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

Reachability-based Trajectory Design with Neural Implicit Safety Constraints

Jonathan Michaux[1], Qingyi Chen[2], Yongseok Kwon[3] and Ram Vasudevan[1,3]

***Abstract*—Generating safe motion plans in real-time is a key requirement for deploying robot manipulators to assist humans in collaborative settings. In particular, robots must satisfy strict safety requirements to avoid self-damage or harming nearby humans. Satisfying these requirements is particularly challenging if the robot must also operate in real-time to adjust to changes in its environment. This paper addresses these challenges by proposing Reachability-based Signed Distance Functions (RDFs) as a neural implicit representation for robot safety. RDF, which can be constructed using supervised learning in a tractable fashion, accurately predicts the distance between the swept volume of a robot arm and an obstacle. RDF's inference and gradient computations are fast and scale linearly with the dimension of the system; these features enable its use within a novel real-time trajectory planning framework as a continuous-time collision-avoidance constraint. The planning method using RDF is compared to a variety of state-of-the-art techniques and is demonstrated to successfully solve challenging motion planning tasks for high-dimensional systems faster and more reliably than all tested methods. Code, data, and video demonstrations can be found at https://roahmlab.github.io/RDF/.**

I. Introduction

Robotic manipulators will one day assist humans in a variety of collaborative tasks such as mining, farming, and surgery. However, to ensure that they operate robustly in human-centric environments, they must satisfy several important criteria. First, robots should be autonomous and capable of making their own decisions about how to accomplish specific tasks. Second, robots should be safe and only perform actions that are guaranteed to not damage objects in the environment, nearby humans, or even the robot itself. Third, robots should operate in real-time to quickly adjust their behavior as their environment or task changes.

Modern model-based motion planning frameworks are typically hierarchical and consist of three levels: a high-level planner, a mid-level trajectory planner, and a low-level tracking controller. The high-level planner generates a sequence of discrete waypoints between the start and goal locations of the robot. The mid-level trajectory planner computes time-dependent velocities and accelerations at discrete time instances that move the robot between consecutive waypoints.

This work is supported by the Ford Motor Company via the Ford-UM Alliance under award N022977, National Science Foundation Career Award #1751093 and by the Office of Naval Research under Award Number N00014-18-1-2575

[1]Robotics Institute, University of Michigan, Ann Arbor, MI ⟨`jmichaux,,ramv, chenqy`⟩`@umich.edu`.
[2]Computer Science, University of Michigan, Ann Arbor, MI `chenqy@umich.edu.edu`.
[3]Mechanical Engineering, University of Michigan, Ann Arbor, MI ⟨`ramv, kwonys`⟩`@umich.edu`.

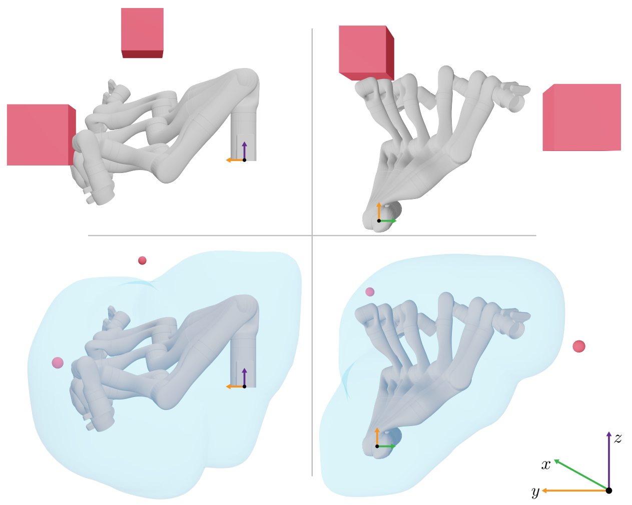

Fig. 1: RDF is a neural implicit safety representation that computes distances between the swept volume of a robotic arm and obstacles within a receding-horizon trajectory planning framework. The top panels show two different orthographic views of a single planning iteration with several intermediate poses of the robot arm in collision with one of the obstacles (red cubes). The bottom panels show orthographic views of the 3D reconstruction of RDFs signed distance field (transparent blue) with one of the obstacle centers (red spheres) interior to RDF's zero-level set.

The low-level tracking controller attempts to track the trajectory as closely as possible. While variations of this motion planning framework have been shown to work on multiple robotic platforms [1], there are still several limitations that prevent wide-scale, real-world adoption of this method. For instance, this approach can be computationally expensive especially as the robot complexity increases, which can make it impractical for real-time applications. By introducing heuristics such as reducing the number of discrete time instances where velocities and accelerations are computed at the mid-level planner, many algorithms achieve real-time performance at the expense of robot safety. Unfortunately, this increases the potential for the robot to collide with obstacles.

To resolve this challenge, this paper proposes Reachability-based Signed Distance Functions (RDFs), a neural implicit representation for robot safety that can be effectively embedded within a mid-level trajectory optimization algorithm. RDF is a novel variation of the signed distance function (SDF) [2]–[4] that computes the distance between the swept volume (*i.e.*, reachable set) of robot manipulator and obstacles in its environment. We use RDFs within a receding-horizon trajectory planning framework to enforce safety by implicitly encoding obstacle-avoidance constraints. RDF has advantages over traditional model-based representations for obstacle-avoidance. First, by approximating the swept volume

RDF learns a continuous-time representation for robot safety. Second, RDF replaces the need to do computationally expensive collision checking at every planning iteration with a rapidly computable forward pass of the network. Third, RDF's fast inference and gradient computations make it ideal as a constraint in trajectory optimization problems. Fourth, as we illustrate in this paper, RDF scales better than existing planning algorithms with respect to the dimension of the system.

A. Related Work

Our approach lies at the intersection of swept volume approximation, neural implicit shape representation, and robot motion planning. We summarize algorithms in these areas and highlight their computational tractability.

Swept volume computation [5]–[9] has a rich history in robotics [10] where it has been used for collision detection during motion planning [11]. Because computing exact swept volumes of articulated robots is analytically intractable [10], many algorithms rely on approximations using convex polyhedra, occupancy grids, and CAD models [12]–[14]. However, these methods often suffer from high computational costs and are generally not suitable when generating complex robot motions and as a result, are only applied while performing offline motion planning [15]. To address some of these limitations, a recent algorithm was proposed to compute a single probabilistic road map offline whose safety was verified online by performing a parallelized collision check with a precomputed swept volume at run-time [16]. However, this method was not used for real-time trajectory generation.

An alternative to computing swept volumes is to buffer the robot and perform collision-checking at discrete time instances along a given trajectory. This is common with state-of-art trajectory optimization approaches such as CHOMP [17] and TrajOpt [18]. Although these methods have been demonstrated to generate robot motion plans in real-time by reducing the number of discrete time instances where collision checking occurs, they cannot be considered safe as they enforce collision avoidance via soft penalties in the cost function.

A recent approach called Reachability-based Trajectory Design (RTD) [19] combines both swept volume approximation and trajectory optimization to generate safe motion plans in real-time. At runtime, RTD uses zonotope arithmetic to build a differentiable reachable set representation that overapproximates the swept volume corresponding to a continuum of possible trajectories the robot could follow. It then solves an optimization problem to select a feasible trajectory such that the subset of the swept volume encompassing the robot's motion is not in collision. Importantly, the reachable sets are constructed so that obstacle avoidance is satisfied in continuous-time. While extensions of RTD have demonstrated real-time, certifiably-safe motion planning for robotic arms [1], [20] with seven degrees of freedom, applying RTD to higher dimensional systems is challenging because, as we illustrate in this paper, it is unable to construct reach sets rapidly.

A growing trend in machine learning and computer vision is to implicitly represent 3D shapes with learned neural networks. One seminal work in this area is DeepSDF [2], which was the first approach to learn a continuous volumetric field that reconstructs the zero-level set of entire classes of 3D objects. Gropp et. al [3] improved the training of SDFs by introducing an *Eikonal* term into their loss function that acts as an implicit regularizer to encourage the network to have unit $l2$-norm gradients. Neural implicit representations have also been applied to robotics problems including motion planning [21], [22], mapping [23], [24], and manipulation [25]–[27]. Particularly relevant to our current approach is the work by Koptev et. al [28], which learned an SDF as an obstacle-avoidance constraint for safe reactive motion planning. Similar to approaches described above, [28] only enforces safety constraints at discrete time points.

B. Contributions

The present work investigates learned neural implicit representations with reachability-based trajectory design for fast trajectory planning. The contributions of this paper are as follows:

1. A neural implicit representation called RDF that computes the distance between a parameterized swept volume trajectory and obstacles;
2. An efficient algorithm to generate training data to construct RDF; and
3. A novel, real-time optimization framework that utilizes RDF to construct a collision avoidance constraint.

To illustrate the utility of this proposed optimization framework we apply it to perform real-time motion planning on a variety of robotic manipulator systems and compare it to several state of the art algorithms. In particular, we demonstrate that RDF has advantages over both model-based and learning-based representations of robot safety. Unlike reachability-based methods [1], [19], [20], [29], [30], RDF can tractably scale up to higher dimensional state space models without sacrificing real-time operation. Neural SDFs have also been used as constraints in MPC algorithms for safe motion planning [24], [28]. However, these methods rely on evaluating the safety constraints at discretized time points, thus introducing an undesirable trade-off between safety and computation speed. More importantly it is unclear how finely to discretize an SDF to simultaneously ensure safety and real-time operation. RDF overcomes the challenges inherent in each of these prior approaches.

The remainder of the paper is organized as follows: Section II summarizes the set representations used throughout the paper; Section III describes the safe motion planning problem of interest; Section IV defines several distance functions that are used to formulate RDF; Section V formulates the safe motion planning problem; Section VI describes how to build RDF and use it within a safe motion planning framework; Sections VII and VIII summarize the evaluation of the proposed method on a variety of different example problems.

II. Preliminaries

This section describes the representations of sets and operations on these representations that we use throughout the paper. Note all norms that are not explicitly specified are the 2-norm.

A. Zonotopes and Polynomial Zonotopes

We begin by defining zonotopes, matrix zonotopes, and polynomial zonotopes.

Definition 1. *A* zonotope $Z \subset \mathbb{R}^n$ *is a convex, centrally-symmetric polytope defined by a* center $c \in \mathbb{R}^n$*,* generator matrix $G \in \mathbb{R}^{n \times n_g}$*, and* indeterminant vector $\beta \in \mathbb{R}^{n_g}$*:*

$$Z = \{z \in \mathbb{R}^n \mid z = c + G\beta, ||\beta||_\infty \leq 1\} \tag{1}$$

where there are $n_g \in \mathbb{N}$ generators. When we want to emphasize the center and generators of a zonotope, we write $Z = (c, G)$.

