure changes as the cost objective, uncertainty environment, δ_s and/or δ_p is modified. On another environment, extensive simulation trials are performed, providing statistical data on the duration, risk, and computation of paths yielded by each algorithm, including how those parameters vary as a function of tree size. Finally, the algorithm is tested for more complex dynamics and scenarios, including double integrator dynamics, Dubins vehicles, and pursuit-evasion games.

5.3.1 Illustrative Scenario

Consider the 2D single integrator dynamics

$$\begin{aligned} x_{t+1} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} dt & 0 \\ 0 & dt \end{bmatrix} u_t + \begin{bmatrix} dt & 0 \\ 0 & dt \end{bmatrix} w_t, \\ x_t &= \begin{bmatrix} p_t^x & p_t^x \end{bmatrix}^T, \\ u_t &= \begin{bmatrix} v_t^x & v_t^y \end{bmatrix}^T, \end{aligned}$$

where $dt = 0.1$ s. The position variables (p_t^x, p_t^y) are constrained within a bounded, two-dimensional environment, containing four obstacles with uncertain placement (Figure 5-1). The velocity inputs $u_t = (u_t^x, u_t^y)$ are modeled as a linear approximation of a 2-norm constraint $\|u_t\|_2 \leq \bar{v} = 0.5$ m/s, taking the form

$$\cos\left(\frac{2\pi d}{D_U}\right) u_t^x + \sin\left(\frac{2\pi d}{D_U}\right) u_t^y \leq \bar{v}, \ \forall\, d \in \mathbb{Z}_{1,D_U}, \ \forall\, t,$$

where $D_U = 36$ is the discretization level.

The system is subject to three forms of uncertainty. First, the initial state x_0 is

subject to localization error (3.2) more prominent in the y-direction,

$$x_0 \in \mathcal{N}(\widehat{x}_0, P_{x_0}^{(1)}), \quad P_{x_0}^{(1)} = 10^{-5} \begin{bmatrix} 5 & 0 \\ 0 & 30 \end{bmatrix}.$$

At each timestep, the system is subject to process noise (3.1) more prominent in the x-direction,

$$w_t \in \mathcal{N}(0, P_w^{(1)}), \quad P_w^{(1)} = 10^{-5} \begin{bmatrix} 30 & 0 \\ 0 & 5 \end{bmatrix}.$$

Finally, the placement of each obstacle is itself uncertain. At all timesteps, the displacement of the jth obstacle is governed by (3.4), where

$$c_{jt} \in \mathcal{N}(\widehat{c}_{jt}, P_{c_j}^{(1)}), \quad P_{c_j}^{(1)} = \begin{bmatrix} \sigma_j & 0 \\ 0 & \sigma_j \end{bmatrix}, \quad \forall\, t$$

and $\sigma_j > 0$. In this environment (Figure 5-1 – obstacles are placed at their means $\widehat{c}_{jt}$), $\sigma_j = 0.2$ for the upper-left obstacle, $\sigma_j = 0.1$ for the bottom-right obstacle, and $\sigma_j = 0.001$ for the other two obstacles. Thus, the upper-left obstacle is the most uncertain, followed by the bottom-right obstacle. The system is required to satisfy a minimum probability of constraint violation at each timestep of $\delta_s = 0.9$; no path-wise probability bound is imposed (*i.e.*, $\delta_p = 0$). Note that the state uncertainty will grow without bound over time, ensuring that Assumption 5.4.3 is satisfied.

The 2D position is sampled uniformly within the bounds of the 2D environment, with any positions that are in collision with the nominal obstacle placements being excluded. The steering law draws a line connecting the old and new positions; the system traverses this line at speed $\bar{v}$. A path is considered to reach the goal is the final position is within 0.25m of the goal location; the lowest-cost path about those paths meeting this criterion is selected for execution. In the absence of any such path, the path bringing the system closest to the goal is used. Finally, the nearby node function uses a maximum radius $\mu = 1$ m.

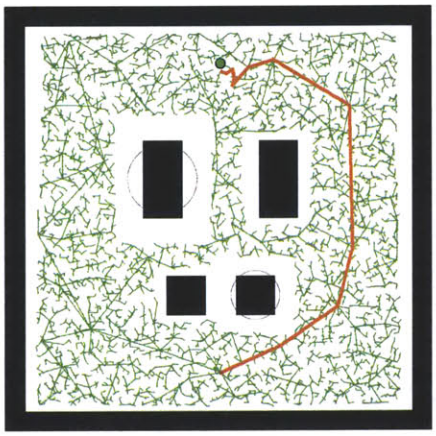
Figure 5-1 shows typical trees and solution paths returned by each algorithm after 5000 nodes of tree growth (via Algorithm 9). The objective is to plan a path from the start (brown dot) to the goal (lime green circle); the minimum-cost path after 5000 nodes is shown in orange. The 2-σ uncertainty ellipse is shown for the placement uncertainty of each obstacle.

As expected, the RRT-based algorithms (Figures 5-1(a), 5-1(c), and 5-1(e)) generate trees which are relatively random and unorganized in their path structure, yielding non-smooth paths. As RRT-based algorithms seek only to connect the nearest node(s) in the tree to each sample, without any rewiring, most node path segments in the tree are relatively short. In contrast, the RRT*-based algorithms (Figures 5-1(b), 5-1(d), and 5-1(f)) attempt to rewire the tree to reduce path costs every time a new sample is added. The resulting trees will then generate paths from the root (brown dot) which minimize the cost of all paths from the root node, not just the one ultimately selected. This yields a much more organized tree with longer node path segments. Thus the cost-minimizing paths identified by each tree tend to be relatively smooth, as demonstrated in the images.

Both RRT (Figure 5-1(a)) and RRT* (Figure 5-1(b)) identify short paths to the goal which take the system between all obstacles. However, both algorithms do not consider the risk of constraint violation, and thus are likely to select risky behaviors in order to minimize path duration. In particular, the uncertainty in the placement of the bottom-right obstacle presents a high chance of collision for the system as it passes nearby to the left. The robust algorithms, however, only add trajectories to the tree as long as they satisfy the time-step-wise chance constraint (3.5) with $\delta_s = 0.8$, as enforced via chance constraints (Figures 5-1(c) through 5-1(f)). A standoff distance containing no (probabilistically feasible) tree paths can be seen around each obstacle and the environment boundaries, reflecting regions where the cumulative effect of the internal and obstacle uncertainties violates the probabilistic feasibility constraint. This buffer is larger for the more uncertain obstacles, and increases with distance from the starting position as process noise accumulates – for example, compare the standoff from the bottom-left and upper-right obstacles. In all cases, no probabilistically

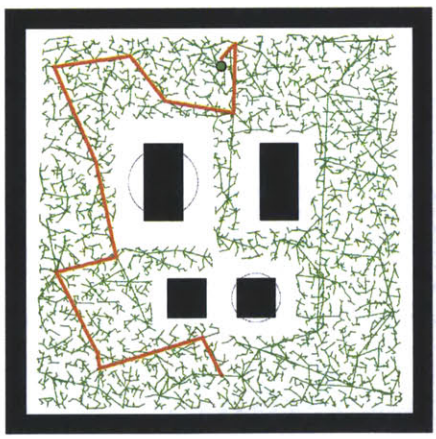
(a) RRT

(b) RRT*



(c) CC-RRT

(d) CC-RRT*



(e) CC-RRT-Risk

(f) CC-RRT*-Risk

Figure 5-1: Demonstrative 5000-node CC-RRT* trees and minimum-cost paths, 2D single integrator, simple scenario

feasible path exists that takes the system between the two lower obstacles, implying the paths chosen by RRT and RRT* violate probabilistic feasibility.

CC-RRT* (Figure 5-1(d)) identifies a minimum-cost solution path which takes the system around both lower obstacles to the left, to avoid the uncertain bottom-right obstacle. It then minimizes the cost of the remaining path to the goal, without bringing it close enough to the highly uncertain upper-left obstacle to violate probabilistic feasibility bounds. Such a path would also be feasible for CC-RRT (Figure 5-1(c)); however, such a path is unlikely to be sampled. For example, in Figure 5-1(c)), the minimum-time path to goal passes around all obstacles to the right, even though a path between them would have been probabilistically feasible if identified.

Both CC-RRT* and CC-RRT*-Risk are subject to the same probabilistic constraints; however, the shapes of both the resulting trees and solution paths are significantly affected by the use of different cost objectives. Whereas CC-RRT* seeks the shortest path subject to the robustness constraints (Figure 5-1(d)), CC-RRT*-Risk instead identifies a path which passes around all obstacles to the right, by a significant distance (Figure 5-1(f)). A particularly large distance is maintained relative to the bottom-right obstacle, while the upper-left obstacle is avoided entirely. Though the resulting path is similar to the path generated in the CC-RRT example (Figure 5-1(c)), the path in Figure 5-1(f) is the result of a rewiring process to minimize the cost, with asymptotically optimal convergence. Repeated executions of the algorithm for this scenario would yield very similar final paths (Section 5.3.2). (Using the risk-based objective function has little effect on CC-RRT, as observed by CC-RRT-Risk in Figure 5-1(e).)

In comparing CC-RRT* to CC-RRT*-Risk, each tree has been rewired in a different way, corresponding to the different cost functions, yielding trees with significant qualitative distinctions. For example, paths passing around obstacles in the CC-RRT* tree (Figure 5-1(d)) tend to travel parallel to the obstacle surfaces in order to minimize path duration, since this algorithm is only concerned with satisfying the minimum time-step feasibility δ_s. On the other hand, paths passing around obstacles in the CC-RRT*-Risk tree (Figure 5-1(f)) tend to travel *perpendicular* to the obstacle

(a) CC-RRT* (b) CC-RRT*-Risk

Figure 5-2: Figures 5-1(d) and 5-1(f), with homotopic boundaries marked

surfaces, in order to minimize the time spent by trajectories in higher-risk (and thus higher-cost) regions.

Of particular note are the "homotopic boundaries" of each tree: those boundaries separating portions of the tree that pass around obstacles in each direction (and thus belonging to different homotopies). Such boundaries do not appear in RRT-based trees, which do not optimize over time via rewiring. Figure 5-2 reproduces Figures 5-1(d) and 5-1(f) with homotopic boundaries marked in blue. The boundaries of the CC-RRT* tree (Figure 5-2(a)) largely follow the Voronoi minimum-distance boundaries; there is a slight preference for passing around the lower obstacles on the left, as passing on the right requiring maintaining a wider berth around the more uncertain bottom-right obstacle. For CC-RRT*-Risk, these boundaries have shifted significantly. For example, along the top of the environment (*i.e.*, above all obstacles), nearly all tree paths approach from the right-hand side.

Governing these changes is the trade-off between minimizing path length and minimizing risk, as dictated by the cost coefficients in the objective. Consider the ratio of cost coefficients $\gamma = C_R/C_T$, where it assumed[1] that $C_M = C_R$; Figure 5-3

[1]In empirical simulation results, performance was nearly identical in all cases where at least one

shows how the resulting CC-RRT*-Risk trees evolve as γ is varied. For low values of γ (*e.g.*, Figure 5-3(a)), the resulting tree behavior is essentially the same as CC-RRT*, for which $\gamma = 0$. As γ is increased (Figure 5-3(b)) toward $10^0 = 1$ (Figure 5-3(c)), the solution path slowly increases its distance from the upper-left obstacle. Additionally, the upper-right homotopic boundary can be seen to moving to the left. When γ is increased to 10^1 (Figure 5-3(d)), this boundary sweeps past the goal, causing the solution path to shift toward passing all obstacles on the outside. The set of paths between the two upper obstacles is also approached from the right, rather than from the left for lower values of γ, as the risk of nearing the upper-left obstacle becomes more significant. This effect becomes more prevalent as γ increases further (Figures 5-3(e) and 5-3(f)).

Regardless, it is essential that $C_T > 0$; including a cost term which minimizes path duration acts as a regularization parameter for CC-RRT*, especially in low-risk regions. Figure 5-4 shows the trees and paths that typically result when $C_T = 0$. In regions where the risk bounds are near zero, tree behavior essentially reverts to the unorganized nature of RRT, resulting in risk-averse paths that are often significantly less smooth.

Figure 5-5 provides several more images of trees generated using CC-RRT*-Risk which demonstrate the effect of the uncertainty environment, δ_s, and/or δ_p. Figure 5-5(a) shows a 500-node tree with visible $2-\sigma$ uncertainty ellipses corresponding to localization and process noise for each tree node; the accumulation of uncertainty over time due to the process noise is clear. This effect is even more pronounced in Figure 5-5(b), in which the covariances of the localization uncertainty $P_{x_0}^{(1)}$ and model uncertainty $P_w^{(1)}$ have each been scaled up by a factor of 10. The distance the system must maintain from the room boundaries clearly increases as paths move toward the goal. Additionally, due to the increased uncertainty, paths that pass between the obstacles are no longer probabilistically feasible.

Figure 5-5(c) shows the resulting tree and solution path for CC-RRT*-Risk when the time-step-wise feasibility bound is changed from $\delta_s = 0.9$ to $\delta_s = 0.999$, respec-

of C_M and C_R were positive, regardless of which specific coefficients were positive.

(a) $\gamma = 10^{-5}$

(b) $\gamma = 10^{-3}$

(c) $\gamma = 10^{0}$

(d) $\gamma = 10^{1}$

(e) $\gamma = 10^{3}$

(f) $\gamma = 10^{5}$

Figure 5-3: Demonstrative 5000-node CC-RRT*-Risk trees for various cost ratios γ

(a) $C_R = 10^5, C_M = 0$ (b) $C_R = C_M = 10^5 (\gamma = \infty)$

Figure 5-4: Demonstrative 5000-node CC-RRT*-Risk trees with $C_T = 0$

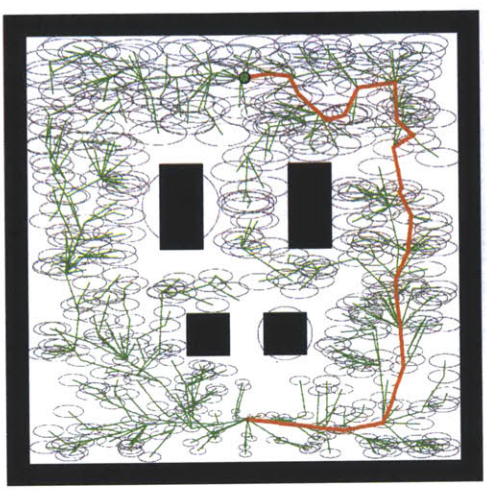
tively. While the size and shape of the trees varies considerably, relative to Figure 5-1(f) (where $\delta_s = 0.9$), the final path selected is nearly identical. This final path invariance is due to the risk-based cost objective encouraging risk-averse behavior, even as the hard probabilistic feasibility constraints are varied.

Finally, Figure 5-5(d) adds a path-wise probabilistic feasibility bound of $\delta_p = 0.9$, in addition to a reduced time-step-wise bound $\delta_s = 0.5$. Due to the additional chance constraint (3.6), the space of feasible solutions here is significantly reduced compared to $\delta_s = 0.9$ (Figure 5-1(f)), as expected.

5.3.2 Simulation Trials

Consider the same single integrator dynamics of Section 5.3.1 now applied to a different environment containing four obstacles (Figure 5-7). The environment is 37 feet (11.3 m) long and 18 feet (5.5 m) wide, while the upper and lower corridors are 2.5 feet (0.76 m) wide each. This will be referred to henceforth as the full-corridor scenario.

