**Robotics: Science and Systems 2023**
**Daegu, Republic of Korea, July 10-July 14, 2023**

# RADIUS: Risk-Aware, Real-Time, Reachability-Based Motion Planning

Jinsun Liu*†, Challen Enninful Adu*†, Lucas Lymburner†, Vishrut Kaushik†,
Lena Trang‡, and Ram Vasudevan†

***Abstract*—Deterministic methods for motion planning guarantee safety amidst uncertainty in obstacle locations by trying to restrict the robot from operating in any possible location that an obstacle could be in. Unfortunately, this can result in overly conservative behavior. Chance-constrained optimization can be applied to improve the performance of motion planning algorithms by allowing for a user-specified amount of bounded constraint violation. However, state-of-the-art methods rely either on moment-based inequalities, which can be overly conservative, or make it difficult to satisfy assumptions about the class of probability distributions used to model uncertainty. To address these challenges, this work proposes a real-time, risk-aware reachability-based motion planning framework called RADIUS. The method first generates a reachable set of parameterized trajectories for the robot offline. At run time, RADIUS computes a closed-form over-approximation of the risk of a collision with an obstacle. This is done without restricting the probability distribution used to model uncertainty to a simple class (*e.g.*, Gaussian). Then, RADIUS performs real-time optimization to construct a trajectory that can be followed by the robot in a manner that is certified to have a risk of collision that is less than or equal to a user-specified threshold. The proposed algorithm is compared to several state-of-the-art chance-constrained and deterministic methods in simulation, and is shown to consistently outperform them in a variety of driving scenarios. A demonstration of the proposed framework on hardware is also provided. Readers can find the paper project page here[1].**

## I. Introduction

For mobile robots to safely operate in unstructured environments, they must be able to sense their environments and develop plans to dynamically react to changes while avoiding obstacles. Unfortunately, it is challenging to accurately and precisely estimate and predict an obstacle's movement. For mobile robots to operate robustly, the uncertainty within these estimations and predictions must be accounted for while generating motion plans. Various approaches to account for this uncertainty have been proposed in the literature, but these methods either (1) have difficulty being utilized in real-time, (2) make strong assumptions on the class of probability distributions used to model the uncertainty, or (3) generate motion plans that are overly conservative and thereby restrict motion. This paper develops an algorithm called **RADIUS**: **R**isk-**A**ware, real-time trajectory **D**esign **I**n **U**ncertain **S**cenarios (introduced in Figure 1) for **risk-aware, real-time motion planning** while addressing each of the aforementioned challenges.

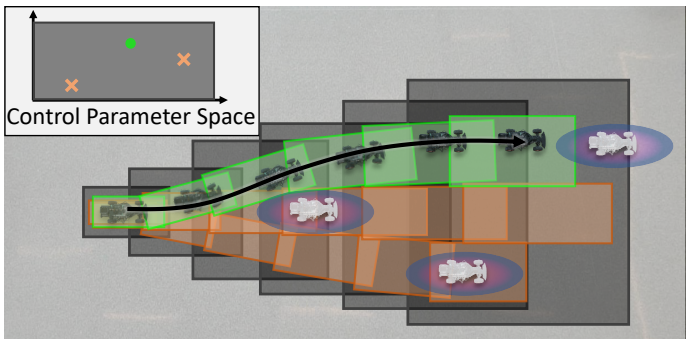



Fig. 1: An illustration of the motion planning framework RADIUS that is developed in this paper. RADIUS first performs offline reachability analysis using a closed-loop full-order vehicle dynamics to construct a series of control-parameterized, zonotope reachable sets (shown as dark gray boxes) that over-approximate all possible behaviors of the ego vehicle over the planning horizon. During online planning, given some user-defined risk of collision ($\epsilon$) for the motion plan, RADIUS constructs a trajectory by solving an optimization problem that selects subsets of pre-computed zonotope reachable sets that are certified to have a no greater than $\epsilon$ risk of colliding with any obstacles. In this figure, the moving obstacles are shown in white and the 3-$\sigma$ regions of the corresponding probability distributions for the obstacle locations are shown as the purple and blue ellipses where the probability density from low to high is illustrated from blue to purple. The subsets of the dark gray zonotope reachable sets corresponding to the trajectory parameter shown in green ensure a collision-free path that is guaranteed to have a no greater than $\epsilon$ risk of colliding with all obstacles, while the two trajectory parameters and their corresponding reachable sets shown orange may have a greater than $\epsilon$ risk of collision with the moving obstacles.

We first summarize related algorithms for safe motion planning under uncertainty. Approaches to address this problem can be categorized as deterministic or stochastic. The deterministic approaches often assume some bounded level of uncertainty and solve for motion plans that are robust to this bounded uncertainty. Reachability-based methods [1][2] for instance, over-approximate the uncertain regions using polynomial level sets or polytopes and plan paths that do not intersect with these regions. As we illustrate in this paper, such methods may be overly conservative as they must over-approximate the uncertain region to include events that may have an exceedingly small probability of occurring to ensure safety.

Stochastic methods incorporate probabilistic information about the environment to reduce the level of conservativeness of their motion plans. Many of these methods leverage chance constraints that allow for some user-specified amount of constraint violation. For instance, sampling-based algorithms for chance-constrained motion planning apply numerical integration techniques to compute the risk of collision between the agent and obstacles in the environment [3], [4]. In this context, random samples are drawn from the uncertain region, and the number of samples that cause a collision is used to

*These authors contributed equally to this work.
†Robotics, University of Michigan, Ann Arbor, MI. <jinsunl, enninful, llymburn, vishrutk, ramv>@umich.edu.
‡College of Engineering, University of Michigan, Ann Arbor, MI. ltrang@umich.edu.
[1]https://roahmlab.github.io/RADIUS/

approximate the risk of collision. These methods are simple to implement, but can be time-consuming as they require large numbers of samples to converge to a good estimate of the risk of collision. To address these limitations, moment-based methods upper bound the risk of collision using moments of the probability distribution [5]–[7]. Recent works leverage MPC and SOS programming in conjunction with these moment-based upper bounds to perform motion planning. Unfortunately, these methods can be conservative as the bound must be valid for any distribution with the given moments. Coherent risk measures like conditional value-at-risk (CVaR) and Entropic value-at-risk (EVaR) have been used to regulate safety by limiting the expectation of the distance to the safe region using a worst-case quantile representation of a distribution [8]–[10]. However, CVaR constraints are difficult to construct directly due to the computationally expensive multi-dimensional integration that is required. As a result, they are often approximated using sampling methods, which as mentioned earlier can be computationally expensive. Branch-and-bound methods like the Chance-Constrained Parallel Bernstein Algorithm (CCPBA) use reachability-based methods in conjunction with a branch-and-bound style algorithm to compute tight bounds for the risk of collision [11]. To compute these probabilities they assume that they have access to a cumulative distribution function, *a priori*, that can be evaluated efficiently during online optimization. However, such an assumption may not hold for arbitrary distributions.

This paper makes the following contributions. First, it provides a novel, parallelizable, closed-form over-approximation of the risk of collision given an arbitrary probability distribution with a twice-differentiable density function (Section VI-A). Second, it describes the analytical derivative of the probability over-approximation with respect to the control parameter (Section VI-C). Third, this paper presents a general, real-time risk-aware motion planning framework that guarantees robot safety up to a user-specified risk threshold over time intervals rather than just at discrete time instances (Section VI-D). Lastly, this paper demonstrates RADIUS in simulation and on hardware and compares its performance to CCPBA, and Cantelli MPC on a variety of scenarios (Section VII). The rest of this manuscript is organized as follows: Section II gives necessary notations for the manuscript. Section III presents the parameterized vehicle dynamics and environment representation. Section IV describes obstacle uncertainty, risk-aware vehicle safety and online planning. Section V discusses offline reachability analysis of the vehicle dynamics. Section VIII discusses the generalizability of the proposed algorithm and highlights the assumptions that need to be satisfied to apply RADIUS to other robotic systems. Section IX concludes the paper.

## II. Notation

This section formalizes notations used throughout the paper. Sets and subspaces are typeset using calligraphic font. Subscripts are primarily used as an index or to describe a particular coordinate of a vector. Let $\mathbb{R}$ and $\mathbb{N}$ denote the spaces of real numbers and natural numbers respectively. The Minkowski sum between two sets $\mathcal{A}$ and $\mathcal{A}'$ is $\mathcal{A} \oplus \mathcal{A}' = \{a + a' \mid a \in \mathcal{A},\ a' \in \mathcal{A}'\}$. Given a set $\mathcal{A}$, denote its power set as $P(\mathcal{A})$. Given vectors $\alpha, \beta \in \mathbb{R}^n$, let $[\alpha]_i$ denote the $i$-th element of $\alpha$, let $\texttt{diag}(\alpha)$ denote the diagonal matrix with $\alpha$ on the diagonal, and let $\texttt{int}(\alpha, \beta)$ denote the $n$-dimensional box $\{\gamma \in \mathbb{R}^n \mid [\alpha]_i \leq [\gamma]_i \leq [\beta]_i,\ \forall i = 1, \ldots, n\}$. Given arbitrary matrix $A \in \mathbb{R}^{n \times n}$, and let $\det(A)$ be the determinant of $A$. Let $\texttt{Prob(E)}$ denote the probability of occurrence of some event $\texttt{E}$, and let $\texttt{rotate}(a)$ denote the 2-dimensional rotation matrix $\begin{bmatrix} \cos(a) & -\sin(a) \\ \sin(a) & \cos(a) \end{bmatrix}$ for arbitrary $a \in \mathbb{R}$.

Next, we introduce a subclass of polytopes, called zonotopes, that are used throughout this paper:

**Definition 1.** *A* zonotope $\mathcal{Z}$ *is a subset of* $\mathbb{R}^n$ *defined as*

$$\mathcal{Z} = \left\{ x \in \mathbb{R}^n \mid x = c + \sum_{k=1}^{\ell} \beta_k g_k, \quad \beta_k \in [-1, 1] \right\} \tag{1}$$

*with* center $c \in \mathbb{R}^n$ *and* $\ell$ generators $g_1, \ldots, g_\ell \in \mathbb{R}^n$. *For convenience, we denote* $\mathcal{Z}$ *by* $<c,\ G>$ *where* $G = [g_1, g_2, \ldots, g_\ell] \in \mathbb{R}^{n \times \ell}$.

Note that an $n$-dimensional box is a zonotope because

$$\texttt{int}(\alpha, \beta) = <\frac{1}{2}(\alpha + \beta),\ \frac{1}{2}\texttt{diag}(\beta - \alpha)>. \tag{2}$$

By definition the Minkowski sum of two arbitrary zonotopes $\mathcal{Z}_1 = <c_1,\ G_1>$ and $\mathcal{Z}_2 = <c_2,\ G_2>$ is still a zonotope as $\mathcal{Z}_1 \oplus \mathcal{Z}_2 = <c_1 + c_2,\ [G_1, G_2]>$. Note that one can define the multiplication of a matrix $A$ of appropriate size with a zonotope $\mathcal{Z} = <c,\ G>$ as

$$A\mathcal{Z} = \left\{ x \in \mathbb{R}^n \mid x = Ac + \sum_{k=1}^{\ell} \beta_k A g_k,\ \beta_k \in [-1, 1] \right\}. \tag{3}$$

Note that $A\mathcal{Z}$ is equal to the zonotope $<Ac,\ AG>$.

