2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 27 - October 1, 2021. Prague, Czech Republic

# Risk-Averse RRT* Planning with Nonlinear Steering and Tracking Controllers for Nonlinear Robotic Systems Under Uncertainty

Sleiman Safaoui*, Benjamin J. Gravell*, Venkatraman Renganathan*, Tyler H. Summers

2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-1714-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IROS51168.2021.9636834

*Abstract*— **We propose a two-phase risk-averse architecture for controlling stochastic nonlinear robotic systems. We present Risk-Averse Nonlinear Steering RRT* (RANS-RRT*) as an RRT* variant that incorporates nonlinear dynamics by solving a nonlinear program (NLP) and accounts for risk by approximating the state distribution and performing a distributionally robust (DR) collision check to promote safe planning. The generated plan is used as a reference for a low-level tracking controller. We demonstrate three controllers: finite horizon linear quadratic regulator (LQR) with linearized dynamics around the reference trajectory, LQR with robustness-promoting multiplicative noise terms, and a nonlinear model predictive control law (NMPC). We demonstrate the effectiveness of our algorithm using unicycle dynamics under heavy-tailed Laplace process noise in a cluttered environment.**

## I. Introduction

Safe deployment of mobile robots in uncertain dynamic environments, such as urban streets and crowded airspaces, requires a systematic accounting of various risks, both within and across layers in an autonomy stack. These autonomy stacks are naturally partitioned into a hierarchy of i) a high-level planner which generates a reference trajectory (often) offline before system operation, and ii) a low-level controller whose purpose is to track the reference trajectory in an online fashion and incorporate feedback to mitigate the effect of disturbances. The survey [12] examines several approaches for motion planning and control of autonomous ground vehicles and suggests two additional upper layers in the hierarchy, namely route planning and behavioral decision-making. In this paper, we assume such route plans and behavioral decisions are encapsulated by the motion planning and control problems.

Many motion planning algorithms have been developed under deterministic settings and assume linear robot dynamics in order to simplify their analysis and design. However, in practice, robotic systems are inherently both nonlinear and stochastic in nature due to external disturbances and noisy onboard sensors. In the presence of model uncertainty or process noise, the resulting trajectory is only a nominal reference and there are no guarantees of its safety. To account for the stochastic components and to provide probabilistic guarantees, motion planning under uncertainty has been considered in several lines of recent research [2], [18]. Specifically, risk-aware motion planning algorithms for linear robot dynamics were developed recently using CVaR- [6] and Wasserstein metric- [8] based formulations.

A chance-constrained version of RRT and RRT* respectively were proposed in [9], [10], where chance constraints were used to encode the risk of constraint violation to provide probabilistic feasibility guarantees for robots with linear dynamics under additive uncertainties. On the other hand, these approaches made questionable assumptions of Gaussianity for system uncertainties ostensibly to maintain computational tractability. It was shown in [16] that such assumptions can lead to significant miscalculations of risk, and hence moment-based ambiguity sets were formulated to propose a distributionally robust variant of RRT called DR-RRT. This approach was extended in [13] to design an asymptotically optimal RRT* using output feedback with linear quadratic regulator and Kalman filter-based state estimation. Here we take a first step towards designing risk-aware nonlinear steering-based motion plans for nonlinear robotic systems. This is closely aligned with the problem addressed by authors in [7].

A low-level tracking controller is implemented to successfully track a given reference trajectory in the presence of uncertain process disturbances. In this work, we assume perfect state estimates are available and consider full-state feedback controllers. The linear-quadratic regulator (LQR) controller being the most common can be obtained through dynamic programming where a quadratic cost involving state deviation and control effort is minimized. The LQR controller can further be generalized to achieve robust stability under parametric model uncertainties by designing it to mean-square stabilize the system with the inclusion of multiplicative noises as described in [5] (LQRm). However, both LQR controllers cannot handle state and input constraints. On the other hand, NMPC explicitly considers both state and input constraints [11]. The authors in [4] used the nonlinear model-predictive control (NMPC) to track the LQR-RRT* trajectory to make up for linearization error. By contrast, we use NMPC to track a risk-averse RRT* trajectory generated by a nonlinear program (NLP)-based steering function which much more closely resembles the low-level NMPC controller.

**Contributions:**

1) We present RANS-RRT*, a new sampling-based motion planner for nonlinear robotic systems which constructs dynamically feasible trajectories that satisfy distributionally robust state constraints to promote safety.
2) We demonstrate our proposed approach on unicycle

This work was supported in part by the Army Research Office under Grant W911NF-17-1-0058, in part by the United States Air Force Office of Scientific Research under Grant FA2386-19-1-4073, and in part by the National Science Foundation under Grant ECCS-2047040.

*Equal contribution of these authors. S. Safaoui is with the department of Electrical Engineering, and B. J. Gravell, V. Renganathan, and T. H. Summers are with the department of Mechanical Engineering at The University of Texas at Dallas, Richardson, TX, USA. E-mail: {sleiman.safaoui, benjamin.gravell, vrengana, tyler.summers}@utdallas.edu.

dynamics under heavy-tailed Laplace process noise in a cluttered environment. We provide a comparative study of the collision-avoidance rate, state deviation and control costs, and computational expense of three low-level reference tracking controllers i) LQR, ii) LQRm and iii) NMPC, across a range of disturbance strengths, through Monte Carlo simulations.

The rest of the paper is organized as follows. The notation, preliminaries and problem formulations are presented in §I. The proposed nonlinear dynamics-based high level motion planner is elucidated in §II. Low-level tracking controllers are described in §III. Simulation results are reported and analyzed in §IV. Finally, the paper is closed in §V along with directions for future research.

## NOTATIONS, PRELIMINARIES & PROBLEM FORMULATION

The set of real numbers and natural numbers are denoted by $\mathbb{R}, \mathbb{N}$ respectively. The subset of natural numbers between $a, b \in \mathbb{N}$ with $a < b$ is denoted by $[a : b]$. The operator $\setminus$ denotes set subtraction and $|\, C \,|$ denotes the cardinality of the set $C$. An identity matrix in dimension $n$ is denoted by $I_n$. The operator $(\cdot)^c$ denotes the set complement.

