2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

ZAPP! Zonotope Agreement of Prediction and Planning for Continuous-Time Collision Avoidance with Discrete-Time Dynamics

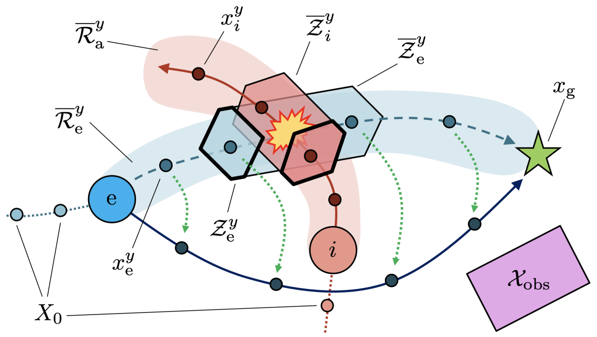
Luca Paparusso[1], Shreyas Kousik[2], Edward Schmerling[3], Francesco Braghin[1], Marco Pavone[3,4]

Abstract— **The past few years have seen immense progress on two fronts that are critical to safe, widespread mobile robot deployment: predicting uncertain motion of multiple agents, and planning robot motion under uncertainty. However, the numerical methods required on each front have resulted in a mismatch of representation for prediction and planning. In prediction, numerical tractability is usually achieved by coarsely discretizing time, and by representing multimodal multi-agent interactions as distributions with infinite support. On the other hand, safe planning typically requires very fine time discretization, paired with distributions with compact support, to reduce conservativeness and ensure numerical tractability. The result is, when existing predictors are coupled with planning and control, one may often find unsafe motion plans. This paper proposes ZAPP (Zonotope Agreement of Prediction and Planning) to resolve the representation mismatch. ZAPP unites a prediction-friendly coarse time discretization and a planning-friendly zonotope uncertainty representation; the method also enables differentiating through a zonotope collision check, allowing one to integrate prediction and planning within a gradient-based optimization framework. Numerical examples show how ZAPP can produce safer trajectories compared to baselines in interactive scenes.**

I. Introduction

The widespread deployment of autonomous mobile robots near and around people is steadily increasing. To ensure safety (i.e., collision avoidance), such robots require predictive models of other agents' uncertain motion [1], [2], plus motion planning methods that can quickly generate collision-avoiding trajectories [3], [4]. However, there is often a mismatch in the numerical representation required to implement predictors and to implement planners and controllers. In particular, predictors typically leverage a coarse time discretization and represent uncertain agent motion via Gaussian mixtures or other unbounded distributions [5]–[10]. In contrast, planning typically requires a fine time discretization to avoid excessive conservativeness and accurately represent dynamics [11]–[13]. Furthermore, since safety assurances are nominally incompatible with unbounded representations of agent motion, planning methods typically consider bounded disturbances (e.g., [14]–[17]). Thus, it remains unclear how best to bridge the representation gap between prediction and motion planning for mobile robots in multi-agent scenes.

We propose a Zonotope Agreement of Prediction and Planning (ZAPP) to address the representation gap between prediction and planning. ZAPP seeks to preserve as many of the learned properties of multi-agent interaction from a predictor while producing a computationally-tractable, planning-compatible representation. We reformulate a generalized notion of prediction using zonotopes, a convex, symmetric polytope representation amenable to robust planning and control, which lets us represent uncertainty and continuous-time motion. ZAPP combines these components into a trajectory optimization framework inspired by recent successes in gradient-based safe motion planning [18]–[21]. We also note that our approach is predictor-agnostic, meaning it can be paired with a variety of existing discrete-time predictors.


$\overline{\mathcal{R}}^y_{\mathrm{a}}$
x^y_i
$\overline{\mathcal{Z}}^y_i$
$\overline{\mathcal{Z}}^y_{\mathrm{e}}$
x_{g}
$\overline{\mathcal{R}}^y_{\mathrm{e}}$
e
$\mathcal{Z}^y_{\mathrm{e}}$
x^y_{e}
i
$\mathcal{X}_{\mathrm{obs}}$
X_0
Fig. 1: Overview of approach and notation. Existing interaction prediction approaches typically use coarse time discretizations to enable long-term predictions, but this can lead to missed collision detections in motion planning. We use zonotopes (blue and red polygons) to represent uncertain, continuous-time reachable sets (light blue and red tubes) for agents in interactive scenes (ego agent in blue, other agent i in red). Note that approximating the reachable sets in discrete time (polygons with thick outlines) may still be insufficient for detecting collisions. To find an ego motion plan, we propose a numerical trajectory optimization approach with zonotope collision-avoidance constraints. The zonotopes are backed out from the outputs of a neural network predictive model, which also provides gradients for trajectory optimization (green dashed arrows). We also illustrate the ego goal x_{g}, agents' state history X_0, and static obstacles $\mathcal{X}_{\mathrm{obs}}$.

Related Work. A wide variety of methods predict future agent motion for navigation and interaction. To achieve high-quality predictions, it is typically necessary to explicitly model interactions, known dynamics, and multimodality [22]–[24]. A further need across prediction architectures is to represent uncertainty in predicted states (e.g., [5]–[7], [25]–[28]). For the sake of numerical tractability, it is common to either assume a Gaussian mixture model [5], [6], or to associate occupancy probabilities with road network structure [25], [29]. A key aspect of these methods is that they predict in *discrete time* for numerical tractability (with the exception

[1]Dept. of Mechanical Engineering, Politecnico di Milano, Milan, Italy.
[2]School of Mechanical Engineering, Georgia Tech, Atlanta, GA, USA.
[3]NVIDIA Autonomous Vehicle Research Group, Santa Clara, CA, USA.
[4]Aeronautics and Astronautics, Stanford University, Stanford, CA, USA.
Toyota Research Institute provided funds to support this work. Corresponding author: Luca Paparusso, `luca.paparusso@polimi.it`.

of [25], which requires known road structure).

Given a prediction model, one can then perform planning and control. In structured settings, one can adopt specific rules of the road as constraints for trajectory planning [17], [30]–[36]. One can also consider *risk tolerance* to allow a user-specified level of constraint violation [32], [37]. In less structured scenes, one can robustly consider interaction dynamics [18], [38]; robustness can introduce conservativeness, which one may mitigate by estimating parameters of other agents' motion [39]. While most methods use discrete time, many planning methods also explicitly consider continuous time in unstructured [34] or structured [17], [33], [35], [40] settings. However, these continuous-time methods typically only consider an *open-loop* prediction model, wherein predictions may be incorrect because they are not conditioned on an ego trajectory plan. Thus, the challenge we address is continuous-time, closed-loop collision avoidance in a numerically-tractable way.