## III. Preliminaries

The goal of this work is to plan trajectories for a mobile robot to navigate through environments with uncertainty in the locations of obstacles while having guarantees on the risk of collision. We illustrate the proposed method on an autonomous vehicle in this work. This section discusses the vehicle model, parameterized trajectories, and lastly the environment.

### A. Open Loop Vehicle Dynamics

#### 1) High-Speed Model

This work adopts the Front-Wheel-Drive vehicle model from [2], and can be generalized to All-Wheel-Drive and Rear-Wheel-Drive vehicle models. Let the ego vehicle's states at a time $t$ be given by $z^{\text{hi}}(t) = [x(t), y(t), h(t), u(t), v(t), r(t)]^\top \in \mathbb{R}^6$, where $x(t)$ and $y(t)$ are the position of the ego vehicle's center of mass in the world frame, $h(t)$ is the heading of the ego vehicle in the world frame, $u(t)$ and $v(t)$ are the longitudinal and lateral speeds of the ego vehicle in its body frame, and $r(t)$ is the yaw rate of the vehicle center of mass. To simplify

exposition, we assume vehicle weight is uniformly distributed and ignore the aerodynamic effect while modeling the flat ground motion of the vehicles by the following dynamics:

$$\dot{z}^{\text{hi}}(t) = \begin{bmatrix} u(t)\cos h(t) - v(t)\sin h(t) \\ u(t)\sin h(t) + v(t)\cos h(t) \\ r(t) \\ \frac{1}{m}\big(F_{\text{xf}}(t) + F_{\text{xr}}(t)\big) + v(t)r(t) + \Delta_u(t) \\ \frac{1}{m}\big(F_{\text{yf}}(t) + F_{\text{yr}}(t)\big) - u(t)r(t) + \Delta_v(t) \\ \frac{1}{I_{\text{zz}}}\big(l_{\text{f}}F_{\text{yf}}(t) - l_{\text{r}}F_{\text{yr}}(t)\big) + \Delta_r(t) \end{bmatrix}, \quad (4)$$

where $l_{\text{f}}$ and $l_{\text{r}}$ are the distances from the center of mass to the front and back of the vehicle, $I_{\text{zz}}$ is the vehicle's moment of inertia, and $m$ is the vehicle's mass. Note: $l_{\text{f}}$, $l_{\text{r}}$, $I_{\text{zz}}$ and $m$ are all assumed to be known constants. The tire forces along the longitudinal and lateral directions of the vehicle at time $t$ are $F_{\text{xi}}(t)$ and $F_{\text{yi}}(t)$ respectively, where the 'i' subscript can be replaced by 'f' for the front wheels or 'r' for the rear wheels. $\Delta_u, \Delta_v, \Delta_r$ are modeling error signals that are unknown and account for imperfect state estimation and tire models. To ensure the system is well-posed (i.e., its solution exists and is unique), we make the following assumption:

**Assumption 2.** *$\Delta_u, \Delta_v, \Delta_r$ are all square-integrable functions and are bounded (i.e., there exist real numbers $M_u, M_v, M_r \in [0, +\infty)$ such that $\|\Delta_u(t)\|_\infty \leq M_u$, $\|\Delta_v(t)\|_\infty \leq M_v$, $\|\Delta_r(t)\|_\infty \leq M_r$ for all $t$).*

Note that $\Delta_u, \Delta_v, \Delta_r$ can be computed using real-world data [2, Section IX-C]. In this work, we assume linear tire models and assume that we can directly control the front tire forces. Such assumptions are validated in [2, Section V.B and VIII.B]. Additionally, we treat rear tire forces as observed signals.

*2) Low-Speed Model*

When the vehicle speed lowers below some critical value $u^{\text{cri}} > 0$, applying the model described in (4) becomes intractable as explained in [2, Section III-B]. As a result, in this work when $u(t) \leq u^{\text{cri}}$, the dynamics of a vehicle are modeled using a steady-state cornering model [12, Chapter 6], [13, Chapter 10]. Note that the critical velocity $u^{\text{cri}}$ can be found according to [14, (5) and (18)].

The steady-state cornering model or low-speed vehicle model is described using four states, $z^{\text{lo}}(t) = [x(t), y(t), h(t), u(t)]^\top \in \mathbb{R}^4$ at time $t$. This model ignores transients on lateral velocity and yaw rate. Note that the dynamics of $x$, $y$, $h$ and $u$ in the low-speed model are the same as in the high-speed model (4); however, the steady-state cornering model describes the yaw rate and lateral speed as

$$r^{\text{lo}}(t) = \frac{\delta(t)u(t)}{l + C_{\text{us}}u(t)^2}, \; v^{\text{lo}}(t) = l_{\text{r}}r^{\text{lo}}(t) - \frac{ml_{\text{f}}}{\bar{c}_{\alpha\text{r}}l}u(t)^2 r^{\text{lo}}(t) \quad (5)$$

with understeer coefficient

$$C_{\text{us}} = \frac{m}{l}\left(\frac{l_{\text{r}}}{\bar{c}_{\alpha\text{f}}} - \frac{l_{\text{f}}}{\bar{c}_{\alpha\text{r}}}\right), \quad (6)$$

where $\delta(t)$ is the front tire steering angle at time $t$, $\bar{c}_{\alpha\text{f}}$ and $\bar{c}_{\alpha\text{r}}$ are cornering stiffness of the front and rear tires respectively.

As we describe in Section III-C, the high-speed and low-speed models can be combined together as a hybrid system to describe the vehicle behavior across all longitudinal speeds. In short, when $u$ transitions past the critical speed $u^{\text{cri}}$ from above at time $t$, the low speed model's states are initialized as:

$$z^{\text{lo}}(t) = \pi_{1:4}(z^{\text{hi}}(t)) \quad (7)$$

where $\pi_{1:4} : \mathbb{R}^6 \to \mathbb{R}^4$ is the projection operator that projects $z^{\text{hi}}(t)$ onto its first four dimensions via the identity relation. If $u$ transitions past the critical speed from below at time $t$, the high-speed model's states are initialized as

$$z^{\text{hi}}(t) = [z^{\text{lo}}(t)^\top, v^{\text{lo}}(t), r^{\text{lo}}(t)]^\top. \quad (8)$$

### *B. Trajectory Parameterization*

In this work, each trajectory plan is specified over a compact time interval of a fixed duration $t_f$. Because RADIUS performs receding-horizon planning, we make the following assumption about the time available to construct a new plan:

**Assumption 3.** *During each planning iteration starting from time $t_0$, the ego vehicle has $t_{plan}$ seconds to find a control input that is applied during the time interval $[t_0 + t_{plan}, t_0 + t_{plan} + t_f]$, where $t_f \geq 0$ is some user-specified constant. In addition, the vehicle state at time $t_0 + t_{plan}$ is known at time $t_0$.*

This assumption requires RADIUS to generate plans in real-time. This means that the ego vehicle must create a new plan before it finishes executing its previously planned trajectory.

In each planning iteration, RADIUS chooses a *desired trajectory* to be followed by the ego vehicle. The desired trajectory is chosen from a pre-specified continuum of trajectories, with each uniquely determined by an $n_p$-dimensional *trajectory parameter* $p \in \mathcal{P} \subset \mathbb{R}^{n_p}$. We adapt the definition of trajectory parametrization from [2, Definition 7], and note two important details about the parametrization: First, all desired trajectories share a time instant $t_{\text{m}} \in [t_0 + t_{\text{plan}}, t_0 + t_{\text{plan}} + t_f)$ such that every desired trajectory consists of a *driving maneuver* during $[t_0 + t_{\text{plan}}, t_0 + t_{\text{plan}} + t_{\text{m}})$ and a *contingency braking maneuver* during $[t_0 + t_{\text{plan}} + t_{\text{m}}, t_0 + t_{\text{plan}} + t_f]$. Second, the contingency braking maneuver slows down the ego vehicle's longitudinal speed to 0 by $t_f$. Note this latter property is used to ensure safety as we describe in Section IV-B. There are many choices of trajectory parametrizations, and the trajectory parametrization utilized in this work is provided in Appendix B-A.

### *C. Closed Loop Vehicle Dynamics*

A partial feedback linearization controller that tracks a parameterized desired trajectory robustly and accounts for the modeling error described by $\Delta_u$, $\Delta_v$ and $\Delta_r$ is provided in [2, Section V]. Using this controller, the closed loop dynamics of the high and low-speed systems can be written as:

$$\dot{z}^{\text{hi}}(t) = f^{\text{hi}}(t, z^{\text{hi}}(t), p), \quad (9)$$

$$\dot{z}^{\text{lo}}(t) = f^{\text{lo}}(t, z^{\text{lo}}(t), p), \quad (10)$$

respectively. Moreover, because the vehicle dynamics changes depending on $u$, we model the ego vehicle as a hybrid system $HS$ [15, Section 1.2] as is done in [2, Section V-C]. The hybrid system $HS$ contains a high-speed mode and a low-speed mode with state $z := z^{\text{hi}}$, whose dynamics can be written as

$$z(t) = \begin{cases} f^{\text{hi}}(t, z^{\text{hi}}(t), p), \text{ if } u(t) > u^{\text{cri}}, \\ \begin{bmatrix} f^{\text{lo}}(t, z^{\text{lo}}(t), p) \\ 0_{2\times 1} \end{bmatrix}, \text{ if } u(t) \leq u^{\text{cri}}, \end{cases} \tag{11}$$

Instantaneous transition between the two modes within $HS$ is described using the notion of a *guard* and *reset map*. The guard triggers a transition and is defined as $\{z(t) \in \mathbb{R}^6 \mid u(t) = u^{\text{cri}}\}$. Once a transition happens, the reset map resets the first $z(t)$ via (7) if $u(t)$ approaches $u^{\text{cri}}$ from above and via (8) if $u(t)$ approaches $u^{\text{cri}}$ from below.

### D. Ego Vehicle, Environment, and Sensing

To provide guarantees about vehicle behavior in a receding horizon planning framework, we define the ego vehicle's footprint similarly to [2, Definition 10]:

**Definition 4.** *Given $\mathcal{W} \subset \mathbb{R}^2$ as the world space, the ego vehicle is a rigid body that lies in a rectangle $\mathcal{O}^{ego} := \texttt{int}([-0.5L, -0.5W]^T, [0.5L, 0.5W]^T) \subset \mathcal{W}$ with width $W > 0$ and length $L > 0$ at time $t = 0$. $\mathcal{O}^{ego}$ is called the* footprint *of the ego vehicle.*

For arbitrary time $t$, given state $z(t)$ of the ego vehicle that starts from initial condition $z_0 \in \mathcal{Z}_0 \subset \mathbb{R}^6$ and applies a control input parameterized by $p \in \mathcal{P}$, the ego vehicle's *forward occupancy* at time $t$ can be represented as

$$\mathcal{E}\big(t, z_0, p\big) := \texttt{rotate}(h(t)) \cdot \mathcal{O}^{\text{ego}} + [x(t), y(t)]^\top, \tag{12}$$

which is a zonotope by (3). We define the obstacles as follows:

**Definition 5.** *An* obstacle *is a set $\mathcal{O}_i^{obs}(t) \subset \mathcal{W}$ that the ego vehicle should not collide with at time $t$, where $i \in \mathcal{I}$ is the index of the obstacle and $\mathcal{I}$ contains finitely many elements.*

The dependency on $t$ in the definition of an obstacle allows the obstacle to move as $t$ varies. However, if the $i$-th obstacle is static, then $\mathcal{O}_i^{\text{obs}}(t)$ is a constant. Note that in this work, we assume that we do not have perfect knowledge of obstacle locations and motion as is normally the case in real-life scenarios. We assume that this uncertainty in the locations of obstacles is represented by some arbitrary probability distribution. This is described in the next section. Assuming that the ego vehicle has a maximum speed $\nu^{\text{ego}}$ and all obstacles have a maximum speed $\nu^{\text{obs}}$ for all time, we make the following assumption on planning and sensing horizon.