This environment is geometrically symmetric along the centerline of its long axis; however, the uncertainty environment is *not* symmetric. In this environment, only



(a) $2-\sigma$ uncertainty ellipses shown for tree (500 nodes)

(b) $P_{x_0}^{(1)}, P_w^{(1)} \times 10$ (5000 nodes)

(c) $\delta_s = 0.999$ (5000 nodes)

(d) $\delta_s = 0.5$, $\delta_p = 0.9$ (5000 nodes)

Figure 5-5: Demonstrative CC-RRT*-Risk trees for various values of δ_s, δ_p, $P_{x_0}^{(1)}$, and $P_w^{(1)}$

the bottommost obstacle has placement uncertainty,

$$c_{jt} \in \mathcal{N}(\widehat{c}_{jt}, P_{c_j}^{(2)}), \quad P_{c_j}^{(2)} = \begin{bmatrix} 0.05 & 0 \\ 0 & 0.05 \end{bmatrix}, \quad \forall\, t;$$

all other obstacle locations are known precisely. The system is additionally subject to the same process noise as before, $P_w^{(2)} = P_w^{(1)}$, and an initial state error 10 times as large, $P_{x_0}^{(2)} = 10P_{x_0}^{(1)}$. A time-step-wise probabilistic feasibility bound of $\delta_s = 0.8$ is enforced. A path is considered to reach the goal is the final position is within 0.5m of the goal location. All other algorithm quantities are the same as in Section 5.3.1

Fifty simulations were performed of each algorithm, each growing a tree of 2500 nodes. Three quantities were evaluated for every 10 nodes:

- **Path Duration:** Time duration of the lowest-cost path in the tree, in seconds.
- **Maximum Risk Bound:** Largest value of $\Delta_t(\widehat{x}_t, P_{x_t})$ encountered at any timestep of the lowest-cost path, representing a bound on the maximum risk encountered at any timestep along that path.
- **Accumulated Risk:** A measure of the total risk encountered by the system along its lowest-cost path. This quantity is evaluated as the path integral of $\Delta_t(\widehat{x}_t, P_{x_t})$, approximated as

$$\Delta_A \approx dt \sum_{t=0}^{N} \Delta_t(\widehat{x}_t, P_{x_t}).$$

  Note that this is distinct from the path-wise risk bound (3.6).

Figure 5-6 charts the evolution of each of these properties as a function of the number of tree nodes. In each figure, the median over all 50 trials for each algorithm is indicated as a solid line, while the shaded region surrounding it denotes the 10th-to-90th percentiles. The percentiles are excluded when comparing the RRT-based algorithms (Figures 5-6(b), 5-6(d), and 5-6(f)), where the huge variation in these properties otherwise obscures the overall trends. Additionally, data is only shown

Table 5.1: Properties of solution path after 2500 nodes, full-corridor scenario, 50 trials

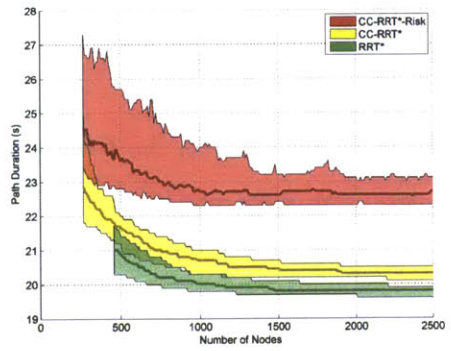
Algorithm	Path Duration (s)				Maximum Risk Bound				Accumulated Risk (mean)
	Mean	SD	Min	Max	Mean	SD	Min	Max	
CC-RRT*-Risk	22.7	0.40	22.1	24.2	0.002	0.002	0.001	0.013	0.004
CC-RRT*	20.3	0.14	20.0	20.6	0.189	0.010	0.158	0.200	0.782
RRT*	19.8	0.12	19.4	20.0	0.472	0.025	0.401	0.500	2.717
CC-RRT-Risk	27.2	3.46	21.4	40.5	0.143	0.047	0.029	0.200	0.421
CC-RRT	27.1	3.97	22.2	40.1	0.126	0.061	0.004	0.198	0.368
RRT	26.6	3.65	21.0	42.6	0.357	0.112	0.019	0.491	1.058

once all trials have found at least one feasible path to goal. Table 5.1 gives additional statistical properties of the solution path after 2500 nodes for each algorithm.

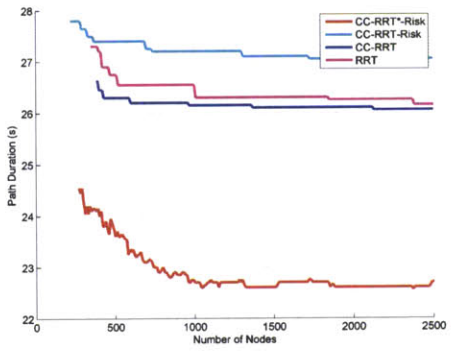
First, consider the evolution of the path duration as a function of the number of nodes (Figure 5-6(a) for the RRT*-based algorithms, Figure 5-6(b) for the RRT-based algorithms). RRT* consistently achieves shorter path durations than CC-RRT* and CC-RRT*-Risk, since it is not subject to the same robustness requirements. CC-RRT* also consistently achieves shorter path durations than CC-RRT*-Risk, since CC-RRT* only seeks to minimize path durations, while CC-RRT*-Risk includes it as one term in a multi-objective optimization. For the same reason, CC-RRT*-Risk's path duration does not decrease monotonically with the number of nodes like CC-RRT* and RRT*, though it does clearly trend in the same direction.

On the other hand, CC-RRT*-Risk is able to identify much shorter paths than any of the RRT-based algorithms, including RRT, which does not have to consider robustness requirements. This is largely due to the significant variation in path duration returned by the RRT-based algorithms due to their lack of asymptotically optimality. For example, Table 5.1 notes that the standard deviation of all RRT-based algorithms is over an order of magnitude larger than their RRT*-based counterparts, while the maximum path durations returned are twice as large.

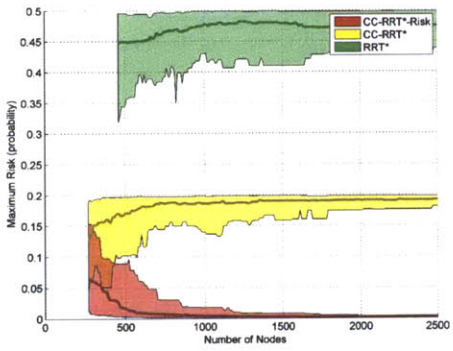
From Table 5.1, the final solution path returned by RRT* is consistently the shortest; its maximum path duration over all trials is the same or shorter than the path durations returned in all other algorithm trials. However, the mean path duration of CC-RRT* is only 2.5% larger than RRT*, and its maximum path duration is only 3% larger than RRT*, despite additionally enforcing robustness guarantees via δ_s. The



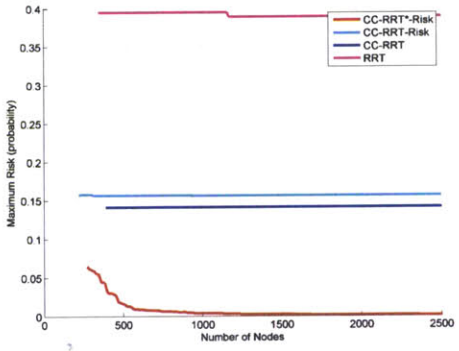
(a) Path Duration: CC-RRT*-Risk vs. RRT*-based algorithms



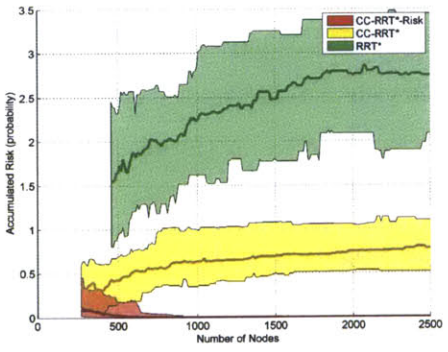
(b) Path Duration: CC-RRT*-Risk vs. RRT-based algorithms



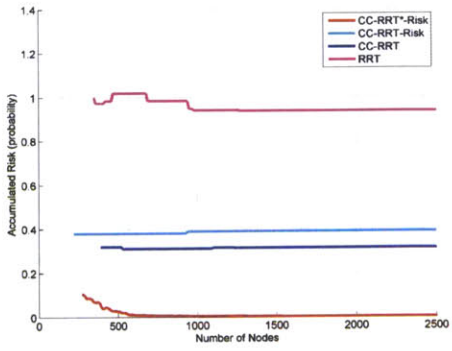
(c) Maximum Risk Bound: CC-RRT*-Risk vs. RRT-based algorithms



(d) Maximum Risk Bound: CC-RRT*-Risk vs. RRT-based algorithms



(e) Accumulated Risk: CC-RRT*-Risk vs. RRT*-based algorithms



(f) Accumulated Risk: CC-RRT*-Risk vs. RRT-based algorithms

Figure 5-6: Evolution of path duration, maximum risk bound, and accumulated risk for each algorithm, full-corridor scenario, 50 trials

paths returned by CC-RRT*-Risk are only slightly longer; its mean path duration is 15% larger than RRT* and 12% larger than CC-RRT*.

Next, consider the evolution of the maximum risk bound as a function of the number of nodes (Figure 5-6(c) for the RRT*-based algorithms, Figure 5-6(d) for the RRT-based algorithms). The most obvious distinction between the RRT*-based algorithms is that maximum risk tends to *increase* with more nodes for RRT* and CC-RRT*, while it tends to *decrease* with more nodes for CC-RRT*-Risk. Because RRT* and CC-RRT* do not include any risk-based terms in their cost objectives, the rewiring process will leverage any margin between the current maximum risk value and the allowable bound to decrease path duration. This is utilized by selecting paths which bring the system closer to obstacles, thus increasing the maximum risk. As a result, as more nodes are added, the maximum risk value tends to converge to its bound. For CC-RRT*, that bound is $1 - \delta_s = 0.2$; for RRT*, that bound approaches 0.5, *i.e.,* the value of the risk bound if the state distribution mean were exactly on a constraint boundary, and thus exactly half-feasible with respect to that constraint. CC-RRT*-Risk, however, quickly drives the maximum risk to nearly zero, with very little variation past 1000 nodes, even though any value less than 0.2 satisfies the probabilistic constraints.

Unlike the RRT*-based algorithms (Figure 5-6(c)), the RRT-based algorithms (Figure 5-6(d)) show very little change in their risk as more nodes are added. This is due to the non-optimal nature of RRT-based algorithms, where the placement of the initial tree samples has a disproportionate impact on the nature of paths generated. Notably, CC-RRT-Risk (which uses risk terms in its objective) performs slightly worse on average than CC-RRT (which only minimizes path duration in its objective) in both path duration and maximum risk bound, as supported by Table 5.1 for the final solution paths. This reinforces the idea that adding risk-based terms to the objective is only useful in an asymptotically optimal algorithm which is able to redesign the tree in accordance with those terms.

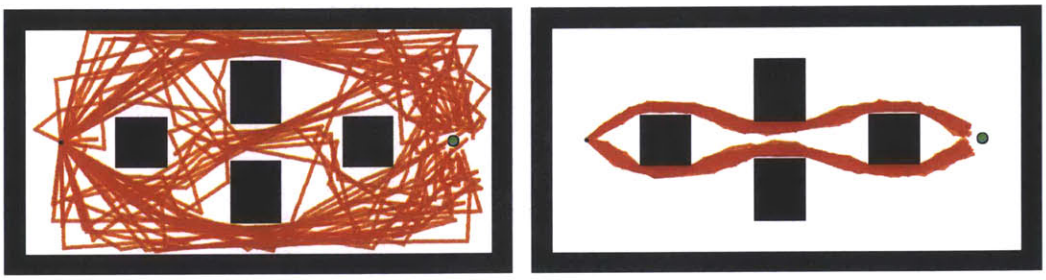
Consider the maximum risk bound in Table 5.1; the mean and maximum values of that quantity are 1–2 orders of magnitude smaller for CC-RRT*-Risk than all

other algorithms. Additionally, for each algorithm except CC-RRT*-Risk, one trial brings the maximum risk bound very close to or at the allowable bound (0.2 for CC-RRT-based algorithms, 0.5 for RRT-based algorithms). Finally, both CC-RRT* and CC-RRT*-Risk have significantly less variation in incurred risk than RRT*. The latter relationship is obvious, since CC-RRT*-Risk explicitly minimizes risk as part of its objective. The former relationship is less obvious, but is related to the fraction of paths coming near the uncertain bottommost obstacle (Figure 5-7, discussed further below).

The trends for accumulated risk (Figure 5-6(a) for the RRT*-based algorithms, Figure 5-6(b) for the RRT-based algorithms) are very similar to those for the maximum risk bound. In particular, the mean accumulated risk for CC-RRT*-Risk is two orders of magnitude lower than all other algorithms, as seen in the rightmost column of Table 5.1.

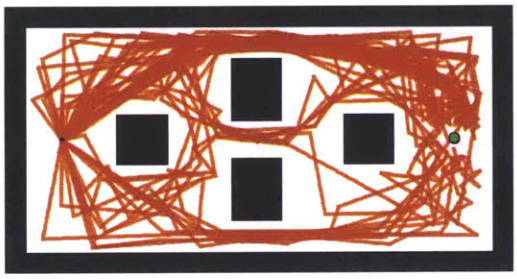
Figure 5-7 shows an overlay of the final solution paths returned by each algorithm after 2500 nodes over the same 50 trials for this scenario. The non-optimal algorithms (Figures 5-7(a), 5-7(c), and 5-7(e)) vary wildly in the qualitative nature of the paths returned, as suggested by their large corresponding variations in Table 5.1. There is a slight decrease in the number of paths which pass near the uncertain bottommost obstacle for CC-RRT (Figure 5-7(c) and CC-RRT-Risk (Figure 5-7(e)), which are risk-aware, compared to RRT (Figure 5-7(a)), which is not. However, paths in all homotopic classes are still occasionally selected by all three algorithms, largely depending on the initial node sample placement for that trial.

The final solution paths returned by the asymptotically optimal algorithms (Figures 5-7(b), 5-7(d), and 5-7(f)) are much more smooth and consistent; however, they do not all necessarily fall into the same homotopies. All of the paths returned by the RRT* algorithm (Figure 5-7(b)) come very close to the boundaries of the leftmost and rightmost obstacles, with very little variation within each homotopy. However, the paths are split between passing those obstacles from above (19 out of 50) or from below (31 out of 50). Because the environment is geometrically symmetric, and RRT* is not a risk-aware algorithm, RRT* is likely to consider paths in both homotopies,

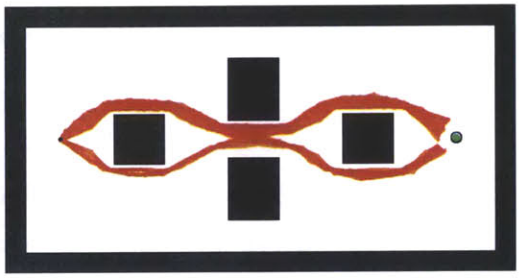


(a) RRT

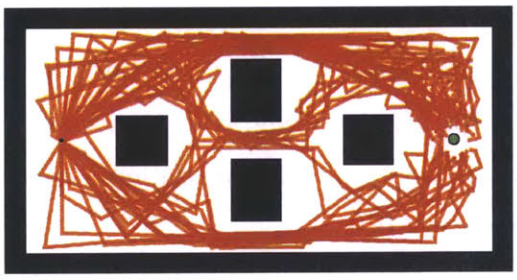
(b) RRT*



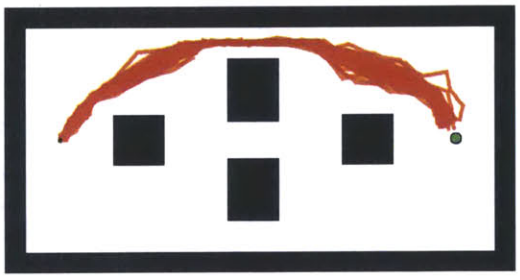
(c) CC-RRT



(d) CC-RRT*



(e) CC-RRT-Risk



(f) CC-RRT*-Risk

Figure 5-7: Overlay of final solution paths returned in all 50 trials for each algorithm, full-corridor scenario

even though one is subject to a much higher risk of collision than the other.

Comparing RRT* to CC-RRT* (Figure 5-7(d)), which is risk-aware, it can be seen that CC-RRT* is much less likely to select paths which pass near the uncertain obstacle. Of the 50 trials, 41 pass the leftmost and rightmost obstacles from above. Additionally, all paths chosen maintain sufficient distance from the obstacles, especially the bottommost obstacle, to ensure probabilistic feasibility constraints are satisfied. Regardless, the paths returned by the algorithm are not consistent: 3 of the 50 trials pass the leftmost and rightmost obstacles from below, while the remaining 6 trials alternate.