Contributions. Our key insight is that typical discrete-time predictions can be readily extended to continuous-time via zonotopes, enabling a differentiable framework for optimization-based motion planning. Our Zonotope Agreement of Prediction and Planning (ZAPP) is enabled by the following contributions. First, we convert unbounded, multi-modal, discrete-time predictions into zonotopes suitable for robust planning. Second, we extend discrete-time zonotope predictions to continuous time while enabling differentiable collision checking for trajectory optimization. Third, we provide a simulation framework for constructing interactive scenes (see Fig. 2), and we show that ZAPP outperforms a variety of baselines through numerical experiments.

Notation. The real numbers are $\mathbb{R}$ and the natural numbers are $\mathbb{N}$. The floor operator $\lfloor \cdot \rfloor$ rounds down to the nearest integer. The cardinality of $\mathcal{A}$ is $|\mathcal{A}|$. The Minkowski sum is $\mathcal{A} + \mathcal{B} = \{a + b \mid a \in \mathcal{A}, b \in \mathcal{B}\}$. A random variable w, drawn from an n-dimensional normal distribution with mean μ and covariance Σ, is $w \sim \mathcal{N}(\mu, \Sigma)$. The $n \times n$ identity matrix is I_n. An $n \times m$ array of zeros (resp. ones) is $0_{n\times m}$ (resp. $1_{n\times m}$). For a vector v, the i^{th} element is $v[i]$. For an array A, the element at row i, column j is $A[i, j]$; the i^{th} row is $A[i, :]$. The diag operator places its arguments block-diagonally in a matrix of zeros.

II. Problem Formulation

We now pose a generic tightly-coupled prediction and planning problem (see Program (3)). This formulation requires solving a differential equation interaction model and collision checking in continuous time, necessitating a careful choice of representation for practical implementation.

A. Setup

1) Agents, Modes, and Dynamics: Consider a control scenario with an ego agent (which we control) and $m \in \mathbb{N}$ other interacting agents (which we do not control, but we can predict their motion). We assume an upper bound on m is known *a priori*.

We consider multi-modal interaction dynamics, where each mode $y \in \mathcal{Y} = \{1, 2, \cdots, n_y\}$ is a unique homotopy class of the multi-agent system's trajectories, meaning that trajectories from one mode cannot be continuously defomed into those of another mode [41]. For example, in a scene with two agents crossing paths (see Fig. 1), there are at least two modes: one where agent i stops to let the ego agent pass, and one where the ego agent stops to let agent i pass.

The ego agent has state $x_{\text{e}}^y(t) \in \mathbb{R}^{n_\text{e}}$ in mode y at time t. The state vectors of the other m agents in mode y are $x_1^y(t), \cdots, x_m^y(t)$, with state dimensions $n_i \in \mathbb{N}$ for $i = 1, \cdots, m$ respectively. The combined state of the multi-agent system is $x^y(t) \in \mathcal{X} \subset \mathbb{R}^{n_x}$, with $n_x = n_\text{e} + \sum n_i$. We consider a finite time horizon $t_\text{f} > 0$, so $t \in \mathcal{T} = [0, t_\text{f}]$.

We assume that all agents occupy a 2-D workspace; our approach can extend to 3-D by applying techniques similar to [19]–[21]. We further assume that each agent's state x_i^y includes a 2-D *position*, $p_i^y(t) = \text{pos}_i(x^y(t))$. We denote static obstacles $\mathcal{X}_\text{obs} \subset \mathcal{X}$, and assume they are known to the ego agent, since the focus of this work is planning.

We control the ego agent with a signal $u^y : \mathcal{T} \to \mathcal{U}$. We associate the control signal with a particular mode to make this formulation general. We apply a receding-horizon strategy wherein u is reoptimized every $\Delta_t > 0$ seconds. We let t be reset to 0 every Δ_t seconds at the beginning of each plan without loss of generality, so we only consider the time interval $\mathcal{T} = [0, t_\text{f}]$ going forward.

Finally, we denote the multi-agent dynamics in mode y as

$$\dot{x}^y(t) = f^y(t, x^y(t), u^y(t)) + w^y(t). \tag{1}$$

We assume the noise $w^y(t) \sim \mathcal{W}$ is drawn from a distribution with compact support.

2) Reachable Sets: We seek to find safe ego motion plans given the uncertain dynamics (1). To do so, we consider the reachable set of the multiagent system:

$$\mathcal{R}^y(t) = \left\{x^y(t) \mid \dot{x}^y = f^y + w^y \text{ and } x^y(0) \in \mathcal{R}(0)\right\}, \tag{2}$$

where $\mathcal{R}(0)$ is an uncertain set of initial states of all agents, which we assume is known. Note, our proposed zonotope framework in Sec. III can be readily extended to account for occupancy (nonzero volume) of each agent [19]–[21], [40], so we omit occupancy to simplify exposition.

B. Problem Statement

We seek to approximate the following trajectory optimization problem in a numerically tractable way:

$$\begin{aligned}
\min_{\{u^y\}_{y\in\mathcal{Y}}} \quad & J(y, x^y, u^y) && \text{(3a)}\\
\text{s.t.} \quad & \text{pos}_\text{e}(\mathcal{R}^y(t)) \cap \text{pos}_i(\mathcal{R}^y(t)) = \emptyset \;\forall\; t, y, i && \text{(3b)}\\
& \text{pos}_\text{e}(\mathcal{R}^y(t)) \cap \mathcal{X}_\text{obs} = \emptyset \;\forall\; t, y && \text{(3c)}\\
& u^a(\tau) = u^b(\tau) \;\forall\; \tau \in [0, t_\text{c}], a, b \in \mathcal{Y} && \text{(3d)}
\end{aligned}$$

where J is an arbitrary cost function, $t \in \mathcal{T}$, $y \in \mathcal{Y}$, and $i \in \{1, \cdots, m\}$. Program (3) seeks a set of $|\mathcal{Y}|$ control signals $\{u^y : \mathcal{T} \to \mathcal{U}\}_{y\in\mathcal{Y}}$ for which the ego agent does not collide with any other agent (Constraint (3b)) or any obstacles

(Constraint (3c)). Per (2), the reachable set constraints (3b) and (3c) implicitly require the multi-agent system to start from the initial condition set $\mathcal{R}(0)$ and obey the interaction dynamics f. Finally, the control signals must agree up to a *consensus horizon* $t_\mathrm{c} \in [\Delta_t, t_\mathrm{f}]$ (Constraint (3d)); this means that the ego agent must apply the same control at least up to time t_c for all modes. We adopt this strategy because we do not necessarily know the actual mode when solving (3), so we must be able to simultaneously plan a control strategy for each mode. This idea, Contingency MPC [42], has been applied successfully in multi-agent settings [6].