Definition 2. *A* matrix zonotope $M \subset \mathbb{R}^{n \times m}$ *is a zonotope with center $C \in \mathbb{R}^{n \times m}$ and generators $G_i \in \mathbb{R}^{n \times m}$ such that*

$$M = \left\{ A \in \mathbb{R}^{n \times m} \mid A = C + \sum_{i=1}^{n_g} G_i \beta_i, \beta_i \in [-1,1] \right\}. \tag{2}$$

Definition 3 (Polynomial Zonotope)**.** *A* polynomial zonotope $\mathbf{P} \subset \mathbb{R}^n$ *is given by its generators $g_i \in \mathbb{R}^n$ (of which there are n_g), exponents $\alpha_i \in \mathbb{N}^{n_g}$, and indeterminates $x \in [-1,1]^{n_g}$ as*

$$\mathbf{P} = \mathcal{PZ}(g_i, \alpha_i, x) = \left\{ z \in \mathbb{R}^n \mid z = \sum_{i=0}^{n_g} g_i x^{\alpha_i}, x \in [-1,1]^{n_g} \right\}. \tag{3}$$

We refer to x^{α_i} as a monomial*. A* term $g_i x^{\alpha_i}$ *is produced by multiplying a monomial by the associated generator g_i.*

Note that one can think of a zonotope as a special case of polynomial zonotope where one has an exponent made up of all zeros and the remainder of exponents only have one non-zero element that is equal to one. As a result, whenever we describe operations on polynomial zonotopes they can be extended to zonotopes. When we need to emphasize the generators and exponents of a polynomial zonotope, we write $\mathbf{P} = \mathcal{PZ}(g_i, \alpha_i, x)$. Throughout this document, we exclusively use bold symbols to denote polynomial zonotopes.

B. Operations on Zonotopes and Polynomial Zonotopes

This section describes various set operations.

1) Set-based Arithmetic: Given a set $\Omega \subset \mathbb{R}^{n_d}$, let $\partial\Omega \subset \mathbb{R}^{n_d}$ be its boundary and $\Omega^c \subset \mathbb{R}^{n_d}$ denote its complement.

Definition 4. *The* convex hull operator $conv: \mathbb{R}^{n_d} \to \mathbb{R}^{n_d}$ *is defined by*

$$conv(\Omega) = \bigcap C_\alpha \tag{4}$$

where C_α is a convex set containing Ω.

Let U, $V \subset \mathbb{R}^n$. The *Minkowski Sum* is $U \oplus V = \{u + v \mid u \in U, v \in V\}$; the *Multiplication* of $UV = \{uv \mid u \in U, v \in V\}$ where all elements in U and V must be appropriately sized to ensure that their product is well-defined.

Operation	Computation
$\mathbf{P}_1 \oplus \mathbf{P}_2$ (PZ Minkowski Sum) ([20], eq. (19))	Exact
$\mathbf{P}_1\mathbf{P}_2$ (PZ Multiplication) ([20], eq. (21))	Exact
$\texttt{slice}(\mathbf{P}, x_j, \sigma)$ ([20], eq. (23))	Exact
$\texttt{inf}(\mathbf{P})$ ([20], eq. (24))	Overapproximative
$\texttt{sup}(\mathbf{P})$ ([20], eq. (25))	Overapproximative
$f(\mathbf{P}_1) \subseteq \mathbf{P}_2$ (Taylor expansion) ([20], eq. (32))	Overapproximative

TABLE I: Summary of polynomial zonotope operations.

2) Polynomial Zonotope Operations: As described in Tab. I, there are a variety of operations that we can perform on polynomial zonotopes (*e.g.*, minkowski sum, multiplication, etc.). The result of applying these operations is a polynomial zonotope that either exactly represents or over approximates the application of the operation on each individual element of the polynomial zonotope inputs. The operations given in Tab. I are rigorously defined in [20]. A thorough introduction to polynomial zonotopes can be found in [31].

One desirable property of polynomial zonotopes is the ability to obtain subsets by plugging in values of known indeterminates. For example, say a polynomial zonotope $\mathbf{P}$ represented a set of possible positions of a robot arm operating near an obstacle. It may be beneficial to know whether a particular choice of $\mathbf{P}$'s indeterminates yields a subset of positions that could collide with the obstacle. To this end, we introduce the operation of "slicing" a polynomial zonotope $\mathbf{P} = \mathcal{PZ}(g_i, \alpha_i, x)$ by evaluating an element of the indeterminate x. Given the j^{th} indeterminate x_j and a value $\sigma \in [-1,1]$, slicing yields a subset of $\mathbf{P}$ by plugging σ into the specified element x_j:

$$\texttt{slice}(\mathbf{P}, x_j, \sigma) \subset \mathbf{P} = \left\{ z \in \mathbf{P} \mid z = \sum_{i=0}^{n_g} g_i x^{\alpha_i}, x_j = \sigma \right\}. \tag{5}$$

One particularly important operation that we require later in the document, is an operation to bound the elements of a polynomial zonotope. It is possible to efficiently generate these upper and lower bounds on the values of a polynomial zonotope through overapproximation. In particular, we define the $\texttt{sup}$ and $\texttt{inf}$ operations which return these upper and lower bounds, respectively, by taking the absolute values of generators. For $\mathbf{P} \subseteq \mathbb{R}^n$, these return

$$\texttt{sup}(\mathbf{P}) = g_0 + \sum_{i=1}^{n_g} |g_i|, \tag{6}$$

$$\texttt{inf}(\mathbf{P}) = g_0 - \sum_{i=1}^{n_g} |g_i|. \tag{7}$$

Note that for any $z \in \mathbf{P}$, $\texttt{sup}(\mathbf{P}) \geq z$ and $\texttt{inf}(\mathbf{P}) \leq z$, where the inequalities are taken element-wise. These bounds may not be tight because possible dependencies between indeterminates are not accounted for, but they are quick to compute.

III. Arm and Environment

This section summarizes the robot and environmental model that is used throughout the remainder of the paper.

A. Robotic Manipulator Model

Given an n_q degree of freedom serial robotic manipulator with configuration space Q and a compact time interval $T \subset \mathbb{R}$ we define a trajectory for the configuration as $q: T \to Q \subset \mathbb{R}^{n_q}$. The velocity of the robot is $\dot{q}: T \to \mathbb{R}^{n_q}$. Let $N_q = \{1, \ldots, n_q\}$. We make the following assumptions about the structure of the robot model:

Assumption 5. *The robot operates in an n_d-dimensional workspace, which we denote $W \subset \mathbb{R}^{n_d}$. The robot is composed of only revolute joints, where the j^{th} joint actuates the robot's j^{th} link. The robot's j^{th} joint has position and velocity limits given by $q_j(t) \in [q^-_{j,\lim}, q^+_{j,\lim}]$ and $\dot{q}_j(t) \in [\dot{q}^-_{j,\lim}, \dot{q}^+_{j,\lim}]$ for all $t \in T$, respectively. Finally, the robot is fully actuated, where the robot's input is given by $u: T \to \mathbb{R}^{n_q}$.*

One can make the one-joint-per-link assumption without loss of generality by treating joints with multiple degrees of freedom (*e.g.*, spherical joints) as links with zero length. Note that we use the revolute joint portion of this assumption to simplify the description of forward kinematics; however, these assumptions can be easily extended to more complex joint using the aforementioned argument or can be extended to prismatic joints in a straightforward fashion.

Note that the lack of input constraints means that one could apply an inverse dynamics controller [32] to track any trajectory of of the robot perfectly. As a result, we focus on modeling the kinematic behavior of the manipulator. Note, that the approach presented in this paper could also be extended to deal with input limits using a dynamic model of the manipulator; however, in the interest of simplicity we leave that extension for future work.

1) Arm Kinematics: Next, we introduce the robot's kinematics. Suppose there exists a fixed inertial reference frame, which we call the *world* frame. In addition suppose there exists a *base* frame, which we denote the 0th frame, that indicates the origin of the robot's kinematic chain. We assume that the j^{th} reference frame $\{\hat{x}_j, \hat{y}_j, \hat{z}_j\}$ is attached to the robot's j^{th} revolute joint, and that $\hat{z}_j = [0, 0, 1]^\top$ corresponds to the j^{th} joint's axis of rotation. Then for a configuration at a particular time, $q(t)$, the position and orientation of frame j with respect to frame $j-1$ can be expressed using homogeneous transformations [32, Ch. 2]:

$$H^{j-1}_j(q_j(t)) = \begin{bmatrix} R^{j-1}_j(q_j(t)) & p^{j-1}_j \\ 0 & 1 \end{bmatrix}, \tag{8}$$

where $R^{j-1}_j(q_j(t))$ is a configuration-dependent rotation matrix and p^{j-1}_j is the fixed translation vector from frame $j-1$ to frame j.

With these definitions, we can express the forward kinematics of the robot. Let $\text{FK}_j : Q \to \mathbb{R}^{4\times 4}$ map the robot's configuration to the position and orientation of the j^{th} joint in the world frame:

$$\text{FK}_j(q(t)) = \prod_{l=1}^{j} H^{l-1}_l(q_l(t)) = \begin{bmatrix} R_j(q(t)) & p_j(q(t)) \\ 0 & 1 \end{bmatrix}, \tag{9}$$

where

$$R_j(q(t)) := R^0_j(q(t)) = \prod_{l=1}^{j} R^{l-1}_l(q_l(t)) \tag{10}$$

and

$$p_j(q(t)) = \sum_{l=1}^{j} R_l(q(t)) p^{l-1}_l. \tag{11}$$

B. Arm Occupancy

Next, we define the forward occupancy of the robot by using the arm's kinematics to describe the volume occupied by the arm in the workspace. Let $\mathbf{L_j} \subset \mathbb{R}^3$ denote a polynomial zonotop overapproximation to the volume occupied by the j^{th} link with respect to the j^{th} reference frame. The forward occupancy of link j is the map $\text{FO}_j : Q \to \mathcal{P}(W)$ defined as

$$\text{FO}_j(q(t)) = p_j(q(t)) \oplus R_j(q(t)) L_j, \tag{12}$$

where the first expression gives the position of joint j and the second gives the rotated volume of link j. The volume occupied by the entire arm in the workspace is given by the function $\text{FO} : Q \to \mathcal{P}(W)$ that is defined as

$$\text{FO}(q(t)) = \bigcup_{j=1}^{n_q} \text{FO}_j(q(t)) \subset W. \tag{13}$$

For convenience, we use the notation $\text{FO}(q(T))$ to denote the forward occupancy over an entire interval T.

C. Environment

Next, we describe the arm's environment and its obstacles.