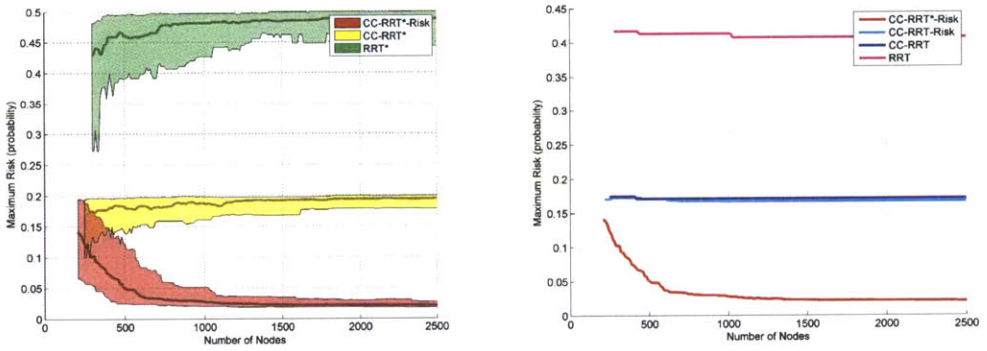
On the other hand, CC-RRT*-Risk (Figure 5-7(f)), which not only acknowledges the risk posed by uncertainty but explicitly optimizes against it via rewiring, consistently identifies paths in the uppermost homotopy in all 50 trials. It is the only algorithm of the six to fully recognize the asymmetry of this environment due to its uncertainty in all trials.

In parts of the environment where the risk is relatively low, *i.e.,* the upper-left and upper-right regions of Figure 5-7(f), CC-RRT*-Risk exhibits more variation in its paths than seen from either RRT* or CC-RRT*. On the other hand, as the system passes the uppermost obstacle, there is very little variation in the paths at all: all 50 paths pass very near the centerline of that corridor. Because CC-RRT*-Risk incorporates risk bounds within its objectives, regions with higher risk correspond to larger gradients in the algorithm's objective function. Thus, the paths returned by CC-RRT*-Risk can be expected to show lower variation in regions of higher risk, and vice versa.

Alternatively, suppose the upper boundary of this environment is moved downward 1.25 feet (0.38 m), such that the width of that uppermost corridor is reduced by half. This is referred to as the half-corridor scenario. Under this modification, any path that would pass over all obstacles is no longer probabilistically feasible for $\delta_s = 0.8$, due to the narrow corridor above the uppermost obstacle. Table 5.2 gives the final path results of running 50 simulation trials of each algorithm in this environment, while Figure 5-8 plots the evolution of the maximum risk bound as a function

Table 5.2: Properties of solution path after 2500 nodes, half-corridor scenario, 50 trials

Algorithm	Path Duration (s)				Maximum Risk Bound				Accumulated Risk (mean)
	Mean	SD	Min	Max	Mean	SD	Min	Max	
CC-RRT*-Risk	23.6	0.51	22.5	25.0	0.022	0.004	0.017	0.035	0.094
CC-RRT*	20.3	0.14	19.9	20.5	0.191	0.009	0.167	0.200	0.832
RRT*	19.8	0.14	19.4	20.2	0.477	0.025	0.376	0.500	2.774
CC-RRT-Risk	27.8	2.93	22.7	33.7	0.150	0.041	0.054	0.200	0.648
CC-RRT	28.7	4.13	21.5	42.5	0.162	0.034	0.071	0.200	0.581
RRT	26.0	3.08	20.8	35.5	0.369	0.102	0.164	0.498	1.306



(a) CC-RRT*-Risk vs. RRT-based algorithms (b) CC-RRT*-Risk vs. RRT-based algorithms

Figure 5-8: Evolution of maximum risk bound for each algorithm, half-corridor scenario, 50 trials

of tree nodes for each algorithm. (The corresponding figures for path duration and accumulated risk are omitted for brevity.)

By making the environment more restrictive, the maximum risk bound and accumulated risk generally increase for each algorithm, as expected. The most notable change is in CC-RRT*-Risk, for which the maximum risk bound and accumulated risk each increase by approximately an order of magnitude. Of the three asymptotically optimal algorithms, CC-RRT*-Risk was the only one to utilize the risk-minimizing upper corridor, which is no longer probabilistically feasible; thus its risk increases significantly as it is forced to consider alternatives, though it remains well below enforced bounds. This is most visible in Figure 5-8 where the maximum risk for CC-RRT*-Risk is seen to be bounded away from zero.

Table 5.3: Number of nodes sampled until feasible path to goal found, 50 trials

Algorithm	Full-Corridor		Half-Corridor	
	Mean	Max	Mean	Max
CC-RRT*-Risk	73	270	84	210
CC-RRT*	80	270	99	250
RRT*	91	460	77	300
CC-RRT-Risk	78	220	76	220
CC-RRT	89	390	98	250
RRT	88	350	64	280

The risk statistics for RRT* and CC-RRT*, on the other hand, are nearly identical to the full-corridor scenario (Figure 5-6(c) and Table 5.1), as the paths those algorithms converge to are unaffected by the shifted environment boundary. The increases in risk faced by CC-RRT-Risk, CC-RRT, and RRT are slightly larger; the final paths for these algorithms tend to vary among all feasible homotopies, but the risk-minimizing homotopy is no longer feasible.

On the other hand, no algorithm experiences a major increase in path duration as a result of the change. Most notably, even though CC-RRT*-Risk is now incurring 10 times as much risk, its mean path duration has only increased by 4%.

An important question in these comparisons is considering whether the refinements beyond the nominal RRT algorithm result in more nodes being required to find a feasible path to the goal. This is particularly important in real-time applications, where it is often critical to quickly identify a feasible solution, especially if it can be refined later online. Table 5.3 shows the mean and maximum number of nodes required to find a feasible path to goal across all trials for each scenario considered above. The data shows that the number of nodes required is fairly consistent across all algorithms and scenarios: a feasible path is found on average in under 100 nodes, while a feasible path is always found within 500 nodes. This implies that the effect of these modifications on finding feasible paths is limited, at best.

Table 5.4 shows the average time required to generate a feasible node for each algorithm across all trials. This calculation includes any time spent on failed attempts to connect the tree to samples, *i.e.,* tree expansion steps that fail to generate additional nodes. The most notable increase in computation comes from switching

Table 5.4: Per-node computation results for each algorithm, 50 trials

Algorithm	Runtime per Node (ms)
CC-RRT*-Risk	17.84
CC-RRT*	17.04
RRT*	7.17
CC-RRT-Risk	1.36
CC-RRT	1.36
RRT	0.83

from an RRT-based algorithm to its RRT*-based equivalent; this results in a runtime increase by a factor of 9–13 per node. This factor can be significantly affected by the complexity of the dynamics and the steering law that is used, as RRT-based algorithms do not require use of a steering law. These considerations should be taken into account when determining which algorithm(s) to apply.

By comparison, introducing robustness via chance constraints only results in a runtime increase by a factor of 1.6–2.5 per node, which is consistent with previous results [46]. The impact of introducing risk-based terms into the cost function is minimal. Regardless, both CC-RRT*-Risk and CC-RRT* are able to generate over 50 nodes a second on average, making them suitable for real-time use.

Figure 5-9 shows an evolution of the computation time required per node, as a function of tree size, for the full-corridor scenario; computation has been averaged every 10 nodes to smooth the plots. The median value is indicated as a solid line, while the shaded region denotes the 10th-to-90th percentiles. The computation required per node for the RRT-based algorithms tends to slowly increase with tree size. On the other hand, the computation required per node for the RRT*-based algorithms increases at a faster rate up to about 700 nodes or so, where it begins to level off and even decrease slightly. This is largely due to the radius r_n used to check nearby nodes for possible connections decreasing over time. At the point where the direction of growth rate change shifts, the computational effort of searching a larger tree becomes offset by the few number of connections to nearby nodes being attempted during each cycle.

Finally, sets of 50 trials were performed for each algorithm in which each trial

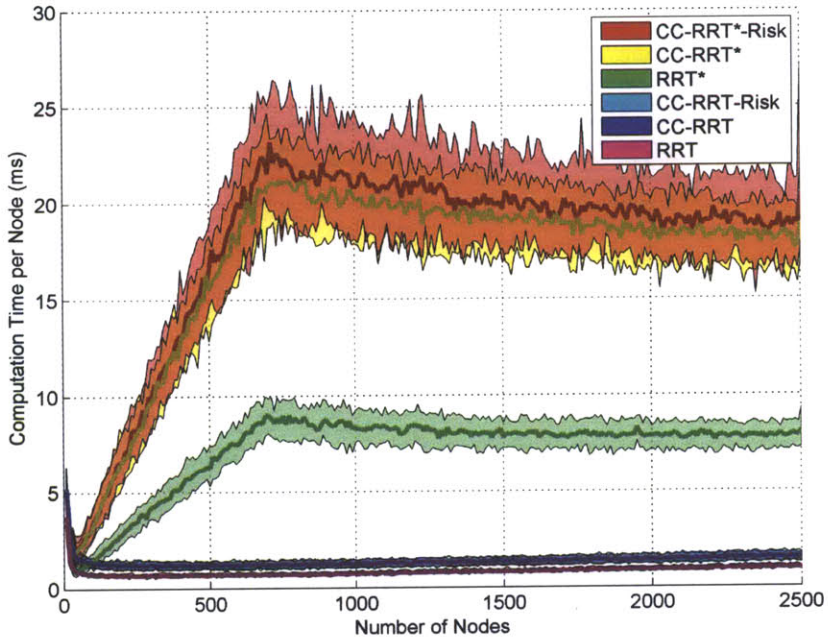


Figure 5-9: Evolution of computation per node for each algorithm, full-corridor scenario, 50 trials

consists of growing a tree for 30 seconds, with no upper bound on the number of nodes. Figure 5-10 shows the evolution of the mean path duration, as a function of computation time elapsed, for these trials. This figure clearly shows that even with the robustness modifications of CC-RRT* and CC-RRT*-Risk, both algorithms find higher-quality paths (in terms of duration) than any RRT-based algorithms within the initial seconds of computation.

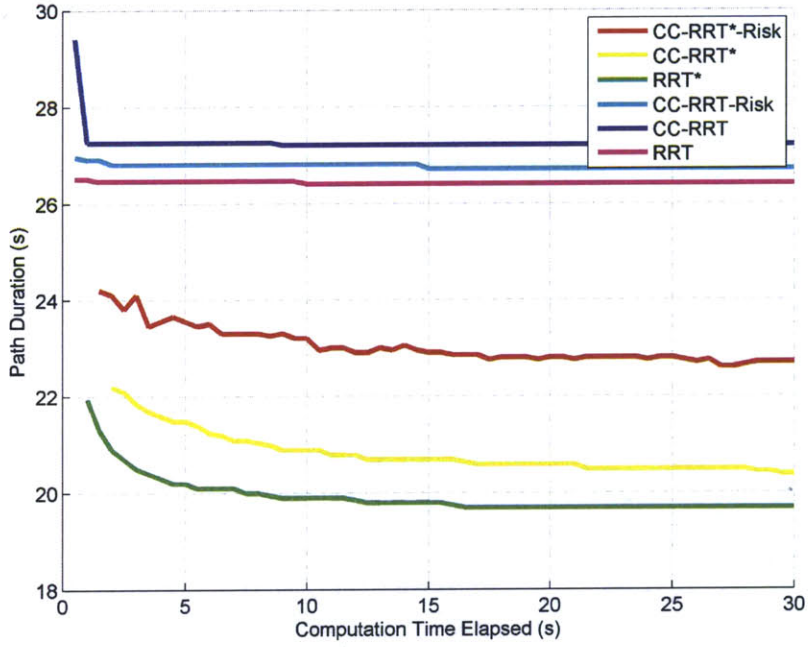


Figure 5-10: Evolution of mean path duration as function of computation time for each algorithm, full-corridor scenario, 50 trials

5.3.3 Double Integrator

Consider the more complex 2D double integrator dynamics,

$$
\begin{aligned}
x_{t+1} &= \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} \frac{1}{2}dt^2 & 0 \\ 0 & \frac{1}{2}dt^2 \\ dt & 0 \\ 0 & dt \end{bmatrix} u_t + \begin{bmatrix} \frac{1}{2}dt^2 & 0 \\ 0 & \frac{1}{2}dt^2 \\ dt & 0 \\ 0 & dt \end{bmatrix} w_t, \\
x_t &= \begin{bmatrix} p_t^x & p_t^y & v_t^x & v_t^y \end{bmatrix}^T, \\
u_t &= \begin{bmatrix} a_t^x & a_t^y \end{bmatrix}^T,
\end{aligned}
$$

where, again, $dt = 0.1$ s. The system operates in the same environment as Section 5.3.2, with the same obstacle placement and start and goal locations. In addition to the previous state constraints, this problem is also subject to the velocity state

constraints $|v_t^x| \leq v_{\max}$, $|v_t^y| \leq v_{\max}$, where $v_{\max} = 2.5$ m/s, as well as the input constraints $|a_t^x| \leq \bar{a}$, $|a_t^y| \leq \bar{a}$, where $\bar{a} = 5.0$ m/s^2.

The system is subject to the same three forms of uncertainty, though their values have changed. First, the initial state x_0 is subject to *position-based* initial state error (3.2) of equal value in all directions,

$$x_0 \in \mathcal{N}(\widehat{x}_0, P_{x_0}^{(3)}), \quad P_{x_0}^{(3)} = 10^{-3} \begin{bmatrix} I_2 & 0_2 \\ 0_2 & 0_2 \end{bmatrix}.$$

At each timestep, the system is subject to *velocity-based* process noise (3.1) of equal value in all directions,

$$w_t \in \mathcal{N}(0, P_w^{(3)}), \quad P_w^{(3)} = 10^{-6} \begin{bmatrix} 0_2 & 0_2 \\ 0_2 & I_2 \end{bmatrix}.$$

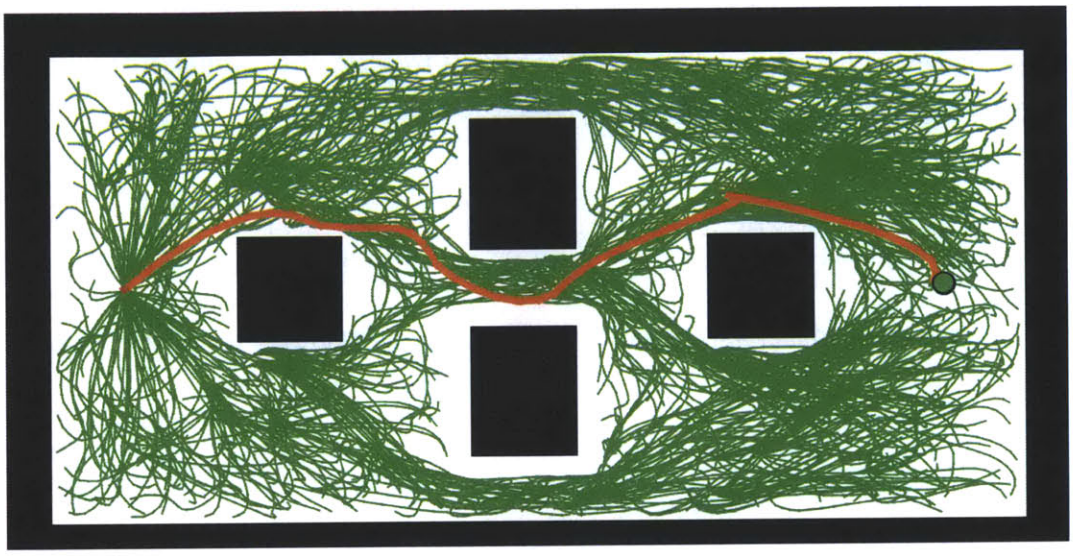
Finally, the placement of the bottommost obstacle is subject to the same position-based uncertainty (3.4) as in Section 5.3.2,

$$c_{jt} \in \mathcal{N}(\widehat{c}_{jt}, P_{c_j}^{(3)}), \quad P_{c_j}^{(3)} = 0.05 \begin{bmatrix} I_2 & 0_2 \\ 0_2 & 0_2 \end{bmatrix}, \quad \forall\, t;$$

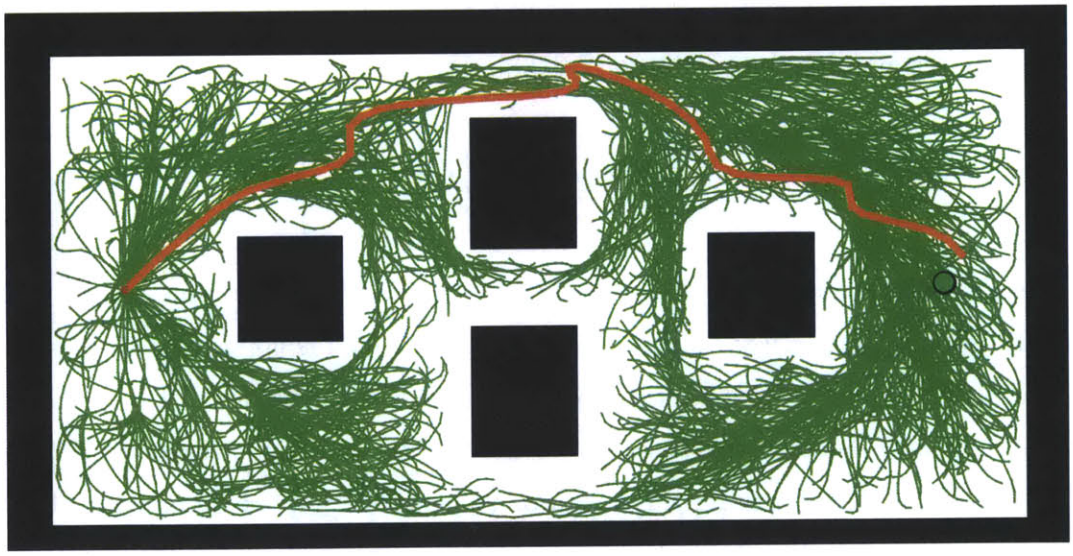
The system is required to satisfy a minimum probability of constraint violation at each timestep of $\delta_s = 0.8$; no path-wise probability bound is imposed (*i.e.*, $\delta_p = 0$). The sampling strategy and steering law are the same as in Section 2.5.4.