The key challenge is that one can never perfectly represent the reachable sets $\mathcal{R}_i^y(t)$. Furthermore, to enable a numerical solution, we require an efficient, differentiable representation of collision detection per (3b) and (3c).

III. Proposed Method

We now propose ZAPP to represent Problem (3) in a numerically-tractable way. We begin by introducing zonotopes and operations that we use to implement Problem (3) Then, we reformulate Problem (3) as Problem (7) and detail our implementation.

A. Zonotope Preliminaries

We use zonotopes to represent reachable sets. A zonotope $\mathcal{Z} \subset \mathbb{R}^n$ is a convex, symmetrical polytope parameterized by a *center* $c \in \mathbb{R}^n$ and a *generator matrix* $G \subset \mathbb{R}^{n \times n_G}$ as

$$\mathcal{Z} = \mathscr{Z}(c, G) = \{c + G\beta \mid \|\beta\|_\infty \leq 1\}, \tag{4}$$

where $\beta \in \mathbb{R}^{n_G}$, and n_G is the number of *generators* (i.e., the columns of G). This is called the *G-representation*.

Zonotopes enable efficient implementation of many set operations via well-known analytical formulas (see [43], [44]). We use the Minkowski sum to construct continuous-time reachable sets and collision avoidance constraints. The Minkowski sum of $\mathcal{Z}_1 = \mathscr{Z}(c_1, G_1)$ and $\mathcal{Z}_2 = \mathscr{Z}(c_2, G_2)$ is $\mathcal{Z}_1 + \mathcal{Z}_2 = \mathscr{Z}(c_1 + c_2, [G_1, G_2])$. We use the Cartesian product to construct multi-agent reachable sets. The Cartesian product is $\mathcal{Z}_1 \times \mathcal{Z}_2 = \mathscr{Z}([\begin{smallmatrix} c_1 \\ c_2 \end{smallmatrix}], \mathrm{diag} G_1, G_2)$.

We project zonotopes to lower dimensions to extract position information. Suppose the multi-agent system is at state $x(t) \in \mathscr{Z}(c(t), G(t)) \subset \mathbb{R}^{n_x}$, where both $c(t)$ and $G(t)$ have n_x rows. We write $p_i(t) \in \mathrm{pos}_i(\mathscr{Z}(c(t), G(t)))$, which selects the rows of $c(t)$ and $G(t)$ corresponding to the position coordinates of state x_i.

To detect if zonotope reachable sets are in collision, one can check if the center of one zonotope lies in the other zonotope Minkowski summed with the generators of the first:

Proposition 1 ([44, Lemma 5.1]). *Consider the zonotopes $\mathcal{Z}_1 = \mathscr{Z}(c_1, G_2)$ and $\mathcal{Z}_2 = \mathscr{Z}(c_2, G_2)$. Then, $\mathcal{Z}_1 \cap \mathcal{Z}_2 = \emptyset$ if and only if $c_1 \notin \mathscr{Z}(c_2, [G_1, G_2])$.*

Note, checking if a point lies within a zonotope of arbitrary dimension typically requires either an iterative approach [44] or solving a linear program [45], [46]. In this work, we avoid this issue by leveraging two facts: (i) our occupancy sets are in 2-D, and (ii) since zonotopes are polytopes, they can also be represented as a collection of linear inequalities, also know as an H-representation, and represented by a matrix A and a vector b such that $x \in \mathcal{Z} \iff \max(Ax - b) \leq 0$:

Proposition 2 ([43, Theorem 2.1]). *Let $\mathcal{Z} = \mathscr{Z}(c, G) \subset \mathbb{R}^2$, with n_G generators. Assume that G has no generators of length 0. Let $\ell_G \in \mathbb{R}^m$ be a vector of the lengths of each generator: $\ell_G[i] = \|G[:, i]\|_2$. Let $C = \begin{bmatrix} -G[2,:] \\ G[1,:] \end{bmatrix}$. Then*

$$A[:, i] = \frac{1}{\ell_G[i]} \cdot \begin{bmatrix} C \\ -C \end{bmatrix} \in \mathbb{R}^{(2n_G) \times 2}, \quad \text{and} \tag{5a}$$

$$b = C^\top c + \left|C^\top G\right| 1_{m \times 1} \in \mathbb{R}^{2n_G}, \tag{5b}$$

where $|\cdot|$ denotes the absolute value taken elementwise. Then

$$x \notin \mathcal{Z} \iff \min\left(Ax - b\right) > 0. \tag{6}$$

B. Reformulation for Implementation

Let $k_\mathrm{f} = \lfloor t_\mathrm{f}/\Delta_t \rfloor$ and $k_\mathrm{c} = \lfloor t_\mathrm{c}/\Delta_t \rfloor$. Suppose each mode y is associated with a probability $\gamma^y \in [0, 1]$ of occurring; in practice, we use a Gaussian mixture model (GMM) to learn multi-agent uncertain dynamics, so γ^y are the coefficients of the GMM. We implement the following program:

$$\min_{\{\delta U_\mathrm{f}^y\}_{y \in \mathcal{Y}}} \sum_y \gamma^y \cdot \left(\lambda_\mathrm{f} \left\| x^y(k_\mathrm{f}) - x_\mathrm{g} \right\|_2^2 + \sum_{k=0}^{k_\mathrm{f}} \lambda_\mathrm{r} \left\| u^y(k) \right\|_2^2 \right) \tag{7a}$$

$$\text{s.t.} \quad \min\left(A_i^y(k) p_\mathrm{e}^y(k) - b_i^y(k)\right) > 0 \;\forall\; i, k, y \tag{7b}$$

$$\min\left(A_j^y(k) p_\mathrm{e}^y(k) - b_j^y(k)\right) > 0 \;\forall\; j, k, y \tag{7c}$$

$$u^a(k) = u^b(k) \;\forall\; a, b \in \mathcal{Y},\; k = 0, \cdots, k_\mathrm{c} \tag{7d}$$