**Assumption 6.** *The ego vehicle senses all obstacles within a sensor radius greater than $(t_f + t_{plan}) \cdot (\nu^{ego} + \nu^{obs}) + 0.5\sqrt{L^2 + W^2}$ around its center of mass.*

Assumption 6 ensures that any obstacle which may cause a collision between times $t \in [t_0 + t_{\text{plan}}, t_0 + t_{\text{plan}} + t_f]$ can be detected by the vehicle [16, Theorem 15]. Note one could treat sensor occlusions as obstacles that travel at the maximum obstacle speed [17], [18]. To simplify notation, we reset time to 0 whenever a feasible control policy is about to be applied, *i.e.*, $t_0 + t_{\text{plan}} = 0$. Finally to aid in the descriptions of obstacle uncertainty and system trajectory over-approximation in Sections IV and V, we partition the planning horizon $[0, t_f]$ into $t_f/\Delta_t$ *time intervals* with some positive number $\Delta_t$ that divides $t_f$, and denote $\mathcal{T}_j$ the $j$-th time interval $[(j-1)\Delta_t, j\Delta_t]$ for any $j \in \mathcal{J} := \{1, 2, \ldots, t_f/\Delta_t\}$.

## IV. Online Planning Under Uncertainty

This section constructs a chance-constrained optimization problem to generate risk-aware motion plans. First, we describe the representations of the obstacle uncertainty used in this work. Then, we define risk-aware vehicle safety and conclude by illustrating how to formulate online planning as a chance-constrained optimization that limits the risk of collision.

### A. Obstacle Uncertainty

To incorporate uncertainty in both obstacle sensing and obstacle motion prediction into the motion planning framework, let $w_{i,j}^{\text{obs}}$ be a random variable that takes values in $\mathcal{W}$. $w_{i,j}^{\text{obs}}$ describes the possible locations of the center of the $i$-th obstacle, $\mathcal{O}_i^{\text{obs}}(t)$, for any time $t \in \mathcal{T}_j$. We then make the following assumption about how $w_{i,j}^{\text{obs}}$ is distributed:

**Assumption 7.** *For any $i \in \mathcal{I}$ and $j \in \mathcal{J}$, $w_{i,j}^{obs}$'s Probability Density Function (PDF), $q_{i,j} : \mathcal{W} \to [0, +\infty)$, exists and is twice-differentiable. In addition, there exists some $w_{i,j}^{obs}$ sampled according to $q_{i,j}$ such that $\mathcal{O}_i^{obs}(t) \subseteq <w_{i,j}^{obs}, G^{obs}>$ at some $t \in \mathcal{T}_j$, where $G^{obs}$ is a 2-row constant matrix.*

According to Assumption 7, $<w_{i,j}^{\text{obs}}, G^{\text{obs}}> = w_{i,j}^{\text{obs}} + <0, G^{\text{obs}}>$ has an uncertain center $w_{i,j}^{\text{obs}}$ with probability density $q_{i,j}(w_{i,j}^{\text{obs}})$, and has invariant shape and size with respect to $i$ and $j$ due to the constant generator matrix $G^{\text{obs}}$ that accounts for the footprint of any obstacle. Such probability density functions $q_{i,j}$ can be generated by, for example, performing variants of Kalman Filter [19], [20] on the $i$-th obstacle given its dynamics or detecting the $i$-th obstacle with a Bayesian confidence framework [21] during $\mathcal{T}_j$. The generator matrix $G^{\text{obs}}$ can be generated as the union of footprints of all obstacles.

### B. Risk-Aware Vehicle Safety

In dynamic environments, it may be difficult to avoid collisions in all scenarios (*e.g.*, a parked ego vehicle can be run into). As a result, we instead develop a trajectory synthesis technique that ensures that the ego vehicle is not-at-fault [16, Definition 11]. We define not-at-fault safety in this work from a probabilistic perspective by bounding the probability of the ego vehicle running into any obstacles.

**Definition 8.** *Let the ego vehicle start from initial condition $z_0 \in \mathcal{Z}_0$ with control parameter $p \in \mathcal{P}$. Given a user-specified*

allowable risk threshold $\epsilon \in [0,1]$, *the ego vehicle is* not-at-fault *with a risk of collision of at most* $\epsilon$*, if it is stopped, or if*

$$\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\int_{\cup_{t\in\mathcal{T}_j}(\mathcal{E}(t,z_0,p)\oplus<0,\ G^{obs}>)} q_{i,j}(w)\ dw \le \epsilon, \qquad (13)$$

*while it is moving during* $[0, t_f]$.

Note that the domain of integration in (13) is the ego vehicle's forward occupancy buffered by the obstacle's footprint. Thus to satisfy (13), the probability that the buffered ego vehicle's forward occupancy intersects with an obstacle's center must be bounded by epsilon over all time intervals and all possible obstacles. In particular, this ensures that the probability of the ego vehicle (including its footprint) intersecting with any obstacle (including its footprint) over the planning horizon is bounded by $\epsilon$ via [22, Lem. 5.1].

### C. Online Optimization

To construct a motion plan that ensures the ego vehicle is not-at-fault with a risk of collision at most $\epsilon$ during online planning, one could solve the following chance-constrained optimization:

$$\min_{p\in\mathcal{P}} \texttt{cost}(z_0,p) \qquad (\texttt{Opt})$$
$$\text{s.t.} \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\int_{\cup_{t\in\mathcal{T}_j}(\mathcal{E}(t,z_0,p)\oplus<0,\ G^{\text{obs}}>)} q_{i,j}(w)\ dw \le \epsilon$$

where $\texttt{cost} : \mathcal{Z}_0 \times \mathcal{P} \to \mathbb{R}$ is a user-specified cost function, and the constraint is a chance constraint that ensures the vehicle is not-at-fault with a maximum risk of collision $\epsilon$ during the planning horizon as stated in Definition 8.

To achieve real-time motion planning, (Opt) must be solved within $t_{\text{plan}}$ seconds. However, efficiently evaluating the chance constraint in (Opt) in a closed form can be challenging in real applications for two reasons. First, the exact information of the ego vehicle's location $\mathcal{E}(t, z_0, p)$ at any time is usually inaccessible due to the nonlinear and hybrid nature of the vehicle dynamics. Second, the probability distribution that describes the uncertain observation of an obstacle's location as described in Assumption 7 can be arbitrary. As illustrated in Appendix C-B, even approximating this chance constraint accurately using Monte-Carlo is challenging for real-time motion planning. Therefore to achieve real-time performance, RADIUS seeks a closed-form approximation of the risk of collision for evaluation efficiency. This paper focuses on constructing an over-approximation to ensure that RADIUS does not underestimate the true risk of collision or generate plans that violate the not-at-fault condition from Definition 8. Moreover, it is preferable that this approximation is differentiable, as providing the gradient of the constraint can speed up the solving procedure of online optimization. We describe how we generate an approximation that satisfies these requirements in the next two sections.

## V. Offline Reachability Analysis

Due to the nonlinear and hybrid nature of the vehicle dynamics, it is challenging to compute the trajectory of vehicle state exactly when evaluating the chance constraint in (Opt). To resolve this challenge, RADIUS over-approximates the ego vehicle's trajectory using zonotopes as stated below:

**Assumption 9.** *Let* $z$ *be a solution to* (11) *starting from initial condition* $z_0 \in \mathcal{Z}_0$ *with control parameter* $p \in \mathcal{P}$*. For each* $j \in \mathcal{J}$*, there exists a map* $\xi_j : \mathcal{Z}_0 \times \mathcal{P} \to P(\mathcal{W})$ *such that*

1) $\xi_j(z_0, p)$ *contains the ego vehicle's footprint during* $\mathcal{T}_j$*, i.e.,* $\cup_{t\in\mathcal{T}_j}\mathcal{E}(t, z_0, p) \subseteq \xi_j(z_0, p)$*, and*
2) $\xi_j(z_0, p)$ *is a zonotope of the form* $<c_j(z_0)+A_j\cdot p,\ G_j>$ *with some linear function* $c_j : \mathcal{Z}_0 \to \mathbb{R}^2$*, some matrix* $A_j \in \mathbb{R}^{2\times n_p}$ *and some 2-row matrix* $G_j$*.*

The collection of maps $\{\xi_j\}_{j\in\mathcal{J}}$ can be constructed by applying existing techniques for offline reachability analysis [2, Section VI]. In particular, $\{\xi_j\}_{j\in\mathcal{J}}$ can be generated by first applying the open-source toolbox CORA [23] to over-approximate the trajectory of the ego vehicle's state with initial condition $z_0$ and control parameter $p$ using a collection of zonotopes, where each zonotope over-approximates a segment of the trajectory during one small time interval among $\{\mathcal{T}_j\}_{j\in\mathcal{J}}$, and then accounts for the ego vehicle's footprint. The proof of Lemma 26 in [2] provides explicit formulas for $\xi_j$, $c_j$, $A_j$ and $G_j$. Note formulas provided in [2, Lemma 26] assume the computation in the body frame of the ego vehicle, i.e., assuming $x(0) = y(0) = h(0) = 0$. In the case when the initial position and heading of the ego vehicle are not zeros, one can represent $\xi_j(z_0, p)$ in the world frame via frame transformation based on $z_0$. In the remainder of this manuscript, we assume $\xi_j(z_0, p)$ is represented in the world frame. We refer to $\xi_j(z_0, p)$ as the zonotope reachable set.

## VI. An Implementable Alternative to (Opt)

This section describes an implementable alternative to (Opt) that can be solved rapidly. In particular, we discuss how to relax the chance constraint in (Opt) in a conservative fashion.

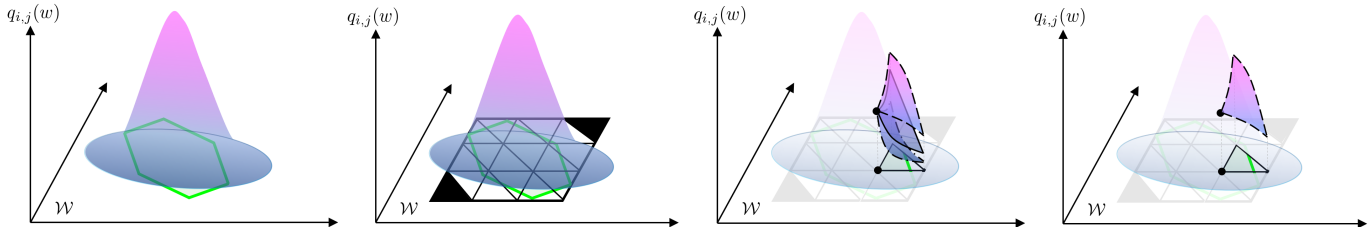
### A. Chance Constraint Relaxation

The chance constraint in (Opt) is relaxed conservatively as illustrated in Figure 2. First, the risk of collision during $[0, t_f]$ is over-approximated using Assumption 9. Then we relax the domain of integration into a collection of right-angled triangles and relax the PDF as a quadratic polynomial. Finally, we describe a closed-form equation for the relaxed integral.