Figure 5-11 demonstrates typical trees and final solution paths generated by CC-RRT* and CC-RRT* risk for this scenario. Similar behaviors are observed for both algorithms as in the single-integrator scenarios, though the degree of suboptimality in the solution paths is higher due to both the reduced number of nodes and the increased state dimension. Regardless, all paths in each tree satisfy probabilistic feasibility requirements for $\delta_s = 0.8$, with additional buffers induced by using a risk-based objective in Figure 5-11(b).

Figure 5-12 shows a 1000-node CC-RRT*-Risk tree and solution path generated



(a) CC-RRT*



(b) CC-RRT*-Risk

Figure 5-11: Demonstrative 1500-node CC-RRT* trees and minimum-cost paths, double integrator, full-corridor scenario

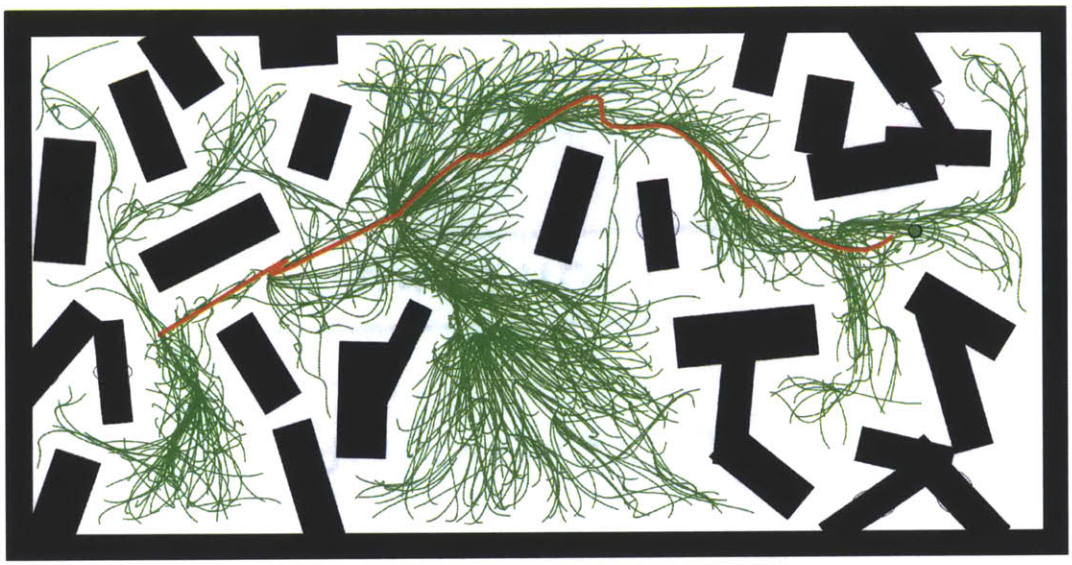


Figure 5-12: Demonstrative 1000-node CC-RRT* tree and minimum-cost path, double integrator, cluttered scenario

for the double integrator dynamics (subject to the same initial state error and process noise) in a cluttered 20m $\times$ 10m environment, consisting of 30 obstacles with randomized placement uncertainty. This demonstrates the scalability of the CC-RRT* algorithm to very complex environments, in both the number of obstacles and the uncertainty characterization.

5.3.4 Higher-Dimensional Planning

Figure 5-13 demonstrates a tree of trajectories and minimum-cost path for a 3D single integrator operating in a 10m $\times$ 10m $\times$ 10m environment. The system (magenta diamond) must find a path around the large central obstacle (gray, left) to the goal (green circle) on the other side. Given the environment constraints, the only feasible way to do so is to pass over the obstacle in the foreground, then under the obstacle in the background. The planner tree is shown in green, while the minimum-cost path which successfully reaches the goal is shown in orange.

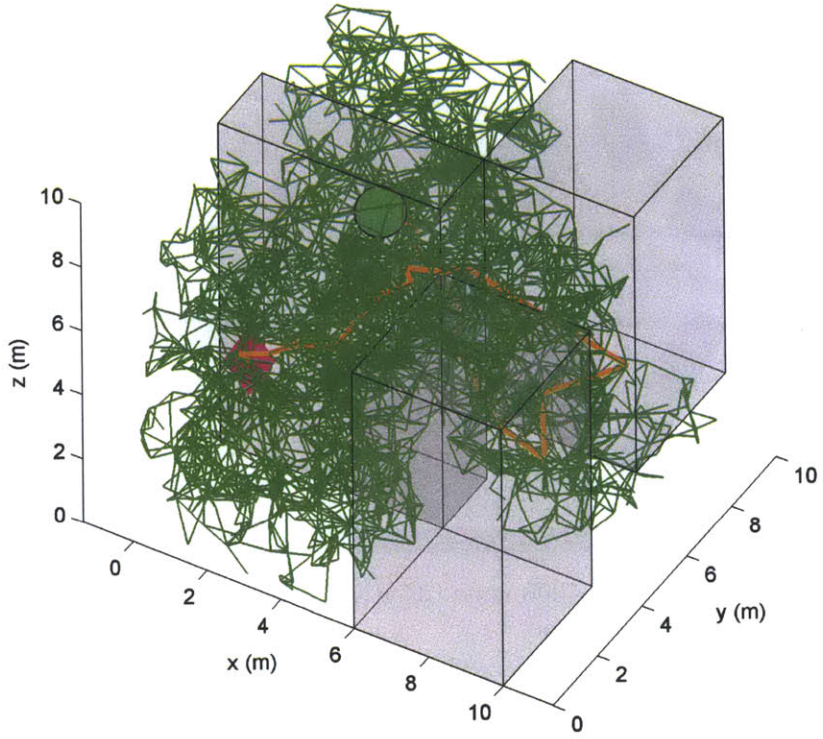


Figure 5-13: Demonstrative CC-RRT* tree, single integrator, three-dimensional environment

5.3.5 Dubins Vehicle

Figure 5-14 provides an example of how CC-RRT* can be utilized on more complex dynamics, as long as a suitable steering law is available. In this case, the system is a 2D car with Dubins dynamics [9] and corresponding steering law. Though the dynamics are nonlinear, the uncertainty environment acts on the system linearly, such that analytic uncertainty distributions can be maintained (Section 4.4). In this case, the system is subject to a small process noise and a dynamic obstacle (magenta) with uncertain intentions (blue), as discussed in Section 4.2. The tree identifies a relatively smooth path that guides the vehicle (brown) as close as it can safely get to the goal (green circle).

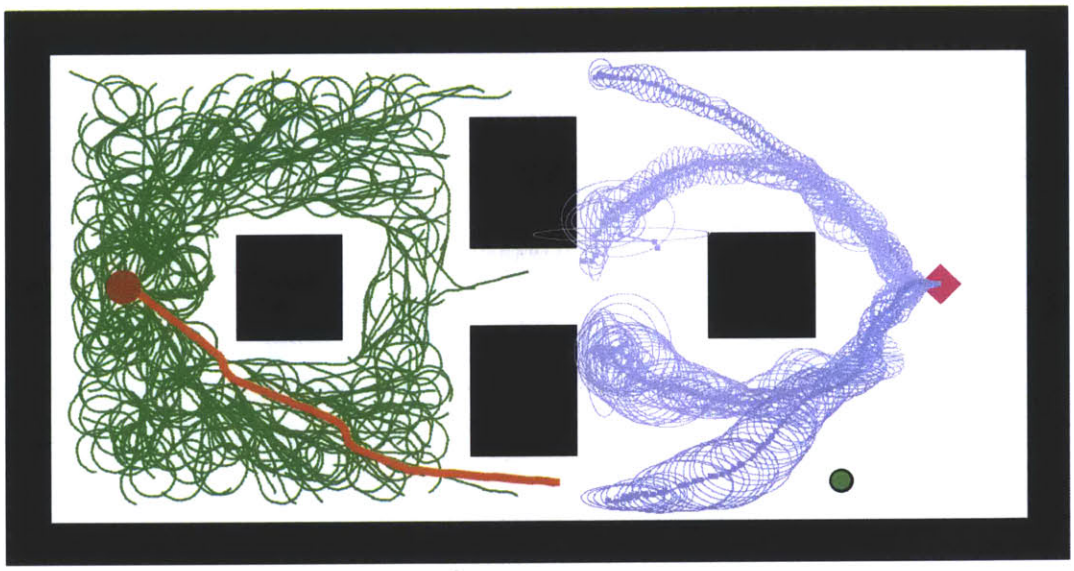


Figure 5-14: Demonstrative CC-RRT* tree, Dubins vehicle, dynamic obstacle scenario

5.3.6 Pursuit-Evasion Games

The CC-RRT* algorithm can also be applied to a variety of planning scenarios with more complex uncertainty assumptions, particularly in characterizing the uncertainty of dynamic obstacles [113]. One possible scenario with real-world implications is the pursuit-evasion problem. In this scenario, it is assumed that there are other dynamic "pursuer" agents in the environment, which will act in a worst-case manner to try to collide with the planning "evader" agent.

By adapting an RRT* algorithm variant for this scenario [60], the CC-RRT* algorithm can handle this problem even in the presence of multiple sources of uncertainty. This variant grows CC-RRT* trees for the evader and all pursuers, removing evader nodes when a pursuer can cause a collision by arriving at that node at the same or previous timestep. The results below show probabilistically feasible open-loop solutions (*i.e.,* not using feedback on pursuer movement) to this adapted [60] problem when the evader and some obstacles are uncertain. Though worst-case behavior is assumed for the pursuers, the actual motion of those agents is deterministic – though it could be made uncertain, as well, without additional modification.

In this example, consider the single integrator from Sections 5.3.1 and 5.3.2 with

the same constraints and apply CC-RRT*-Risk with $\delta_s = 0.9$. The environment now takes the form of Figure 5-15; note that the uppermost obstacle is uncertain, while all other obstacle placements are precisely known. The uncertainty environment is characterized as

$$S_x^{PE} = \begin{bmatrix} 0.0005 & 0 \\ 0 & 0.0030 \end{bmatrix}, \; S_w^{PE} = \begin{bmatrix} 0.000060 & 0 \\ 0 & 0.000010 \end{bmatrix}, \; S_c^{PE} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}.$$

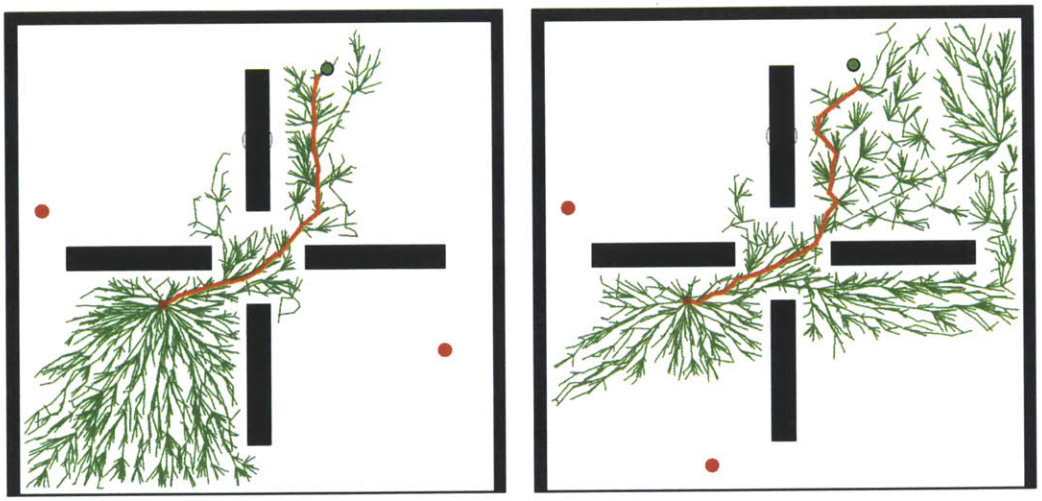
There are two pursuing agents in the environment; their dynamics are assumed to be deterministic, but otherwise identical to those of the evader, including their maximum speed.

Figures 5-15(a) and 5-15(b) show the resulting CC-RRT*-Risk tree and current solution path for the evading agent, for two different initial placements of the pursuing agents. The evading agent and chosen path are shown in orange, while the pursuing agents are shown in red. The 2-σ uncertainty ellipse is shown for the uppermost obstacle, whose placement is uncertain. The trees for the pursuers are suppressed for clarity.

As the lower pursuer is moved behind the evading agent in going from Figure 5-15(a) to Figure 5-15(b), a significant additional region of feasible nodes is added to the evader tree in the lower-right corner. Ultimately, these nodes are not used to select a path, as the cost is higher in this region, and thus they are not connected directly to the goal. The tree maintains a large buffer with respect to the uncertain obstacle, with the evader ultimately choosing a risk-averse path that maintains an even larger distance from that obstacle.

5.4 Conclusions

This chapter has introduced the CC-RRT* algorithm for robust, scalable, and asymptotically optimal path planning. The algorithm efficiently computes bounds on risk of constraint violation using a chance constraint formulation – expanded in this work to consider environmental uncertainty, path-wise feasibility bounds, and probabilistic



(a) Placement 1 of pursuers

(b) Placement 2 of pursuers

Figure 5-15: Demonstrative 1500-node CC-RRT*-Risk tree, 2D single integrator, pursuit-evasion scenario

environmental boundaries – to ensure all trajectories considered are both dynamically and probabilistically feasible. These risk bounds are also utilized within a novel, risk-based cost function, shown to be admissible for RRT*. The CC-RRT algorithm has been shown to be probabilistically complete and asymptotically optimal under appropriate assumptions. Simulation results have demonstrated that CC-RRT* can be utilized to identify smooth, robust trajectories, displaying a level of risk-averse behavior specified by the user.

Chapter 6

Experimental Results

This chapter presents hardware results demonstrating the ability of the CC-RRT algorithm to plan and execute safe trajectories in real-time. In these experiments, the CC-RRT planner is used to guide an autonomous rover through dynamic, uncertain environments. There are many sources of uncertainty tested in these experiments, including uncertainty in the motion of the rover; uncertainty in the location of detected pedestrians; and uncertainty in the future motion of dynamic robots. A variety of sensing infrastructure is used to gather data about the vehicle state and the location of obstacles, including motion-capture camera data and onboard 2D lidar data, facilitating true perception-driven planning.

First, the implementation of the experimental setup is presented (Section 6.1), including the testbed environment, autonomous vehicle, available sensors, and planning software. The first set of experiments (Section 6.2.1) involve the rover driving around small robots – both static and dynamic with uncertain intentions – tracked by motion capture. In the second set of experiments (Section 6.2.2), the rover must plan and execute online paths to avoid one or more pedestrians, identified and tracked by onboard lidar.