### A. Environment and Obstacles Specification

Consider a robot in an environment $\mathcal{X} \subseteq \mathbb{R}^n$ with static obstacles. It is expected to navigate the environment $\mathcal{X}$ while safely avoiding obstacles at all times. We denote the set of all obstacles by $\mathcal{B}$ with $|\mathcal{B}| = F > 0$. The environment and obstacles (assumed disjoint) are convex polytopes and hence can each be represented as a conjunction of halfspace constraints. The space occupied by the $i^{th}$ obstacle in $\mathcal{B}$ is denoted $\mathcal{O}_i$. The union of the space occupied by all obstacles is $\mathcal{C} := \cup_{i=1}^{F} \mathcal{O}_i$. Hence the free space in the environment is given by $\mathcal{X}_{free} := \mathcal{X} \backslash \mathcal{C} = \mathcal{X} \setminus \bigcup_{i=1}^{F} \mathcal{O}_i$ where

$$\mathcal{X} = \{x \mid A_{env} x \leq b_{env}\}, \quad \mathcal{O}_i = \{x \mid A_{ob_i} x \leq b_{ob_i}\} \forall \mathcal{O}_i \in \mathcal{B}.$$

For a given deterministic state $s \in \mathbb{R}^n$, the condition for collision avoidance with all obstacles is $s \notin \mathcal{C} \Leftrightarrow \wedge_{i=1}^{F} (s \notin \mathcal{O}_i)$ where each individual obstacle avoidance constraint can be expressed as

$$s \notin \mathcal{O}_i \Leftrightarrow \neg \left(A_{ob_i} s \leq b_{obs_i}\right) \Leftrightarrow \vee_{j=1}^{n_{ob_i}} \left(a_{ob_i,j}^{\mathsf{T}} s \geq b_{ob_i,j}\right),$$

and the condition for collision avoidance with the environment bounds is

$$s \in \mathcal{X} \Leftrightarrow \wedge_{j=1}^{n_{env}} \left(a_{env,j}^{\mathsf{T}} s \leq b_{env,j}\right),$$

where $n_{ob_i}$ are the number of constraints for obstacle $\mathcal{O}_i$ and $n_{env}$ are those of the environment. The total number of constraints is denoted $n_{total} = n_{env} + \sum_{i=1}^{F} n_{ob_i}$.

### B. Robot System Dynamics

For all time instances $k \in \mathbb{N}$, we model the robot as a discrete-time nonlinear dynamical system given by:

$$x[k+1] = f(x[k], u[k]) + w[k], \qquad x[0] = x_0, \tag{1}$$

where $x, w \in \mathbb{R}^n$, $u \in \mathbb{R}^m$ are the system state, additive disturbance, and control input, respectively, at the time step indexed in the brackets, $x_0 \in \mathbb{R}^n$ is the initial state, and $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the robot dynamics that represents the nonlinear transformation. The disturbances $w[k]$ are assumed independent and identically distributed according to some prescribed distribution $\mathbb{P}_k^w \sim (0, \Sigma_k^w)$.

### C. Unscented Transformation and Moment Estimation

The unscented transformation (UT) can be used to estimate the statistics of a random variable which undergoes a nonlinear transformation. An ensemble of $2n+1$ samples called sigma points are generated deterministically [17] and propagated individually through the nonlinear transformation to yield an ensemble of transformed sigma points. The weighted statistics of the transformed sigma points approximate the statistics of the transformed random variable. Though parameters that generate the sigma points can be tailored for specific distributions, there is no pre-defined set of rules that work in general for all distributions. For a discrete-time nonlinear robot system as in (1), we can use the UT to estimate the mean and covariance of the next state given the current one. To do so, the ensemble of $2n+1$ sigma points are obtained as follows

$$\chi_i[k] = \begin{cases} \hat{x}[k-1], & i = 0, \\ \hat{x}[k-1] + \left(\sqrt{(n+\lambda)\Sigma_x[k-1]}\right)_i, & i = [1:n] \\ \hat{x}[k-1] - \left(\sqrt{(n+\lambda)\Sigma_x[k-1]}\right)_{i-n}, & i = [n+1:2n]. \end{cases}$$

where $(\sqrt{(n+\lambda)\Sigma_x[k-1]})_i$ is the $i^{th}$ row or column of the matrix $\sqrt{(n+\lambda)\Sigma_x[k-1]}$ obtained through Cholesky decomposition. Then, the weights below are used to scale $\chi_i[k]$ in the estimation of the mean and covariance

$$W_0^{(m)} = \frac{\lambda}{n+\lambda}, W_0^{(c)} = \frac{\lambda}{n+\lambda} + 1 - \hat{\alpha}^2 + \hat{\beta}, \tag{2}$$
$$W_i^{(m)} = W_i^{(c)} = \frac{\lambda}{2(n+\lambda)}, i = [1:2n].$$

$\lambda = \hat{\alpha}^2(n+\kappa) - n$ is a scaling parameter where $\hat{\alpha}, \hat{\beta}, \kappa$ are used to tune the unscented transformation. Usually $\hat{\beta} = 2$ is a good choice for Gaussian uncertainties, $\kappa = 3 - n$ is a good choice for $\kappa$, and $0 \leq \hat{\alpha} \leq 1$ is an appropriate choice for $\hat{\alpha}$, where a larger value for $\hat{\alpha}$ spreads the sigma points further from the mean. Using the above-obtained sigma points and the weights defined in (2), the estimated mean and covariance of the random variable $x[k]$ under the dynamics (1), assuming Gaussian noise $w$, are computed as follows.

$$\xi_i[k] = f(\chi_i[k], u_i[k-1]), \quad i = [0:2n],$$
$$\hat{x}[k] \simeq \sum_{i=0}^{2n} W_i^{(m)} \xi_i[k],$$
$$\hat{\Sigma}_x[k] \simeq \sum_{i=0}^{2n} W_i^{(c)} (\xi_i[k] - \hat{x}[k])(\xi_i[k] - \hat{x}[k])^{\mathsf{T}} + \Sigma_k^w.$$

### D. Moment-Based Ambiguity Set for The State Distribution

The state $x[k]$ $\forall k \in \mathbb{N}_{>0}$ is a random vector. The state $x[k-1]$, under input $u[k-1]$ and the noise $w[k-1]$, evolves to $x[k]$. Due to the difficulty in estimating the distribution of a random variable under a nonlinear transformation, we will assume that a state $x[k]$ belongs to an unknown distribution $\mathbb{P}_k^x$. Since we can estimate its first two moments, we can consider a moment-based ambiguity set $\mathscr{P}_k^x$ with the estimated moments. This will guarantee robustness to errors in propagating the state distribution due to the nonlinear dynamics. It can also provide robustness to moment estimation errors. The mean and covariance we consider for the ambiguity set are the ones we estimate through the UT and thus we get the following estimate for the ambiguity set:

$$\mathscr{P}_k^x = \{\mathbb{P}_k^x \,|\mathbb{E}[x[k]] = \hat{x}[k], \\ \mathbb{E}[(x[k]-\hat{x}[k])(x[k]-\hat{x}[k])^\intercal] = \hat{\Sigma}_x[k]\}.$$

For Gaussian inputs, the above moment estimates from UT are accurate up to the third-order approximation and for the case of non-Gaussian, the approximations are accurate to at least the second-order as described in [17].

### E. Nonlinear Motion Planning Problem

The planning problem provides a high-level solution that can then be used by low-level tracking controllers. This plan can be computed offline and requires finding an optimal reference trajectory that satisfies the robot dynamics, state and input constraints, and risk constraints. In this work, we consider the following planning problem.