#### 1) PDF Integration

Recall the ego vehicle's performance during the planning horizon is over-approximated by a collection of maps $\{\xi_j\}_{j\in\mathcal{J}}$, then the chance constraint in (Opt) is relaxed according to the following lemma which follows directly from the first property in Assumption 9:

**Lemma 10.** *Suppose the ego vehicle starts with initial condition* $z_0 \in \mathcal{Z}_0$ *and control parameter* $p \in \mathcal{P}$*. Let probability density functions* $\{q_{i,j}\}_{(i,j)\in\mathcal{I}\times\mathcal{J}}$ *and matrix* $G^{obs}$ *be as assumed in Assumption 7. Let maps* $\{\xi_j\}_{j\in\mathcal{J}}$ *be as assumed in*




(a) PDF Integration (Sec. VI-A1). (b) Domain Relaxation (Sec. VI-A2). (c) Integrand Relaxation (Sec. VI-A3). (d) Closed-form Computation (Sec. VI-A4).

Fig. 2: An illustration of chance constraint relaxation. Given arbitrary $(i,j) \in \mathcal{I} \times \mathcal{J}$, in (a) the risk of collision between the ego vehicle and the $i$-th obstacle during time interval $\mathcal{T}_j$ is relaxed as the integration of probability density function $q_{i,j}$ (shown in purple and blue) over zonotope $\xi_j(z_0,p) \oplus <0,\ G^{\text{obs}}> \subset \mathcal{W}$ (shown in green). In (b), $\xi_j(z_0,p) \oplus <0,\ G^{\text{obs}}>$ is over-approximated by a collection of right-angled triangles colored in white and black depending on if a triangle has a nontrivial intersection with $\xi_j(z_0,p) \oplus <0,\ G^{\text{obs}}>$ or not. In (c), Interval Arithmetic is used to generate an over- and under-approximation of $q_{i,j}$ over each right-angled triangle. And in (d), the integration of the over-approximation of $q_{i,j}$ over each right-angled triangle is computed in closed form.

*Assumption 9. Then for arbitrary $\epsilon \in [0,1]$, (13) holds if*

$$\sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} \int_{\xi_j(z_0,p)\oplus<0,\ G^{obs}>} q_{i,j}(w)\ dw \le \epsilon. \tag{14}$$

*2) Domain Relaxation*

Recall by Assumption 9 that $\xi_j(z_0,p)$ can be rewritten as $<c_j(z_0) + A_j \cdot p,\ G_j> = <c_j(z_0),\ G_j> + A_j \cdot p$. Then to relax the domain of integration $\xi_j(z_0,p) \oplus <0,\ G^{\text{obs}}>$, we start by constructing a $k$-by-$k$ grid that covers zonotope $\xi_j(z_0,0) \oplus <0,\ G^{\text{obs}}> = <c_j(z_0),\ [G_j, G^{\text{obs}}]>$ where $k$ is some user-specified positive integer. Each cell in the grid shares the same size and is indexed by its row and column index in the grid. Each cell in the grid is further divided into two simplexes as right-angled triangles that are indexed by 1 or -1 corresponding to the lower or upper triangles of the cell, respectively. For notational ease, denote $\mathcal{S}_{j,k_1,k_2,k_3}(z_0) \subset \mathcal{W}$ as the simplex indexed by $k_3 \in \{-1,1\}$ in the cell on the $k_1$-th row and $k_2$-th column of the grid that covers $<c_j(z_0),\ [G_j, G^{\text{obs}}]>$. Define

$$\begin{aligned}\overline{\mathcal{S}}_j(z_0) := \{&\mathcal{S}_{j,k_1,k_2,k_3}(z_0) \mid k_1,k_2 \in \{1,2,\dots,k\},\\ &|k_3| = 1,\ \mathcal{S}_{j,k_1,k_2,k_3}(z_0) \cap <c_j(z_0),\ [G_j, G^{\text{obs}}]> \neq \emptyset\}.\end{aligned} \tag{15}$$

as the collection of every possible $\mathcal{S}_{j,k_1,k_2,k_3}(z_0)$ that intersects with $<c_j(z_0),\ [G_j, G^{\text{obs}}]>$, thus

$$<c_j(z_0),\ [G_j, G^{\text{obs}}]> \subset (\cup_{\mathcal{S}\in\overline{\mathcal{S}}_j(z_0)} \mathcal{S}). \tag{16}$$

Notice $\xi_j(z_0,p) = \xi_j(z_0,0) + A_j \cdot p$, therefore $\xi_j(z_0,p) \oplus <0,\ G^{\text{obs}}> \subset \Big((\cup_{\mathcal{S}\in\overline{\mathcal{S}}_j(z_0)} \mathcal{S}) + A_j \cdot p\Big)$ and

$$\int_{\xi_j(z_0,p)\oplus<0,\ G^{\text{obs}}>} q_{i,j}(w)\ dw \le \sum_{\mathcal{S}\in\overline{\mathcal{S}}_j(z_0)} \int_{\mathcal{S}+A_jp} q_{i,j}(w)\ dw. \tag{17}$$

**Remark 11.** *Note that for any $\mathcal{S} \in \overline{\mathcal{S}}_j(z_0)$, $\mathcal{S}$ depends on $z_0$ and $j$. However, to reduce notational complexity, we drop its dependency on $z_0$ and $j$ in the remainder of this manuscript.*

*3) Integrand Relaxation*

Next, we present a lemma, whose proof is provided in Appendix A, to conservatively approximate the integral in the chance constraint by relaxing the integrand.

**Lemma 12.** *Suppose the ego vehicle starts with initial condition $z_0 \in \mathcal{Z}_0$ and control parameter $p \in \mathcal{P}$. Let $\mathcal{S} \in \overline{\mathcal{S}}_j(z_0)$ and let $w_{\mathcal{S}}^{ctr}$ denote the vertex of the right angle in $\mathcal{S}$. Define the function $q_{i,j,\mathcal{S}} : \mathcal{W} \times \mathcal{P} \to [0,\infty)$ as*

$$\begin{aligned}q_{i,j,\mathcal{S}}(w,p) := q_{i,j}(w_{\mathcal{S}}^{ctr} + A_jp) + \frac{\partial q_{i,j}}{\partial w}(w_{\mathcal{S}}^{ctr} + A_jp)\cdot\\ \cdot (w - w_{\mathcal{S}}^{ctr} - A_jp) + \frac{1}{2}(w - w_{\mathcal{S}}^{ctr} - A_jp)^\top\cdot\\ \cdot H_{\mathcal{S}} \cdot (w - w_{\mathcal{S}}^{ctr} - A_jp),\end{aligned} \tag{18}$$

*where $H_{\mathcal{S}} \in \mathbb{R}^{2\times 2}$ is generated by taking element-wise supremum of the Hessian of $q_{i,j}$ over $\mathcal{S} \oplus A_j\mathcal{P}$ using Interval Arithmetic [24]. Then for all $w \in \mathcal{S} + A_jp$:*

$$q_{i,j}(w) \le q_{i,j,\mathcal{S}}(w,p). \tag{19}$$

Note the inequality in (19) flips if $H_{\mathcal{S}}$ is generated by taking an element-wise infimum of the Hessian of $q_{i,j}$. This would create an under-approximation to $q_{i,j}$.

*4) Closed-form Computation*

As a result of (17) and (19), the following inequality holds:

$$\begin{aligned}\int_{\xi_j(z_0,p)\oplus<0,\ G^{\text{obs}}>} q_{i,j}(w)\ dw \le\\ \le \sum_{\mathcal{S}\in\overline{\mathcal{S}}_j(z_0)} \int_{\mathcal{S}+A_jp} q_{i,j,\mathcal{S}}(w,p)\ dw.\end{aligned} \tag{20}$$

Notice that $q_{i,j,\mathcal{S}}(w,p)$ defined in (19) is indeed a quadratic polynomial of $w$, and $\mathcal{S} + A_jp$ is a simplex in $\mathbb{R}^2$. One can then compute $\int_{\mathcal{S}+A_jp} q_{i,j,\mathcal{S}}(w,p)\ dw$ in closed-form as follows:

**Theorem 13.** *For any $i \in \mathcal{I}$, $j \in \mathcal{J}$, $z_0 \in \mathcal{Z}_0$, $p \in \mathcal{P}$ and $\mathcal{S} \in \overline{\mathcal{S}}_j(z_0)$, let $A_j \in \mathbb{R}^{2\times n_p}$ be defined as in Assumption 9, let $q_{i,j,\mathcal{S}}$ be defined as in (19), and let $\texttt{idx}_{k_3}(\mathcal{S})$ denote the last index of its argument (i.e., $\texttt{idx}_{k_3}(\mathcal{S}_{j,k_1,k_2,k_3}(z_0)) = k_3$). Assume that positive numbers $l_1$ and $l_2$ give the lengths of*

*horizontal and vertical right angle sides of $\mathcal{S}$ respectively, then*

$$\int_{\mathcal{S}+A_jp} q_{i,j,\mathcal{S}}(w,p)dw = \frac{1}{2\det(A_\mathcal{S})}\Bigg(q_{i,j}(w^{ctr}_\mathcal{S}+A_jp)+ \\ + \begin{bmatrix}\frac{1}{4\sqrt{3}} & \frac{1}{4\sqrt{3}}\end{bmatrix}\left(\hat{H}_\mathcal{S}\odot\begin{bmatrix}2&1\\1&2\end{bmatrix}\right)\begin{bmatrix}\frac{1}{2\sqrt{3}}\\ \frac{1}{2\sqrt{3}}\end{bmatrix}+ \\ + \frac{\partial q_{i,j}}{\partial w}(w^{ctr}_\mathcal{S}+A_jp)\cdot A_\mathcal{S}^{-1}\cdot\begin{bmatrix}\frac{1}{3}\\ \frac{1}{3}\end{bmatrix}\Bigg) \tag{21}$$

*where $\odot$ denotes the element-wise multiplication and*

$$A_\mathcal{S} = \begin{bmatrix}\texttt{idx}_{k_3}(\mathcal{S})/l_1 & 0\\ 0 & \texttt{idx}_{k_3}(\mathcal{S})/l_2\end{bmatrix}, \tag{22}$$

$$\hat{H}_\mathcal{S} = A_\mathcal{S}^{-\top}H_\mathcal{S}A_\mathcal{S}^{-1}. \tag{23}$$

*Proof:* The claim follows from [25, Theorem 1.1] and the fact that $A_\mathcal{S}\cdot\big((\mathcal{S}+A_jp)-(w^{\text{ctr}}_\mathcal{S}+A_jp)\big)$ equals the canonical simplex $\Delta := \{(a,b)\in\mathbb{R}^2 \mid a+b\le 1, a\ge 0, b\ge 0\}$. ∎

### B. Tractable Online Optimization

The computation in Section VI-A provides a tractable way to enforce not-at-fault behavior compared to the original chance constraint in (Opt). As a result, RADIUS solves the following optimization during online planning:

$$\min_{p\in\mathcal{P}} \texttt{cost}(z_0,p) \tag{Opt-E}$$
$$\text{s.t.} \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\sum_{\mathcal{S}\in\overline{\mathcal{S}}_j(z_0)}\int_{\mathcal{S}+A_jp} q_{i,j,\mathcal{S}}(w,p)\ dw \le \epsilon.$$

(Opt-E) is a strengthened version of (Opt) because satisfaction of the chance constraint in (Opt-E) implies satisfaction of the chance constraint in (Opt) by Lemma 10, (16), and (19). Thus, the following lemma holds based on Definition 8.