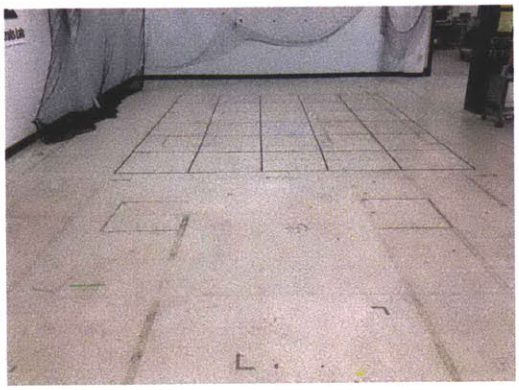


Figure 6-1: RAVEN testbed environment

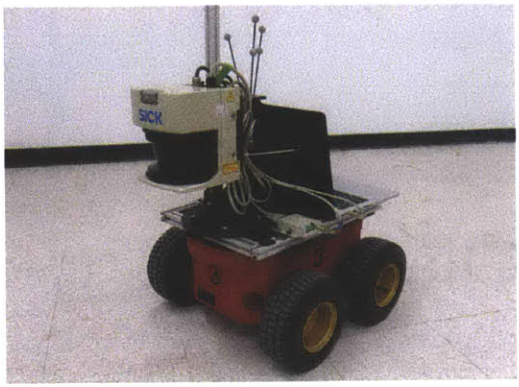


Figure 6-2: Pioneer 3-AT rover

6.1 Implementation

Hardware experiments are performed in the Real-time indoor Autonomous Vehicle test ENvironment, or RAVEN [114]. The primary RAVEN testbed area is approximately 12 m $\times$ 6 m in size (Figure 6-1), and contains a set of Vicon motion-capture cameras designed to track the location of any operational vehicles [115]. In these experiments, a smaller subset of that environment, approximately 9.75 m $\times$ 4.25 m, is used as the environmental boundaries for the rover.

A Pioneer 3-AT rover is used as the autonomous vehicle in all experiments. The Pioneer 3-AT is a skid-steered vehicle, with a maximum speed of 0.7 m/s for each wheel in either direction. Its payload includes a 2D SICK LMS-291 lidar for onboard pedestrian detection [96] and an Intel Core i5 laptop with 6GB RAM for computation, as well as a configuration of reflective dots for motion-capture tracking.

The online planning and control algorithms implement the full CC-RRT algorithm, including tree expansion and the online execution loop, onboard the rover laptop using a multi-threaded, real-time Java application. The software implementation consists of four primary modules, each in a separate thread. The vehicle thread manages the overall simulation, including all simulation objects. The CC-RRT thread implements Algorithms 4-5, growing the CC-RRT tree while periodically sending the current best path in the tree to the vehicle thread. Where appropriate, the DPGP thread builds predictions on the likelihoods and future state distributions of possible

behaviors for each dynamic obstacle [96]. Finally, the communications thread receives state/obstacle data and sends waypoint plans via the Robotic Operating System (ROS) [116]. High-fidelity localization is provided for the rover via the motion-capture cameras [114].

A pure pursuit controller [117] is used both in planning to generate robustly feasible trajectories within CC-RRT, and onboard the rover laptop to execute waypoint plans provided by CC-RRT. The pure pursuit controller uses a desired velocity of 0.3 m/s, an L1 distance of 0.6m, and an anchor point 3cm ahead of the vehicle front wheel axes.

6.2 Hardware Experiments

6.2.1 Robust Avoidance of Dynamic Robots

In these experiments, the autonomous rover must safely navigate around one or more small iRobot Create [118] robots to reach a sequence of goal waypoints. Each robot is detected and tracked through Vicon motion capture. Within the planner, each robot is represented as an 8-sided obstacle.

The planner provides the rover with a fixed sequence of goal waypoints to reach, one goal at a time, located at the four corners of the testing environment. Rather than waiting for the rover to reach each goal precisely, the planner switches to its next goal anytime the rover gets within approximately one meter of a goal waypoint. Whenever the goal is switched, the existing CC-RRT tree is preserved, but the costs are recomputed based on the new goal location. Doing so enables the rover to move continuously through the waypoint sequence.

In these experiments, the rover is initially subject to two primary forms of uncertainty. First, the rover's driving is not precise – its controller occasionally overshoots or undershoots waypoints on its trajectory plan. Within the CC-RRT planner, this model uncertainty is represented as a process noise in position. Second, the placement of each robot is assumed to be uncertain. The CC-RRT algorithm grows a tree of

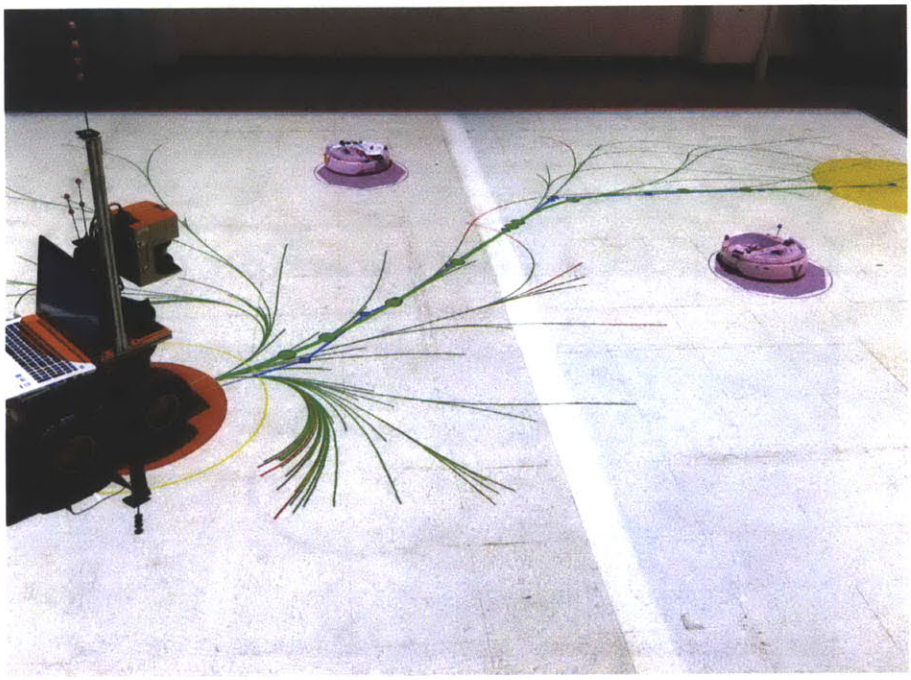
trajectories that are probabilistically feasible for the time-step-wise chance constraint $\delta_s = 0.9$, subject to both of these uncertainties.

To demonstrate CC-RRT planning and execution, we have set up a visual overlay which maps the CC-RRT situational awareness directly onto the physical environment it represents (Figure 6-3). This is accomplished through the use of a pair of overhead projectors which project directly onto the floor, and act as a monitor on which the CC-RRT visualization can be run. In the figure, the white region on the overlay represents feasible portions of the environment for the rover. They do not represent the *entire* feasible space, however. The projectors span the full width of the feasible space, but not its length; thus, the rover will occasionally have feasible plans that take it outside the boundaries of the projected overlay to reach goal waypoints.

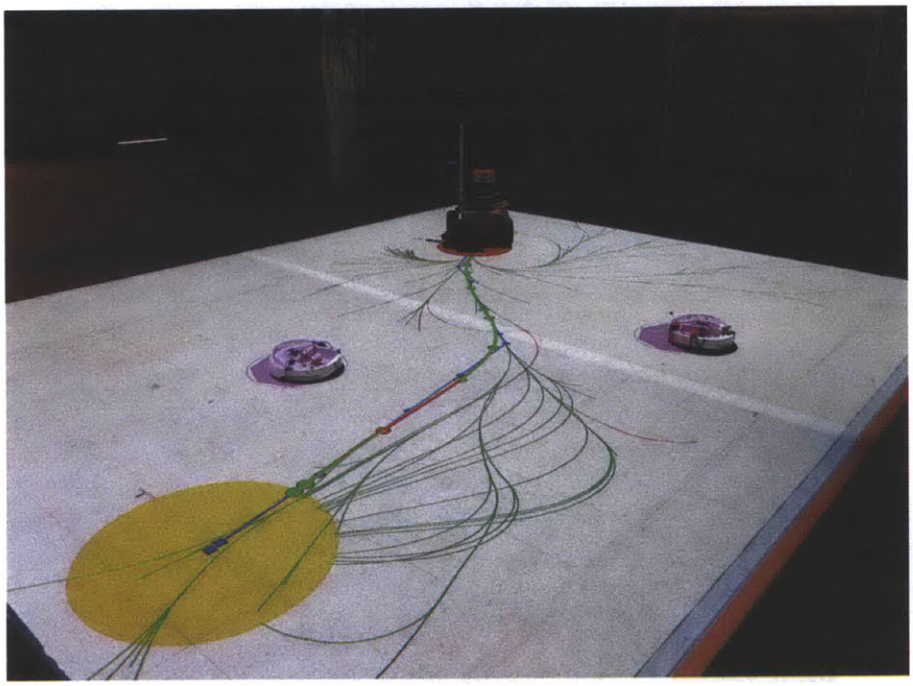
Figures 6-3(a) and 6-3(b) show representative CC-RRT trees grown for the rover to reach a goal waypoint while avoiding two static robots. In the overlay, the rover is represented by a large orange circle, while each robot is represented as an 8-sided magenta polygon with uncertainty ellipse (gray) shown. The goal region for the rover is marked in yellow. The CC-RRT tree (thin edges) and selected path for execution (thicker edges, with circular nodes) are nominally colored green; however, as the risk bound for trajectory segments on the tree/path approaches $1 - \delta_s$, the color changes from green to red, indicating higher-risk segments of the tree.

The planner runs the CC-RRT execution loop (Algorithm 5) online at $\Delta t = 1s$. During each iteration, it uses updated information on obstacle locations to update the risk bounds throughout the tree, removing any nodes that are no longer probabilistically feasible. Video 1 (Table 6.1) and Figure 6-4 demonstrate how these risk bound updates allow the CC-RRT planner to quickly update its planning tree and choose paths to execute as changes are observed. In this example, the rover is stationary, but four static robots are manually placed and moved within the environments, causing probabilistically feasible regions to shift dynamically. In fact, once the fourth robot is placed, the rover can no longer identify a probabilistically feasible path to the goal waypoint (Figure 6-4(d)); only a partial path can be found.

In Video 2 (Table 6.1), four robots are placed in fixed locations, but the rover

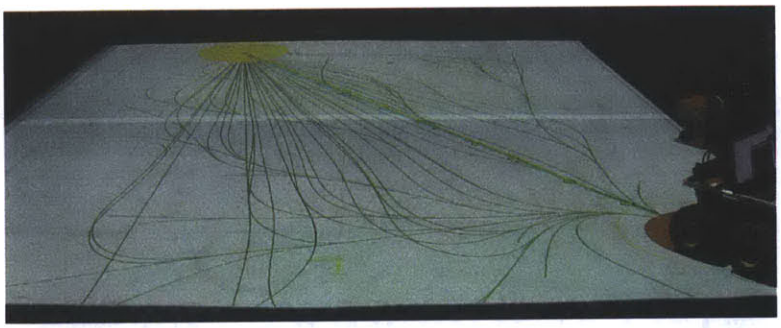


(a) Front view

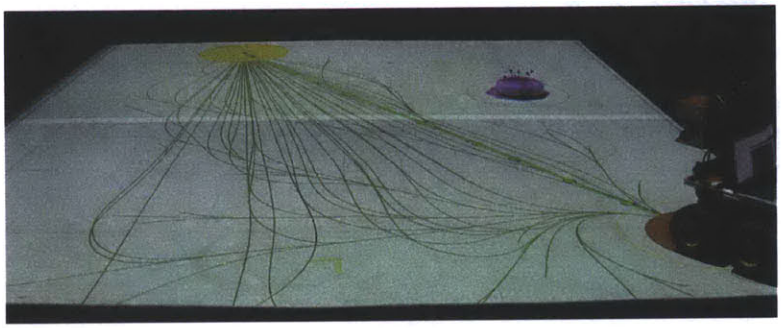


(b) Back view

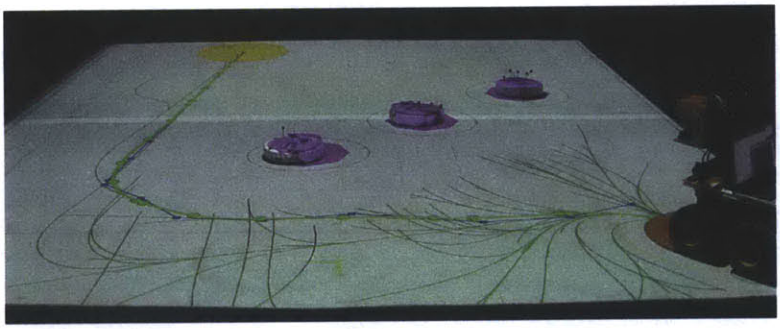
Figure 6-3: Static obstacle environment, overlaid with planner visualization



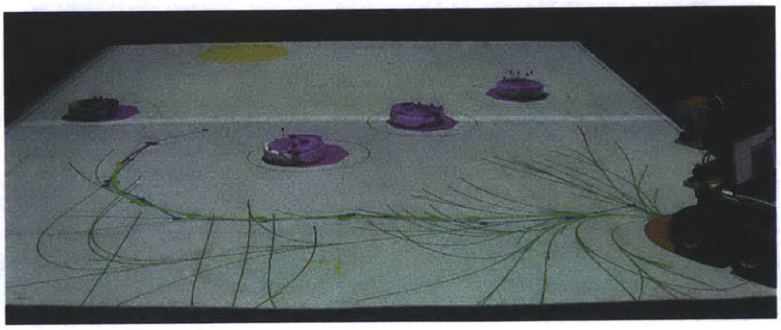
(a)



(b)



(c)



(d)

Figure 6-4: Stationary rover planning paths around 4 static robots

Table 6.1: Summary of hardware experiment videos

#	Filename	Description
1	`video1.mov`	Stationary rover, 4 static robots
2	`video2.mov`	Moving rover, 4 static robots
3	`video3.mov`	Moving rover, 1 dynamic robot
4	`video4.mov`	Moving rover, 2 dynamic robots
5	`video5.mov`	Stationary rover, 3 pedestrians
6	`video6.mov`	Moving rover, 1 pedestrian
7	`video7.mov`	Moving rover, 2 pedestrians

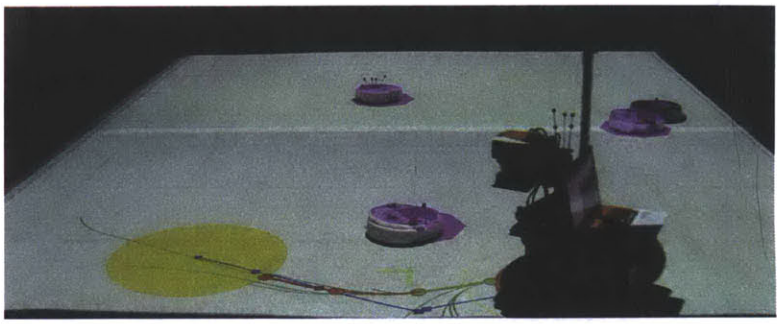
All videos are located at `http://acl.mit.edu/luders-phd/video#.mov` - insert # from table above. Copies were also included in the submission of this thesis to MIT.

is now moving autonomously using the CC-RRT planner. The rover safely passes through 17 goal waypoints; examples of some of the safe motion plans generated by CC-RRT for the rover are shown in Figure 6-5.

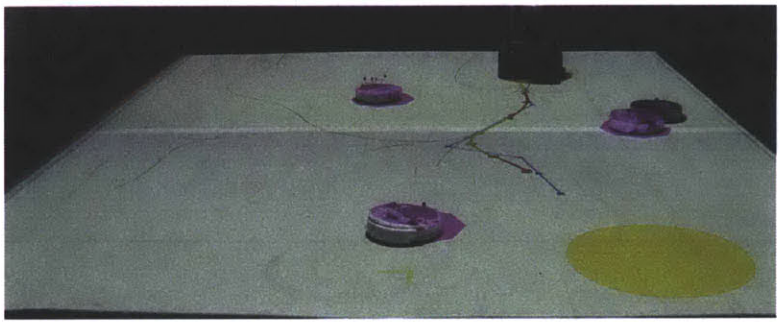
Experiments were also performed in which the rover safely avoids one or more dynamic robots with uncertain intentions, introducing the third type of uncertainty. In these examples, the dynamic robots can follow one of three possible behaviors, visualized as gray outlines on the testbed overlay (Figure 6-6). The DPGP algorithm [96] is trained on these behaviors, such that it can predict the likelihood and trajectory distribution for each online and provide that information to the CC-RRT planner (Section 4.2.3).