1) Obstacles: The arm must avoid obstacles in the environment while performing motion planning. These obstacles satisfy the following assumption:

Assumption 6. *The transformation between the world frame of the workspace and the base frame of the robot is known, and obstacles are represented in the base frame of the robot. At any time, the number of obstacles $n_{\mathcal{O}} \in \mathbb{N}$ in the scene is finite. Let $\mathcal{O}$ be the set of all obstacles $\{O_1, O_2, \ldots, O_{n_{\mathcal{O}}}\}$. Each obstacle is convex, bounded, and static with respect to time. The arm has access to a zonotope that overapproximates the obstacle's volume in workspace. Each zonotope overapproximation of the obstacle has the same volume and is an axis-aligned cube.*

A convex, bounded object can always be overapproximated as a zonotope [33]. In addition, if one is given a non-convex bounded obstacle, then one can outerapproximate that obstacle by computing its convex hull. If one has an obstacle that is larger than the pre-fixed axis-aligned cube, then one can introduce several axis-aligned cubes whose union is an overapproximation to the obstacle. Note because we use the zonotope overapproximation during motion planning, we conflate the obstacle with its zonotope overapproximation throughout the remainder of this document.

Dynamic obstacles may also be considered within the RDF framework by introducing a more general notion of safety [29, Definition 11], but we omit this case in this paper to ease exposition. Finally, if a portion of the scene is occluded then one can treat that portion of the scene as an obstacle. We say that the arm is *in collision* with an obstacle if $\mathrm{FO}_j(q(t)) \cap O_\ell \neq \emptyset$ for any $j \in N_q$ or $\ell \in N_{\mathscr{O}}$ where $N_{\mathscr{O}} = \{1, \ldots, n_{\mathscr{O}}\}$.

D. Trajectory Design

Our goal is to develop an algorithm to compute safe trajectories in a receding-horizon manner by solving an optimization program over parameterized trajectories at each planning iteration. These parameterized trajectories are chosen from a pre-specified continuum of trajectories, with each uniquely determined by a *trajectory parameter* $k \in K \subset \mathbb{R}^{n_k}$, $n_k \in \mathbb{N}$. K is compact and can be designed in a task dependent or robot morphology-specific fashion [1], [19], [30], [34], as long as it satisfies the following properties.

Definition 7 (Trajectory Parameters)**.** *For each $k \in K$, a* parameterized trajectory *is an analytic function $q(\,\cdot\,;k) : T \to Q$ that satisfies the following properties:*

I. The parameterized trajectory starts at a specified initial condition $(q_0, \dot{q}_0)$, so that $q(0;k) = q_0$, and $\dot{q}(0;k) = \dot{q}_0$.
II. $\dot{q}(t_f;k) = 0$ (i.e., each parameterized trajectory brakes to a stop, and at the final time has zero velocity).

The first property allows for parameterized trajectories to be generated online. In particular, recall that RDF performs real-time receding horizon planning by executing a desired trajectory computed at a previous planning iteration while constructing a desired trajectory for the subsequent time interval. The first property allows parameterized trajectories that are generated by RDF to begin from the appropriate future initial condition of the robot. The second property ensures that a fail safe braking maneuver is always available.

IV. Reachability-based Signed Distance Functions

This section defines the signed distance function between sets. Signed distance functions are used in robotics in a variety of applications including representing collision avoidance constraints. This section describes how to extend the signed distance function to a distance function between the forward occupancy of a robot and an obstacle. This novel distance function, which we call the reachability-based signed distance function (RDF), enables us to formulate the collision avoidance problem between a parameterized reachable set and an obstacle as an optimization problem.

A. Overview of Signed Distance Fields

We begin by defining an unsigned distance function:

Definition 8. *Given a set $\Omega \subset \mathbb{R}^{n_d}$, the* distance function *associated with Ω is defined by*

$$d(x;\Omega) = \min_{y \in \partial\Omega} \|x - y\|. \tag{14}$$

The distance between two sets $\Omega_1, \Omega_2 \subset \mathbb{R}^{n_d}$ *is defined by*

$$d(\Omega_1, \Omega_2) = \min_{\substack{x \in \partial\Omega_1 \\ y \in \partial\Omega_2}} \|x - y\|. \tag{15}$$

Notice that this distance function is zero for sets that have non-trivial intersection. As a result, this distance function provides limited information for such sets (i.e., it is unclear how much they are intersecting with one another). To address this limitation, we consider the following definition:

Definition 9. *Given a subset Ω of $\mathbb{R}^{n_d}$, the* signed distance function *s between a point is a map $s : \mathbb{R}^{n_d} \to \mathbb{R}$ defined as*

$$s(x;\Omega) = \begin{cases} d(x, \partial\Omega) & \text{if } x \in \Omega^c \\ -d(x, \partial\Omega) & \text{if } x \in \Omega. \end{cases} \tag{16}$$

The signed distance between two sets $\Omega_1, \Omega_2 \subset \mathbb{R}^{n_d}$ *is defined as*

$$s(\Omega_1, \Omega_2) = \begin{cases} d(\Omega_1, \Omega_2) & \text{if } \Omega_1 \cap \Omega_2 = \emptyset \\ -d(\Omega_1, \Omega_2) & \text{otherwise.} \end{cases} \tag{17}$$

Note that signed distance functions are continuous [35], differentiable almost everywhere [35], [36], and satisfy the *Eikonal* equation:

Definition 10. *Suppose s is the signed distance function associated with a set $\Omega \subset R^{n_d}$. Then the gradient of s satisifes the Eikonal Equation which is defined as*

$$\|\nabla s(x)\| = 1. \tag{18}$$

We use this property to construct our loss term in VI-C

B. Reachability-Based Signed Distance Functions

This subsection describes the reachability-based distance function as the signed distance function associated with forward occupancy of a robot.

Definition 11. *The* reachability-based distance *function associated with the forward occupancy reachable set $\mathrm{FO}(q(T;k))$ is a mapping defined by*

$$r(x, \mathrm{FO}_j(q(T;k))) = \min_{j \in N_q} r_j(x; \mathrm{FO}_j(q(T;k))) \tag{19}$$

where r_j is the signed distance function associated with the j^{th} forward occupancy FO_j such that

$$r_j(x; \mathrm{FO}_j(q(T;k))) = s(x; \mathrm{FO}_j(q(T;k))). \tag{20}$$

The reachability-based distance *between an obstacle $O \subset \mathbb{R}^d$ and the robot's forward occupancy reachable set FO is defined by*

$$r(O, \mathrm{FO}(q(T;k))) = \min_{j \in N_q} s(O, \mathrm{FO}_j(q(T;k))). \tag{21}$$

One can use this distance function to formulate trajectory optimization problems as we describe next.

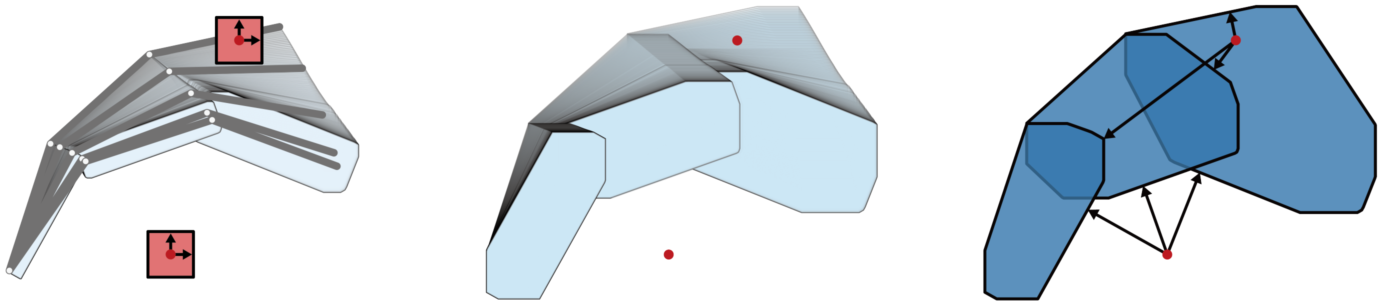

Fig. 2: Visual illustration of Alg. 1. (Left) Forward occupancy reachable set as a function of some initial conditions (Alg. 1, line 6). (Middle) Forward occupancy reachable set buffered by the obstacle generators (Alg. 1, line 7). (Right) Convex hull of each link's forward occupancy reach set (Alg. 1, line 11).

V. Formulating the Motion Planning Problem Using Polynomial Zonotopes

To construct a collision free trajectory in a receding-horizon fashion, one could try to solve the following nonlinear optimization problem at each planning iteration:

$$\min_{k\in K} \quad \texttt{cost}(k) \tag{22}$$
$$q_j(t;k) \subseteq [q^-_{j,\mathrm{lim}}, q^+_{j,\mathrm{lim}}] \qquad \forall j \in N_q, t \in T \tag{23}$$
$$\dot{q}_j(t;k) \subseteq [\dot{q}^-_{j,\mathrm{lim}}, \dot{q}^+_{j,\mathrm{lim}}] \qquad \forall j \in N_q, t \in T \tag{24}$$
$$r(O_\ell, \mathrm{FO}(q(T;k))) > 0 \qquad \forall \ell \in N_{\mathscr{O}} \tag{25}$$

The cost function (22) specifies a user-defined objective, such as bringing the robot close to some desired goal. Each of the constraints guarantee the safety of any feasible trajectory parameter. The first two constraints ensure that the trajectory does not violate the robot's joint position and velocity limits. The last constraint ensures that the robot does not collide with any obstacles in the environment. Note in this optimization problem, we have assumed that the robot does not have to deal with self-intersection constraints.

Implementing a real-time optimization algorithm to solve this problem is challenging for several reasons. First, the constraints associated with obstacle avoidance are non-convex. Second, the constraints must be satisfied for all time t in an uncountable set T. To address these challenges, a recent paper proposed to represent the trajectory and the forward occupancy of the robot using a polynomial zonotope representation [20]. We summarize these results below.

A. Time Horizon and Trajectory Parameter PZs

We first describe how to create polynomial zonotopes representing the planning time horizon T. We choose a timestep Δt so that $n_t := \frac{T}{\Delta t} \in \mathbb{N}$. Let $N_t := \{1, \dots, n_t\}$. Divide the compact time horizon $T \subset \mathbb{R}$ into n_t time subintervals. Consider the i^{th} time subinterval corresponding to $t \in [(i-1)\Delta t, i\Delta t]$. We represent this subinterval as a polynomial zonotope $\mathbf{T_i}$, where

$$\mathbf{T_i} = \left\{ t \in T \mid t = \tfrac{(i-1)+i}{2}\Delta t + \tfrac{1}{2}\Delta t x_{t_i},\ x_{t_i} \in [-1,1] \right\} \tag{26}$$

with indeterminate $x_{t_i} \in [-1,1]$.

Now we describe how to create polynomial zonotopes representing the set of trajectory parameters K. In this work, we choose $K = \bigtimes_{i=1}^{n_q} K_i$, where each K_j is the compact one-dimensional interval $K_j = [-1,1]$ and $\bigtimes_{i=1}^{n_q} K_i$ is the Cartesian product. We represent the interval K_j as a polynomial zonotope $\mathbf{K_j} = x_{k_j}$ where $x_{k_j} \in [-1,1]$ is an indeterminate.