**Definition 1** (*Optimal Risk-Based Path Planning Problem*)**.** Given an initial state $x[0] \in \mathcal{X}$ and a set of final goal locations $X_{goal} \subset \mathcal{X}$, find a measurable state-and-input-history-dependent control policy $\pi = [\pi[0], \dots \pi[T-1]]$ with $u[k] = \pi[k](x[0:k], u[0:k-1])$ that minimizes the finite-horizon additive cost function subject to constraints:

$$J_{hl} = \min_{\pi} \sum_{k=0}^{T-1} g_{hl}[k]\left(\mathbb{E}\left[x[k]\right], u[k]\right) + g_{hl}[T]\left(\mathbb{E}\left[x[T]\right]\right) \quad (3)$$

$$\text{s.t. } (1)$$

$$w[k] \sim \mathbb{P}_k^w \quad (4)$$

$$u[k] \in \mathcal{U}[k], \quad (5)$$

$$\inf_{\mathbb{P}_k^x \in \mathscr{P}_k^x} \mathbb{P}_k^x\left(x[k] \in \mathcal{X}_{free}\right) \geq 1 - \alpha_k, \quad (6)$$

$$\mathbb{E}\left[x[T]\right] \in \mathcal{X}_{goal}. \quad (7)$$

Here, $g_{hl}[k]\ \forall k \in [1:T]$ is a stage cost function, (5) is the inputs constraint, (6) is the states risk constraint that ensures that the state, under the worst-case distribution, is in the free space with high probability specified through the stage risk bound $\alpha_k \in (0, 0.5]$, and (7) is the goal constraint requiring the final state to be in the goal region.

**Remark 1.** *The risk constraint* (6) *is infinite dimensional and generally non-convex which makes this problem challenging.*

To understand the stage risk constraints (6) in the context of trajectory safety, let $P^S$ denote the event that plan $P$ succeeds and $P^F$ be the complementary event (i.e. failure). Consider the plan succeeding with high probability $\mathbb{P}(P^S) \geq 1-\beta,\ \beta \in [0, 0.5]$ or equivalently it failing with low probability $\mathbb{P}(P^F) \leq \beta$. Failure requires at least one stage risk constraints to be violated. Using the fact that $\inf_{\mathbb{P}_k^x \in \mathscr{P}_k^x} \mathbb{P}_k^x\left(x[k] \in \mathcal{X}_{free}\right) \geq 1-\alpha_k$ is equivalent to $\sup_{\mathbb{P}_k^x \in \mathscr{P}_k^x} \mathbb{P}_k^x\left(x[k] \notin \mathcal{X}_{free}\right) \leq \alpha_k$ and Boole's law, the probability of the success event can be lower bounded as follows:

$$\begin{aligned}
\mathbb{P}(P^S) &= 1 - \mathbb{P}(P^F) = 1 - \mathbb{P}\left(\bigcup_{k=0}^{T} x[k] \notin \mathcal{X}_{free}\right) \\
&\geq 1 - \sup_{\mathbb{P}_k^x \in \mathscr{P}_k^x \forall k} \mathbb{P}\left(\bigcup_{k=0}^{T} x[k] \notin \mathcal{X}_{free} \;\middle|\; x[k] \sim \mathbb{P}_k^x\right) \\
&\geq 1 - \sum_{k=0}^{T} \sup_{\mathbb{P}_k^x \in \mathscr{P}_k^x} \mathbb{P}_k^x\left(x[k] \notin \mathcal{X}_{free}\right) \geq 1 - \sum_{k=0}^{T} \alpha_k = 1 - \beta
\end{aligned}$$

If the stage risks $\forall k \in [0:T]$ are equal, meaning $\alpha_k = \alpha$, then $\beta = (T+1)\alpha$. Furthermore, if the stage risk $\alpha_k = \alpha$ is equally distributed over all $n_{total}$ constraints, then, the risk bound for a single constraint is $\alpha/n_{total}$ and the risk bound for a single obstacle $\mathcal{O}_i$ (or the environment $\mathcal{X}$) is $\alpha n_{ob_i}/n_{total}$ (or $\alpha n_{env}/n_{total}$).

**Remark 2.** *In general, $T$ may not be known ahead of time. For sampling-based planners, $T$ depends on the random nodes sampled. In such cases, an upper bound can be used $T_{max} \geq T$. The choices of $T_{max}$, the risk bound on the plan's failure $\beta$, and the risk budget allocation across time steps and constraints are design parameters. Alternatively, it is also possible to build up the risk bounds from the individual constraints into a risk bound on the whole plan [15].*

### F. Nonlinear Reference Trajectory Tracking Problem

Given a reference trajectory $\bar{x}[k]$ for $k \in [0:T]$ generated by the high-level motion planner, the reference tracking problem involves minimizing deviations of the state $x[k]$ from the reference state $\bar{x}[k]$ subject to the nonlinear dynamics and realizations of all system uncertainties. This problem is formally presented below.

**Definition 2** (*Optimal reference trajectory tracking problem*)**.** Given reference trajectory $\bar{x}[k]$ for $k \in [0:T]$ such that $\bar{x}[0] = x[0]$ and $\bar{x}[T] \in X_{goal}$, find a measurable state-and-input-history-dependent control policy $\pi = [\pi[0], \dots \pi[T-1]]$ with $u[k] = \pi[k](x[0:k], u[0:k-1])$ that minimizes the finite-horizon additive cost function subject to constraints:

$$J_{ll} = \min_{\pi} \quad \mathbb{E}\left[\sum_{k=0}^{T-1} \left(g_{ll}[k](\bar{x}[k], x[k], u[k])\right) + g_{ll}[T](\bar{x}[T], x[T])\right],$$

$$\text{s.t.} \quad (1), (4), (5)$$

where $g_{ll}[k] \forall k \in [1:T]$ is a stage cost function that penalizes the control effort and deviations of the robot state from those of the reference trajectory at each time step.

## II. High Level Planner: RANS-RRT*

The high-level planner finds an optimal (or approximately optimal) plan for the low-level controller to execute. If the plan gets close to obstacles, tracking might fail due to process noise. By incorporating uncertainty in the high-level planner, a more conservative, but safe, trajectory is designed. In this work, we present an approximate solution to the problem in Definition 1 using rapidly exploring random trees. We propose RANS-RRT*: a Risk-Averse, Nonlinear Steering RRT* planner. Below, we discuss: 1) the NLP problem used to steer between tree nodes, 2) the mean and covariance propagation along such a trajectory segment, 3) the treatment of uncertainty and risk, and 4) the RANS-RRT* algorithm.

### A. NLP Steering

Our RANS-RRT* algorithm employs a nonlinear steering law to compute a trajectory $\mathscr{T}$ of length $N$ consisting of state and input pairs $\mathscr{T} = \{(s_0,u_0),\dots,(s_N,u_N)\}$ that drive an initial state $s_{init}$ to a final one $s_{des}$. The first state belongs to the RANS-RRT* tree $\mathcal{T}$ and the other is either a sampled state or another tree state. NLP steering is defined below.