In Video 3 (Table 6.1), the rover safely navigates through a sequence of 16 goal waypoints while avoiding one dynamic robot. Figure 6-8 provides an example of an interesting rover/robot interaction in this scenario. Initially, the rover's path to its goal in the back-right corner of the environment is partially pruned, due to an anticipated overlap with one of the dynamic robot's possible behaviors (Figure 6-8(a)). After some time, the rover begins to execute a new, feasible path to goal that it identifies (Figure 6-8(b)). However, updates in the robot position cause that path to also become infeasible. The rover comes to a stop and waits for the robot to pass (Figure 6-8(c)); once the robot has done so, the planner identifies a new path reaching the goal (Figure 6-8(d)).

In Video 4 (Table 6.1), the rover safely navigates through a sequence of 10 goal

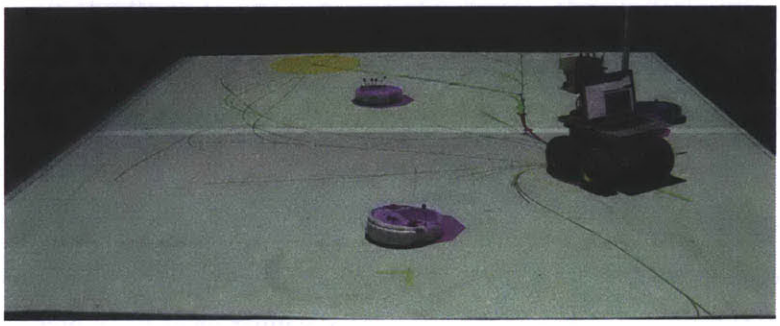


(a)



(b)

(c)



(d)

Figure 6-5: Moving rover planning paths around 4 static robots

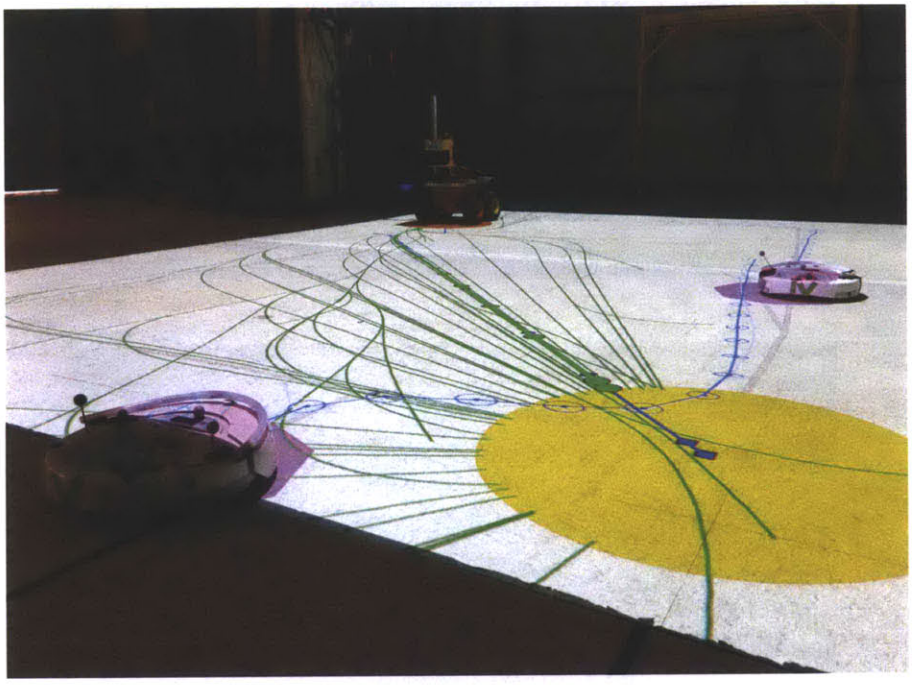


Figure 6-6: Dynamic obstacle environment

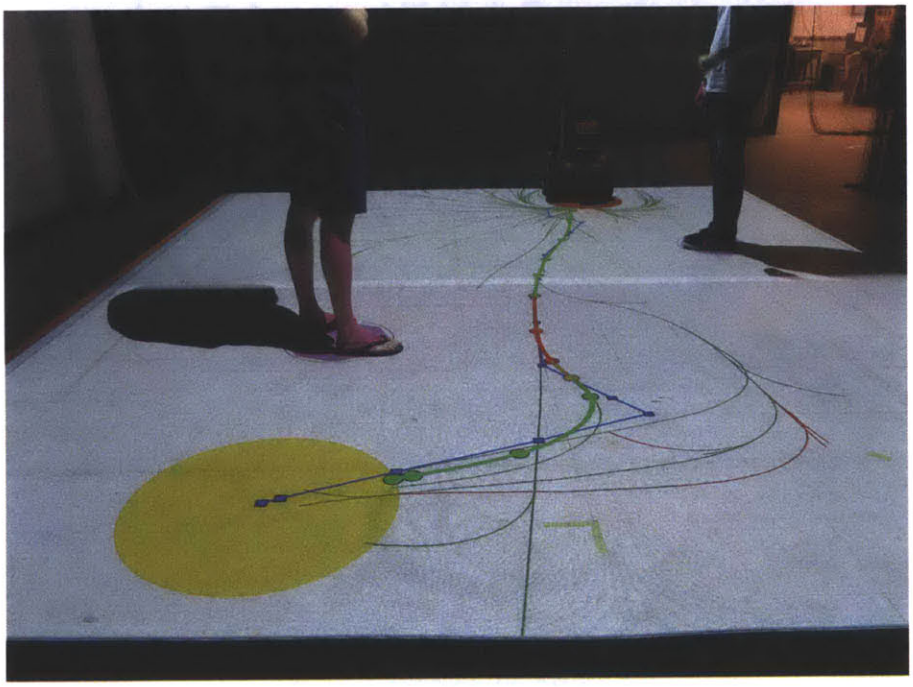
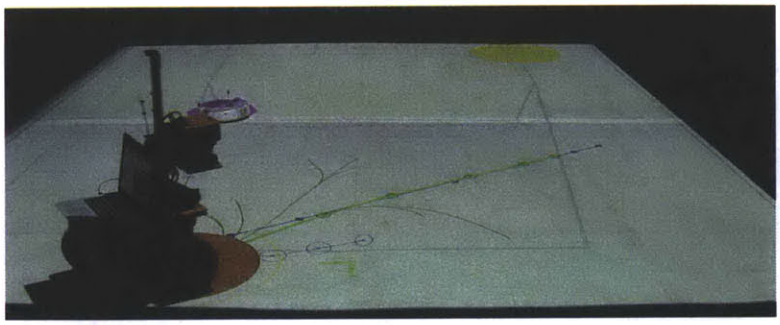
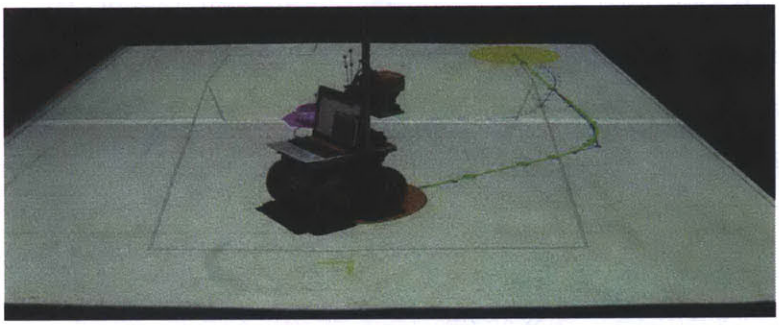


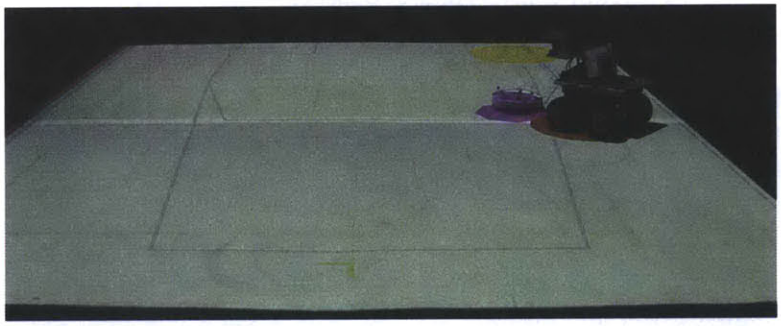
Figure 6-7: Pedestrian environment



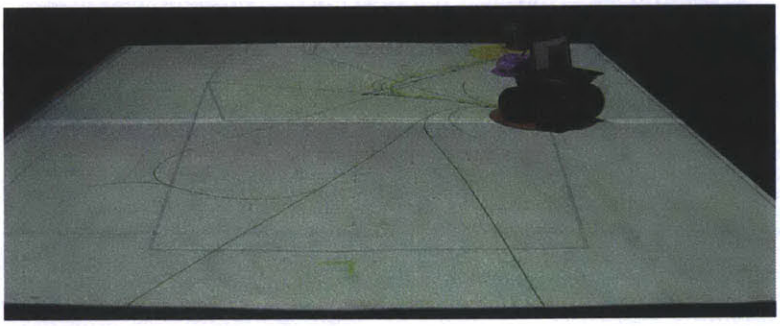
(a)



(b)



(c)



(d)

Figure 6-8: Moving rover planning paths around 1 dynamic robot

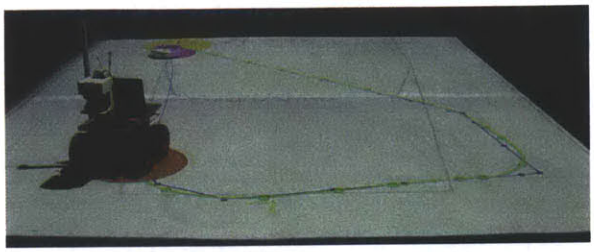
waypoints while avoiding two dynamic robots. Figure 6-9 provides an example of a particularly interesting interaction that involves the rover and both dynamic robots. The rover's initial planned path to reach the goal directly behind it takes it far to the right side of the environment (Figure 6-9(a)). By doing so, the planner is attempting to avoid both possible behaviors of the near robot. A few seconds later, DPGP has predicted that the near rover is not taking the middle crossing behavior, and so the planner identifies a more direct path to the goal (Figure 6-9(b)). While it is executing it, the DPGP expected behavior of the *far* robot begins to overlap with the rover's planned path, causing the portion near the goal to be pruned as too risky (Figure 6-9(c)). The planner then identifies a new path which maintains a larger standoff from the far rover (Figure 6-9(d)). While executing that path, the planner eventually switches to a new path taking it even further from the far rover (Figure 6-9(e)).

6.2.2 Robust Avoidance of Pedestrians

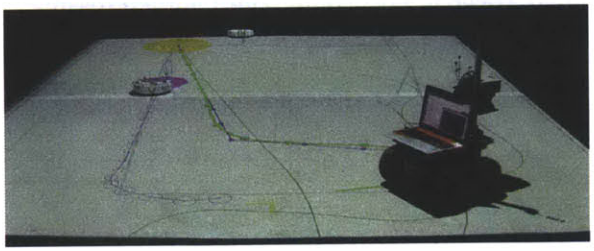
In these experiments, the autonomous rover must safely navigate around one or more pedestrians to reach a sequence of goal waypoints (Figure 6-7). Each pedestrian is detected and tracked through the onboard 2D lidar as a potential obstacle; multiple pedestrians can be tracked simultaneously [96]. The pedestrian location is mapped to a point in the environment using the rover's Vicon-based localization, which continues to be used for navigation in these experiments.

However, the pedestrians introduce two additional forms of uncertainty: sensor noise and incomplete knowledge of the environment. In the latter case, the lidar has a limited field of view, meaning that the planner's obstacle data can become out-of-date or be missing pedestrians entirely. As such, the planner may generate paths that would intersect with pedestrians because it cannot see them, and thus relies on tree updates and replanning (Algorithm 5) to safely avoid such threats. More advanced strategies are available for planning under such conditions, but are beyond the scope of this work.

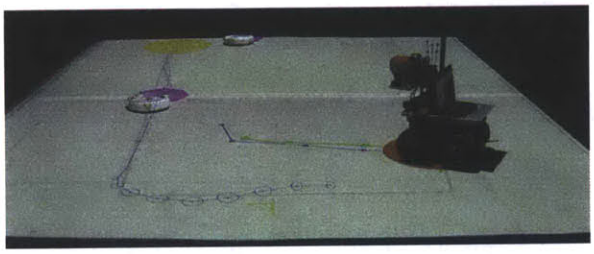
Each pedestrian is treated as a static obstacle at its sensed location. If a pedestrian drops out of the lidar field-of-view, the obstacle associated with that pedestrian will



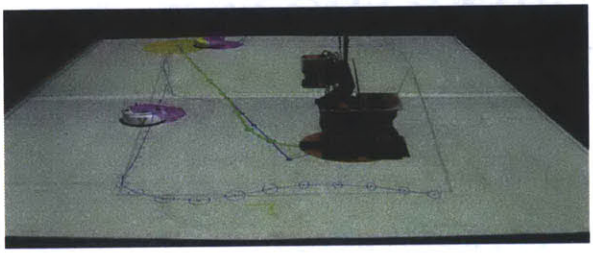
(a)



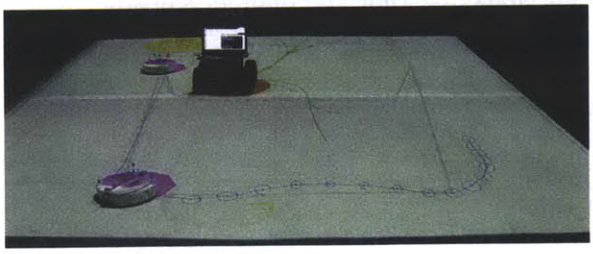
(b)



(c)



(d)



(e)

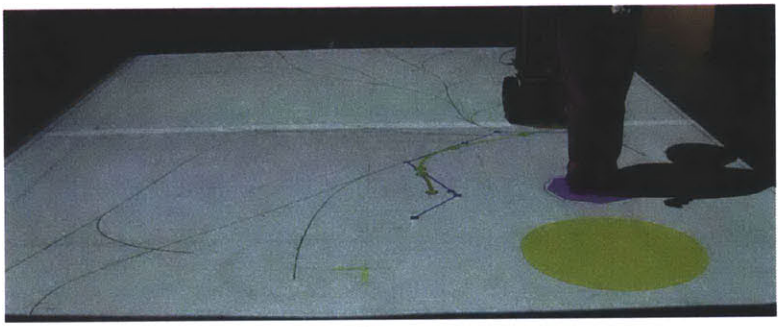
Figure 6-9: Moving rover planning paths around 2 dynamic robots

persist for several seconds before disappearing. Video 5 (Table 6.1) shows how the CC-RRT tree and motion plan evolve over time due to the presence of up to three pedestrians coming into view.

In Video 6 (Table 6.1), the rover safely navigates through a sequence of 5 goal waypoints while avoiding a pedestrian. Figure 6-10 provides an example where the rover correctly identifies a safe path to goal (Figure 6-10(a)), but is later blocked by the pedestrian moving into their path (Figure 6-10(b)). In this case, the planner correctly constructs a new path taking the rover around the pedestrian (Figure 6-10(c)), reaching the goal from the side (Figure 6-10(d)).

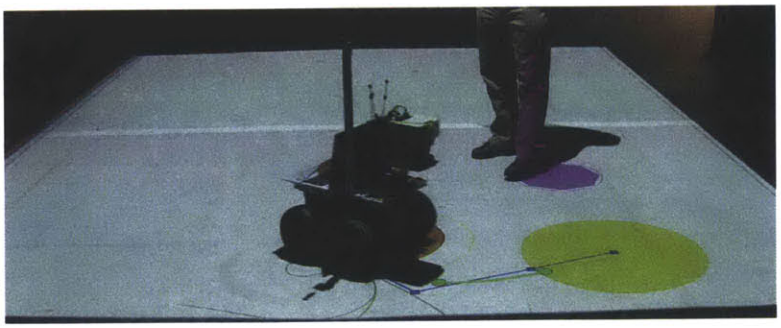
Finally, in Video 7 (Table 6.1), the rover safely navigates through a sequence of 6 goal waypoints while avoiding 2 pedestrians. Figure 6-11 provides an example of some of the planning challenges introduced by onboard sensing, and how the CC-RRT execution loop can mitigate those challenges. In this case, the planner identifies a path to the next goal, but is unaware of a pedestrian standing on that path, due to the pedestrian being out of the lidar's field of view (Figure 6-11(a)). As the rover turns, the second pedestrian is detected by the lidar, and the original path is ruled probabilistically infeasible (Figure 6-11(b)). However, the planner execution loop quickly identifies an alternate route for the rover to reach the goal (Figures 6-11(c) and 6-11(d)).