B. Parameterized Trajectory and Forward Occupancy PZs

The parameterized position and velocity trajectories of the robot, defined in Def. 7, are functions of both time t and the trajectory parameter k. Using the time partition and trajectory parameter polynomial zonotopes described above, we create polynomial zonotopes $\mathbf{q_j}(\mathbf{T_i};\mathbf{K})$ that overapproximate $q_j(t;k)$ for all t in the i^{th} time subinterval and $k \in K$ by plugging the polynomial zonotopes $\mathbf{T_i}$ and $\mathbf{K}$ into the formula for $q_j(t;k)$.

Recall that $\mathbf{T_i}$ and $\mathbf{K_j}$ have indeterminates x_{t_i} and x_{k_j}, respectively. Because the desired trajectories only depend on t and k, the polynomial zonotopes $\mathbf{q_j}(\mathbf{T_i};\mathbf{K})$ and $\dot{\mathbf{q}}_\mathbf{j}(\mathbf{T_i};\mathbf{K})$ depend only on the indeterminates x_{t_i} and x_k. By plugging in a given k for x_k via the $\texttt{slice}$ operation, we obtain a polynomial zonotope where x_{t_i} is the only remaining indeterminate. Because we perform this particular slicing operation repeatedly throughout this document, if we are given a polynomial zonotope, $\mathbf{q_{d,j}}(\mathbf{T_i};\mathbf{K})$, we use the shorthand $\mathbf{q_j}(\mathbf{T_i};k) = \texttt{slice}(\mathbf{q_j}(\mathbf{T_i};\mathbf{K}), x_k, k)$. Importantly, one can apply [20, Lemma 17] to prove that the sliced representation is over approximative as we restate below:

Lemma 12 (Parmaeterized Trajectory PZs)**.** *The parameterized trajectory polynomial zonotopes $\mathbf{q_{d,j}}(\mathbf{T_i};\mathbf{K})$ are overapproximative, i.e., for each $j \in N_q$ and $k \in \mathbf{K}$,*

$$q_j(t;k) \in \mathbf{q_j}(\mathbf{T_i};k) \quad \forall t \in \mathbf{T_i} \tag{27}$$

One can similarly define $\dot{\mathbf{q}}_\mathbf{j}(\mathbf{T_i};\mathbf{K})$ that are also overapproximative.

Next, we describe how to use this lemma to construct an overapproximative representation to the forward occupancy. In particular, because the rotation matrices $R_j^{j-1}(q_j(t;k))$ depend on $\cos(q_j(t;k))$ and $\sin(q_j(t;k))$ one can compute $\cos(\mathbf{q_j}(\mathbf{T_i};\mathbf{K}))$ and $\sin(\mathbf{q_j}(\mathbf{T_i};\mathbf{K}))$ using Taylor expansions as in ([20], eq. (32)). By using this property and the fact that all operations involving polynomial zonotopes are either exact or overapproximative, the polynomial zonotope forward occupancy can be computed and proven to be overapproximative:

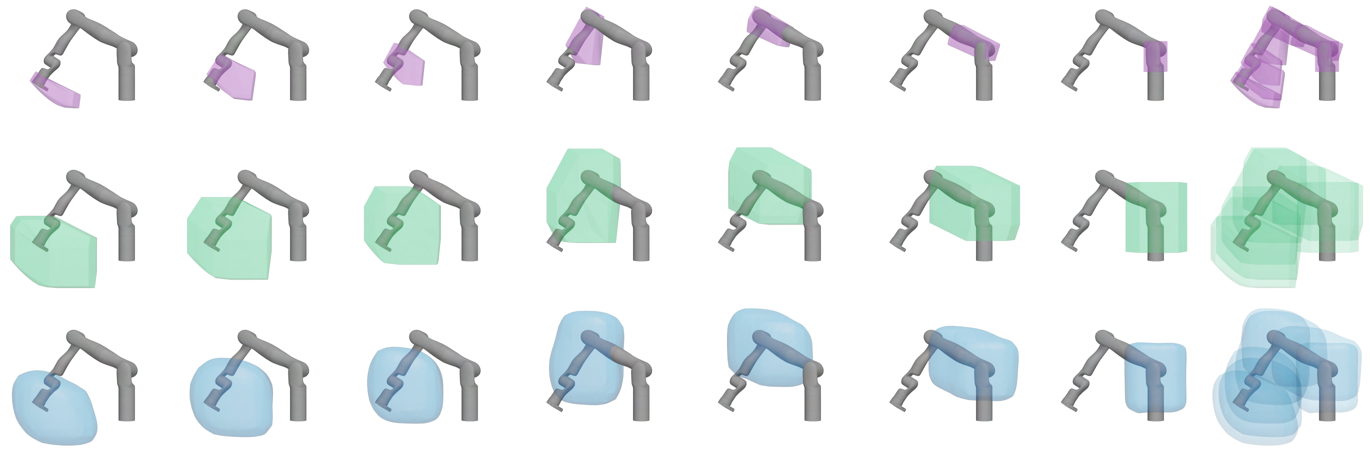
Fig. 3: 3D Reconstruction of RDF's zero-level sets compared to Alg. 1. (Top) Convex hull of each link's forward occupancy reach set (purple) before buffering by obstacle generators (Alg. 1, line 6). (Middle) Following buffering, the size of the new forward occupancy (green) is increased (Alg. 1, line 7). (Bottom) RDF's zero-level set (blue) is shown to be a smooth approximation to that of the convex hull buffered forward occupancy reach set (middle, green).

Lemma 13 (PZ Forward Occupancy)**.** *Let the polynomial zonotope forward occupancy reachable set for the j^{th} link at the i^{th} time step be defined as:*

$$\mathbf{FO_j}(\mathbf{q}(\mathbf{T_i};\mathbf{K})) = \mathbf{p_j}(\mathbf{q}(\mathbf{T_i};\mathbf{K})) \oplus \mathbf{R_j}(\mathbf{q}(\mathbf{T_i};\mathbf{K}))\mathbf{L_j}, \tag{28}$$

then for each $j \in N_q$, $k \in \mathbf{K}$, $\mathrm{FO}_j(q(t;k)) \in \mathbf{FO_j}(\mathbf{q}(\mathbf{T_i};k))$ for all $t \in \mathbf{T_i}$.

For convenience, let

$$\mathbf{FO}(\mathbf{q}(\mathbf{T_i};\mathbf{K})) = \bigcup_{j=1}^{n_q} \mathbf{FO_j}(\mathbf{q}(\mathbf{T_i};\mathbf{K})). \tag{29}$$

C. PZ-based Optimization Problem

Rather than solve the optimization problem described in (22) – (25), [20] uses these polynomial zonotope over approximations to solve the following optimization problem:

$$\min_{k \in K} \quad \mathtt{cost}(k) \tag{30}$$

$$\mathbf{q_j}(\mathbf{T_i};k) \subseteq [q^-_{j,\lim}, q^+_{j,\lim}] \qquad \forall j \in N_q, i \in N_t \tag{31}$$

$$\dot{\mathbf{q}}_\mathbf{j}(\mathbf{T_i};k) \subseteq [\dot{q}^-_{j,\lim}, \dot{q}^+_{j,\lim}] \qquad \forall j \in N_q, i \in N_t \tag{32}$$

$$r(O_\ell, \mathbf{FO}(\mathbf{q}(\mathbf{T_i};k))) > 0 \qquad \forall \ell \in N_{\mathscr{O}}, i \in N_t. \tag{33}$$

This formulation of the trajectory optimization problem has the benefit of being implementable without sacrificing any safety requirements. In fact, as shown in [20, Lemma 22], any feasible solution to this optimization problem can be applied to generate motion that is collision free. Though this method can be applied to 7 degree of freedom systems in real-time, applying this method to perform real-time planning for more complex systems is challenging as we show in Sec. VIII.

VI. Modeling RDF with Neural Networks

This section presents RDF, a neural implicit representation that can encode obstacle-avoidance constraints in continuous-time. In particular, RDF predicts the distance between obstacles and the entire *reachable set* of a robotic arm. To construct this neural implicit representation, we require training data. Unfortunately computing the exact distance to the reachable set of multi-link articulated robotic arm is intractable because that multi-link arm is a non-convex set. To build this training data, we rely on the polynomial zonotope-based representations presented in the previous section.

Importantly, we show that we can conservatively approximate the distance between an obstacle and sliced polynomial zonotope-based representation as the solution to a convex program. This allows us to efficiently generate the training data required to construct our neural implicit representation. Subsequently, we give an overview of the neural network architecture and loss function used for training. Finally, we describe how to reformulate the trajectory optimization problem using the neural network representation of the reachability-based signed distance function.

A. Derivation of RDF Approximation

This subsection derives an approximation to the reachability-based signed distance function defined in Def. 11. The core idea is to approximate the distance between an obstacle and the polynomial zonotope forward occupancy $\mathbf{FO}(\mathbf{q}(\mathbf{T_i};\mathbf{K}))$ (29) over of time for an entire trajectory. Note that slicing a polynomial zonotope of all of its *dependent* coefficients results in a zonotope [31]. This allows RDF to approximate both positive and negative (*i.e.* signed) distances by leveraging the zonotope arithmetic described in A-A. We now present the main theorem of the paper whose proof can be found in supplementary material Appendix A:

Theorem 14. *Suppose a robot is following a parameterized trajectory $q(t;k)$ for all $t \in T$. Consider an obstacle O with center c_O and generators G_O and $\mathbf{FO_j}(\mathbf{q}(\mathbf{T_i};k))$ with center c_F and generators G_F. Let $\mathcal{P}_j := \bigcup_{i=1}^{n_t} \mathbf{FO^{buf}_j}(\mathbf{q}(\mathbf{T_i};k))$ where $\mathbf{FO^{buf}_j}(\mathbf{q}(\mathbf{T_i};k)) = (c_F, G_O \cup G_F)$. Define the function $\tilde{r}_j$ as follows:*

$$\tilde{r}_j(c_O, \mathcal{P}_j) = \begin{cases} d(c_O, \partial \mathcal{P}_j) & \textit{if } c_O \notin \mathcal{P} \\ -d(c_O, \partial \mathcal{P}_j) & \textit{otherwise}, \end{cases} \tag{34}$$

and define the function $\tilde{r}$ as follows:

$$\tilde{r}(c_O, \cup_{j\in N_q}\mathcal{P}_j) = \min_{j\in N_q} \tilde{r}_j(c_O, \mathcal{P}_j). \tag{35}$$

If $\mathbf{FO_j}(\mathbf{q}(\mathbf{T_i};k)) \cap O \neq \emptyset$, then $\tilde{r}(c_O, \mathcal{P}_j) \geq r(O, \mathrm{FO}(q(T;k)))$. If $\mathbf{FO_j}(\mathbf{q}(\mathbf{T_i};k)) \cap O = \emptyset$, then $\tilde{r}(c_O, \mathcal{P}_j) \leq r(O, \mathrm{FO}(q(T;k)))$.