**Definition 3** (NLP Steering)**.** Given an initial state $s_{init}$, a desired state $s_{des}$, and a steering horizon $N$, the NLP steering solution is the set of states $\{s_0,\dots,s_N\}$ and controls $\{u_0,\dots,u_{N-1}\}$ to the following deterministic optimization problem without any risk constraints.

$$J_{nlp} = \min_{u[0:N-1]} \sum_{k=0}^{N-1} u_k^\mathsf{T} R[k] u_k \tag{8}$$
$$\text{s.t} \quad s_0 = s_{init},\ s_N = s_{des},\ s[k+1] = f(s[k],u[k]).$$

### B. Mean and Covariance Propagation

NLP steering returns the trajectory $\mathscr{T} = \{(s_0,u_0),\dots,(s_{N-1},u_{N-1}),(s_N)\}$. To use this in the RANS-RRT* tree $\mathcal{T}$, as a trajectory between two nodes, we need the mean state, covariance of the state, and control law. Since the dynamics are nonlinear and the NMPC control policy is the solution of a constrained optimization problem without an explicit form as a function of the state, it is extremely challenging to incorporate the NMPC feedback law into the RANS-RRT* trajectories. As such, we use the open-loop controls of the NLP trajectory $\mathscr{T}$. As a byproduct, we also require the mean states in the RANS-RRT* trajectory to match the states of the NLP trajectory $\mathscr{T}$. With the mean states and inputs fixed according to $\mathscr{T}$, we only need to estimate the covariance at the mean states. To do so, we use the UT with the NLP controls and match the UT mean states with the NLP states. This returns covariances associated with every state that are used in enforcing risk bounds. Thus, for every computed NLP trajectory $\mathscr{T}$ we obtain a RANS-RRT* trajectory of the form $\text{traj}_k = \{(\hat{x}[Nk],\hat{\Sigma}[Nk],u[Nk]),\dots,(\hat{x}[N(k+1)],\hat{\Sigma}[N(k+1)])\}$ where $\hat{x}[Nk+i] = s_i$ and $u[NK+i] = u_i$ for all $i$.

### C. Risk Treatment

Consider a mean state and covariance pair in the RANS-RRT* trajectories $(\hat{x}[k],\hat{\Sigma}[k])$. The risk constraint associate with this time step has the form $\inf_{\mathbb{P}_k^x\in\mathscr{P}_k^x}\mathbb{P}_k^x(x[k]\in\mathcal{X}_{free}) \geq 1-\alpha_k \Leftrightarrow \sup_{\mathbb{P}_k^x\in\mathscr{P}_k^x}\mathbb{P}_k^x(x[k]\in\mathcal{C}\cup\mathcal{X}^c) \leq \alpha_k$. Since $\mathcal{C}$ is a union of the obstacle sets, we use Boole's law to get:

$$\sup_{\mathbb{P}_k^x\in\mathscr{P}_k^x}\mathbb{P}_k^x(x[k]\in\mathcal{C}\cup\mathcal{X}^c) \leq \sum_{i=1}^{F}\sup_{\mathbb{P}_k^x\in\mathscr{P}_k^x}\mathbb{P}_k^x(x[k]\in\mathcal{O}_i) + \sup_{\mathbb{P}_k^x\in\mathscr{P}_k^x}\mathbb{P}_k^x(x[k]\in\mathcal{X}^c).$$

We allocate the risk bound equally among the constraints by setting the risk bound for being in $\mathcal{O}_i$ to $\alpha_k n_{ob_i}/n_{total}$ and that of not being in the environment to $\alpha_k n_{env}/n_{total}$. Thus:

$$\sum_{i=1}^{F}\sup_{\mathbb{P}_k^x\in\mathscr{P}_k^x}\mathbb{P}_k^x(x[k]\in\mathcal{O}_i) + \sup_{\mathbb{P}_k^x\in\mathscr{P}_k^x}\mathbb{P}_k^x(x[k]\in\mathcal{X}^c) \leq \sum_{i=1}^{F}\alpha_k n_{ob_i}/n_{total} + \alpha_k n_{env}/n_{total} = \alpha_k,$$

and hence, the desired risk constraint is enforced. Now we turn our attention to satisfying:

$$\sup_{\mathbb{P}_k^x\in\mathscr{P}_k^x}\mathbb{P}_k^x(x[k]\in\mathcal{X}^c) \leq \alpha_k n_{env}/n_{total} \tag{9}$$
$$\sup_{\mathbb{P}_k^x\in\mathscr{P}_k^x}\mathbb{P}_k^x(x[k]\in\mathcal{O}_i) \leq \alpha_k n_{ob_i}/n_{total} \tag{10}$$

Since $x[k]\in\mathcal{X}^c \Leftrightarrow A_{env}x[k] > b_{env} \Leftrightarrow \wedge_{j=1}^{n_{env}} a_{env,j}^\mathsf{T} x[k] > b_{env,j}$, we can apply Boole's inequality to (9) and get

$$\sum_{j=1}^{n_{env}}\sup_{\mathbb{P}_k^x\in\mathscr{P}_k^x}\mathbb{P}_k^x\left(a_{env,j}^\mathsf{T} x[k] > b_{env,j}\right) \leq \alpha_k n_{env}/n_{total}.$$

This bound is satisfied if $\sup_{\mathbb{P}_k^x\in\mathscr{P}_k^x}\mathbb{P}_k^x\left(a_{env,j}^\mathsf{T} x[k] > b_{env,j}\right) \leq \alpha_k/n_{total}\ \forall j\in[1:n_{env}]$. As for (10), we have the following:

$$\begin{aligned}
(10) &\Leftrightarrow \inf_{\mathbb{P}_k^x\in\mathscr{P}_k^x}\mathbb{P}_k^x(x[k]\notin\mathcal{O}_i) > 1-\alpha_k n_{ob_i}/n_{total} \\
&\Leftrightarrow \inf_{\mathbb{P}_k^x\in\mathscr{P}_k^x}\mathbb{P}_k^x\left(\vee_{j=1}^{n_{ob_i}} a_{ob_i,j}^\mathsf{T} x > b_{ob_i,j}\right) > 1-\alpha_k n_{ob_i}/n_{total}, \\
\text{and} \quad & \inf_{\mathbb{P}_k^x\in\mathscr{P}_k^x}\mathbb{P}_k^x\left(\vee_{j=1}^{n_{ob_i}} a_{ob_i,j}^\mathsf{T} x > b_{ob_i,j}\right) \\
&> \inf_{\mathbb{P}_k^x\in\mathscr{P}_k^x}\max_j \mathbb{P}_k^x\left(a_{ob_i,j}^\mathsf{T} x > b_{ob_i,j}\right) \qquad (11) \\
&\geq \inf_{\mathbb{P}_k^x\in\mathscr{P}_k^x}\mathbb{P}_k^x\left(a_{ob_i,j}^\mathsf{T} x > b_{ob_i,j}\right)
\end{aligned}$$