$$\delta u^y(k) + u_\mathrm{nom}^y(k) = u^y(k) \;\forall\; k, y \tag{7e}$$

where $i \in \{1, \cdots, m\}$, $k \in \{0, \cdots, k_\mathrm{f}\}$, $y \in \mathcal{Y}$, and $k_\mathrm{c} = t_\mathrm{c}\Delta_t$. We set $x_\mathrm{g} \in \mathcal{X}$ as a user-specified goal state and $\lambda_\mathrm{f}, \lambda_\mathrm{r} > 0$ as user-specified weights. The optimization variables $\delta U_\mathrm{f}^y \subset \mathcal{V} \subset \mathbb{R}^{n_u}$ are perturbations of the control signal $u^y : \mathcal{T} \to \mathcal{U}$ with respect to a fixed nominal control signal $u_\mathrm{nom}^y : \mathcal{T} \to \mathcal{U}$, as in (7e); this decision variable formulation is detailed below in Sec. III-D. The collision avoidance constraints for dynamic obstacles (7b) and static obstacles (7c) both leverage an H-representation, where the $A_\star^\star(k)$ and $b_\star^\star(k)$ matrices are constructed using Prop. 2 applied to zonotopes constructed in Sec. III-G. Static obstacles in (7c) are indexed by $j = 1, \cdots, n_\mathrm{obs}$. To implement Program (7), we use a gradient-based solver. In the following, we are careful to ensure that (7b)–(7d) are differentiable with respect to $\{\delta U_\mathrm{f}^y\}_{y \in \mathcal{Y}}$.

Remark 3. *We make a minor abuse of notation to write k instead of $k\Delta_t$ as arguments to time-varying quantities.*

To proceed, first, we detail our decision variable implementation. Then, we approximate discrete-time reachable sets using a prediction model. Next, we extend from discrete to continuous time. Finally, we discuss how to collision check our reachable sets.

C. Dynamics via a Prediction Model

1) Reformulation: It is often numerically intractable to perfectly represent the dynamics f^y, and thus the reachable sets $\mathcal{R}^y(t)$, especially in arbitrary multi-agent settings. Instead, we train a predictor model $\mathscr{P}$ to estimate discretized solutions to (1) by maximizing the likelihood that $x^y(k) \sim \mathscr{P}(y, k, X_0, U_\mathrm{f}^y; \theta)$, where $\theta \in \mathbb{R}^{n_\theta}$ are the trained parameters (e.g., neural network weights). The sequence $X_0 = \big(x(k)\big)_{k=-h}^{0}$ is a finite, discrete state history of length $h \in \mathbb{N}$. Note that the states $x(k)$ in the state history are not associated with a specific mode, since the past mode may not be known and the mode may change in the future. Finally, $U_\mathrm{f}^y \subset \mathcal{U}$ is the planned (future) control signal $u^y : \mathcal{T} \to \mathcal{U}$.

2) Implementation: We use Trajectron++ [5], a state-of-the-art prediction model implemented as a Conditional Variational Auto-Encoder (CVAE). Trajectron++ assumes a known form of dynamics and control for each agent (e.g., 2-D integrator or kinematic unicycle). The model outputs a multimodal distribution over each agent's applied controls at each time step, represented as a Gaussian mixture, with one multivariate Gaussian per mode: $u_i^y(k) \sim \mathscr{N}(\mu_{i,u}^y(k), \Sigma_{i,u}^y(k))$. Here, $\mu_{i,u}^y(k)$ is the mean over controls for agent i in mode y, and $\Sigma_{i,u}^y(k)$ is the covariance matrix. Trajectron++ produces a distribution over each agent's state by propagating the control and previous state distributions forward according to each agent's dynamics, which we denote by the mappings:

$$g_\mu :\ (y, k, X_0, U_\mathrm{f}^y) \mapsto \mu_i^y(k) \text{ and} \tag{8a}$$
$$g_\Sigma :\ (y, k, X_0, U_\mathrm{f}^y) \mapsto \Sigma_i^y(k), \tag{8b}$$

where $\mu_i^y(k) \in \mathbb{R}^{n_i}$ is a mean state and $\Sigma_i^y(k) \in \mathbb{R}^{n_i \times n_i}$ is a state covariance matrix, such that $x_i^y(k) \sim \mathscr{N}(\mu_i^y(k), \Sigma_i^y(k))$ with high likelihood. Note that μ_i^y (mean state) is denoted differently from $\mu_{i,u}^y$ (mean control), and similarly for standard deviation.

D. Decision Variable Implementation

When solving Problem (7) numerically, our predictor would need to be re-evaluated after each solver iteration, because the output prediction is conditioned on the future ego motion. In other words, one must perform a forward-pass of a neural network multiple times for a single MPC solve, which can be computationally burdensome [18]. To avoid this issue, we implement our decision variables as follows.

First, the predictor is evaluated once at the beginning of an MPC execution using $U_{\mathrm{f,nom}}^y \subset \mathcal{U}$, which is a fixed nominal control $u_\mathrm{nom}^y : \mathcal{T} \to \mathcal{U}$. In practice, we create this nominal control as an open-loop sequence of controls that drives the ego agent in a straight line towards its global goal (ignoring static or dynamic obstacles). We obtain nominal mean states $\mu_{i,\mathrm{nom}}^y$ and nominal state covariance matrices $\Sigma_{i,\mathrm{nom}}^y$ as

$$\mu_{i,\mathrm{nom}}^y(k) = g_\mu(y, k, X_0, U_{\mathrm{f,nom}}^y), \tag{9a}$$
$$\Sigma_{i,\mathrm{nom}}^y(k) = g_\Sigma(y, k, X_0, U_{\mathrm{f,nom}}^y), \tag{9b}$$

which are created by rolling out the Trajectron++ predictor dynamics [5]. Then, we consider a first-order Taylor expansion around the nominal control, so our decision variables are δU_f^y such that

$$\hat{\mu}_i^y(k) := \mu_{i,\mathrm{nom}}^y(k) + \left.\frac{\partial g_\mu}{\partial U_\mathrm{f}^y}\right|_{(k, X_0, U_{\mathrm{f,nom}}^y)} \delta U_\mathrm{f}^y, \tag{10}$$

where $\hat{\mu}_i^y(k)$ is the new perturbed mean state.

Optimizing and constraining the perturbations of the control sequence, instead of the control sequence itself, ensures that the updated solutions provided by the solver remain fairly close to the nominal control sequence, which is used for the evaluation of the neural network. Since we use a receding-horizon MPC framework, we take advantage of warmstarting each time we call our solver (i.e., to initialize our decision variable, we take the optimal control sequence from the previous solution, discard the value of the first timestep, and duplicate the value of the last timestep).