**Lemma 14.** *If the ego vehicle applies any feasible solution, $p^*\in\mathcal{P}$, of (Opt-E) beginning from $z_0\in\mathcal{Z}_0$ at $t=0$, then it is not-at-fault with a risk of collision at most $\epsilon$ during $[0,t_f]$.*

### C. Constraint Gradient and Parallelization

To improve the solving procedure of (Opt-E), we provide the derivative of its chance constraint. To compute the gradient of the chance constraint in (Opt-E), it suffices to compute the derivative of $\int_{\mathcal{S}+A_jp} q_{i,j,\mathcal{S}}(w,p)\ dw$ in (21) with respect to $p$. Notice that $A_\mathcal{S}$ and $\hat{H}_\mathcal{S}$ are invariant over $\mathcal{P}$, then

$$\frac{\partial}{\partial p}\int_{\mathcal{S}+A_jp} q_{i,j,\mathcal{S}}(w,p)\ dw = \\ = \frac{1}{2\det(A_\mathcal{S})}\cdot\Bigg(\frac{\partial q_{i,j}}{\partial w}(w^{\text{ctr}}_\mathcal{S}+A_jp)\cdot A_j + \\ + \begin{bmatrix}\frac{1}{3} & \frac{1}{3}\end{bmatrix}A_\mathcal{S}^{-\top}\texttt{Hess}_{q_{i,j}}(w^{\text{ctr}}_\mathcal{S}+A_jp)\cdot A_j\Bigg), \tag{24}$$

where $\texttt{Hess}_{q_{i,j}}(w^{\text{ctr}}_\mathcal{S}+A_jp)$ is the evaluation of the Hessian of $q_{i,j}$ at $w^{\text{ctr}}_\mathcal{S}+A_jp$.

Notice the computations of $\int_{\mathcal{S}+A_jp} q_{i,j,\mathcal{S}}(w,p)\ dw$ and its gradient can be parallelized for all $(i,j)\in\mathcal{I}\times\mathcal{J}$ and all $\mathcal{S}\in\overline{\mathcal{S}}_j(z_0)$ as indicated in Algorithm 1. $\overline{\mathcal{S}}_j(z_0)$ is generated

**Algorithm 1** Chance Constraint Parallelization

**Require:** $z_0\in\mathcal{Z}_0$, $p\in\mathcal{P}$, $\{q_{i,j}\}_{(i,j)\in\mathcal{I}\times\mathcal{J}}$, $\{\xi_j\}_{j\in\mathcal{J}}$
1: **Generate** $\{\overline{\mathcal{S}}_j(z_0)\}_{j\in\mathcal{J}}$ as in (15) using $\{\xi_j\}_{j\in\mathcal{J}}$
2: **Parfor** $(i,j)\in\mathcal{I}\times\mathcal{J}$ **do**
3: **Parfor** $\mathcal{S}\in\overline{\mathcal{S}}_j(z_0)$ **do**
4: **Compute** $\int_{\mathcal{S}+A_jp} q_{i,j,\mathcal{S}}(w,p)\ dw$ as in (21)
5: **Compute** $\frac{\partial}{\partial p}\int_{\mathcal{S}+A_jp} q_{i,j,\mathcal{S}}(w,p)\ dw$ as in (24)
6: **End Parfor**
7: **End Parfor**

in Line 1 for all $j\in\mathcal{J}$. The outer parallel for loop that starts at Line 2 iterates every element in $\mathcal{I}\times\mathcal{J}$, and the inner parallel for loop that starts at Line 3 iterates over each simplex in $\overline{\mathcal{S}}_j(z_0)$. The integral and its gradient over each simplex are computed from Lines 4 to 5. Then the chance constraint and its gradient in (Opt-E) can be computed as the summation of all computed $\int_{\mathcal{S}+A_jp} q_{i,j,\mathcal{S}}(w,p)\ dw$ and $\frac{\partial}{\partial p}\int_{\mathcal{S}+A_jp} q_{i,j,\mathcal{S}}(w,p)\ dw$ respectively.

### D. Online Operation

Algorithm 2 summarizes the online operation of RADIUS. It begins by sensing and constructing predictions for obstacle locations as in Assumption 7 in Line 1. OnlineOpt then solves (Opt-E) to search for a not-at-fault plan with a risk of collision at most $\epsilon$ in Line 2. If (Opt-E) is infeasible, then RADIUS terminates planning in Line 3, otherwise it enters the planning loop in Line 4. Within the loop, RADIUS first resets time to 0 in Line 5 and executes the entire driving maneuver corresponding to $p^*$ in Line 6. Meanwhile, SenseObstacles updates the predictions for obstacle locations in Line 7 and StatePrediction predicts the ego vehicle state at $t=t_\text{m}$ as in Assumption 3 in Line 8. In the case when the predicted vehicle state $z_0\notin\mathcal{Z}_0$, then RADIUS breaks the planning loop in Line 9. Otherwise (Opt-E) is solved again using the updated obstacle information and $z_0$ in Line 10, and RADIUS breaks the planning loop if (Opt-E) is infeasible or takes longer than $t_\text{plan}$ to find a solution in Line 11. Finally, once RADIUS leaves the planning loop, the contingency braking maneuver corresponding to $p^*$ is executed. This braking maneuver by construction brings the vehicle to a stop. Note that the risk of collision of this maneuver was already verified to be less than $\epsilon$ during the previous planning iteration. Subsequently, RADIUS terminates planning in Line 13. Note, we are able to obtain the following theorem by iteratively applying Lemma 14:

**Theorem 15.** *Suppose the ego vehicle can sense and predict the surrounding obstacles as in Assumption 7, and starts from rest with an initial condition $z_0\in\mathcal{Z}_0$ at $t=0$. Then by performing planning and execution as in Algorithm 2, the ego vehicle is not-at-fault with a risk of collision at most $\epsilon$ for all time.*

## VII. Experiments and Results

All experiments are conducted in MATLAB R2023a on a Ubuntu 22.04 machine with an AMD Ryzen 9 5950X CPU,

**Algorithm 2** RADIUS Online Planning

**Require:** $z_0 \in \mathcal{Z}_0$ and $\epsilon \in [0, 1]$

**Initialize:** $\{q_{i,j}\}_{(i,j)\in\mathcal{I}\times\mathcal{J}} \leftarrow$ `SenseObstacles()`
**Try** $p^* \leftarrow$ `OnlineOpt`$(z_0, \{q_{i,j}\}_{(i,j)\in\mathcal{I}\times\mathcal{J}}, \epsilon)$
**Catch** terminate planning
**Loop:** *// Line 6 executes simultaneously with Lines 7-11*
  **Reset** $t$ to 0
  **Execute** $p^*$ during $[0, t_{\mathrm{m}})$
  $\{q_{i,j}\}_{(i,j)\in\mathcal{I}\times\mathcal{J}} \leftarrow$ `SenseObstacles()`
  $z_0 \leftarrow$ `StatePrediction`$(z_0, p^*, t_{\mathrm{m}})$
  **If** $z_0 \notin \mathcal{Z}_0$, **then** break
  **Try** $p^* \leftarrow$ `OnlineOpt`$(z_0, \{q_{i,j}\}_{(i,j)\in\mathcal{I}\times\mathcal{J}}, \epsilon)$
  **Catch** break
**End**
**Execute** $p^*$ during $[t_{\mathrm{m}}, t_f]$, **then** terminate planning

two NVIDIA RTX A6000 48GB GPUs, and 64GB RAM. Parallelization is achieved using CUDA 12.1. RADIUS invokes C++ for online planning using IPOPT. Additional details on experimental setup can be found in Appendix B An additional experiment detailing the single planning iteration performance of RADIUS, CCPBA, Cantelli MPC and REFINE can be found in Appendix C-A. An ablation study illustrating the importance of the analytical gradient and closed-form over-approximation of the risk of collision for real-time performance can be found in Appendix C-B. Readers can find our implementation[2] and the video[3] of simulations and hardware demos utilising RADIUS online.

### A. Tightness and Generality of Risk Approximation

For effective motion planning, we desire a tight over-approximation of the risk of collision. Additionally, we want RADIUS to be able to generalize to arbitrary probability distributions as well to accommodate other uncertainty representations. Thus, to evaluate the tightness of our over-approximation of $\int_{\xi_j(z_0,p)\oplus<0,\ G^{\mathrm{obs}}>} q_{i,j}(w)\ dw$, we compare the proposed method's approximation of the risk of collision against Monte-Carlo integration [26, Chapter 4], the Cantelli inequality [5, Section IV B], and the Chance-Constrained Parallel Bernstein Algorithm (CCPBA) [11, Chapter 6] on 9000 randomly generated test cases.

In each test case, $(i, j, z_0, p)$ is randomly chosen from $\mathcal{I} \times \mathcal{J} \times \mathcal{Z}_0 \times \mathcal{P}$. Additionally, $q_{i,j}$ is set to be the probability density function of either a randomly generated 2-dimensional (2D) Gaussian distribution, 2D Beta distribution [27] or a 2D Multimodal distribution. Each type of distribution is evaluated in 3000 test cases. Note that because CCPBA is unable to handle a distribution that is not Gaussian, we only evaluate RADIUS and the Cantelli inequality on these non-Gaussian probability distributions. We perform the Monte-Carlo integration of $q_{i,j}$ over $\xi_j(z_0, p) \oplus <0,\ G^{\mathrm{obs}}>$ with $10^6$ samples and treat that as the ground truth.

[2]https://github.com/roahmlab/RADIUS
[3]https://youtu.be/8eU9fiA39sE

| Method | Gaussian Error (Mean, Max.) | Beta Error (Mean, Max.) | Multimodal Error (Mean, Max.) |
|---|---|---|---|
| RADIUS | **(0.0073, 0.0523)** | **(0.0065, 0.0489)** | **(0.0079, 0.0262)** |
| Cantelli | (0.1774, 0.3420) | (0.2221, 0.5062) | (0.4734, 0.5851) |
| CCPBA | (0.1881, 0.2149) | - | - |

TABLE I: Results for the risk of collision estimation error of RADIUS and the Cantelli Inequality when the obstacle location is represented by three types of probability distributions. Note that CCPBA can only handle Gaussian distributions so does not have results for the other types of distributions.

The risk approximation error, which is the difference between the Monte-Carlo integration and each method, is illustrated in Table I. The results from Table I show that RADIUS provides significantly tighter upper bounds to the ground truth than CCPBA and the Cantelli inequality for the tested probability distributions. The associated average and maximum times to compute the over-approximation for each method are shown in Table IV of Appendix B-B.