where (11) follows from the Fréchet-Boole lower bound. If $\inf_{\mathbb{P}_k^x\in\mathscr{P}_k^x}\mathbb{P}_k^x\left(a_{ob_i,j}^\mathsf{T} x[k] > b_{ob_i,j}\right) > 1-\alpha_k/n_{total}\ \forall j$ then, $\inf_{\mathbb{P}_k^x\in\mathscr{P}_k^x}\mathbb{P}_k^x(x[k]\notin\mathcal{O}_i) > 1-\alpha_k/n_{total} \geq 1-\alpha_k n_{ob_i}/n_{total}$ which holds as $n_{ob_i}\geq 1$. Thus, (6) is satisfied if

$$\sup_{\mathbb{P}_k^x\in\mathscr{P}_k^x}\mathbb{P}_k^x\left(a_{env,j}^\mathsf{T} x[k] > b_{env,j}\right) \leq \alpha_k/n_{total}\ \forall j$$
$$\inf_{\mathbb{P}_k^x\in\mathscr{P}_k^x}\mathbb{P}_k^x\left(a_{ob_i,j}^\mathsf{T} x[k] > b_{ob_i,j}\right) > 1-\alpha_k/n_{total}\ \forall i,j$$

Using [3, Theorem 3.1] these are equivalent to:

$$a_{env,j}^\mathsf{T}\hat{x}[k] \leq b_{env,j} - \sqrt{\frac{1-\alpha_k/n_{total}}{\alpha_k/n_{total}}}\left\|\hat{\Sigma}[k]^{1/2}a_{env,j}\right\|_2 \tag{12}$$
$$a_{ob_i,j}^\mathsf{T}\hat{x}[k] \geq b_{ob_i,j} + \sqrt{\frac{1-\alpha_k/n_{total}}{\alpha_k/n_{total}}}\left\|\hat{\Sigma}[k]^{1/2}a_{ob_i,j}\right\|_2. \tag{13}$$

### D. Algorithm

RANS-RRT* is presented in Algorithm 1. The free space is first sampled and the tree is initialized with the initial state (the root of the tree) and the initial covariance of zero (lines 1-2). Then the sampled states $x_{samples}$ are traversed. For every state $S \in x_{samples}$, the nearest node in the tree $S_{nearest} \in \mathcal{T}$ is found (line 4). If the distance between $S$ and $S_{nearest}$ is larger than some threshold, a closer state $S_{lim}$ is returned along the same direction (line 5). In line 6, NLP steering is performed to find a trajectory from $S_{nearest}$ to $S_{lim}$ within the steering horizon $N$. If steering fails, the sample is skipped. Else, a RANS-RRT* trajectory (mean states, covariances, and inputs) is returned. In line 9, the trajectory is checked for collisions. A collision is detected if any of the risk-tightened constraints are violated at any of the mean states, in which case the sample is skipped. If safe, the trajectory may be added to the tree after checking nearby nodes for a more optimal trajectory as done in RRT* (line 12). Then, the trajectory is added to the tree (line 13) and $\mathcal{T}$ is rewired (line 14). Note that the rewire step also uses the same steering and collision check functions described before. When an optimal trajectory is queried, the trajectory with the smallest total NLP steering cost (output cost of Definition 3) is returned.

**Algorithm 1:** RANS-RRT* - Tree Expansion

```
Result: RANS-RRT* Tree 𝒯
1  S_samples = sample(𝒳_free);
2  𝒯 = [(x̂_0, Σ_0 = 0)];
3  for S ∈ x_samples do
4      S_nearest = nearestNode(S, 𝒯);
5      S_lim = limitDistance(S, S_nearest);
6      (success, traj) = steer(S_nearest, S_lim);
7      if not success then
8          Continue;
9      collision = checkDRCollision(traj);
10     if collision then
11         Continue;
12     traj = connectViaMinCostPath(traj, 𝒯);
13     𝒯.addTrajNode(traj);
14     𝒯.rewire();
```

**Remark 3.** *RANS-RRT* only* approximately *solves the problem in Definition 1 because 1) the risk treatment step* approximately *solves the infinite-dimensional constraint* (6)*, 2) the covariance estimation step using UT is* imperfect *for non-Gaussian distributions, and 3) the covariance propagation assumes the estimated means and NLP means* coincide*, which is used to keep the problem tractable. Hence, we do not make any formal risk-bound guarantees. Nonetheless, as we will see in the experimental results section, the padding added to obstacles through the DR collision check step makes the algorithm significantly robust to disturbances.*

## III. Tracking Controllers

In all low-level controllers we use identical cost functions. This facilitates a fair comparison between controllers, as the same objective is approximately optimized. We use stage costs which are quadratic in the state deviation and input:

$$\begin{aligned} g_{ll}[k](\bar{x}[k], x[k], u[k]) &= \delta_x[k]^\intercal Q[k] \delta_x[k] + u[k]^\intercal R u[k] \\ g_{ll}[T](\bar{x}[T], x[T]) &= \delta_x[T]^\intercal Q[T] \delta_x[T] \end{aligned}$$

where $\delta_x[k] = x[k] - \bar{x}[k]$ and the penalty matrices $Q[k]$ and $R[k]$ are symmetric positive definite for all $k$.

### A. Robot dynamics

We consider the problem of navigating a robot with unicycle dynamics from an initial state to a final set of states. The discrete-time unicycle nonlinear dynamics obtained through forward-Euler discretization of the corresponding continuous-time dynamics are given by:

$$\begin{aligned} p_x[k+1] &= p_x[k] + \cos(\theta[k]) v[k] \Delta t + w_x[k] \Delta t \\ p_y[k+1] &= p_y[k] + \sin(\theta[k]) v[k] \Delta t + w_y[k] \Delta t \\ \theta[k+1] &= \theta[k] + \omega[k] \Delta t + w_\theta[k] \Delta t \end{aligned} \tag{14}$$

where $p_x[k], p_y[k] \in \mathbb{R}$ are the horizontal and vertical positions of the robot, $\theta[k] \in \mathbb{R}$ is the heading of the robot relative to the $x$-axis of an assigned world frame, $v[k], \omega[k] \in \mathbb{R}$ are the linear and angular velocity control inputs, and $w_x[k], w_y[k], w_\theta[k]$ are the disturbances affecting each state, all expressed at timestamp $k$. The quantity $\Delta t$ is the sampling time in seconds between any two timestamps $k, k+1$. With short-hand notations, $x[k] = \begin{bmatrix} p_x[k] & p_y[k] & \theta[k] \end{bmatrix}^\intercal, u[k] = \begin{bmatrix} v[k] & \omega[k] \end{bmatrix}^\intercal, w[k] = \begin{bmatrix} w_x[k] & w_y[k] & w_\theta[k] \end{bmatrix}^\intercal$, the dynamics in (14) and corresponding Jacobians can be written in the compact form as

$$x[k+1] = f(x[k], u[k]) + w[k],$$

$$\frac{\partial f}{\partial x}\Big|_{\bar{x},\bar{u}} = \begin{bmatrix} 1 & 0 & -v \sin(\theta) \Delta t \\ 0 & 1 & v \cos(\theta) \Delta t \\ 0 & 0 & 1 \end{bmatrix}, \quad \frac{\partial f}{\partial u}\Big|_{\bar{x},\bar{u}} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & \Delta t \end{bmatrix}.$$