E. Discrete-Time Reachable Sets via a Prediction Model

1) Reformulation: We seek to approximate each discrete-time reachable set $\mathcal{R}^y(k)$ as an α-confidence region of the predictor. We extend this to continuous time in Sec. III-F.

2) Implementation: Given the distribution from Trajectron++, we represent the reachable set as a zonotope over-approximating a confidence region of the distribution:

$$\mathcal{R}_i^y(k) \approx \mathcal{Z}_i^y(k) = \mathscr{Z}\Big(\hat{\mu}^y(k), \hat{G}_\Sigma^y(k)\Big), \tag{11}$$

with $\hat{\mu}^y(k) = (\hat{\mu}_\mathrm{e}^y(k), \hat{\mu}_1^y(k), \cdots, \hat{\mu}_m^y(k))$, and $\hat{G}_\Sigma^y(k) = \mathrm{diag}(\hat{G}_\mathrm{e}^y(k), \hat{G}_1^y(k), \cdots, \hat{G}_m^y(k))$. In particular, we construct the generators in each $\hat{G}_i^y(k)$ as the principal axes of a confidence ellipsoid of the distribution $\mathscr{N}(\hat{\mu}_i^y(k), \hat{\Sigma}_i^y(k))$, per [47] and [48, Proposition 2]. We set $\hat{G}_i^y(k) = \varepsilon_i \left[(\lambda_{i,1}^y)^{1/2} v_{i,1}^y, \cdots, (\lambda_{i,n_i}^y)^{1/2} v_{i,n_i}^y\right]$ and $\varepsilon_i = (\mathrm{chi2inv}_{n_i}(\mathrm{erf}(\alpha/\sqrt{2})))^{1/2}$, where $\lambda_{i,j}^y$ is the j^th eigenvalue, $v_{i,j}^y$ is the j^th eigenvector for agent i in mode y, $\mathrm{chi2inv}_{n_i}$ is the inverse of the χ^2 (chi squared) distribution function with n_i degrees of freedom, and erf is the error function [49, Sec. 3.1]. Note that ε_i can be precomputed for a user-specified confidence bound for each agent, and is constant when solving Problem (7). We set $\alpha = 1.0$.

F. Continuous-time Reachable Set Approximation

We now approximate the continuous-time reachable sets from (3b) and (3c) using zonotopes, which we use later in Sec. III-G to generate the constraints (7b) and (7c).

1) Reformulation: We seek to model the continuous-time reachable set for each agent, denoted

$$\overline{\mathcal{R}}_i^y(k) = \bigcup_{\tau \in [k\Delta_t,\, (k+1)\Delta_t]} \mathcal{R}_i^y(x^y(\tau)). \tag{12}$$

2) Implementation: We propose a geometric approximation of (12). We construct $\overline{c}_i^y(k)$ and $\overline{G}_i^y(k)$ such that

$$\overline{\mathcal{R}}_i^y(k) \approx \overline{\mathcal{Z}}_i^y(k) = \mathscr{Z}\Big(\overline{c}_i^y(k), \overline{G}_i^y(k)\Big) \tag{13}$$

The joint continuous-time reachable set for all agents is then

$$\overline{\mathcal{R}}^y(k) \approx \overline{\mathcal{Z}}^y(k) = \overline{\mathcal{Z}}_\mathrm{e}^y(k) \times \overline{\mathcal{Z}}_1^y(k) \times \cdots \times \overline{\mathcal{Z}}_m^y(k). \tag{14}$$

To approximate $\overline{\mathcal{R}}_i^y(k)$ geometrically, we first represent the line segment between $x_i^y(k)$ and $x_i^y(k+1)$ as a zonotope:

$$\mathcal{L}_i^y(k) = \mathscr{Z}\Big(c_{\mathcal{L},i}^y(k), G_{\mathcal{L},i}^y(k)\Big), \text{ with} \tag{15}$$

$$c_{\mathcal{L},i}^y(k) = \tfrac{1}{2}\big(\mu_i^y(k) + \mu_i(k+1)\big) \text{ and} \tag{16}$$

$$G_{\mathcal{L},i}^y(k) = \tfrac{1}{2}\big(\mu_i^y(k+1) - \mu_i^y(k)\big). \tag{17}$$

Then, we approximate $\overline{\mathcal{R}}_i^y$ by extending the reachable set at each time step halfway towards the reachable sets at the previous and subsequent time steps:

$$\begin{aligned}\overline{\mathcal{Z}}_i^y(k) = &\big(\mathcal{Z}_i^y(k) + \tfrac{1}{2}\big(\mathcal{L}_i^y(k) - x_i^y(k)\big)\big) \cup \\ &\big(\mathcal{Z}_i^y(k+1) + \tfrac{1}{2}\big(\mathcal{L}_i^y(k) - x_i^y(k+1)\big)\big),\end{aligned} \tag{18}$$

where $\mathcal{Z}_i^y(k) = \mathscr{Z}\Big(\hat{\mu}_i^y(k), \hat{G}_i^y k\Big)$, and we have used the Minkowski sum of zonotopes as per Sec. III-A.

We leave alternative methods of approximating continuous time to future work. For example, one could compute the convex hull between timesteps with constrained zonotopes [45], [50], but the resulting numerical representations are typically large. One could also apply standard zonotope reachability methods [40], [43], though the resulting set representations may not be differentiable.

G. Collision Avoidance Constraints

We now use our reachable set approximation to create the collision avoidance constraints (7b) and (7c).

1) Reformulation: We reformulate the collision avoidance constraints as

$$\mathrm{pos}_{\mathrm{e}}(\overline{\mathcal{R}}^y(k)) \cap \mathrm{pos}_i(\overline{\mathcal{R}}^y(k)) = \emptyset \quad \text{and} \tag{19a}$$

$$\mathrm{pos}_{\mathrm{e}}(\overline{\mathcal{R}}^y(k)) \cap \mathcal{X}_{\mathrm{obs}} = \emptyset \tag{19b}$$

for each $k = 0, \cdots, k_{\mathrm{f}} - 1$.