This theorem is useful because it allows us to conservatively approximate r using $\tilde{r}$ which computes the distance between a point and a convex set. As we show next, this distance computation can be done by solving a convex program. Recall that a convex hull can be represented as the intersection of a finite number of half planes, *i.e.,*

$$\mathcal{P}_j = \bigcap_{h\in N_{h,j}} \mathcal{H}_j^{(h)} \tag{36}$$

where $N_{h,j} = \{1, \cdots, n_{h,j}\}$ and $n_{h,j}$ is the number of half-spaces $\mathcal{H}_j^{(h)}$ [37, Ch. 1]. As a result, one can determine whether a point $p \in \mathbb{R}^{n_d}$, is in side of $\mathcal{P}_j$ using the following property [37, Ch. 1]:

$$p \in \mathcal{P}_j \Longleftrightarrow A_j^{(h)} p - b_j^{(h)} \leq 0 \quad \forall i = N_{h,j}, \tag{37}$$

where $A_j^{(h)} \in \mathbb{R}^{n_d}, b_j^{(h)} \in \mathbb{R}$ represents each half-space, $\mathcal{H}_j^{(h)}$. As a result, one can compute the distance in (35) as:

$$d(c_O, \partial\mathcal{P}_j) = \min_p \quad ||p - c_O|| \tag{38}$$

$$A_j^{(h)} p - b_j^{(h)} \leq 0 \qquad \forall h \in N_{h,j}, \tag{39}$$

where depending upon the norm chosen in the cost function one can solve the optimization problem using a linear or a convex quadratic program. In the instance that there is non-trivial intersection between an obstacle and $\mathcal{P}_j$, one can apply the Euclidean projection [38, p.398] to directly calculate the distance between c_O and every half-space $\mathcal{H}_j^{(h)}$ supporting $\mathcal{P}_j$:

$$d(c_O, \partial\mathcal{P}_j) = \min_{h\in N_{h,j}} d(c_O, \mathcal{H}_j^{(h)}) \tag{40}$$

$$= -\max_{h\in N_{h,j}} \frac{A_j^{(h)} c_O - b_j^{(h)}}{||A_j^{(h)}||}. \tag{41}$$

B. Model Architecture

We follow the RDF model design of [3] and use an Multi-Layer Pereceptron (MLP) with 8 hidden layers and a jump connection in the middle between the 4th and 5th hidden layers. The network takes as input $x = (q_0, \dot{q}_0, k, c_o) \in \mathbb{R}^{3n_q+n_d}$, which consists of a concatenation of vectors corresponding to the initial joint positions q_0, initial joint velocities $\dot{q}_0$, trajectory parameters k, and obstacle center c_O. As described in Section III-D each $q_0, \dot{q}_0, k$ corresponds to particular desired trajectory and reachable set $\mathrm{FO} \subset \mathbb{R}^{n_d}$ that may or may not be in collision. The network outputs an approximation $\hat{y} = (\hat{r}_1, \hat{r}_2, \cdots, \hat{r}_{n_q})$ of the reachability-based signed distance between the obstacle center c_O and forward occupancy of each link FO_j.

Algorithm 1: $\texttt{RDF}(q_0, \dot{q}_0, k, O)$

```
1:  $\{\mathbf{q}(\mathbf{T_i};\mathbf{K}) : i \in N_t\} \leftarrow (q_0, \dot{q}_0)$
2:  $c_O, G_O \leftarrow O$
3:  for $i = 1 : n_t$
4:      for $j = 1 : n_q$
5:          $\mathbf{FO_j}(\mathbf{q}(\mathbf{T_i};\mathbf{K})) \leftarrow \texttt{PZFO}(\mathbf{q}(\mathbf{T_i};\mathbf{K}))$
6:          $\mathbf{FO_j}(\mathbf{q}(\mathbf{T_i};k)) \leftarrow \texttt{slice}(\mathbf{FO_j}(\mathbf{q}(\mathbf{T_i};\mathbf{K})), k)$
7:          $\mathbf{FO_j^{buf}}(\mathbf{q}(\mathbf{T_i};k)) \leftarrow \mathbf{FO_j}(\mathbf{q}(\mathbf{T_i};k)) \oplus Z_O^g$
8:      end for
9:  end for
10: for $j = 1 : n_q$
11:     $\mathcal{P}_j \leftarrow \texttt{conv}(\bigcup_{i=1}^{n_t} \mathbf{FO_j^{buf}}(\mathbf{q}(\mathbf{T_i};k)))$
12:     if $c_O \in \mathcal{P}_j$ then
13:         $\tilde{r}_j \leftarrow -d(c_O, \partial\mathcal{P}_j)$ // negative distance
14:     else
15:         $\tilde{r}_j \leftarrow d(c_O, \partial\mathcal{P}_j)$ // positive distance
16:     end if
17: end for
18: Return $\{\tilde{r}_j : j \in N_q\}$
```

C. Loss Function

This section describes the loss function used to train the RDF model. We apply a mean square error loss added to an Eikonal loss term similar to [3]. The mean square error loss forces the network to learn to predict the distance while the Eikonal loss regularizes the gradient of the RDF prediction. Given an input, ground-truth RDF distance pair $x = (q, \dot{q}, k, c_o)$, $y = (\tilde{r}_1, \tilde{r}_2, \cdots, \tilde{r}_{n_q})$ from a batched dataset sample (X_{batch}, Y_{batch}), our network, parameterized by its weights θ, computes the output batch $\hat{Y}_{batch} = \{\hat{y} | \hat{y} = f_\theta(x), x \in X_{batch}\}$ and results in the loss:

$$\mathcal{L} = \mathcal{L}_{MSE} + \alpha \cdot \mathcal{L}_{Eikonal} \tag{42}$$

where

$$\mathcal{L}_{MSE} = \frac{1}{|\hat{Y}|} \sum_{\hat{y}\in\hat{Y}} \left(\frac{1}{n_q} \sum_{i=1}^{nq} (\hat{r}_i - \tilde{r}_i)^2\right) \tag{43}$$

$$\mathcal{L}_{Eikonal} = \frac{1}{|\hat{Y}|} \sum_{\hat{y}\in\hat{Y}} \left(\frac{1}{n_q} \sum_{i=1}^{n_q} (\|\nabla_{c_O} \hat{r}_i\| - 1)^2\right), \tag{44}$$

while α is a hyperparameter that denotes the coefficient of Eikonal loss $\mathcal{L}_{Eikonal}$ used in the total loss $\mathcal{L}$.

D. RDF-based Trajectory Optimization

After training, we generate a model $\tilde{r}_{NN|\theta}$ that takes in $(q_0, \dot{q}_0, k, c_{O,\ell})$ and predicts the reachability based distance between the obstacle and the robot's forward occupancy. Using this representation, we can reformulate the motion planning optimization problem described by (30)–(33) into:

$$\min_{k\in K} \quad \texttt{cost}(k) \tag{45}$$

$$\mathbf{q_j}(\mathbf{T_i};k) \subseteq [q_{j,\lim}^-, q_{j,\lim}^+] \qquad \forall j \in N_q, i \in N_t \tag{46}$$

$$\mathbf{\dot{q}_j}(\mathbf{T_i};k) \subseteq [\dot{q}_{j,\lim}^-, \dot{q}_{j,\lim}^+] \qquad \forall j \in N_q, t \in N_t \tag{47}$$

$$\tilde{r}_{NN|\theta}(q_0, \dot{q}_0, k, c_{O,\ell}) > \delta \qquad \forall \ell \in N_{\mathscr{O}} \tag{48}$$

where δ is a buffer threshold of the RDF collision-avoidance constraint, equation (48). Note in particular that one computes the gradients of the last constraint by performing back-propagation through the neural network representation. The gradient of the first two constraints can be computed by applying [20, Section IX.D].

VII. RDF Experimental Setup

This section describes our experimental setup including simulation environments, how the training and test sets were generated, and how the network hyperparameters were selected.

A. Implementation Details

We use Gurobi's quadratic programming solver [39] to construct the ground truth reachability-based positive distance in Alg. 1. The RDF model is built and trained with Pytorch [40]. In the motion planning experiment, we ran trajectory optimization with IPOPT [41]. A computer with 12 Intel(R) Core(TM) i7-6800K CPU @ 3.40GHz and an NVIDIA RTX A6000 GPU was used for experiments in Sec. VIII-A and VIII-B. A computer with 12 Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz and an NVIDIA RTX A6000 GPU was used for the motion planning experiment in Sec. VIII-C.

B. Simulation and Simulation Environment

Each simulation environment has dimensions characterized by the closed interval $[-1,1]^{n_d}$ and the base of the robot arm is located at the origin. For each 2D environment, every link of the robotic arm is of same size and is adjusted according to the number of links n_q to fit into the space. We consider planar robot arms with 2, 4, 6, 8, and 10 links, respectively. In each environment all obstacles are static, axis-aligned, and fixed-size where each side has length $\frac{0.2}{1.2n_q}$. For example, the 2D 6-DOF arm has a link length of 0.139m while obstacles are squares with side-length 0.028m. In 3D we use the Kinova Gen3 7-DOF serial manipulator [42]. The volume of each Kinova's link is represented as the smallest bounding box enclosing the native link geometry.

We also ensure that each robot's initial configuration is feasible and does not exceed its position limits. Each planning trial is considered a success if the l2-distance between the arm's configuration and goal configuration is within 0.1 rad. A planning trial is considered a failure if the robot arms collides with an obstacle, if the trajectory planner fails to find a feasible trajectory in two successive steps, or if the robot does not reach the goal within 400 planning steps.