### B. LQR

Finite-horizon linear-quadratic regulation (LQR) about a reference trajectory is a standard optimal control approach wherein the system dynamics are linearized about the reference trajectory at each time $k$

$$A[k] = \frac{\partial f}{\partial x}\Big|_{\bar{x}[k],\bar{u}[k]}, \quad B[k] = \frac{\partial f}{\partial u}\Big|_{\bar{x}[k],\bar{u}[k]},$$

and thereby yielding the LQR problem

$$\begin{aligned} &\text{minimize} && J_{ll} \\ &\text{subject to} && \delta_x[k+1] = A[k] \delta_x[k] + B[k] \delta_u[k]. \end{aligned}$$

The solution is time-varying affine state-feedback characterized by a sequence of gain matrices and vectors $\{K[k], L[k], e[k]\}_{k=0}^{T-1}$ which are pre-computed before running the system using linear-algebraic operations according to dynamic programming relations. For brevity, we defer the details to the supplementary report [14]. At runtime, the control inputs are computed as

$$u[k] = K[k] \delta_x[k] + L[k] \begin{bmatrix} \bar{x}[k] \\ \bar{u}[k] \end{bmatrix} + e[k] + \bar{u}[k],$$

which is the summation of a closed-loop feedback term of the state-deviation and an open-loop feedforward term.

### C. Robust LQR

We seek to promote robustness against errors in the state-space matrices due to linearization about states other than the reference trajectory, which occurs due to the process disturbance. Observing the Jacobians, only mis-specifications in heading $\theta$ change the entries. We assume that the heading deviation from reference is uniformly upper bounded as

$$|\theta[k]-\theta_r[k]| \leq \delta\theta_{\max} \leq \pi/2.$$

For the unicycle model, it is straightforward to construct appropriate 2D bounding boxes in the space of A and B matrices that fully contain every possible Jacobian. Consider the $1,1$- and $2,1$-entries of the $B$ matrix

$$b = \begin{bmatrix} B_{11} \\ B_{21} \end{bmatrix} = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}.$$

Through trigonometric relations depicted by Figure 1, we obtain the robustness regions

$$A[k] \in \{A: \ A = \overline{A}[k] + \mu_{A_1}[k]A_1[k] + \mu_{A_2}[k]A_2[k]\}$$
$$B[k] \in \{B: \ B = \overline{B}[k] + \mu_{B_1}[k]B_1[k] + \mu_{B_2}[k]B_2[k]\}$$

determined by directions

$$A_1[k] = \begin{bmatrix} 0 & 0 & -s[k] \\ 0 & 0 & c[k] \\ 0 & 0 & 0 \end{bmatrix}, A_2[k] = \begin{bmatrix} 0 & 0 & -c[k] \\ 0 & 0 & -s[k] \\ 0 & 0 & 0 \end{bmatrix},$$
$$B_1[k] = \begin{bmatrix} -s[k] & 0 \\ c[k] & 0 \\ 0 & 0 \end{bmatrix}, \qquad B_2[k] = \begin{bmatrix} c[k] & 0 \\ s[k] & 0 \\ 0 & 0 \end{bmatrix}$$

where $c[k] = \cos(\theta[k]), s[k] = \sin(\theta[k])$ and where the scales are bounded as

$$|\mu_{A_i}[k]| \leq \sigma_{A_i}[k], \quad |\mu_{B_i}[k]| \leq \sigma_{B_i}[k]$$

where

$$\sigma_{A_1}[k] = v[k]\Delta t \sin(\delta\theta_{\max}), \quad \sigma_{A_2}[k] = v[k]\Delta t\,[1-\cos(\delta\theta_{\max})],$$
$$\sigma_{B_1}[k] = \sin(\delta\theta_{\max}), \qquad \sigma_{B_2}[k] = 1-\cos(\delta\theta_{\max})$$

Note that the robustness region is conservative as it extends in the $A_2$ and $B_2$ directions twice as far as strictly necessary. This is done merely as a matter of convenience so that the center of the robustness region remains at ($A[k]$, $B[k]$) regardless of $\delta\theta_{\max}$. It has been proved in [5] that, in the infinite-horizon time-invariant setting, the inclusion of (fictitious) multiplicative noise in the control design induces robustness to static model perturbations in the same directions as the multiplicative noise. This inspires the inclusion of such multiplicative noises with variance and directions related to the desired robustness region over which we desire to minimize quadratic cost. Please refer to the supplementary report [14] for details on how the multiplicative noise is incorporated into the LQR problem. We include the robust LQR in our comparison to demonstrate the benefit of our NMPC controller over a simpler robust control method.

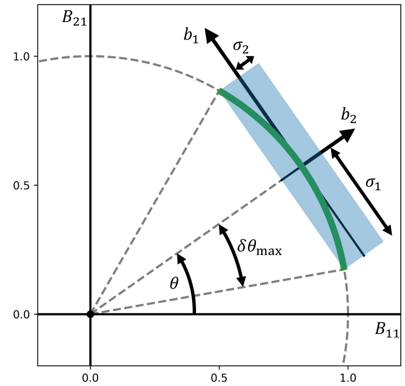



Fig. 1. Geometry of $B$ matrices under linearization about various states. The thick circular arc segment is the locus of all possible $B$ when $\theta$ is interval bounded. The shaded box represents the set of $B$ on which the controller is designed to achieve low cost.

### D. NMPC

NMPC can be used to approximate the nonlinear optimal trajectory tracking problem. NMPC accomplishes that by reducing the problem into a sequence of open-loop optimization problems over a horizon $N_{ll}$ where after solving an NLP to track a reference trajectory, only the first input is applied and the horizon is shifted back. The NMPC tracking NLP is given below.

**Definition 4** (NMPC Tracking NLP)**.** Given an reference trajectory $\overline{x}[k]$ for $k \in [0:T]$ and a planning horizon $N_{ll}$, find a control sequence at time step $t$ that minimizes the $N_{ll}$-horizon additive cost function subject to constraints:

$$\min_{u[t:T+N_{ll}-1]} \quad \sum_{k=t}^{t+N_{ll}-1} g_{ll}[k](\overline{x}[k], x[k], u[k]) + g_{ll}[T](\overline{x}[T], x[T])$$
$$(5), \ x[k] \in \mathcal{X}, \ x[k+1] = f(x[k], u[k])$$

After solving this problem at step $t$, only the first input $u[t]$ is applied and the horizon is shifted to $t+1$ for the problem to be solved again. The final set of control inputs is thus $u^*_{0:T-1} = [u[0], \ldots u[T-1]]$.