2) Implementation: We construct the dynamic collision avoidance constraints as follows. Consider the ego agent and agent i in mode y at time step k. Recall that the continuous time occupancy approximation is given as $\overline{\mathcal{Z}}_i^y(k) = \mathscr{Z}(\overline{c}_i^y(k), \overline{G}_i^y(k))$ for agent i. We apply Prop. 1 and project each such zonotope to the agent's position dimensions to construct a collision check zonotope between the ego and each i^{th} agent:

$$\mathcal{P}_i^y(k) = \mathscr{Z}\big(p_i^y(k), \big[\mathrm{pos}_{\mathrm{e}}(\overline{G}_{\mathrm{e}}^y(k)), \mathrm{pos}_i(\overline{G}_i^y(k))\big]\big), \tag{20}$$

where pos_i extracts the position coordinates of the generator matrix. We convert $\mathcal{P}_i^y(k)$ to an H-representation $(A_i^y(k), b_i^y(k))$ with Prop. 2 to get the constraint in (7b):

$$\min\big(A_i^y(k)p_{\mathrm{e}}^y(k) - b_i^y(k)\big) > 0. \tag{21}$$

We formulate the static obstacle collision avoidance constraints similarly. First, we assume the static obstacles are overapproximated by a finite union of zonotopes: $\mathcal{X}_{\mathrm{obs}} \subseteq \bigcup_{j=1}^{n_{\mathrm{obs}}} \mathscr{Z}(p_{\mathrm{obs},j}, G_{\mathrm{obs},j})$. This is a reasonable assumption for a variety of common obstacle representations, such as occupancy grids or compact polygons. Then, for each obstacle j, we apply Prop. 1 to construct a zonotope $\mathcal{P}_j^y(k) = \mathscr{Z}\big(p_{\mathrm{obs},j}, \big[\mathrm{pos}_{\mathrm{e}}(\overline{G}_{\mathrm{e}}^y(k)), G_{\mathrm{obs},j}\big]\big)$. We apply Prop. 2 to convert to H-representation, $(A_j^y(k), b_j^y(k))$, as in (7c).

Importantly, our collision avoidance constraints are subdifferentiable [51, Ch. 5.1.4], meaning we can use them with a gradient-based optimization solver. This is because we leverage Props. 1 and 2; notice that Prop. 1 only requires matrix concatenation, and Prop. 2 uses arithmetic, concatenations, an absolute value, and a max. In Prop. 2, generator lengths appear in the denominator of (5), but we assume there are no generators of length 0. In practice, we compute derivatives of the constraints with respect to the ego controls using autodifferentiation in PyTorch [52].

Next, we present a numerical experiment to illustrate the utility of our proposed ZAPP approach.

IV. Numerical Experiments

We now present a numerical study to illustrate the utility of our proposed method for uncertainty-aware planning in multi-agent scenes. As part of our codebase, we contribute a framework for generating interactive scenes. We created this because we found that, in preliminary experiments with traffic data [53], [54], traffic agents mostly follow predefined paths with little interaction. Our code is open-source[1]. We implement zonotopes in python using the numpy package [55]. We solve (7) using the Interior Point Method (IPM) solver cyipopt[2]. The cost and constraints gradients are computed via PyTorch [52] automatic differentiation.

We generate our dataset using the environment shown in Figs. 2 and 3. The goal is to navigate a hallway while avoiding 10 non-ego agents. All agents are modeled as double integrators subject to nonlinear forces (note that Trajectron++ [5] can readily handle more complex nonlinear dynamics; we deliberately chose dynamics that allowed us to construct highly interactive scenes). Each (i, j) pair of non-ego agents repels each other proportional to the inverse squared Euclidean distance $1/\left\|p_i^y - p_j^y\right\|_2^2$. They are also repelled, but with much lower magnitude, from the ego agent; this mimics social forces while allowing collisions (see Table I). Finally, non-ego agents are repelled from static obstacles proportional to their relative velocity and inverse of relative distance. This setup renders the prediction problem control-dependent and prevents the surrounding agents from easily avoiding the ego agent, ensuring interaction. All agents have a maximum absolute velocity of 4 m/s and a maximum absolute acceleration of 3 m/s^2 in both the Cartesian directions. We consider the ego agent to have no size (i.e., a point mass), and the surrounding agents to be squares with a side length of 1 m.

We train the predictor on 300 scenes of 8 seconds each. In the training scenes, the ego controls are simulated by creating an artificial potential field that attracts the ego agent towards the global goal, resulting in realistic and smooth control policies.

A. Experiment Setup

1) Task: We consider the control task complete when the ego agent has traveled 28 m horizontally from the start

[1]`https://github.com/lpaparusso/ZAPP`
[2]`https://cyipopt.readthedocs.io/en/stable/`

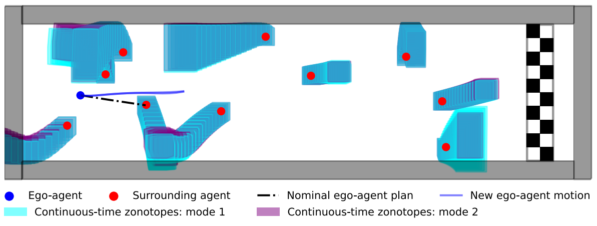


Fig. 2: ZAPP takes as input a nominal ego-agent trajectory (black dashed line), which is used to predict the nominal continuous-time predictions (blue tubes), and finally steers the motion plan to avoid collision (blue line).

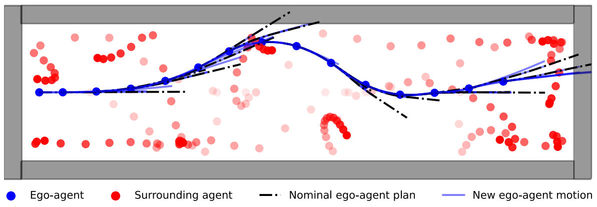


Fig. 3: Example of a completed scenario in the robustness test (w/o DC). ZAPP is able to lead ego to the goal (from left to right) without crashes, implementing receding horizon motion planning (trajectories at each time step shown in blue). To ease visualization, the surrounding agents' trajectories are shown at discrete times with color fading from dark to light as time passes.

point. If a collision involving the ego agent occurs, we register a crash and consider the task not complete. Note, we collision check the robot approximately in continuous time by checking at a much finer time discretization than is used for our MPC planner.

2) Planner Details: Our planner considers the 2 most probable modes for each of the 3 closest surrounding agents. It runs at 2 Hz in simulation time, with the 16 step predictive horizon discretized at 10 Hz. We set $|\delta u^y(k)| \in [-3, 3]$ m/s^2 to ensure that the linearization of the prediction model is a fair assumption (see (7e)). We set the maximum number of solver iterations per MPC execution to 10.