C. Desired Trajectory

We parameterize our trajectory with piece-wise linear velocity. We design *trajectory parameter* $k=(k_1,\cdots,k_{n_q})\in\mathbb{R}^{n_q}$ as a constant acceleration over $[0,t_p)$. Then, the rest of the trajectory takes a constant braking acceleration over $[t_p,t_\text{f}]$

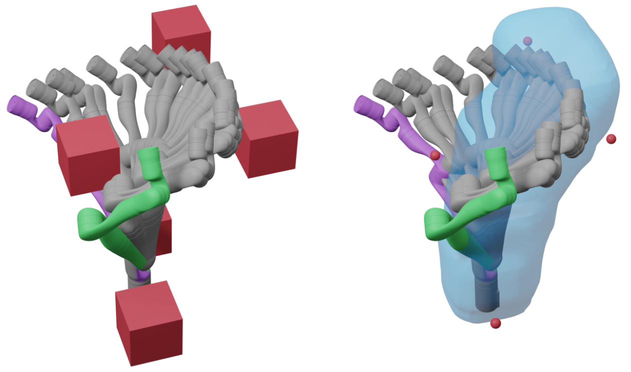

Fig. 4: Real-time receding-horizon trajectory planning with RDF in a cluttered environment. (Left) The arm safely moves from the start pose (purple) through intermediate configurations (grey) to reach the goal (green) while avoiding obstacles (red cubes). (Right) During the highlighted planning iteration all obstacle centers (red spheres) remain outside RDF's zero-level set (blue).

to reach stationary at t_f. Given an initial velocity $\dot{q}_0$, the parameterized trajectory is given by

$$\dot{q}(t;k)=\begin{cases}\dot{q}_0+kt, & t\in[0,t_p)\\ \frac{\dot{q}_0+kt_p}{t_\text{f}-t_p}(t_\text{f}-t), & t\in[t_p,t_\text{f}].\end{cases} \tag{49}$$

D. Dataset

We compute the dataset for RDF by randomly sampling data points consisting of the initial joint position q_0, initial joint velocity $\dot{q}_0$, and trajectory parameter k. For each initial condition, we then randomly sample $n_o=16$ obstacle center positions in $[-1,1]^{n_d}$ and compute the ground truth distances between the forward reachable set of the robot and obstacles using Alg. 1. The input to the network is then $x=(q_0,\dot{q}_0,k,c_O)$ where $(q_0,\dot{q}_0,k)$ specifies the desired trajectory and the reachable set and c_O defines position of the center of the obstacle. The corresponding label is $y=(\tilde{r}_1,\tilde{r}_2,\cdots,\tilde{r}_{n_q})$ where $\tilde{r}_j$ is the the approximation of reachability-based signed distance to each link outlined in Alg. 1. For the 2D tasks, the datasets consist of 2.56 million samples while the 3D datasets consist of 5.12 million sample. 80% of samples in each case are used for training and 20% are used for validation. Another set of the same size as the validation set is generated for testing.

E. Network Hyperparameters

We train models using all combinations of the following hyperparameters: learning rates $lr=(0.001,0.0001)$, Eikonal loss coefficients $\alpha=(0.0,0.001,0.0001)$, and $\beta=(0.9,0.999)$ and weight decay $\gamma=0.01$ for the Adam optimizer. We train the 2D and 3D models for 300 and 350 epochs, respectively. The model that performs best on the validation set is chosen for further evaluation.

VIII. Results

This section evaluates the performance of the trained RDF network in terms of its accuracy, inference time, the time required to compute its gradient, and its ability to safely solve motion planning tasks. We compare RDF's safety and success rate on motion planning tasks to ARMTD [1], CHOMP [17], and the method presented in [28].

Env. Dim.	DOF	Mean Error (cm) ↓
2	2	0.16 ± 0.15
	4	0.26 ± 0.30
	6	0.37 ± 0.36
	8	0.39 ± 0.48
	10	0.51 ± 0.52
3	7	0.45 ± 0.48

TABLE II: Mean $l1$-norm error of each RDF model evaluated on test set

A. RDF Accuracy and Runtime Compared to Alg. 1

This section compares RDF's distance prediction accuracy to the distances computed by the Alg. 1. We perform these comparisons for 2D planar multi-link robot arms and the Kinova Gen3 7DOF arm on the test sets that were not used to either train or validate RDF. As shown in Table II, each model has a mean prediction error of < 1cm in the $l1$-norm. These results are supported by Fig. 3, which shows RDF's zero-level sets are smooth approximations to Alg.1.

We then compared the mean runtime of RDF's inference and gradient computations to the computation time of Alg. 1 and its first-order numerical gradient. These comparisons are done over a random sample of 1000 feasible data points $(q_0, \dot{q}_0, k, c_o)$. As shown in Table III, RDF computes both distances and gradients at least an order of magnitude faster than Alg. 1. This result holds even when considering only the time required to solve the quadratic program in Alg. 1. Note also that RDF's runtime appears to grow linearly with the DOF of the system, while Alg. 1's grows quadratically.

B. Accuracy & Runtime Comparison with SDF

We compared RDF's distance prediction accuracy and runtime to that of an SDF-based model similar to [28] over an entire trajectory in 3D. To train a discrete-time, SDF-based model similar to that presented in [28], we generated a dataset of 5.12 million examples. Each input to this SDF takes the form $x = (q_d(t;k), c_O)$ and the corresponding label is $y = (\tilde{s}_1, \tilde{s}_2, \cdots, \tilde{s}_{n_q})$, where $\tilde{s}_j$ is the distance between c_O and a polynomial zonotope overapproximation of j^{th} link of the robot. Note that, in principle, this is equivalent to evaluating RDF at stationary configurations by specifying $\dot{q}_0 = 0$ and $k = 0$. Following [28], we also ensure that the number of collision and non-collision samples are balanced for each link.

For RDF, we generated 1000 samples where the i^{th} sample is of the form $(q_0, \dot{q}_0, k, c_O)_i$. Because SDF is a discrete-time model, its corresponding i^{th} sample is the minimum distance between the obstacle and a set of robot configurations $\{q_d(t_n;k) : n \in N_t\}$ sampled at timepoints t_n evenly separated by a given Δt. Note that for SDF, we considered multiple time discretizations ($\Delta t = 0.01s, 0.02, 0.1s, 0.5s, 1.0s$). During the implementation of SDF, we allow the forward pass through the network to be batched and evaluate all time steps for a given discretization size, simultaneously. As shown in Table IV, RDF has lower mean and max $l1$-norm error compared to SDF. Similarly, RDF has a lower run time than SDF across all time discretizations.

C. Receding Horizon Motion Planning

This subsection describes the application of RDF to real-time motion planning and compares its performance to several state of the art algorithms. We evaluate each method's performance on a reaching task where the robot arm is required to move from an initial configuration to a goal configuration while avoiding collision with obstacles and satisfying joint limits. Note that the planner is allowed to perform receding-horizon trajectory planning. We evaluate each planner's success rate, collision rate, and mean planning times under various planning time limits. If the planner was unable to find a safe solution, the arm will execute the fail-safe maneuver from the previous plan.

1) 2D Results: In 2D, we compare the performance of RDF to ARMTD [1] across a variety of different arms with varying degrees of freedom from 2-10 DOF to better understand the scalability of each approach. In each instance, the robot is tasked with avoiding 2 obstacles and is evaluated over 500 trials. In the interest of simplicity, we select δ in (48) to be 3cm to 3.5cm which is approximately 10 times larger than the mean RDF error prediction as described in Tab. II. Because our goal is to develop a planning algorithm that can operate in real-time, we also evaluate the performance of these algorithms when each planning iteration is restricted to compute a solution within $5, 0.3$, and 0.033s. Note that each planning algorithm can only be applied for 400 planning iterations per trial.

Tables V and VI summarize the results. Across all experiments, when both algorithms are given 5s per planning iteration, ARMTD was always able to arrive at the goal more frequently than RDF. A similar pattern seems to hold as the number of degrees of freedom increase when each algorithm is given 0.3s per planning iteration; however, in the 10 DOF case ARMTD's success rate drastically decreases while RDF's performance is mostly unaffected. This is because the computation time of ARMTD grows dramatically as the number of DOFs increases as depicted in Table VII. This observation is more pronounced in the instance where both planning algorithms are only allowed to take 0.033s per planning iteration. In that instance RDF's performance is unaffected as the number of DOFs increases while ARMTD is unable to succeed beyond the 2DOF case. Note across all experiments, none of the computed trajectories ended in collision.

2) 3D Results: In 3D, we compare the performance of RDF to ARMTD and an SDF-based version of the obstacle-avoidance constraints within a receding-horizon trajectory planning framework. Note that for RDF and SDF, we buffer their distance predictions with buffer size 3cm, which is approximately five times larger than the mean prediction error in Tab. IV. We also compare the performance of each of the aforementioned methods in 3D to CHOMP [17]. Because our goal is to develop a planning algorithm that can operate in real-time, we also evaluate the performance of these algorithms when each planning iteration is restricted to compute a solution within $5, 0.3$, 0.033, and 0.025s. We also consider the case of

Env. Dim.	DOF	Alg.1 Distance	Alg.1 Gradient	QP Distance	QP Gradient	RDF Distance	RDF Gradient
2	2	0.08 ± 0.01	0.16 ± 0.01	0.009 ± 0.006	0.015 ± 0.005	**0.0008 ± 0.00003**	**0.003 ± 0.0003**
	4	0.18 ± 0.01	0.70 ± 0.03	0.020 ± 0.011	0.065 ± 0.021	**0.0009 ± 0.00004**	**0.005 ± 0.0008**
	6	0.28 ± 0.01	1.65 ± 0.05	0.028 ± 0.012	0.132 ± 0.033	**0.0009 ± 0.00004**	**0.006 ± 0.0011**
	8	0.40 ± 0.02	3.19 ± 0.12	0.033 ± 0.015	0.238 ± 0.065	**0.0010 ± 0.00005**	**0.007 ± 0.0016**
	10	0.85 ± 0.02	8.53 ± 0.11	0.030 ± 0.008	0.300 ± 0.044	**0.0010 ± 0.00018**	**0.008 ± 0.0019**
3	7	1.59 ± 0.11	11.1 ± 0.80	0.251 ± 0.117	1.755 ± 0.807	**0.0011 ± 0.00013**	**0.006 ± 0.0015**

TABLE III: Mean Runtime (s) of Distance and Gradient Computation ↓

Method	Mean Error (cm) ↓	Max Error (cm) ↓	Runtime (ms) ↓
RDF	**0.55 ± 0.71**	**11**	**0.75 ± 0.02**
SDF ($\Delta t = 1.0s$)	7.16 ± 11.5	88	1.17 ± 0.03
SDF ($\Delta t = 0.5s$)	0.96 ± 1.57	38	1.18 ± 0.12
SDF ($\Delta t = 0.1s$)	0.79 ± 0.93	17	1.15 ± 0.03
SDF ($\Delta t = 0.02s$)	0.80 ± 0.93	17	1.11 ± 0.03
SDF ($\Delta t = 0.01s$)	0.80 ± 0.93	17	1.12 ± 0.06

TABLE IV: RDF vs. SDF Distance Predictions

ARMTD			
DOF	Time Limit		
	5.0	0.3	0.033
2	**354**	**354**	**338**
4	**354**	**354**	0
6	**358**	**356**	0
8	**382**	**372**	0
10	**380**	6	0

TABLE V: # of successes for ARMTD in 2D planning with 5.0s, 0.3s, and 0.033s time limit ↑

RDF			
DOF	Time Limit		
	5.0	0.3	0.033
2	253	253	253
4	246	245	**243**
6	239	239	**233**
8	239	239	**235**
10	246	**242**	**233**

TABLE VI: # of successes for RDF in 2D planning with 5.0s, 0.3s, and 0.033s time limit ↑

DOF	RDF	ARMTD
2	**0.022 ± 0.241**	0.034 ± 0.127
4	**0.023 ± 0.246**	0.071 ± 0.191
6	**0.037 ± 0.310**	0.108 ± 0.156
8	**0.023 ± 0.231**	0.182 ± 0.166
10	**0.030 ± 0.272**	0.417 ± 0.286

TABLE VII: Mean time for each planning step for RDF and ARMTD in 2D planning experiments with 5.0s time limit ↓

# Obstacles	5			
time limit (s)	5.0	0.3	0.033	0.025
RDF	401	398	384	377
ARMTD	**487**	**479**	0	0
SDF	421	420	339	304
CHOMP	401	392	**386**	**386**

TABLE VIII: # of successes for RDF, ARMTD, SDF, and CHOMP in 3D 7 link planning experiment with 5 obstacles ↑

avoiding 5 obstacles and 10 obstacles. Each obstacle case was evaluated over 500 trials. Note that each planning algorithm can only be applied for 400 planning iterations per trial.