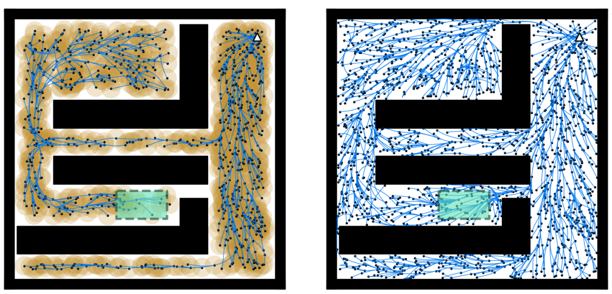
## IV. Numerical Results

RANS-RRT* plans were generated on a machine with an Intel Core i7 6700K CPU and 16 of RAM. The low-level Monte Carlo simulations were performed on a machine with a Ryzen 7 2700X and 64GB of RAM. The NLP is modeled with CasADi Opti and solved with IPOPT [1]. The environment consisted of a root node (white triangle), a goal area (dashed green rectangle), a $10 \times 10$ environment with 4 rectangular obstacles for its sides (black boundary), and 5 rectangular obstacles (black rectangles). The robot is assumed to occupy a single point. Its controls bounds are $\pm 0.5$ units/sec for linear velocity and $\pm\pi$rad/sec for angular velocity. The RANS-RRT* steering horizon is $N = 30$ and the NMPC planning horizon is $N_{ll} = 10$. The discrete time step was $\delta t = 0.2$sec. The planner control cost matrix was $R[k] = \mathrm{diag}([1,1])$. The tracking (LQR, robust LQR, and NMPC) cost matrices were $Q[k] = \mathrm{diag}([100,100,10])$ and $R[k] = \mathrm{diag}([1,1])$ for all $k = [0:T-1]$, and $Q[T] = 10Q[0]$.

We used a high-level plan risk bound of $\beta = 0.1$ divided equally across the time steps $T_{max} = 1000$ and among the obstacle constraints. The process noise distribution $\mathbb{P}_k^w$ was taken as a multivariate Laplace distribution with zero mean and covariance $\Sigma^w = \sigma_w^2 I_n$ where the variance $\sigma_w^2 := 5 \cdot 10^{-7}$ is referred to as the noise level. In the tracking step, we evaluated performance under different (greater) noise levels.

### A. RANS-RRT* Results

Constructing the RANS-RRT* tree in Figure 2a took about 28 minutes. The tree started with 2000 randomly sampled nodes but only 865 nodes were deemed feasible and safe and added to the tree. The RANS-RRT* trajectories were safe in the sense that they avoided getting too close to obstacles. Small gaps, such as to the right of the goal, were implicitly deemed too risky and avoided. On the other hand, a tree grown using the standard RRT* in the same environment in Figure 2b discovered such gaps and thereby returned an unsafe (risky) trajectory.



(a) RANS-RRT* tree with 942 Nodes. (b) RRT* tree with 1284 Nodes.

Fig. 2. Trees grown by our proposed RANS-RRT* algorithm and standard RRT* are demonstrated in Figures 2a and 2b respectively. The DR collision checks are represented by circles. The tree root is indicated by the white triangle and the goal region is the green rectangle with a dashed edge. Obstacles, including environment bounds are the black rectangles.

### B. Low-Level Tracking Results

We used open-loop control and three low-level closed-loop controllers 1) LQR, 2) robust LQR, and 3) NMPC, to track a high-level trajectory in the environment shown in Figure 2a under realization of the Laplace noise. For each noise level, 1000 Monte Carlo simulations were performed. The resulting trajectories are plotted in Figure 3. As expected, the noise level assumed for the high-level plan $\sigma_w^2 = 5 \cdot 10^{-7}$ was insignificant: even open-loop control succeeded. However, as the noise level increased, open-loop control began to fail. At around $\sigma_w^2 = 0.001$, the open-loop control almost always failed, while the other controllers almost always succeeded. From there, the feedback controllers began failing more frequently. Robust LQR did slightly better than LQR with fewer collisions for each noise level. Both were outperformed by NMPC which was better able to reject the more aggressive noise, leading to notably fewer collisions. The largest noise levels tested for which the plan failure risk bound of 10 percent was satisfied were 0.003 for LQR and robust LQR and 0.0035 for NMPC.

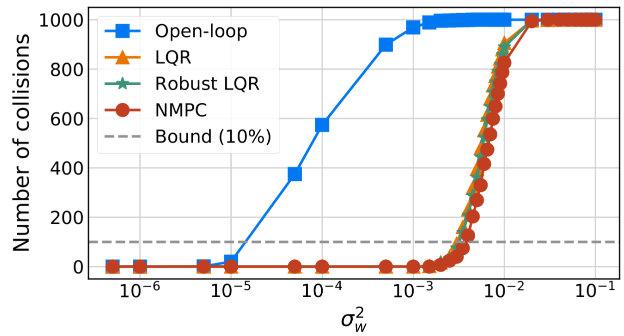


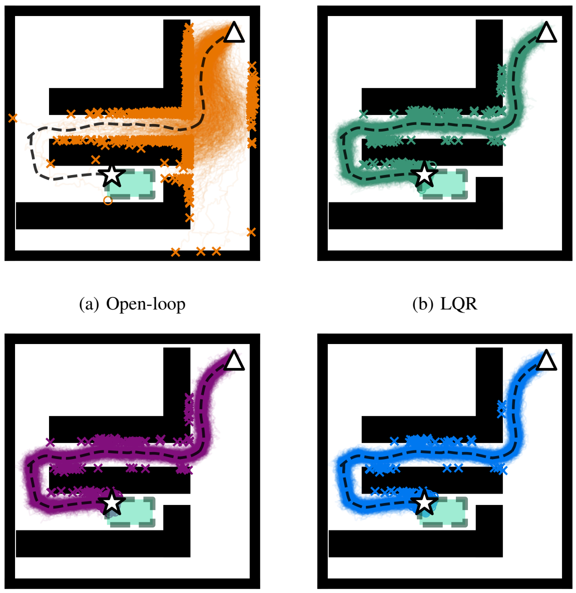

Fig. 3. Number of failures for each controller across a range of noise covariance levels. 1000 Monte Carlo trials are run for each noise value.