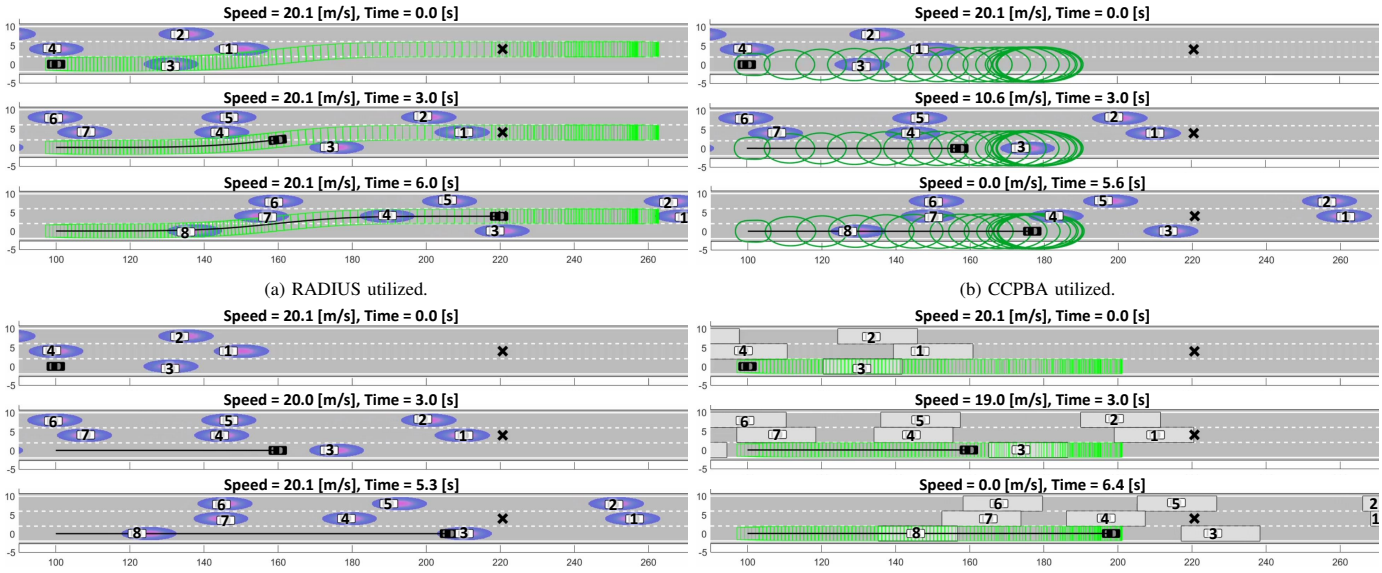
### B. Simulations

This section compares RADIUS to two state-of-the-art chance-constrained motion planning algorithms: CCPBA [11] and Cantelli MPC [5], and one state-of-the-art deterministic motion planning algorithm REFINE [2] in dense highway scenarios. Additionally, we evaluate how varying the allowable risk threshold ($\epsilon$) for RADIUS affects the ego vehicle behavior in various unprotected left turn scenarios. For all simulation experiments, $t_{\text{plan}}$ is set as 3[sec] and $t_f$ is chosen according to [2, Lemma 14]. A description of parameterized desired trajectories that are used in this work is provided in Appendix B-A.

#### 1) 3-Lane Highway

This experiment evaluates the performance of RADIUS in a 3-lane highway environment where it must execute multiple planning iterations in succession and compares its performance to CCPBA, Cantelli MPC and REFINE. We compare the methods over 1000 randomly generated 3-lane highway scenarios in simulation with $\epsilon = 0.05$. In each simulation scenario, the ego vehicle is expected to navigate through dynamic traffic for 1000[m] from a given initial position. Each scenario contains 3 static obstacles and a number of moving vehicles as dynamic obstacles, where the number of moving vehicles in each scenario is randomly selected between 5 and 25. To ensure that we can compare to CCPBA, we choose the $q_{i,j}$ describing each obstacle's location during $\mathcal{T}_j$ as a Gaussian with a standard deviation of $\sigma_{i,j}$ for any $(i, j) \in \mathcal{I} \times \mathcal{J}$. To compare to REFINE, which is a deterministic motion planning algorithm that assumes perfect knowledge of the locations of obstacles, we require that REFINE avoids a box that over-approximates the 5-$\sigma_{i,j}$ region of a Gaussian distribution. Details on the reasoning behind the choice of 5-$\sigma_{i,j}$ are explained in the last paragraph of Appendix B-C.

Each scenario is simulated for 10 trials resulting in a total of 10,000 simulation cases. In each trial, the starting locations of the ego vehicle and obstacles are the same, but the trajectories the obstacles follow are varied to capture the uncertain nature of each scenario. Additional details on how these scenarios and




(a) RADIUS utilized.

(b) CCPBA utilized.

(c) Cantelli MPC utilized.

(d) REFINE utilized.

Fig. 3: Example of a single planning iteration of a simulated scenario in which RADIUS is able to navigate the ego vehicle (black) to the provided waypoint (black cross) through a lane change maneuver solved by one planning iteration, while CCPBA and REFINE execute contingency braking maneuvers as they are unable to find feasible solutions. Cantelli MPC results in a crash, as it has no fail-safe maneuver, so it instead maintains its velocity and continues searching for a feasible solution to execute the lane change. Forward reachable sets are shown in green. Obstacles are shown in white and are marked by their indices to make them trackable among different time instances. Probability distributions are illustrated as purple and blue ellipses where the probability density from low to high is illustrated from blue to purple. Note that for visualization purposes we have buffered the probability distributions shown in this figure by the footprints of the obstacles. Lastly, the convex hull that REFINE uses to over-approximate the 5-$\sigma_{i,j}$ regions of the obstacle probability distributions are shown as gray boxes around the obstacles for all $(i, j) \in \mathcal{I} \times \mathcal{J}$. Note that the distances on the x-axis and y-axis are in [m].

obstacle trajectories are generated can be found in Appendix B-C.

Figure 3 depicts an example of a single planning iteration of a scenario where RADIUS is able to successfully find a solution to execute a lane change to get to the given waypoint, while CCPBA, Cantelli MPC and REFINE are unable to execute the lane change. Table II presents RADIUS, CCPBA, Cantelli MPC, and REFINE's results for all 10,000 cases of the 3-lane highway multiple planning horizon experiment. As seen in Table II, RADIUS is able to achieve a higher success rate than CCPBA and Cantelli MPC. This is in part due to the fact that RADIUS is able to more closely approximate the ground truth risk of collision as shown in Table I. REFINE also has a lower success rate than RADIUS due to its conservative treatment of the uncertain region for the obstacle location. Cantelli MPC has more crashes than CCPBA, and RADIUS because (1) Cantelli MPC, unlike CCPBA and RADIUS can only enforce risk-aware safety at discrete instances so crashes could occur in between these instances, (2) Cantelli MPC does not have a fail-safe stopping maneuver in case it cannot find a solution. Instead, it maintains speed and searches for a new solution and sometimes crashes into the vehicles in front of it.

Notice that despite the fact that $\epsilon = 0.05$, RADIUS has a crash rate that is smaller than 5%. This occurs because even though RADIUS has a significantly tighter over-approximation than the comparison methods, it still over-approximates the true risk of collision. This difference in the over-approximation and the true risk of collision can be attributed to: (1) the multiple relaxations made to compute the closed-form over-approximation highlighted in Section VI-A, and (2) the over-approximation of the actual ego vehicle location within the $j$-th time interval with the collection of maps introduced in Assumption 9. Future work will explore the contribution of each of these relaxations to the extent of over-approximation. **Note that despite this over-approximation, RADIUS is still significantly less conservative than deterministic algorithms like REFINE as seen in Table II**.

| Method | Success [%] | Crash [%] | Safely Stop [%] | Solve Time [s] (Mean, Max.) |
|---|---|---|---|---|
| RADIUS | **80.3** | 0.8 | 18.9 | (0.412, 0.683) |
| CCPBA | 44.5 | 0.4 | 55.1 | **(0.291, 0.417)** |
| Cantelli MPC | 25.6 | 74.4 | 0.0 | (0.743, 3.921) |
| REFINE | 61.0 | **0.0** | 39.0 | (0.341, 0.562) |

TABLE II: Simulation results comparing RADIUS to CCPBA, Cantelli MPC and REFINE on a 1000[m] stretch of dense highway. A successful trial means the ego vehicle was able to successfully travel 1000[m] without crashing or coming to a stop.

*2) Left Turning*

The aim of this experiment is to evaluate how different allowable risk thresholds ($\epsilon$) affect the behavior and performance of RADIUS over a single planning iteration, and unprotected left turns are ideal scenarios for this task. In these scenarios, being less conservative has a large effect on the ability of the ego vehicle to complete the left turn faster by navigating through tight windows in the traffic of oncoming vehicles. In this experiment, the risk threshold is set to 0.01, 0.05, 0.10, 0.20

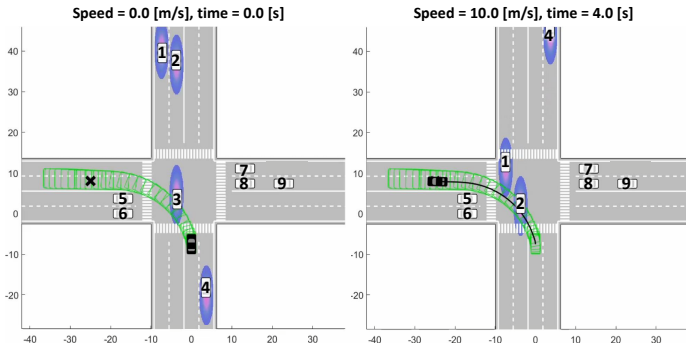



Fig. 4: Example of a simulated unguarded left turning trial where the ego vehicle (black) navigates through a gap in the oncoming traffic to successfully execute the unguarded left turn. Forward reachable sets are shown in green. Obstacles are shown in white and are marked by their indices to make them trackable among different time instances. Probability distributions are illustrated as purple and blue ellipses where the probability density from low to high is illustrated from blue to purple.

| $\epsilon$ | Success [%] | Crash[%] | ATTG [s] | MTTG [s] |
|---|---|---|---|---|
| 0.01 | **100.0** | **0.0** | 8.868 | 13.500 |
| 0.05 | 99.9 | 0.1 | 8.752 | 13.500 |
| 0.10 | 99.5 | 0.5 | 8.187 | 13.400 |
| 0.20 | 98.7 | 1.3 | 6.085 | 10.200 |
| 0.50 | 95.6 | 4.4 | **5.891** | **10.000** |

TABLE III: Simulation results over a single planning iteration for RADIUS executing a left turn at different risk thresholds. Average Time To Goal (ATTG) and Maximum Time To Goal (MTTG) are only taken over successful trials.

and 0.50, and the ego vehicle is tasked with navigating through 100 randomly generated unprotected left turning scenarios. Each scenario is simulated for 10 trials, where in each trial the obstacle trajectories are varied in the same way as mentioned in Section VII-B1. In each scenario, the ego vehicle is tasked with executing an unprotected left turn across two oncoming lanes at a 4-way intersection as depicted in Figure 4. Each scenario contains between 4 to 6 static obstacles occupying the horizontal lanes of the intersection and up to 4 dynamic obstacles that drive through the intersection in the vertical lanes, where the number of static and dynamic obstacles are randomly selected for each scenario. The starting lanes and initial positions of the dynamic obstacles are also randomly selected for each scenario. The initial speeds of the dynamic obstacles are randomly sampled from between 12[m/s] and 17[m/s]. Similar to Section VII-B1, $q_{i,j}$ is chosen to describe an obstacle's location during $\mathcal{T}_j$ as a Gaussian for all $(i.j) \in \mathcal{I} \times \mathcal{J}$.

Table III summarizes the results of the left turning experiment. As the allowable risk threshold $\epsilon$ increases, we see a reduction in the Average Time To Goal (ATTG) and Maximum Time To Goal (MTTG) as the ego vehicle is less conservative with higher $\epsilon$ and thus executes the unprotected left turns more quickly. Additionally, as $\epsilon$ increases, the rate of collision also increases.