(a)



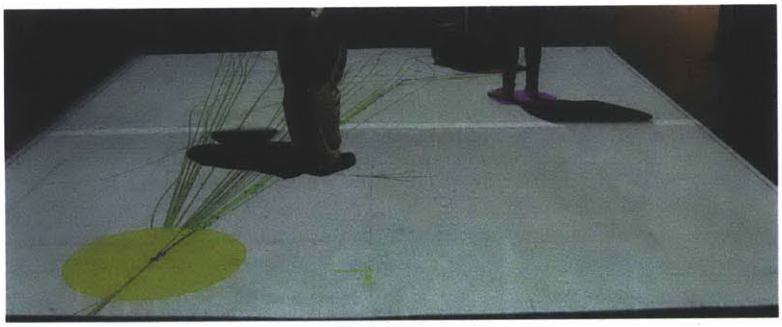
(b)

(c)



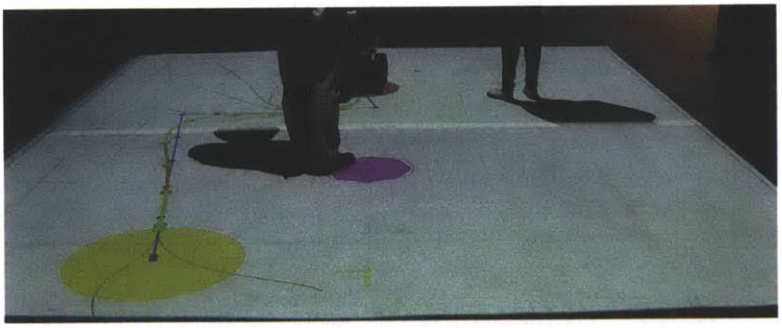
(d)

Figure 6-10: Moving rover planning paths around 1 pedestrian



(a)

(b)



(c)

(d)

Figure 6-11: Moving rover planning paths around 2 pedestrians

Chapter 7

Conclusions

This thesis has developed several novel motion planning algorithms suitable for real-world motion planning problems, which may contain a large number of dynamic and/or uncertain constraints. Unlike existing approaches in the literature, each of the algorithms developed in this work provides guarantees on constraint satisfaction subject to both internal and external uncertainty, even in complex environments, while maintaining the scalability to operate in real-time. In particular, these approach leverage the advantages of sampling-based algorithms, particularly randomly-exploring random trees, to quickly identify robustly feasible solutions that scale well with problem complexity. However, each algorithm developed in this thesis has been designed to incorporate uncertainty models efficiently, such that minimal computational overhead is required to achieve robustness, relative to the nominal motion planning algorithms.

In bounded-uncertainty RRT (BU-RRT) and bounded-uncertainty RRT* (BU-RRT*), absolute feasibility is guaranteed subject to bounded internal and external uncertainty. Problem constraints are tightened incrementally to reflect the uncertainty environment, including the possibility of incorporating disturbance feedback to expand the feasible solution space. Under appropriate assumptions, BU-RRT* has been demonstrated to be probabilistically complete and asymptotically optimal. Simulation results demonstrate that BU-RRT* can quickly grow trees of trajectories and identify low-cost solution paths that are safe for any feasible disturbance realization.

In chance-constrained RRT (CC-RRT) and chance-constrained RRT* (CC-RRT), probabilistic feasibility is guaranteed subject to Gaussian internal and external uncertainty, including dynamic obstacles with uncertain intentions. By leveraging the trajectory-wise constraint checking of RRT, the risk of constraint violation is efficiently computed and bounded, both at each timestep and along entire trajectories in the tree of state distributions, through the use of modified chance constraints. Under appropriate assumptions, CC-RRT* has been demonstrated to be probabilistically complete and asymptotically optimal. Several extensions to the CC-RRT algorithm are also proposed which approximate the chance constraints, including nonlinear dynamics, output modeling, and non-Gaussian uncertainty. Additionally, an admissable, risk-based objective function have been proposed to provide soft constraints on risk for CC-RRT*. Simulation results demonstrate that CC-RRT* can quickly generate safe trajectories with risk-averse behavior for many types of complex environments.

Finally, variations of CC-RRT have been demonstrated for a variety of problem domains and uncertainty models, including urban navigation with RR-GP/DPGP prediction of dynamic obstacles, and parafoil terminal guidance. The CC-RRT algorithm has also been demonstrated for perception-driven planning, via hardware experiments in which an autonomous rover safely navigates to avoid dynamic robots and pedestrians.

7.1 Future Work

This section briefly explores several ways in which the work in this thesis could be extended further in useful directions.

7.1.1 Analytic CC-RRT*

By leveraging the asymptotically optimal nature of RRT*, a formulation of CC-RRT* designed specifically for the parafoil terminal guidance problem (Section 4.4), *i.e.,* analytic CC-RRT*, might be able to significantly improve both the consistency in generated solution paths and the miss distance. However, adapting the parafoil

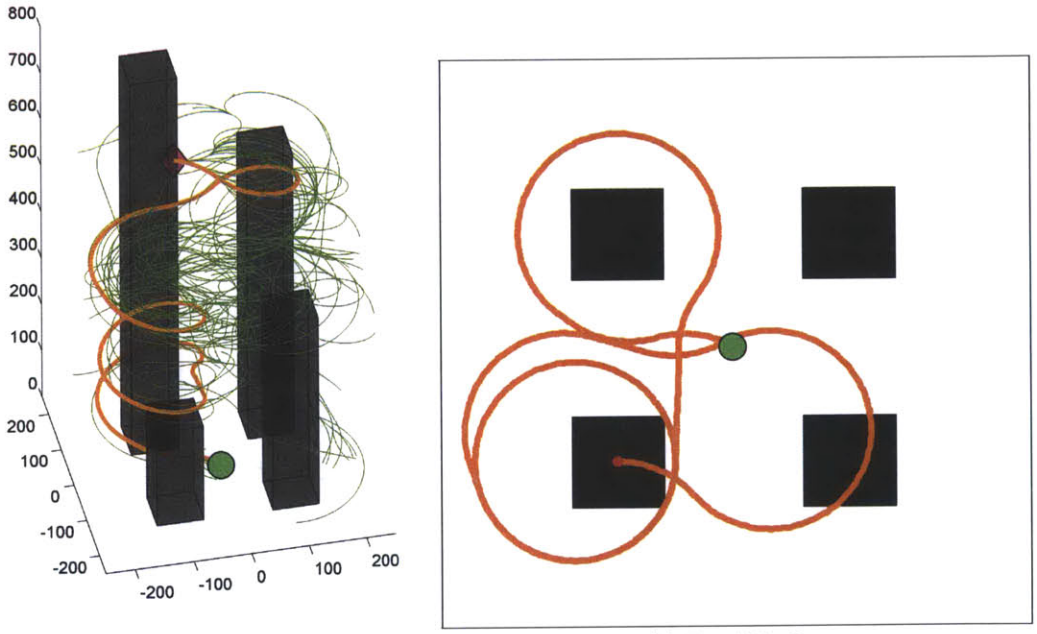
dynamics (4.22)-(4.23) into the CC-RRT* poses several challenges, most notably the development of a steering law for such a complex and underactuated (particularly in terms of altitude) system. Figure 7-1 demonstrates one possible proof-of-concept approach, in which the 3D planning problem (Figure 7-1(a)) is treated as a 2D planning problem with an arrival timing constraint (Figure 7-1(b)). In this case, the parafoil (magenta) grows a tree of trajectories (green) before finding a path (orange) guiding it to the target landing location on the ground (green circle).

7.1.2 Robust Fast Marching Trees

Many of the robustness techniques developed in this thesis for rapidly-exploring random trees could potentially be incorporated into other sampling-based motion planners with desirable properties. One such example is the recently-proposed fast marching trees (FMT) algorithm [119]. FMT utilizes a dynamic-programming-based recursion to generate asymptotically optimal paths without rewiring, and is computationally faster than RRT*, combining features of multiple-query and single-query algorithms. However, *a priori* sampling is required, limiting its usage to offline operation; the algorithm is run on a predetermined, fixed number of nodes. Figure 7-2 demonstrates the growth of a proof-of-concept "robust FMT" tree for the CC-RRT* example in Section 5.3.1, using the same robustness constraints (3.62)-(3.64) with 5000 samples.

7.1.3 Other Forms of Uncertainty

A key emphasis in the development of the algorithms in this thesis has been enabling many different kinds of uncertainty to be represented. With that in mind, there are still several additional forms of uncertainty that could be incorporated, such as parametric uncertainty, probabilities on the likelihood that obstacles exist (*e.g.*, "pop-up obstacles"), or uncertain robot/obstacle interactions.



(a) Side/3D view

(b) Top/2D view

Figure 7-1: Analytic CC-RRT* proof-of-concept

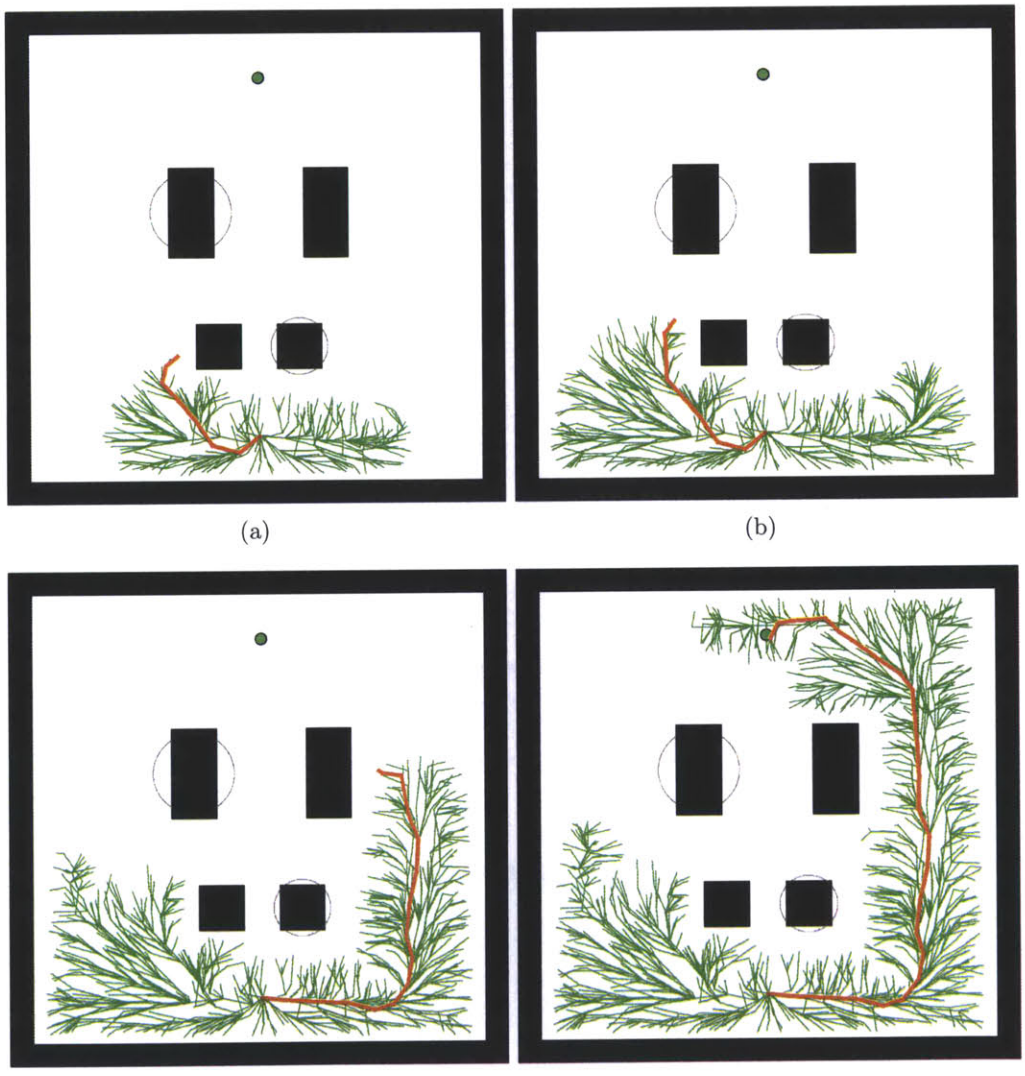


Figure 7-2: Robust FMT proof-of-concept

Bibliography

[1] J. Leonard, J. P. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, O. Koch, Y. Kuwata, D. Moore, E. Olson, S. Peters, J. Teo, R. Truax, M. Walter, D. Barrett, A. Epstein, K. Maheloni, K. Moyer, T. Jones, R. Buckley, M. Antone, R. Galejs, S. Krishnamurthy, and J. Williams. A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25(10):727–774, 2008. 19, 33

[2] Seth Teller, Matthew R. Walter, Matthew Antone, Andrew Correa, Randall Davis, Luke Fletcher, Emilio Frazzoli, Jim Glass, Jonathan P. How, Albert S. Huang, Jeong hwan Jeon, Sertac Karaman, Brandon Luders, Nicholas Roy, and Tara Sainath. A voice-commanded robotic forklift working alongside humans in minimally-prepared outdoor environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, AK, May 2010. 19

[3] S. M. LaValle and R. Sharma. On motion planning in changing, partially-predictable environments. *International Journal of Robotics Research*, 16(6):775–824, 1995. 19, 21, 105

[4] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara

Nefian, and Pamela Mahoney. Stanley: The robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, 23(9), 2006. 19

[5] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, M. N. Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas M. Howard, Sascha Kolski, Alonzo Kelly, Maxim Likhachev, Matt McNaughton, Nick Miller, Kevin Peterson, Brian Pilnick, Raj Rajkumar, Paul Rybski, Bryan Salesky, Young-Woo Seo, Sanjiv Singh, Jarrod Snider, Anthony Stentz, William Red Whittaker, Ziv Wolkowicki, Jason Ziglar, Hong Bae, Thomas Brown, Daniel Demitrish, Bakhtiar Litkouhi, Jim Nickolaou, Varsha Sadekar, Wende Zhang, Joshua Struble, Michael Taylor, Michael Darms, and Dave Ferguson. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008. 19

[6] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementation.* MIT Press, Cambridge, MA, 2005. 20

[7] S. M. LaValle. *Planning Algorithms.* Cambridge University Press, Cambridge, U.K., 2006. 20, 26

[8] B. R. Donald, P. G. Xavier, J. Canny, and J. Reif. Kinodynamic motion planning. *Journal of the Association for Computing Machinery*, 40(5):1048–1066, November 1993. 21

[9] L.E. Dubins. On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents. *American Journal of Mathematics*, 79(3):497–516, 1957. 22, 196

[10] J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–394, 1990. 22

[11] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760, 1998. 22

[12] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collisin avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997. 22

[13] T. Fraichard and H. Asama. Inevitable collision states - a step towards safer robots? *Advanced Robotics*, pages 1001–1024, 2004. 22

[14] Antoine Bautin, Luis Martinez-Gomez, and Thierry Fraichard. Inevitable collision states: a probabilistic perspective. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4022–4027. IEEE, 2010. 22

[15] Daniel Althoff, Matthias Althoff, Dirk Wollherr, and Martin Buss. Probabilistic collision state checker for crowded environments. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1492–1498. IEEE, 2010. 22

[16] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789–814, 2000. 22

[17] T. Schouwenaars, B. de Moor, E. Feron, and J. P. How. Mixed integer programming for multi-vehicle path planning. In *Proceedings of the European Control Conference*, pages 2603–2608, Porto, Portugal, September 2001. European Union Control Association. 23

[18] A. Richards and J. P. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *American Control Conference (ACC)*, volume 3, pages 1936–1941, 2002. 23

[19] A. Stentz. Optimal and efficient path planning for partially-known environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1994. 23

[20] A. Stentz. The focussed D* algorithm for real-time replanning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1652–1659, August 1995. 23

[21] S. Koenig and M. Likhachev. Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics*, 21(3):354–363, June 2005. 23

[22] D. Ferguson and A. Stentz. Field D*: An interpolation-based path planner and replanner. In *Proceedings of the International Symposium on Robotics Research*, 2005. 23

[23] M. Likhachev, G. Gordon, and S. Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. *Advances in Neural Information Processing Systems*, 16:1–8, 2003. 23

[24] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics.* MIT Press, 2005. 23

[25] A. Bemporad and M. Morari. Robust model predictive control: A survey. In *Robustness in Identification and Control*, volume 254, pages 207–226. Lecture Notes in Control and Information Sciences, 1999. 23