In Figure 4 the realized trajectories obtained through the Monte Carlo simulations are plotted for $\sigma_w^2 = 0.0035$. The performance metrics for $\sigma_w^2 = 0.0000005$ and $\sigma_w^2 = 0.0035$ are tabulated in Tables I and II respectively. We use the following metrics to evaluate the effectiveness of each tracking controller:

1) The number of collisions: the number of Monte Carlo trials in which the realized trajectory collided with an obstacle.
2) The average $\delta_x$ and $u$ costs: the realized state-deviation cost $\sum_{k=0}^{T} \delta_x[k]^\intercal Q[k] \delta_x[k]$ and control cost $\sum_{k=0}^{T-1} u[k]^\intercal R u[k]$, conditionally averaged across all collision-free trajectories.
3) The average run time: the time taken to run the entire Monte Carlo trial, conditionally averaged across all collision-free trajectories, which is necessary as simulations terminate immediately upon collision.

With $\sigma_w^2 = 0.0000005$, the trajectory was short enough so that collisions did not occur even with purely open-loop control, although the trajectories began to diverge from the reference. The closed-loop controllers exhibited minimal state deviations, as reflected in the significantly lower average $\delta_x$ cost. The difference in $\delta_x$ and $u$ costs between each closed-loop controller was insignificant; since the state remained extremely close to the reference, the inputs generated by each controller were very similar.

However with $\sigma_w^2 = 0.0035$ as shown in Figure 4, the following observations were made. Almost all open-loop trajectories ended with collisions as shown in Figure 4a. Compared to the standard LQR, robust LQR led to a lower number of collisions and state-deviation cost as shown in Figures 4b, 4c but both were significantly outperformed by NMPC. NMPC trajectories generally remained closer to the reference than LQR or robust LQR along with lower number of collision as the robot moved through the corridor as shown in Figure 4d. However, NMPC's closer reference tracking and collision-avoidance came at a price, as it used more control effort than LQR and robust LQR. Likewise, the more sophisticated computations involved in solving the NLPs in NMPC led to a longer average run time. It is evident that under both the noise settings, the NMPC outperforms other controllers in tracking the given reference trajectory to reach the goal with low failure rate and a better cost.



(a) Open-loop (b) LQR

(c) LQRm (d) NMPC

Fig. 4. The results of 1000 independent Monte Carlo trials with different low-level reference tracking controllers. The reference trajectory is shown as a dashed line. The start and goal locations are marked by triangle and star icons respectively. The goal area is a green rectangular area whose border is marked by a dashed line. The terminal state of failed and successful trajectories are shown by 'X' and 'O' markers respectively. Obstacles and the free space are represented by solid black and white regions, respectively. The disturbance variance was $\sigma_w^2 = 0.0035$.

| Controller | Num. collisions | $\delta_x$ cost | $u$ cost | Run time (s) |
|---|---|---|---|---|
| Open-loop | 0 | 30.869 | 54.989 | 0.0103 |
| LQR | 0 | 3.403 | 43.267 | 0.1867 |
| LQRm | 0 | 2.911 | 44.351 | 0.2833 |
| NMPC | 0 | 3.522 | 43.240 | 5.6234 |

TABLE I. Performance metrics for all controllers from Monte Carlo simulation with $\sigma_w^2 = 0.0000005$.

| Controller | Num. collisions | $\delta_x$ cost | $u$ cost | Run time (s) |
|---|---|---|---|---|
| Open-loop | 999 | 5103.148 | 54.989 | 0.0018 |
| LQR | 160 | 827.949 | 110.016 | 0.1751 |
| LQRm | 138 | 820.916 | 114.646 | 0.2946 |
| NMPC | 75 | 732.646 | 131.059 | 6.9159 |

TABLE II. Performance metrics for all controllers from Monte Carlo simulation with $\sigma_w^2 = 0.0035$.

## V. Conclusion and Future Work

We proposed a risk-averse control architecture tailored for safely controlling stochastic nonlinear robotic systems, which combines a novel nonlinear steering-based variant of RRT* called RANS-RRT* that accounts for risk by performing DR collision checks with low-level reference tracking controllers. We performed thorough numerical experiments using unicycle dynamics, compared three controllers, and observed better performance from NMPC than LQR variants. We showed that the despite the usage of very small noise level assumptions in the high-level planner, the low-level controllers performed well under moderate and aggressive disturbance realizations. Future research involves considering a full nonlinear sensor model while incorporating the exact DR risk constraints in the optimization problems of both the levels of autonomy stack for accurate risk assessment. We will also seek to decrease the computation time of NMPC through the usage of code generation tools.

## References


[1] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

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

[3] G. C. Calafiore and L. El Ghaoui, "On distributionally robust chance-constrained linear programs," *Journal of Optimization Theory and Applications*, vol. 130, no. 1, pp. 1–22, 2006.

[4] B. Doerr and R. Linares, "Motion planning and control for on-orbit assembly using LQR-RRT* and nonlinear MPC," *arXiv preprint arXiv:2008.02846*, 2020.

[5] B. Gravell, P. M. Esfahani, and T. Summers, "Robust control design for linear systems via multiplicative noise," in *Proceedings of the IFAC World Congress, Berlin, Germany, 13-17 July 2020*, 2020.

[6] A. Hakobyan, G. C. Kim, and I. Yang, "Risk-aware motion planning and control using CVaR-constrained optimization," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3924–3931, 2019.

[7] A. Hakobyan and I. Yang, "Wasserstein distributionally robust motion control for collision avoidance using conditional value-at-risk," *arXiv preprint arXiv:2001.04727*, 2020.

[8] ——, "Wasserstein distributionally robust motion planning and control with safety constraints using conditional value-at-risk," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 490–496.

[9] B. Luders, M. Kothari, and J. How, "Chance constrained RRT for probabilistic robustness to environmental uncertainty," in *AIAA guidance, navigation, and control conference*, 2010, p. 8160.

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

[11] L. Magni, D. M. Raimondo, and F. Allgöwer, "Nonlinear model predictive control," *Lecture Notes in Control and Information Sciences*, vol. 384, 2009.

[12] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on intelligent vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[13] V. Renganathan, I. Shames, and T. H. Summers, "Towards integrated perception and motion planning with distributionally robust risk constraints," *Proceedings of the IFAC World Congress, Berlin, Germany, 13-17 July 2020*, 2020.

[14] S. Safaoui, B. Gravell, V. Renganathan, and T. H. Summers, "Risk-averse RRT* planning with nonlinear steering and tracking controllers for nonlinear robotic systems under uncertainty," https://personal.utdallas.edu/~tyler.summers/papers/risk_averse_rrt_nmpc.pdf.

[15] S. Safaoui, L. Lindemann, D. V. Dimarogonas, I. Shames, and T. H. Summers, "Control design for risk-based signal temporal logic specifications," *IEEE Control Systems Letters*, vol. 4, no. 4, pp. 1000–1005, 2020.

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

[17] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*. Ieee, 2000, pp. 153–158.

[18] H. Zhu and J. Alonso-Mora, "Chance-constrained collision avoidance for mavs in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 776–783, 2019.