3) Baselines: (i) To understand the relevance of continuous-time approximation, we compare against MATS, a planning-focused predictor that uses linearized dynamics and discrete-time collision avoidance [6]; to ensure a fair comparison, we augment MATS with our discrete-time reachable set approximation. (ii) To test the effect of interaction prediction, we test against ZAPP without interaction gradients (w/o Int.), meaning we do not update the predictions when the ego control changes while solving Problem (7). (iii) To test robustness to model error, we evaluate both MATS and ZAPP without domain consistency (w/o DC), meaning the predictor is trained on different interaction forces from the ones used in the experiment. Future work will compare against additional social navigation baselines [38], [56].

B. Results and Discussion

Quantitative results are shown in Table I. In Figure 2, a qualitative example shows ZAPP in action. A full completed scenario is shown in Figure 3 for the ZAPP w/o DC case.

a) Crash Percentage: ZAPP produces a low crash percentage (approximately 1/10 compared to MATS). This

Method	G/C [%]	AS [m/s]	ST [s]
MATS [6]	70.0 / 30.0	3.81±0.10	2.27±0.62
MATS [6] w/o DC	53.3 / 46.7	3.79±0.14	2.27±0.62
ZAPP w/o DC	86.7 / 13.3	3.75±0.15	4.33±1.72
ZAPP w/o Int.	46.7 / 53.3	**3.86±0.08**	**2.21±0.58**
ZAPP (ours)	**96.7 / 3.3**	3.80±0.14	4.33±1.72

TABLE I: Baseline performance on goals/crashes (G/C), average speed (AS), and average solve time (ST).

shows that collision checking in continuous time has a positive effect on safety and task completion. The fact that ZAPP w/o DC has more crashes than ZAPP indicates that modeling uncertainty (i.e., reachable sets) is necessary, and that improving predictor robustness is critical for future work. This is further supported by the high crash percentage of MATS w/o DC, which is not acceptable in safety-critical control tasks. Overall, these results indicate that the extension of collision checking to continuous time has a positive effect on safety and robustness to perturbations when planning ego motion in dynamic, multi-agent scenes.

b) Average Speed: Table I reports the average speed (towards the goal) of the ego agent over all scenes with no crashes, similar to the metric in [34]. The 4 baselines have similar average speed values, close to the maximum allowed control action of 4 m/s. Therefore, ZAPP improves safety and robustness compared to MATS at practically no expense in task completion speed. However, we stress that the metrics are calculated only for scenarios without crashes. Since ZAPP completes the task almost always, its average speed is more relevant than the one obtained by MATS.

c) Solve Time: MATS baselines, as expected, compute faster because discrete-time collision checking needs half as many zonotopes and constraints. However, all of the obtained solver times are only one order of magnitude slower than real time, even though the predictor and planner are written in non-compiled and non-optimized Python code. Thus, we anticipate that, in future work, our method can readily be made to work in real time with compiled and optimized code, and task-focused prediction architectures (e.g., [9]).

V. Conclusion

This paper presented a Zonotope Agreement of Prediction and Planning (ZAPP) to overcome the gap between prediction and planning numerical representations for mobile robot motion planning in interactive scenes. The method leverages zonotopes to enable continuous-time reasoning for planning, whereas most prediction frameworks are based on discrete time. Numerical experiments show that this enables increased safety, and a significant safety boost over discrete time planning, when generating interactive motion plans in mobile robot settings. Future work will explore alternative methods of extending from discrete to continuous time and alternative representations of reachable sets and uncertainty. We note that ZAPP can extend to include strict safety guarantees with techniques such as [19], [33], [34], [57], [58]. Furthermore, we plan to conduct hardware trials to validate the proposed approach.

References

[1] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, "Human motion trajectory prediction: A survey," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 895–935, 2020.

[2] F. Leon and M. Gavrilescu, "A review of tracking and trajectory prediction methods for autonomous driving," *Mathematics*, vol. 9, no. 6, p. 660, 2021.

[3] M. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," *Robotics and Autonomous Systems*, vol. 100, pp. 171–185, 2018.

[4] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on intelligent transportation systems*, vol. 17, no. 4, pp. 1135–1145, 2015.

[5] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *European Conference on Computer Vision*, Springer, 2020, pp. 683–700.

[6] B. Ivanovic, A. Elhafsi, G. Rosman, A. Gaidon, and M. Pavone, "MATS: An interpretable trajectory forecasting representation for planning and control," in *Conf. on Robot Learning*, Nov. 2020.

[7] H. O. Jacobs, O. K. Hughes, M. Johnson-Roberson, and R. Vasudevan, "Real-time certified probabilistic pedestrian forecasting," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2064–2071, 2017.

[8] S. Shi, L. Jiang, D. Dai, and B. Schiele, "Motion transformer with global intention localization and local movement refinement," *Advances in Neural Information Processing Systems*, vol. 35, pp. 6531–6543, 2022.

[9] A. Kamenev, L. Wang, O. B. Bohan, *et al.*, "Predictionnet: Real-time joint probabilistic traffic prediction for planning, control, and simulation," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 8936–8942.

[10] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "Gohome: Graph-oriented heatmap output for future motion estimation," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 9107–9114.

[11] L. Claussmann, M. Revilloud, D. Gruyer, and S. Glaser, "A review of motion planning for highway autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 5, pp. 1826–1848, 2019.

[12] G. Sánchez and J.-C. Latombe, "On delaying collision checking in prm planning: Application to multi-robot coordination," *The International Journal of Robotics Research*, vol. 21, no. 1, pp. 5–26, 2002.

[13] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *Ieee access*, vol. 2, pp. 56–77, 2014.

[14] C. Danielson, K. Berntorp, A. Weiss, and S. Di Cairano, "Robust motion planning for uncertain systems with disturbances using the invariant-set motion planner," *IEEE Transactions on Automatic Control*, vol. 65, no. 10, pp. 4456–4463, 2020.

[15] 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.

[16] M. Chen, S. L. Herbert, H. Hu, *et al.*, "Fastrack: A modular framework for real-time motion planning and guaranteed safe tracking," *IEEE Transactions on Automatic Control*, vol. 66, no. 12, pp. 5861–5876, 2021.

[17] C. Pek and M. Althoff, "Fail-safe motion planning for online verification of autonomous vehicles using convex optimization," *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 798–814, 2020.

[18] S. Schaefer, K. Leung, B. Ivanovic, and M. Pavone, "Leveraging neural network gradients within trajectory optimization for proactive human-robot interactions," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 9673–9679.

[19] M. Selim, A. Alanwar, S. Kousik, G. Gao, M. Pavone, and K. Johansson, "Safe reinforcement learning using black-box reachability analysis," *IEEE Robotics and Automation Letters*, vol. 7, pp. 10 665–10 672, Oct. 2022.