[26] P. O. M. Scokaert and D. Q. Mayne. Min-max model predictive control for constrained linear systems. *IEEE Transactions on Automatic Control*, 43(8):1136–1142, August 1998. 23

[27] L. Chisci, J. A. Rossiter, and G. Zappa. Systems with persistent disturbances: Predictive control with restricted constraints. *Automatica*, 37:1019 1028, 2001. 23

[28] A. G. Richards and J. P. How. Robust stable model predictive control with constraint tightening. In *Proceedings of the American Control Conference*, pages 1557–1562, Minneapolis, MN, June 2006. 23, 48

[29] Y. Kuwata. *Trajectory Planning for Unmanned Vehicles using Robust Receding Horizon Control.* PhD thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, February 2007. 23

[30] J. Löfberg. *Minimax approaches to robust model predictive control.* PhD thesis, Linköping University, 2003. 23

[31] P. J. Goulart, E. C. Kerrigan, and J. M. Maciejowski. Optimization over state feedback policies for robust control with constraints. *Automatica*, 42:523–533, 2006. 24, 37, 38

[32] P. J. Goulart and E. C. Kerrigan. Robust receding horizon control with an expected value cost. Submitted to Automatica, 2007. 24

[33] P. Li, M. Wendt, and G. Wozny. A probabilistically constrained model predictive controller. *Automatica*, 38:1171 1176, 2002. 24

[34] D. H. van Hessem. *Stochastic inequality constrained closed-loop model predictive control with application to chemcial process operation.* PhD thesis, Technische Universiteit Delft, June 2004. 24

[35] Y. Yan and R. R. Bitmead. Incorporating state estimation into model predictive control and its application to network traffic control. *Automatica*, 41:595–604, 2005. 24

[36] L. Blackmore, H. Li, and B. Williams. A probabilistic approach to optimal robust path planning with obstacles. In *American Control Conference (ACC)*, 2006. 24, 25, 77, 80, 81, 84, 85, 87, 92

[37] L. Blackmore. A probabilistic particle control approach to optimal, robust predictive control. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, 2006. 24, 34

[38] L. Blackmore. A probabilistic particle control approach to optimal robust predictive control. In *American Control Conference (ACC)*, 2007. 24

[39] L. Blackmore. Robust path planning and feedback design under stochastic uncertainty. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Honolulu, HI, August 2008. 24

[40] L. Blackmore and M. Ono. Convex chance constrained predictive control without sampling. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, 2009. 24

[41] M. Ono and B. C. Williams. Iterative risk allocation: A new approach to robust model predictive control with a joint chance constraint. In *Proceedings of the IEEE Conference on Decision and Control*, 2008. 24, 88

[42] M. Ono, L. Blackmore, and B. C. Williams. Chance constrained finite horizon optimal control with nonconvex constraints. In *Proceedings of the American Control Conference*, 2010. 24

[43] M. Ono, B. C. Williams, and L. Blackmore. Probabilistic planning for continuous dynamic systems under bounded risk. *Journal of Artificial Intelligence Research*, 46:511–577, 2013. 25

[44] Michael P. Vitus and Claire J. Tomlin. On feedback design and risk allocation in chance constrained control. In *IEEE Conference on Decision and Control and European Control Conference*, pages 734–739. IEEE, 2011. 25

[45] M. P. Vitus and Tomlin. C. J. A hybrid method for chance constrained control in uncertain environments. In *IEEE Conference on Decision and Control (CDC)*, 2012. 25

[46] Brandon Luders, Mangal Kothari, and Jonathan P. How. Chance constrained RRT for probabilistic robustness to environmental uncertainty. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, Toronto, Canada, August 2010. (AIAA-2010-8160). 25, 31, 34, 77, 155, 156, 190

[47] Noel E. Du Toit and Joel W. Burdick. Robotic motion planning in dynamic, cluttered, uncertain environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 966–973. IEEE, 2010. 25

[48] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99 134, 1998. 25

[49] Jur van den Berg, Sachin Patil, and Ron Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. *The International Journal of Robotics Research*, 31(11):1263–1278, 2012. 26

[50] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, August 1996. 26

[51] J. P. van den Berg, D. Nieuwenhuisen, L. Jaillet, and M. H. Overmars. Creating robust roadmaps for motion planning in changing environments. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2005. 27

[52] P. Cheng, E. Frazzoli, and S. LaValle. Improving the performance of sampling-based motion planning with symmetry-based gap reduction. *IEEE Transactions on Robotics*, 24(2):488–494, April 2008. 27

[53] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report 98-11, Iowa State University, October 1998. 27, 33, 77, 155

[54] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, May 2001. 27, 33, 51, 57, 100

[55] J. J. Kuffner and S. M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 995–1001, San Francisco, CA, April 2000. 27

[56] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7):846–894, June 2011. 27, 33, 35, 37, 51, 52, 53, 59, 60, 61, 64, 101, 122, 155, 156, 158, 166, 167, 168

[57] S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. In *Robotics: Science and Systems (RSS)*, 2010. 27, 33, 35, 51, 122, 155, 157, 160, 166, 168, 169

[58] S. Karaman, M.R. Walter, A. Perez, E. Frazzoli, and S. Teller. Anytime motion planning using the RRT*. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1478–1483. IEEE, 2011. 28

[59] Sertac Karaman and Emilio Frazzoli. Sampling-based optimal motion planning for non-holonomic dynamical systems. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5041–5047. IEEE, 2013. 28

[60] S. Karaman and E. Frazzoli. Incremental Sampling-based Algorithms for a class of Pursuit-Evasion Games. In *Proceedings of the Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2010. 28, 197

[61] Jeong hwan Jeon, Sertac Karaman, and Emilio Frazzoli. Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*. In *IEEE Conference on Decision and Control and European Control Conference*, pages 3276–3282. IEEE, 2011. 28

[62] J. Jeon, R. V. Cowlagi, S. C. Peters, S. Karaman, E. Frazzoli, P. Tsiotras, and K. Iagnemma. Optimal motion planning with the half-car dynamical model for autonomous high-speed driving. In *IEEE Conference on Decision and Control (CDC)*, 2012. 28

[63] Dustin J Webb and Jur van den Berg. Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5054–5061. IEEE, 2013. 28

[64] Alejandro Perez, Robert Platt, George Konidaris, Leslie Kaelbling, and Tomas Lozano-Perez. LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2537–2542. IEEE, 2012. 28

[65] R. Tedrake. LQR-Trees: Feedback motion planning on sparse randomized trees. Technical report, MIT, 2009. 28

[66] J. Barraquand, B. Langlois, and J. C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(2):224–241, March-April 1992. 28

[67] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986. 28

[68] Ron Alterovitz, Sachin Patil, and Anna Derbakova. Rapidly-exploring roadmaps: Weighing exploration vs. refinement in optimal motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3706–3712. IEEE, 2011. 28

[69] L. J. Guibas, D. Hsu, H. Kurniawati, and E. Rehman. Bounded uncertainty roadmaps for path planning. In *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics*, 2008. 28, 30

[70] R. Pepy, M. Kieffer, and E. Walter. Reliable robust path planning. *International Journal of Applied Math and Computer Science*, 1:1 11, 2009. 28

[71] N. A. Melchior and R. Simmons. Particle RRT for path planning with uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2007. 28, 35, 124

[72] G. Kewlani, G. Ishigami, and K. Iagnemma. Stochastic mobility-based path planning in uncertain environments. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pages 1183–1189, St. Louis, MO, USA, October 2009. 28

[73] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2003. 29

[74] S. Prentice and N. Roy. The belief roadmap: Efficient planning in lienar POMDPs by factoring the covariance. In *Proceedings of the International Symposium of Robotics Research*, Hiroshima, Japan, November 2007. 29, 107

[75] R. He, S. Prentice, and N. Roy. Planning in information space for a quadrotor helicopter in a GPS-denied environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1814–1820, Los Angeles, CA, 2008. 29

[76] Ron Alterovitz, Thierry Siméon, and Kenneth Y Goldberg. The stochastic motion roadmap: A sampling framework for planning with Markov motion uncertainty. In *Robotics: Science and Systems*, pages 246–253. Citeseer, 2007. 29

[77] Vu Anh Huynh, Sertac Karaman, and Emilio Frazzoli. An incremental sampling-based algorithm for stochastic optimal control. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2865–2872. IEEE, May 2012. 29, 30

[78] Vu Anh Huynh and Emilio Frazzoli. Probabilistically-sound and asymptotically-optimal algorithm for stochastic control with trajectory constraints. In *IEEE Conference on Decision and Control (CDC)*. IEEE, December 2013. 30

[79] Vu Anh Huynh, Leonid Kogan, and Emilio Frazzoli. A martingale approach and time-consistent sampling-based algorithms for risk management in stochastic optimal control. *arXiv preprint arXiv:1312.7602v1*, December 2013. 30

[80] Brandon D. Luders, Sertac Karaman, and Jonathan P. How. Robust sampling-based motion planning with asymptotic optimality guarantees. In *AIAA Guid-*

ance, Navigation, and Control Conference (GNC), Boston, MA, August 2013. 30, 35, 73

[81] Yin-Lam Chow and Marco Pavone. Stochastic optimal control with dynamic, time-consistent risk constraints. In *American Control Conference (ACC)*. IEEE, June 2013. 30

[82] A. Agha-mohammadi, S. Chakravorty, and N. M. Amato. FIRM: Feedback controller-based information-state roadmap – a framework for motion planning under uncertainty –. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4284–4291, San Francisco, CA, 2011. 30

[83] Ali-akbar Agha-mohammadi, Suman Chakravorty, and Nancy M Amato. FIRM: Sampling-based feedback motion planning under motion uncertainty and imperfect measurements. *The International Journal of Robotics Research*, 2013. 30

[84] Ali-akbar Agha-mohammadi, Saurav Agarwal, Aditya Mahadevan, Suman Chakravorty, Daniel Tomkins, Jory Denny, and Nancy M. Amato. Robust online belief space planning in changing environments: Application to physical mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014. 30

[85] P. E. Missiuro and N. Roy. Adapting probabilistic roadmaps to handle uncertain maps. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1261–1267, Orlando, FL, May 2006. 30

[86] B. Burns and O. Brock. Sampling-based motion planning with sensing uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3313–3318, Roma, Italy, April 2007. 30

[87] J. van den Berg, P. Abbeel, and K. Goldberg. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. *International Journal of Robotics Research*, 30(7):1448–1465, 2011. 30

[88] Adam Bry and Nicholas Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011. 31, 55

[89] Ian Postlethwaite and Mangal Kothari. Multi-agent motion planning for non-linear gaussian systems. *International Journal of Control*, 86(11):2075–2089, 2013. 31

[90] S. Patil, J. van den Berg, and R. Alterovitz. Estimating probability of collision for safe planning under gaussian motion and sensing uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2012. 31, 105, 107, 110

[91] M. R. Walter, M. Antone, E. Chuangsuwanich, A. Correa, R. Davis, L. Fletcher, E. Frazzoli, Y. Friedman, J. Glass, J. P. How, J. Jeon, S. Karaman, B. Luders, N. Roy, S. Tellex, and S. Teller. A Situationally-Aware Voice-Commandable Robotic Forklift Working Alongside People in Unstructured Outdoor Environments. *Journal of Field Robotics*, 2014 (accepted). 33

[92] Brandon D. Luders and Jonathan P. How. An asymptotically optimal sampling-based motion planner with guaranteed robustness to bounded uncertainty. In *American Control Conference (ACC)*, Portland, OR, June 2014 (submitted). 33

[93] P. J. Goulart and E. C. Kerrigan. Robust receding horizon control with an expected value cost. In *Proceedings of the UKACC International Conference on Control*, Glasgow, Scotland, August 2006. 33

[94] Yoshiaki Kuwata. *Trajectory Planning for Unmanned Vehicles using Robust Receding Horizon Control.* PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge MA, February 2007. 33, 37, 38, 42, 49

[95] Georges S. Aoude, Brandon D. Luders, Joshua M. Joseph, Nicholas Roy, and Jonathan P. How. Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns. *Autonomous Robots*, 35(1):51–76, 2013. 35, 123, 125, 126, 127, 128, 129, 130, 136

[96] S. Ferguson, B. Luders, R. C. Grande, and J. P. How. Real-time predictive modeling and robust avoidance of pedestrians with uncertain, changing intentions. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, Istanbul, Turkey, August 2014 (submitted). 35, 123, 127, 135, 202, 203, 207, 211

[97] Brandon Luders and Jonathan P. How. Probabilistic feasibility for nonlinear systems with non-Gaussian uncertainty using RRT. In *AIAA Infotech@Aerospace Conference*, St. Louis, MO, March 2011. (AIAA-2011-1589). 35, 124, 139

[98] Brandon Luders, Ian Sugel, and Jonathan P. How. Robust trajectory planning for autonomous parafoils under wind uncertainty. In *AIAA Infotech@Aerospace Conference*, Boston, MA, August 2013. 35, 124, 145, 146, 147, 149, 150

[99] D. Carter, L. Singh, L. Wholey, S. Rasmussen, T. Barrows, S. George, M. McConley, C. Gibson, S. Tavan, and B. Bagdonovich. Band-limited guidance and control of large parafoils. In *AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar*, 2009 (AIAA 2009-2981). 35, 151

[100] D. Q. Mayne, M. M. Seron, and S. V. Raković. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41:219–224, 2005. 48

[101] S. Prentice and N. Roy. The belief roadmap: Efficient planning in belief space by factoring the covariance. *International Journal of Robotics Research*, 28(11-12):1448–1465, 2009. 55

[102] Daniel Levine, Brandon Luders, and Jonathan P. How. Information-theoretic motion planning for constrained sensor networks. *Journal of Aerospace Information Systems*, 10(10):477–496, 2013. 55

[103] GNU Project. GNU Linear Programming Kit (GLPK), 2012. 62

[104] M. Lukasiewycz. Java ILP – Java Interface to ILP Solvers, 2008. 62

[105] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How. Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology*, 17(5):1105–1118, September 2009. 64, 97, 99, 131, 161

[106] Brandon Luders, Sertac Karaman, Emilio Frazzoli, and Jonathan P. How. Bounds on tracking error using closed-loop rapidly-exploring random trees. In *American Control Conference (ACC)*, pages 5406–5412, Baltimore, MD, June/July 2010. 64, 112, 113

[107] E. Frazzoli, M. A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. *AIAA Journal of Guidance, Control, and Dynamics*, 25(1):116–129, January-February 2002. 95, 97

[108] R. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME, Journal of Basic Engineering*, 82:35–45, 1960. 105

[109] Y. Kuwata, J. Teo, S. Karaman, G. Fiore, E. Frazzoli, and J. P. How. Motion planning in complex environments using closed-loop prediction. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, Honolulu, HI, Aug 2008. (AIAA-2008-7166). 120

[110] C. Fulgenzi, C. Tay, A. Spalanzani, and C. Laugier. Probabilistic navigation in dynamic environment using rapidly-exploring random trees and gaussian processes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1056–1062, Nice, France, September 2008. 128

[111] Joshua Joseph, Finale Doshi-Velez, A. S. Huang, and N. Roy. A Bayesian nonparametric approach to modeling motion patterns. *Autonomous Robots*, 31(4):383–400, 2011. 135

[112] Ian Sugel. Robust planning for autonomous parafoil. Master's thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge, MA, September 2013. 145, 146, 147, 148

[113] Brandon D. Luders, Georges S. Aoude, Joshua M. Joseph, Nicholas Roy, and Jonathan P. How. Probabilistically safe avoidance of dynamic obstacles with uncertain motion patterns. Technical report, Massachusetts Institute of Technology, July 2011. 197

[114] J. P. How, B. Bethke, A. Frank, D. Dale, and J. Vian. Real-time indoor autonomous vehicle test environment. *IEEE Control Systems Magazine*, 28(2):51–64, April 2008. 202, 203

[115] Motion Capture Systems from Vicon, 2011. 14 Minns Business Park, West Way, Oxford OX2 0JB, UK `http://www.vicon.com/`. 202

[116] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A.Y. Ng. ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, volume 3, 2009. 203

[117] S. Park, J. Deyst, and J. P. How. Performance and Lyapunov stability of a nonlinear path-following guidance method. *AIAA Journal on Guidance, Control, and Dynamics*, 30(6):1718–1728, November-December 2007. 203

[118] iRobot Corporation. iRobot Create programmable robot, 2010. 203

[119] L. Janson and M. Pavone. Fast marching trees: a fast marching sampling-based method for optimal motion planning in many dimensions. In *Robotics: Science and Systems (RSS)*, 2013. 219