### C. Hardware

To illustrate the capabilities of RADIUS, we also tested it on a $\frac{1}{10}$th-scale All-Wheel-Drive car-like robot, Rover, based on a Traxxas RC platform. The first experiment shows RADIUS executing a lane change to overtake an obstacle Rover amidst uncertainty in the obstacle's location with varying risk thresholds. At lower risk thresholds RADIUS is more conservative and is not able to execute the lane change, whereas at higher thresholds RADIUS allows the ego Rover to aggressively overtake the obstacle Rover. The second experiment illustrates a scenario where the obstacle Rover is driven laterally into the path of the ego Rover and it must generate a trajectory with a less than $\epsilon$ risk of collision into the obstacle Rover. At lower risk thresholds, RADIUS chooses to be conservative and triggers its contingency braking maneuver before it intersects with the obstacle Rover, unable to reach the set waypoint. However, at higher risk thresholds RADIUS is more aggressive and executes an aggressive lane change in order to get to the waypoint.

## VIII. Extensions To Other Systems

The proposed algorithm, RADIUS, can generalize to robotic systems other than the vehicle model highlighted in this work, including manipulators and quadrotors. Specifically, RADIUS can be applied to any system that satisfies the following assumptions: 1) The system dynamics must have bounded modeling error (Assumption 2); 2) The system must be able to generate a motion plan in $t_{\text{plan}}$ seconds (Assumption 3); 3) The given system must have a large enough sensor radius as described in Assumption 6; 4) Obstacles must be represented as twice-differentiable probability density functions (Assumption 7); 5) There must exist a set of over-approximative zonotope reachable sets of a parameterized dynamical model for the system (Assumption 9). Note that this last assumption is a critical, robot-specific assumption for RADIUS. However, zonotope reachable sets satisfying Assumption 9 have been constructed for a number of robotic systems including autonomous vehicles [2], manipulators [28] and quadrotors [29].

## IX. Conclusion

This work proposes RADIUS as a real-time, risk-aware trajectory planner for autonomous vehicles operating in uncertain and dynamic environments. The method proposes a novel method for computing a tight, differentiable, closed-form over-approximation of the risk of collision with an obstacle. Given some allowable risk threshold ($\epsilon$), RADIUS uses this over-approximation conjunction with reachability-based methods to generate motion plans that have a no greater than $\epsilon$ risk of collision with any obstacles for the duration of the planned trajectory. Furthermore, RADIUS is able to outperform the existing state-of-the-art chance-constrained and deterministic approaches in a variety of driving scenarios while achieving real-time performance.

## References

[1] S. Kousik, S. Vaskov, F. Bu, M. Johnson-Roberson, and R. Vasudevan, "Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots," *The International Journal of Robotics Research*, vol. 39, no. 12, pp. 1419–1469, 2020.

[2] J. Liu, Y. Shao, L. Lymburner, *et al.*, "Refine: Reachability-based trajectory design using robust feedback linearization and zonotopes," *arXiv preprint arXiv:2211.11997*, 2022.

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

[4] S. Asmussen and P. W. Glynn, *Stochastic simulation: algorithms and analysis*. Springer, 2007, vol. 57.

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

[6] A. Wang, X. Huang, A. Jasour, and B. C. Williams, "Fast risk assessment for autonomous vehicles using learned models of agent futures," *Robotics: Science and Systems*, vol. 2, p. 10, 2020.

[7] F. P. Cantelli, "Sui confini della probabilita," in *Atti del Congresso Internazionale dei Matematici: Bologna del 3 al 10 de settembre di 1928*, 1929, pp. 47–60.

[8] 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.

[9] M. Ahmadi, X. Xiong, and A. D. Ames, "Risk-averse control via cvar barrier functions: Application to bipedal robot locomotion," *IEEE Control Systems Letters*, vol. 6, pp. 878–883, 2021.

[10] A. Dixit, M. Ahmadi, and J. W. Burdick, "Risk-sensitive motion planning using entropic value-at-risk," in *2021 European Control Conference (ECC)*, IEEE, 2021, pp. 1726–1732.

[11] S. Vaskov, "Fast and Safe Trajectory Optimization for Autonomous Mobile Robots using Reachability Analysis," *PhD Thesis*, 2020.

[12] T. D. Gillespie, "Fundamentals of vehicle dynamics," SAE Technical Paper, Tech. Rep., 1992.

[13] S Dieter, M Hiller, and R Baradini, *Vehicle dynamics: Modeling and simulation*, 2018.

[14] T.-Y. Kim, S. Jung, and W.-S. Yoo, "Advanced slip ratio for ensuring numerical stability of low-speed driving simulation: Part ii—lateral slip ratio," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of automobile engineering*, vol. 233, no. 11, pp. 2903–2911, 2019.

[15] J. Lunze and F. Lamnabhi-Lagarrigue, *Handbook of hybrid systems control: theory, tools, applications*. Cambridge University Press, 2009.

[16] S. Vaskov, S. Kousik, H. Larson, *et al.*, "Towards provably not-at-fault control of autonomous robots in arbitrary dynamic environments,"

[17] M.-Y. Yu, R. Vasudevan, and M. Johnson-Roberson, "Occlusion-aware risk assessment for autonomous driving in urban environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2235–2241, 2019.

[18] M.-Y. Yu, R. Vasudevan, and M. Johnson-Roberson, "Risk assessment and planning with bidirectional reachability for autonomous driving," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 5363–5369.

[19] "Real-time ego-motion estimation using lidar and a vehicle model based extended kalman filter," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2014, pp. 431–438.

[20] J. Almeida and V. M. Santos, "Real time egomotion of a nonholonomic vehicle using lidar measurements," *Journal of Field Robotics*, vol. 30, no. 1, pp. 129–141, 2013.

[21] J. F. Fisac, A. Bajcsy, S. L. Herbert, *et al.*, "Probabilistically safe robot planning with confidence-based human predictions," *arXiv preprint arXiv:1806.00109*, 2018.

[22] L. J. Guibas, A. T. Nguyen, and L. Zhang, "Zonotopes as bounding volumes.," in *SODA*, vol. 3, 2003, pp. 803–812.

[23] M. Althoff, "An introduction to cora 2015," in *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015.

[24] T. Hickey, Q. Ju, and M. H. Van Emden, "Interval arithmetic: From principles to implementation," *Journal of the ACM (JACM)*, vol. 48, no. 5, pp. 1038–1068, 2001.

[25] J. B. Lasserre, "Simple formula for integration of polynomials on a simplex," *BIT Numerical Mathematics*, vol. 61, no. 2, pp. 523–533, 2021.

[26] R. E. Caflisch, "Monte carlo and quasi-monte carlo methods," *Acta numerica*, vol. 7, pp. 1–49, 1998.

[27] I. Olkin and T. A. Trikalinos, "Constructions for a bivariate beta distribution," *Statistics & Probability Letters*, vol. 96, pp. 54–60, 2015.

[28] J. Michaux, P. Holmes, B. Zhang, *et al.*, "Can't touch this: Real-time, safe motion planning and control for manipulators under uncertainty," Jan. 2023.

[29] S. Kousik, P. Holmes, and R. Vasudevan, "Technical report: Safe, aggressive quadrotor flight via reachability-based trajectory design," *arXiv preprint arXiv:1904.05728*, 2019.

[30] P. Sahoo and T. Riedel, *Mean value theorems and functional equations*. World Scientific, 1998.

[31] S. Manzinger, C. Pek, and M. Althoff, "Using reachable sets for trajectory planning of automated vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 2, pp. 232–248, 2020.

## APPENDIX A
## PROOF OF LEMMA 12

*Proof:* To relax the integrand $q_{i,j}$, we start by computing the 2nd-order Taylor Expansion of $q_{i,j}$ centered at $w_{\mathcal{T}}^{\text{ctr}} + A_j p$ and applying Mean Value Theorem (MVT) [30, Theorem 4.1] to eliminate higher order terms in the Taylor Expansion as:

$$\begin{aligned} q_{i,j}(w) =& q_{i,j}(w_{\mathcal{S}}^{\text{ctr}} + A_j p) + \frac{\partial q_{i,j}}{\partial w}(w_{\mathcal{S}}^{\text{ctr}} + A_j p) \cdot \\ & \cdot (w - w_{\mathcal{S}}^{\text{ctr}} - A_j p) + \frac{1}{2}(w - w_{\mathcal{S}}^{\text{ctr}} - A_j p)^\top \cdot \\ & \cdot \texttt{Hess}_{q_{i,j}}(w') \cdot (w - w_{\mathcal{S}}^{\text{ctr}} - A_j p) \end{aligned} \tag{25}$$

where $w' \in \mathcal{S} + A_j p$ is some point on the line segment joining points $w_{\mathcal{S}}^{\text{ctr}} + A_j p$ and $w$ in $\mathcal{S} + A_j p$, and $\texttt{Hess}_{q_{i,j}}$ gives the Hessian of $q_{i,j}$. Because MVT does not provide $w'$ in a closed-form, we then drop the dependency of $w'$ in (25) by bounding the Hessian of $q_{i,j}$ as a matrix $H_{\mathcal{S}} \in \mathbb{R}^{2\times 2}$ where $H_{\mathcal{S}}$ is generated by taking element-wise supremum of $\texttt{Hess}_{q_{i,j}}$ over $\mathcal{S} \oplus A_j \mathcal{P}$ using Interval Arithmetic [24]. Note by construction of $\mathcal{S}$ and $w_{\mathcal{S}}^{\text{ctr}}$, it is guaranteed that either $w \geq w_{\mathcal{S}}^{\text{ctr}} + A_j p$ for all $w \in \mathcal{S} + A_j p$ or $w \leq w_{\mathcal{S}}^{\text{ctr}} + A_j p$ for all $w \in \mathcal{S} + A_j p$, thus by definition of $H_{\mathcal{S}}$ the desired inequality holds for all $w \in \mathcal{S} + A_j p$. ∎

## APPENDIX B
## ADDITIONAL DETAILS

Specifications of the full-size FWD vehicle and the Rover robot used in this work can be found in [2, Section IX], from which we also adapt control gains and zonotope reachable sets.

### A. Desired Trajectories

In this work we select 4 families of desired trajectories for driving maneuvers often observed during daily driving: speed changes, direction changes, lane changes and left turning. Each desired trajectory is the concatenation of a driving maneuver and a contingency braking maneuver. Note that the duration $t_{\text{m}}$ of driving maneuvers remains constant within each trajectory family, but can vary among different trajectory families. Each desired trajectory is parameterized by $p = [p_u, p_y]^\top \in \mathcal{P} \subset \mathbb{R}^2$ where $p_u$ and $p_y$ decide desired longitudinal speed and lateral displacement respectively. The trajectory families associated with speed change, direction change and lane change are detailed in [2, Section IX-A], thus we describe desired trajectories that achieve left turning here.

To achieve a left turning maneuver, we set the desired trajectory for longitudinal speed as

$$u^{\text{des}}(t,p) = \begin{cases} \frac{11}{2}t, & \text{if } 0 \leq t < \frac{1}{4}t_{\text{m}} \text{ and } \frac{11}{8}t_{\text{m}} < p_u \\ p_u, & \text{if } 0 \leq t < \frac{1}{4}t_{\text{m}} \text{ and } \frac{11}{8}t_{\text{m}} \geq p_u \\ p_u, & \text{if } \frac{1}{4}t_{\text{m}} \leq t < t_{\text{m}} \\ u^{\text{brake}}(t,p), & \text{if } t \geq t_{\text{m}} \end{cases} \tag{26}$$

where $u^{\text{brake}} : [t_{\text{m}}, t_f] \times \mathcal{P} \to \mathbb{R}$ describes the desired trajectory of longitudinal speed during contingency braking and shares the same formulation with the other 3 trajectory families. The desired trajectory of the yaw rate is as follows:

$$r^{\text{des}}(t,p) = \begin{cases} \frac{1}{2}p_y\left(1 - \cos\left(\frac{4\pi}{t_{\text{m}}}t\right)\right), & \text{if } 0 \leq t < \frac{1}{4}t_{\text{m}} \\ p_y, & \text{if } \frac{1}{4}t_{\text{m}} \leq t < \frac{3}{4}t_{\text{m}} \\ \frac{1}{2}p_y\left(1 - \cos\left(\frac{4\pi}{t_{\text{m}}}t\right)\right), & \text{if } \frac{3}{4}t_{\text{m}} \leq t < t_{\text{m}} \\ 0, & \text{if } t \geq t_{\text{m}} \end{cases} \tag{27}$$

And desired trajectory of heading is set as $h^{\text{des}}(t,p) = h_0 + \int_0^t r^{\text{des}}(\tau, p)\, d\tau$ for any $t \in [0, t_f]$ and $p \in \mathcal{P}$, where $h_0 \in [-\pi, \pi]$ is the initial heading of the ego vehicle at time 0.