[20] P. Holmes, S. Kousik, B. Zhang, *et al.*, " Reachable Sets for Safe, Real-Time Manipulator Trajectory Design," Jul. 2020.

[21] S. Kousik, P. Holmes, and R. Vasudevan, "Safe, aggressive quadrotor flight via reachability-based trajectory design," in *Dynamic Systems and Control Conference*, American Society of Mechanical Engineers, vol. 59162, 2019, V003T19A010.

[22] M. Gulzar, Y. Muhammad, and N. Muhammad, "A survey on motion prediction of pedestrians and vehicles for autonomous driving," *IEEE Access*, vol. 9, pp. 137 957–137 969, 2021.

[23] Y. Huang, J. Du, Z. Yang, Z. Zhou, L. Zhang, and H. Chen, "A survey on trajectory-prediction methods for autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 652–674, 2022.

[24] P. Karle, M. Geisslinger, J. Betz, and M. Lienkamp, "Scenario understanding and motion prediction for autonomous vehicles—review and comparison," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 16 962–16 982, 2022.

[25] M. Koschi and M. Althoff, "Set-based prediction of traffic participants considering occlusions and traffic rules," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 2, pp. 249–265, 2020.

[26] T. Zhao, Y. Xu, M. Monfort, *et al.*, "Multi-agent tensor fusion for contextual trajectory prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.

[27] S. Suo, S. Regalado, S. Casas, and R. Urtasun, "Trafficsim: Learning to simulate realistic multi-agent behaviors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 400–10 409.

[28] R. Girgis, F. Golemo, F. Codevilla, *et al.*, "Latent variable sequential set transformers for joint multi-agent motion prediction," in *International Conference on Learning Representations*, 2021.

[29] P. Kaniarasu, G. C. Haynes, and M. Marchetti-Bowick, "Goal-directed occupancy prediction for lane-following actors," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 3270–3276.

[30] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv preprint arXiv:1708.06374*, 2017.

[31] A. Censi, K. Slutsky, T. Wongpiromsarn, *et al.*, "Liability, ethics, and culture-aware behavior specification using rulebooks," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 8536–8542.

[32] A. Undurti and J. P. How, "A decentralized approach to multi-agent planning in the presence of constraints and uncertainty," in *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 2534–2539.

[33] S. Vaskov, H. Larson, S. Kousik, M. Johnson-Roberson, and R. Vasudevan, "Not-at-fault driving in traffic: A reachability-based approach," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2019, pp. 2785–2790.

[34] S. Vaskov, S. Kousik, H. Larson, *et al.*, "Towards provably not-at-fault control of autonomous robots in arbitrary dynamic environments," in *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, Jun. 2019.

[35] S. Söntges and M. Althoff, "Computing possible driving corridors for automated vehicles," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2017, pp. 160–166.

[36] J. Johnson and K. Hauser, "Optimal acceleration-bounded trajectory planning in dynamic environments along a specified path," in *2012 IEEE International Conference on Robotics and Automation*, IEEE, 2012, pp. 2035–2041.

[37] T. Nyberg, C. Pek, L. Dal Col, C. Norén, and J. Tumova, "Risk-aware motion planning for autonomous vehicles with safety specifications," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2021, pp. 1016–1023.

[38] K. Leung, E. Schmerling, M. Zhang, *et al.*, "On infusing reachability-based safety assurance within planning frameworks for human–robot vehicle interactions," *The International Journal of Robotics Research*, vol. 39, no. 10-11, pp. 1326–1345, 2020.

[39] D. Fridovich-Keil, A. Bajcsy, J. F. Fisac, *et al.*, "Confidence-aware motion prediction for real-time collision avoidance1," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 250–265, 2020.

[40] M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 903–918, 2014.

[41] E. Schmitzberger, J.-L. Bouchet, M. Dufaut, D. Wolf, and R. Husson, "Capture of homotopy classes with probabilistic road map," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, vol. 3, 2002, pp. 2317–2322.

[42] J. P. Alsterda, M. Brown, and J. C. Gerdes, "Contingency model predictive control for automated vehicles," in *2019 American Control Conference (ACC)*, IEEE, 2019, pp. 717–722.

[43] M. Althoff, "Reachability analysis and its application to the safety assessment of autonomous cars," Ph.D. dissertation, Technische Universität München, 2010.

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

[45] J. K. Scott, D. M. Raimondo, G. R. Marseglia, and R. D. Braatz, "Constrained zonotopes: A new tool for set-based estimation and fault detection," en, *Automatica*, vol. 69, pp. 126–136, Jul. 2016.

[46] A. Kulmburg and M. Althoff, "On the co-np-completeness of the zonotope containment problem," *European Journal of Control*, vol. 62, pp. 84–91, 2021.

[47] A. Shetty and G. X. Gao, "Trajectory planning under stochastic and bounded sensing uncertainties using reachability analysis," in *Proceedings of the 33rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2020)*, 2020, pp. 1637–1648.

[48] M. Althoff, O. Stursberg, and M. Buss, "Safety assessment for stochastic linear systems using enclosing hulls of probability density functions," in *2009 European Control Conference (ECC)*, IEEE, 2009, pp. 625–630.

[49] B. Wang, W. Shi, and Z. Miao, "Confidence analysis of standard deviational ellipse and its extension into higher dimensional euclidean space," *PloS one*, vol. 10, no. 3, e0118537, 2015.

[50] V. Raghuraman and J. P. Koeln, "Set operations and order reductions for constrained zonotopes," Sep. 2020.

[51] E. Polak, *Optimization: algorithms and consistent approximations*. Springer Science & Business Media, 2012, vol. 124.

[52] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035.

[53] H. Caesar, V. Bankiti, A. H. Lang, *et al.*, "Nuscenes: A multimodal dataset for autonomous driving," in *CVPR*, 2020.

[54] E. Leurent, *An environment for autonomous driving decision-making*, `https://github.com/eleurent/highway-env`, 2018.

[55] C. R. Harris, K. J. Millman, S. J. Van Der Walt, *et al.*, "Array programming with numpy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[56] J. Fisac, A. Bajcsy, S. Herbert, *et al.*, "Probabilistically safe robot planning with confidence-based human predictions," Jun. 2018.

[57] 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.

[58] Y. S. Shao, C. Chen, S. Kousik, and R. Vasudevan, "Reachability-based trajectory safeguard (rts): A safe and fast reinforcement learning safety layer for continuous control," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3663–3670, 2021.