Tables VIII and IX summarize the results of the performance of each algorithm across different time limits for the 5 and 10 obstacles cases, respectively. First observe that as the number of obstacles increases each algorithms performance decreases. Note in particular that for a fixed number of obstacles, the ability of each method to reach the goal decreases as the time limit on planning decreases. Though ARMTD initially performs the best, when the time limit is drastically reduced, RDF begins to perform better. This is because the computation time of ARMTD grows dramatically as the number of DOFs increases as depicted in Table XII. The transition from when ARMTD performs best to when RDF performs best occurs when the time limit is restricted to 0.033s. Note that RDF, ARMTD, and SDF are collision free across all tested trials. However CHOMP has collisions in every instance where it is unable to reach the goal. An example of RDF successfully planning around 5 obstacles is shown in Fig. 4.

IX. Conclusion

This paper introduces the Reachability-based signed Distance Function (RDF), a neural implicit representation useful for safe robot motion planning. We demonstrate RDF's viability as a collision-avoidance constraint within a real-time receding-horizon trajectory planning framework. We show that RDF's distance computation is fast, accurate, and unlike model-based methods like ARMTD, scales linearly with the dimension of the system. RDF is also able to solve challenging motion planning tasks for high DOF robotic arms under limited planning horizons.

We note that the current work has two main limitations. First, RDF lacks theoretical safety guarantees. Second, RDF does not generalize to different morpologies and thus requires training a new model for every robot. We aim to address these shortcomings in future work.

References

[1] P. Holmes, S. Kousik, B. Zhang, *et al.*, " Reachable Sets for Safe, Real-Time Manipulator Trajectory Design," in *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, Jul. 2020.

[2] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, *DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation*, arXiv:1901.05103 [cs], Jan. 2019.

[3] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit geometric regularization for learning shapes," in *Proceedings of the 37th International Conference on Machine Learning*, 2020, pp. 3789–3799.

[4] *Learning Smooth Neural Functions via Lipschitz Regularization - Google Search.*

[5] D. Blackmore and M. Leu, "A differential equation approach to swept volumes," in *[1990] Proceedings. Rensselaer's Second International Conference on Computer Integrated Manufacturing*, May 1990, pp. 143–149.

[6] D. Blackmore and M. Leu, "Analysis of Swept Volume via Lie Groups and Differential Equations," en, *The International Journal of Robotics Research*, vol. 11, no. 6, pp. 516–537, Dec. 1992, Publisher: SAGE Publications Ltd STM.

[7] D. Blackmore, M. C. Leu, and F. Shih, "Analysis and modelling of deformed swept volumes," en, *Computer-Aided Design*, Special Issue: Mathematical methods for CAD, vol. 26, no. 4, pp. 315–326, Apr. 1994.

# Obstacles	10			
time limit (s)	5.0	0.3	0.033	0.025
RDF	321	319	**306**	**293**
ARMTD	**466**	301	0	0
SDF	357	**352**	215	15
CHOMP	312	301	293	**293**

TABLE IX: # of successes for RDF, ARMTD, SDF, and CHOMP in 3D 7 Link planning experiment with 10 obstacles ↑

# Obstacles	5			
time limit (s)	5.0	0.3	0.033	0.025
RDF	**0**	**0**	**0**	**0**
ARMTD	**0**	**0**	**0**	**0**
SDF	**0**	**0**	**0**	**0**
CHOMP	99	108	114	114

TABLE X: # of collisions for RDF, ARMTD, SDF, and CHOMP in 3D 7 link planning experiment with 5 obstacles ↓

# Obstacles	10			
time limit (s)	5.0	0.3	0.033	0.025
RDF	**0**	**0**	**0**	**0**
ARMTD	**0**	**0**	**0**	**0**
SDF	**0**	**0**	**0**	**0**
CHOMP	188	199	207	207

TABLE XI: # of collisions for RDF, ARMTD, SDF, and CHOMP in 3D 7 link planning experiment with 10 obstacles ↓

# Obstacles	5	10
RDF	**0.023 ± 0.183**	**0.037 ± 0.296**
ARMTD	0.172 ± 0.131	0.289 ± 0.358
SDF	0.038 ± 0.212	0.064 ± 0.354
CHOMP	0.086 ± 0.138	0.078 ± 0.211

TABLE XII: Mean time for each planning step for RDF, ARMTD, SDF, and CHOMP in the 3D 7 link planning experiment with 5.0s time limit ↓

[8] D. Blackmore, M. Leu, and L. P. Wang, "The sweep-envelope differential equation algorithm and its application to NC machining verification," en, *Computer-Aided Design*, vol. 29, no. 9, pp. 629–637, Sep. 1997.

[9] D. Blackmore, R. Samulyak, and M. C. Leu, "Trimming swept volumes," en, *Computer-Aided Design*, vol. 31, no. 3, pp. 215–223, Mar. 1999.

[10] K. Abdel-Malek, J. Yang, D. Blackmore, and K. Joy, "Swept volumes: Fundation, perspectives, and applications," *International Journal of Shape Modeling*, vol. 12, no. 01, pp. 87–127, Jun. 2006, Publisher: World Scientific Publishing Co.

[11] Lozano-Perez, "Spatial planning: A configuration space approach," *IEEE Transactions on Computers*, vol. C-32, no. 2, pp. 108–120, 1983.

[12] M. Campen and L. P. Kobbelt, "Polygonal boundary evaluation of minkowski sums and swept volumes," *Computer Graphics Forum*, vol. 29, 2010.

[13] Y. J. Kim, G. Varadhan, M. C. Lin, and D. Manocha, "Fast swept volume approximation of complex polyhedral models," *Comput. Aided Des.*, vol. 36, pp. 1013–1027, 2003.

[14] A. Gaschler, R. P. A. Petrick, T. Kröger, O. Khatib, and A. Knoll, "Robot task and motion planning with sets of convex polyhedra," in *Robotics: Science and Systems Conference*, 2013.

[15] N. Perrin, O. Stasse, L. Baudouin, F. Lamiraux, and E. Yoshida, "Fast humanoid robot collision-free footstep planning using swept volume approximations," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 427–439, 2012.

[16] S. Murray, W. Floyd-Jones, Y. Qi, D. J. Sorin, and G. D. Konidaris, "Robot motion planning on a chip," in *Robotics: Science and Systems*, 2016.

[17] M. Zucker, N. Ratliff, A. D. Dragan, *et al.*, "Chomp: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[18] J. Schulman, Y. Duan, J. Ho, *et al.*, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[19] 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.

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

[21] M. Adamkiewicz, T. Chen, A. Caccavale, *et al.*, "Vision-only robot navigation in a neural radiance world," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4606–4613, 2022.

[22] J. Huh, D. D. Lee, and V. Isler, "Neural cost-to-go function representation for high dimensional motion planning,"

[23] J. Ortiz, A. Clegg, J. Dong, *et al.*, *iSDF: Real-Time Neural Signed Distance Fields for Robot Perception*, arXiv:2204.02296 [cs], May 2022.

[24] G. S. Camps, R. Dyro, M. Pavone, and M. Schwager, "Learning deep sdf maps online for robot navigation and exploration," *arXiv preprint arXiv:2207.10782*, 2022.

[25] J. Ichnowski, Y. Avigal, J. Kerr, and K. Goldberg, "Dex-nerf: Using a neural radiance field to grasp transparent objects," *arXiv preprint arXiv:2110.14217*, 2021.

[26] D. Driess, Z. Huang, Y. Li, R. Tedrake, and M. Toussaint, "Learning multi-object dynamics with compositional neural radiance fields," *arXiv preprint arXiv:2202.11855*, 2022.

[27] L. Yen-Chen, P. Florence, J. T. Barron, T.-Y. Lin, A. Rodriguez, and P. Isola, "Nerf-supervision: Learning dense object descriptors from neural radiance fields," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 6496–6503.

[28] M. Koptev, N. B. Figueroa Fernandez, and A. Billard, "Implicit distance functions: Learning and applications in control," in *International Conference on Robotics and Automation (ICRA 2022)*, 2022.

[29] 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.

[30] 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.

[31] N. Kochdumper and M. Althoff, "Sparse polynomial zonotopes: A novel set representation for reachability analysis," *IEEE Transactions on Automatic Control*, vol. 66, no. 9, pp. 4043–4058, 2020.

[32] M. Spong, S. Hutchinson, and M. Vidyasagar, "Robot modeling and control," 2005.

[33] L. J. Guibas, A. Nguyen, and L. Zhang, "Zonotopes as bounding volumes," in *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics, 2003, pp. 803–812.

[34] 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.

[35] C. Dapogny and P. Frey, "Computation of the signed distance function to a discrete contour on adapted triangulation," *Calcolo*, vol. 49, pp. 193–219, 2012.

[36] L. C. Evans, *Partial differential equations*. American Mathematical Society, 2022, vol. 19.

[37] G. M. Ziegler, *Lectures on polytopes*. New York: Springer-Verlag, 1995.

[38] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[39] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, 2023.

[40] A. Paszke, S. Gross, S. Chintala, *et al.*, “Automatic differentiation in pytorch,” 2017.

[41] A. Wächter and L. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,” *Mathematical programming*, vol. 106, pp. 25–57, Mar. 2006.

[42] Kinova, *User Guide - KINOVA Gen3 Ultra lightweight robot*. 2022.

[43] S. Cameron and R. Culley, “Determining the minimum translational distance between two convex polyhedra,” in *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, vol. 3, 1986, pp. 591–596.

[44] P. K. Agarwal, L. J. Guibas, S. Har-Peled, A. Rabinovitch, and M. Sharir, “Computing the penetration depth of two convex polytopes in 3d,” in *Algorithm Theory - SWAT 2000*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 328–338.

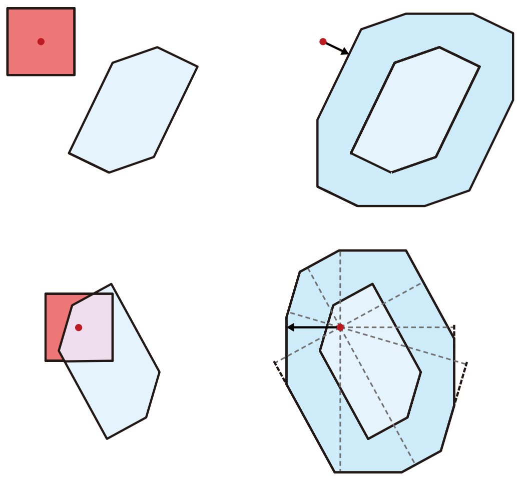
Fig. 5: Illustration of signed distance for convex polytopes. In particular, this works well with zonotopes.