The duration $t_{\text{m}}$ of driving maneuvers for every trajectory family is 3[s] for speed change, 3[s] for direction change, 6[s] for lane change, and 4[s] for left change. Because we do not know which desired trajectory ensures not-at-fault *a priori*, to guarantee real-time performance, $t_{\text{plan}}$ should be no greater than the smallest duration of a driving maneuver. Therefore in this work we set $t_{\text{plan}} = 3$[s].

### B. Computational Efficiency of Risk Approximation

In this section we add to the evaluation presented in Section VII-A of the main text. We compare the computation times of the proposed method's approximation of the risk of collision against the Cantelli inequality, and the Chance-Constrained Parallel Bernstein Algorithm (CCPBA) on the same 3000 randomly generated test cases. In each test case, $(i, j, z_0, p)$ is randomly chosen from $\mathcal{I} \times \mathcal{J} \times \mathcal{Z}_0 \times \mathcal{P}$. Table IV summarizes the computation times for generating the risk of collision by approximating the following the integral: $\int_{\xi_j(z_0,p) \oplus <0,\, G^{\text{obs}}>} q_{i,j}(w)\, dw$ when $q_{i,j}$ is modeled as a Gaussian. Table IV shows that RADIUS is able to compute the approximation of the risk of collision about 60% faster than CCPBA, but not as fast as evaluating the Cantelli inequality.

| Method | Mean Solve Time [ms] | Maximum Solve Time [ms] |
|---|---|---|
| RADIUS | 0.4372 | **2.3150** |
| Cantelli | **0.0181** | 2.5223 |
| CCPBA | 1.1012 | 3.5820 |

TABLE IV: Results comparing the computation times for generating an approximation for the risk of collision using RADIUS, CCPBA, and the Cantelli Inequality when the obstacle uncertainty is represented as a Gaussian distribution.

### C. Additional Simulation details

We discuss more in detail the experimental setup for the simulation experiments discussed in Section VII-B. In both the 3-lane highway and left turning scenarios, the uncertainty in the locations and predictions of obstacle motions for each obstacle is represented as a Gaussian distribution $q_{i,j}$ for the stochastic methods for any $(i, j) \in \mathcal{I} \times \mathcal{J}$. More exactly, $q_{i,j}$ is a Gaussian distribution that represents the possible location of the $i$-th obstacle during the $j$-th time interval $\mathcal{T}_j$, and is constructed such that its mean, $\mu_{i,j}$, is at the center of a lane for all $\mathcal{T}_j$. As

time increases we translate this uncertain region forward at a constant speed while keeping it centered in the lane. In other words, $\mu_{i,j} - \mu_{i,j-1}$ is constant for all $j > 1$ in $J$. Recall, from Definition 4, that $L$ and $W$ are the length and width of each obstacle, respectively. Then for any $(i,j) \in \mathcal{I} \times \mathcal{J}$, the standard deviation $\sigma_{i,j}$ of $q_{i,j}$ is chosen such that the $3\sigma_{i,j}$-region of the Gaussian distribution covers the area of width $3.7 - W$ and length $L + u_i(0) \cdot \Delta_t$, where $u_i(0)$ is the speed of the $i$-th obstacle at time 0 and $\Delta_t$ is the time interval defined in Section III-D. Note the choice of width $3.7 - W$ ensures that the footprint of the obstacle always stays inside the lane.

To capture the stochastic nature of each scenario, we simulate each scenario for 10 trials. For each given scenario and for any $i \in \mathcal{I}$, the $i$-th obstacle is always initialized with the same state, but follows different trajectories among different trials of this scenario. These trajectories are selected such that the locations of the $i$-th obstacle during the $j$-th time interval $\mathcal{T}_j$ are randomly sampled from $q_{i,j}$ for each trial, where $q_{i,j}$ is kept constant for all trials of the same scenario for any $(i,j) \in \mathcal{I} \times \mathcal{J}$. This variability in the trajectories allows us to capture trials where a specific obstacle trajectory may cause a crash, whereas in other trials of the same scenario, different trajectories do not cause crashes.

Because REFINE is a deterministic motion planning algorithm, it assumes perfect knowledge of the locations of obstacles. In scenarios where there is uncertainty in the location of obstacles, deterministic algorithms like REFINE often generate motion plans to avoid the entire uncertain region. However, because Gaussian distributions have non-trivial probability density over the entire space $\mathcal{W}$, applying REFINE naively would require avoiding the entire world space. As such, we require REFINE to avoid a box that over approximates the 5-$\sigma_{i,j}$ region of the Gaussian distribution for any $(i,j) \in \mathcal{I} \times \mathcal{J}$ as shown in Figure 3d. Note such 5-$\sigma_{i,j}$ region accounts for 99.99999% of the probability mass.

## APPENDIX C
## ADDITIONAL EXPERIMENTS

In addition to the experiments reported in the main body of this work, we perform two additional experiments to evaluate the proposed method.

### A. Single Planning Horizon 3-Lane Highway Environment

This experiment compares the performance of RADIUS to CCPBA, Cantelli MPC, and REFINE in a 3-lane highway driving scenario in simulation over 10 randomly generated scenarios. The aim of this experiment is to see how RADIUS performs in comparison to the other methods when trying to execute a single lane change in a small stretch of dense highway. Each scenario contains 1 static obstacle and between 4 to 9 dynamic obstacles, where the number of dynamic obstacles was randomly selected for each scenario. It is important to note that in these scenarios, since it was only over a single planning iteration, these obstacles were all spawned within a 100[m] radius of the ego vehicle. This resulted in scenarios that are denser than the situations typically encountered in the 3-lane experiment described in Section VII-B1. The initial speeds of all dynamic obstacles are also randomly sampled from between 15[m/s] to 20[m/s] in each scenario. In each of these scenarios, the ego vehicle is randomly initialized in a lane, and is commanded to get as close as possible to a given waypoint in a different lane. The ego vehicle is expected to achieve the task in a single planning iteration with allowable risk threshold $\epsilon = 0.05$ and $t_{\text{plan}} = 3$[sec]. Success in this experiment is characterized by being able to find a feasible lane change maneuver during the planning time and execute it without crashing. Each scenario is simulated for 200 trials, resulting in a total of 2000 simulation cases. The obstacle uncertainties for the stochastic methods and REFINE are represented in the same way as in Section VII-B1. Table V summarizes RADIUS, CCPBA, Cantelli MPC, and REFINE's performance on the 3-lane single planning iteration scenarios. In this experiment we see that RADIUS was able to successfully execute this lane change maneuver more often than the comparisons.

| Method | Success [%] | Crash [%] | Other Action [%] |
|---|---|---|---|
| RADIUS | **81.8** | **0.0** | **18.2** |
| CCPBA | 55.2 | **0.0** | 44.8 |
| Cantelli MPC | 9.1 | 35.4 | 55.5 |
| REFINE | 21.2 | 0.0 | 78.8 |

TABLE V: Single planning iteration results using RADIUS, CCPBA and Cantelli MPC. "Success" encompasses the trials where each method was able to successfully execute the lane change. "Other Action" encompasses the trials where a method did not complete the lane change maneuver, but instead either executed a safe stop maneuver or decided to keep driving in lane.

### B. Ablation Study

This section performs an ablation study to evaluate the effectiveness of each component of RADIUS. In this ablation study, we examine the performance of 3 algorithms, each of which has a certain aspect of RADIUS removed. Namely, the 3 algorithms are: RADIUS (No Analytical Gradient), Monte-Carlo RTD (Discrete) and Monte-Carlo RTD (Continuous). For RADIUS (No Analytical Gradient), we examine the performance of RADIUS without providing the constraint gradient described in Section VI-C. Instead, we allow IPOPT to compute a numerical gradient during the optimization. In the other 2 algorithms, instead of leveraging our over-approximation to the risk of collision from Section VI, we instead replace this with a Monte Carlo style sampling method to estimate the risk of collision as defined in (14). For Monte-Carlo RTD (Discrete), to find a viable control parameter, we sample discrete points from the control parameter space $\mathcal{P}$ in a similar fashion to [31]. To approximate the risk of collision in a Monte-Carlo fashion, we sample 100 points from $q_{i,j}$ and count the number of points that fall in the zonotope $\xi_j(z_0, p) \oplus <0,\ G^{\text{obs}}>$ for any $(i,j) \in \mathcal{I} \times \mathcal{J}$, where $p$ corresponds to a sampled control parameter. We then sum up these integrals over all $i \in \mathcal{I}$ and $j \in \mathcal{J}$ to estimate the risk of collision with (14). Monte-Carlo RTD (Continuous) estimates the risk of collision in the same way as Monte-Carlo RTD (Discrete), however it searches for

| Algorithm | Success [%] | Crash [%] | Other Action [%] | Solve Time [s] |
|---|---|---|---|---|
| MCRTD(D) | 59.0 | 0.0 | 41.0 | 4.781 |
| MCRTD(C) | 60.0 | 0.0 | 40.0 | 3.334 |
| RADIUS(NAG) | **80.0** | 0.0 | **20.0** | 4.931 |
| RADIUS | **80.0** | 0.0 | **20.0** | **0.412** |

TABLE VI: Single planning iteration results using the RADIUS variants considered in the ablation study. "Other Action" encompasses the trials where each method did not complete the lane change maneuver, but instead either executed a safe stop maneuver or decided to keep driving in the lane. RADIUS (No Analytical Gradient) is abbreviated as RADIUS (NAG), Monte-Carlo RTD (Discrete) is abbreviated as MCRTD(D) and Monte-Carlo RTD (Continuous) is abbreviated as MCRTD(C).

feasible not-at-fault plans over the continuous control parameter space $\mathcal{P}$ and requires IPOPT to compute a numerical gradient during only planning.

We evaluate these algorithms in another set of 10 randomly generated 3-lane highway scenarios, with the same experimental setup from C-A. Each scenario is simulated for 50 trials resulting in a total of 500 test cases. The results of the ablation study can be found in Table VI. From the ablation study, it is evident that replacing the closed-form over-approximation with Monte-Carlo sampling methods causes an 8-10$\times$ increase in the solve time due to a longer risk of collision estimation time. Additionally, both Monte-Carlo methods achieve a roughly 20% lower success rate than RADIUS, which is too slow to be able to achieve real-time performance. We note that increasing the number of samples from 100 would likely increase the success rate as the estimation of the risk of collision would get more accurate. However, this increase in the number of samples will cause the solve time to be even slower. Without the analytical gradients being provided, RADIUS(NAG) requires an average time of about 10$\times$ slower than RADIUS. It is evident from these results that both the analytical gradient and the closed-form over-approximation play important roles in being able to generate motion plans in real time.