Appendix A
Proof of Theorem 14

Before proving the required result, we prove several intermediate results.

A. Signed Distance Between Zonotopes

Here we derive the signed distance between two zonotopes. To proceed we first require a condition for determining if two zonotopes are in collision:

Lemma 15. *Let $Z_1 = (c_1, G_1)$ and $Z_2 = (c_2, G_2)$ be zonotopes. Then $Z_1 \cap Z_2 \neq \emptyset$ if and only if $c_1 \in Z_{2,buf} = (c_2, G_1 \cup G_2)$.*

Proof. Suppose the intersection of Z_1 and Z_2 is non-empty. This is equivalent to the existence of $z_1 \in Z_1$ and $z_2 \in Z_2$ such that $z_1 = z_2$. Furthermore, by definition of a zonotope (Defn. 1), this is equivalent to the existence of coefficients β_1 and β_2 such that $z_1 = c_1 + G_1\beta_1$, $z_2 = c_2 + G_2\beta_2$, and

$$c_1 + G_1\beta_1 = c_2 + G_2\beta_2 \tag{50}$$
$$\iff c_1 = c_2 + G_2\beta_2 - G_1\beta_1 \tag{51}$$
$$\iff c_1 \in (c_1, G_1 \cup G_2) = Z_{2,\text{buf}}. \tag{52}$$

Thus we have shown that $Z_1 \cap Z_2 \neq \emptyset$ if and only if $c_1 \in (c_1, G_1 \cup G_2) = Z_{2,\text{buf}}$. □

Lemma 15 can then be used to compute the the positive distance between two non-intersecting zonotopes:

Lemma 16. *Let $Z_1 = (c_1, G_1)$ and $Z_2 = (c_2, G_2)$ be non-intersecting zonotopes.*

$$d(Z_1, Z_2) = d(c_1; \partial Z_{2,buf}) \tag{53}$$

where $Z_{2,buf} = (c_2, G_1 \cup G_2)$.

Proof. Suppose Z_1 and Z_2 are non-intersecting zonotopes such that $d(Z_1, Z_2) = r$ for some $r \in \mathbb{R}^+$. That is, there exists $z_1 \in Z_1$ and $z_2 \in Z_2$ such that

$$r = \|z_1 - z_2\| \tag{54}$$
$$= \|c_1 + G_1\beta_1 - (c_2 + G_2\beta_2)\| \tag{55}$$
$$= \|c_1 - (c_2 + G_2\beta_2 - G_1\beta_1)\| \tag{56}$$
$$\geq d(c_1, (c_2, G_1 \cup G_2)) \tag{57}$$
$$= d(c_1, \partial Z_{2,\text{buf}}). \tag{58}$$

Now suppose $d(c_1, Z_{2,\text{buf}}) = r$. Then there exists coefficients β_1 and β_2 such that

$$r = \|c_1 - (c_2 + G_1\beta_1 + G_2\beta_2)\| \tag{59}$$
$$= \|c_1 - G_1\beta_1 - (c_2 + G_2\beta_2)\| \tag{60}$$
$$= \|z_1 - z_2\| \tag{61}$$
$$\geq d(Z_1, Z_2). \tag{62}$$

Together, these inequalities prove that $d(Z_1, Z_2) = d(c_1; Z_{2,buf})$. □

Let $\text{int}(S)$ denote the interior of a set. Before deriving the negative distance we first define the penetration depth [43] of two zonotopes:

Definition 17. *The* penetration depth *of Z_1 and Z_2 in $\mathbb{R}^{n_d}$ is denoted $\pi(Z_1, Z_2)$ and is defined by*

$$\pi(Z_1, Z_2) = \min\{\|t\| : \; int(Z_1 + t \cap Z_2) = \emptyset, \quad t \in \mathbb{R}^{n_d}\}. \tag{63}$$

The penetration depth $\pi(Z_1, Z_2)$ can be interpreted as the minimum translation applied to Z_1 such that the interior of $Z_1 \cap Z_2$ is empty. Following [44] we can now redefine the penetration distance between $Z_1 = (c_1, G_1)$ and $Z_2 = (c_2, G_2)$ in terms of c_1 and $Z_{2,\text{buf}}$.

Lemma 18. *Let $Z_1 = (c_1, G_1)$ and $Z_2 = (c_2, G_2)$ be zonotopes such that $Z_1 \cap Z_2 \neq \emptyset$. Then the penetration depth between Z_1 and Z_2 is given by*

$$\pi(Z_1, Z_2) = d(c_1, \partial Z_{2,buf}). \tag{64}$$

Proof. Suppose $Z_1 \cap Z_2 \neq \emptyset$. Then by Defn. 17, there exists a translation vector $t \in \mathbb{R}^{n_d}$ such that $\text{int}(Z_1 + t \cap Z_2) = \emptyset$. This means that Z_1 is translated by t such that only the boundaries of $Z_1 + t$ and Z_2 are intersecting. Thus, there exists $z_1 \in \partial Z_1$ and $z_2 \in \partial Z_2$ such that $z_1 + t = z_2$. Furthermore, there exist coefficients β_1 and β_2 such that $z_1 = c_1 + G_1\beta_1$, $z_2 = c_2 + G_2\beta_2$, and

$$c_1 + G_1\beta_1 + t = c_2 + G_2\beta_2 \tag{65}$$
$$\iff c_1 + t = c_2 + G_2\beta_2 - G_1\beta_1 \in \partial Z_{2,\text{buf}}. \tag{66}$$

Since $d(c_1, \partial Z_{2,\text{buf}})$ is the minimum distance between c_1 and all points in the boundary of $Z_{2,\text{buf}}$, this means that $d(c_1, \partial Z_{2,\text{buf}}) \leq \|t\|$.

Now suppose $c_1 \in Z_{2,\text{buf}}$ and let $t \in \mathbb{R}^{n_d}$ such that $\|t\|$ is the distance between c_1 and the boundary of $Z_{2,\text{buf}}$. Then there exists coefficients β_1 and β_2 such that $z_2 = c_2 + G_1\beta_1 + G_2\beta_2$ and

$$c_1 + t = c_2 + G_1\beta_1 + G_2\beta_2 \tag{67}$$

$$c_1 - G_1\beta_1 + t = c_2 + G_2\beta_2. \tag{68}$$

This means translating $z_1 = c_1 - G_1\beta_1 \in \partial Z_1$ by t causes the boundary of Z_1 to intersect the boundary of Z_2. Therefore $\pi(Z_1, Z_2) \leq \|t\|$. Together the above inequalities prove the result. □

Theorem 19. *Let $Z_1 = (c_1, G_1)$ and $Z_2 = (c_2, G_2)$ be zonotopes such that $Z_1 \cap Z_2 \neq \emptyset$. Then the signed distance between Z_1 and Z_2 is given by*

$$s(Z_1, Z_2) = \begin{cases} d(c_1; \partial Z_{2,buf}) & \text{if } c_1 \notin Z_{2,buf} \\ -d(c_1, \partial Z_{2,buf}) & \text{if } c_1 \in Z_{2,buf} \end{cases}. \tag{69}$$

Proof. The proof follows readily from Lemmas 16 and 18. □

Note that we provide a visual illustration of this theorem in Fig. 5.

B. The Proof

We prove the desired result by considering two cases. First assume $O \cap \mathrm{FO}(q(T;k)) = \emptyset$ for all $t \in T$. Then

$$\begin{aligned} r(O, \mathrm{FO}(q(T;k))) &= \min_{t \in T} \min_{j \in N_q} s(O, \mathrm{FO}_j(q(t;k))) && (70) \\ &\geq \min_{i \in N_t} \min_{j \in N_q} s(O, \mathbf{FO_j}(\mathbf{q}(\mathbf{T_i};k))) && (71) \\ &= \min_{i \in N_t} \min_{j \in N_q} s(c_O, \mathbf{FO_j^{buf}}(\mathbf{q}(\mathbf{T_i};k))) && (72) \\ &= \min_{j \in N_q} s(c_O, \bigcup_{i \in N_t} \mathbf{FO_j^{buf}}(\mathbf{q}(\mathbf{T_i};k))) && (73) \\ &\geq \min_{j \in N_q} s(c_O, \mathcal{P}_j) && (74) \\ &= \min_{j \in N_q} \tilde{r}_j(c_O, \mathcal{P}_j) && (75) \\ &= \tilde{r}(c_O, \bigcup_{j \in N_q} \mathcal{P}_j), && (76) \end{aligned}$$

where the first equality follows from the definition of r in Def. 11, the second inequality follows from Lemma 13, the third equality follows from Lemma 16, the fourth inequality by flipping the order of minimization, and the fifth inequality follows from Def. 4.

Thus $\tilde{r}$ underapproximates the distance between the forward occupancy and the obstacle. If instead $O \cap \mathrm{FO}_j(q(t)) \neq \emptyset$ for some $t \in T$, replacing $\geq$ with $\leq$ shows that we can overapproximate the penetration distance between an obstacle and the forward occupancy.

Appendix B
SDF Time Discretization

This section shows the performance of the SDF model in the presence of 10 obstacles with time discretizations ranging from 0.001 to 1.0 seconds. We note, in particular, SDF achieves no successes when the planning time limit is 0.025s and the time discretization is 0.001 seconds. These results support our earlier conclusion that RDF is superior to SDF for real-time motion planning.

time discretization (s)	# success	# collision	mean planning time (s)
1.0	149	161	0.020
0.5	258	5	0.058
0.1	353	0	0.055
0.01	357	0	0.065
0.001	354	0	0.210

TABLE XIII: Performance of SDF in 500 trials of 3D 7 link planning experiments with 5.0s time limit and 10 obstacles

time discretization (s)	# success	# collision	mean planning time (s)
1.0	149	161	0.020
0.5	257	5	0.027
0.1	349	0	0.027
0.01	352	0	0.037
0.001	302	0	0.143

TABLE XIV: Performance of SDF in 500 trials of 3D 7 link planning experiments with 0.3s time limit and 10 obstacles

time discretization (s)	# success	# collision	mean planning time (s)
1.0	148	161	0.019
0.5	238	3	0.020
0.1	262	0	0.021
0.01	214	0	0.029
0.001	0	0	>0.033

TABLE XV: Performance of SDF in 500 trials of 3D 7 link planning experiments with 0.033s time limit and 10 obstacles

time discretization (s)	# success	# collision	mean planning time (s)
1.0	148	160	0.019
0.5	227	3	0.020
0.1	224	0	0.020
0.01	4	0	>0.025
0.001	0	0	>0.025

TABLE XVI: Performance of SDF in 500 trials of 3D 7 link planning experiments with 0.025s time limit and 10 obstacles