Conformalized Reachable Sets for Obstacle Avoidance With Spheres

Yongseok Kwon **Jonathan Michaux** **Seth Isaacson** **Bohao Zhang**
Matthew Ejakov **Katherine A. Skinner** **Ram Vasudevan**
Department of Robotics, University of Michigan, Ann Arbor, MI 48109, United States
{kwonys, jmichaux, sethgi, jimzhang, ejakovm, kskin, ramv}@umich.edu

Abstract: Safe motion planning algorithms are necessary for deploying autonomous robots in unstructured environments. Motion plans must be safe to ensure that the robot does not harm humans or damage any nearby objects. Generating these motion plans in real-time is also important to ensure that the robot can adapt to sudden changes in its environment. Many trajectory optimization methods introduce heuristics that balance safety and real-time performance, potentially increasing the risk of the robot colliding with its environment. This paper addresses this challenge by proposing Conformalized Reachable Sets for Obstacle Avoidance With Spheres (CROWS). CROWS is a novel real-time, receding-horizon trajectory planner that generates probablistically-safe motion plans. Offline, CROWS learns a novel neural network-based representation of a sphere-based reachable set that overapproximates the swept volume of the robot's motion. CROWS then uses conformal prediction to compute a confidence bound that provides a probabilistic safety guarantee on the learned reachable set. At runtime, CROWS performs trajectory optimization to select a trajectory that is probabilstically-guaranteed to be collision-free. We demonstrate that CROWS outperforms a variety of state-of-the-art methods in solving challenging motion planning tasks in cluttered environments while remaining collision-free. Code, data, and video demonstrations can be found at https://roahmlab.github.io/crows/.

Keywords: Robot Safety, Conformal Prediction, Reachability Analysis

1 Introduction

Robot manipulators have the potential to transform multiple facets of how humans live and work. This includes using robots to complete daily chores in peoples' homes, replacing humans in performing difficult, dangerous, or repetitive tasks at construction sites, and assisting medical professionals with life-saving surgeries. In each of these settings, it is essential that the robot remain safe at all times to prevent colliding with obstacles, damaging high-value objects, or harming nearby humans. It is also critical that the robot generates motion plans in real-time to ensure that any given task is performed efficiently or the robot can quickly adjust its behavior to react to local changes in its environment.

A modern approach to motion planning typically involves combining a high-level planner, a mid-level trajectory planner, and a low-level tracking controller into a hierarchical framework [1, 2]. The high-level planner generates a path between the robot's start and goal configurations consisting of a sequence of discrete waypoints. The mid-level trajectory planner computes time-dependent positions, velocities, and accelerations that move the robot from one waypoint to the next. The low-level tracking controller generates control inputs that attempt to minimize deviations between the robot's actual motion and the desired trajectory. For example, one may use a sampling-based planner

Workshop on Safe and Robust Robot Learning for Operation in the Real World (SAFE-ROL) at CoRL 2024.

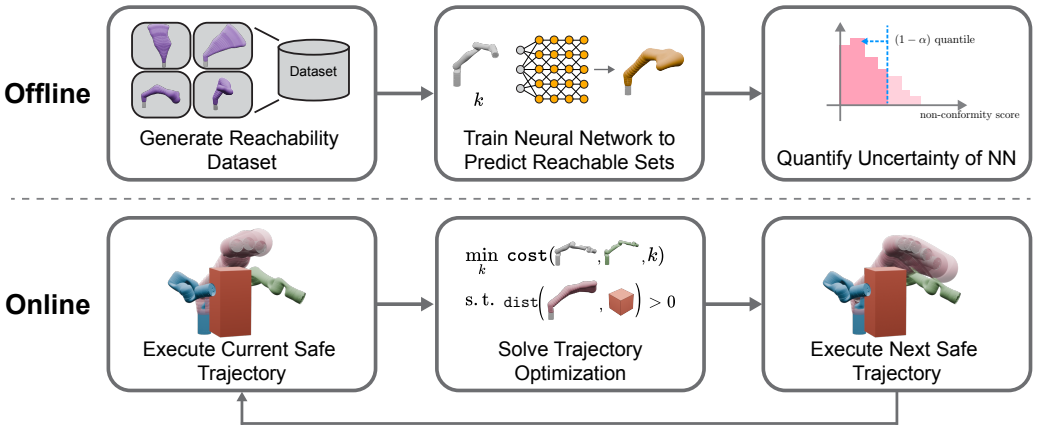

Figure 1: Overview of CROWS, a probabilistically-safe receding-horizon trajectory planner. Offline, CROWS constructs a dataset to train a network that predicts spherical overapproximations of parameterized swept volumes of a robot arm. After training, conformal prediction is used to quantify the uncertainty of the network's predictions. Online, the output of the network is buffered by a nonconformity score and used as a safety constraint during trajectory optimization. Finally, CROWS executes the trajectory while generating a plan for the next iteration.

such as an RRT* to generate a set of discrete waypoints for a trajectory optimization algorithm such as TrajOpt [3], and track the resulting trajectories with an inverse dynamics controller [4].

While variations of this planning framework have been demonstrated to work on various robotic platforms [5, 1, 2], several limitations prevent this approach from being widely deployed in the real-world. For instance, this approach can become computationally expensive as the complexity of the environment or robot increases, making it less practical for applications requiring both real-time performance and safety guarantees. Many algorithms also introduce heuristics to improve computation speed such as reducing the number of collision checks to achieve real-time performance; however, this may come at the expense of robot safety. Furthermore, many algorithms assume that the robot's dynamics are fully known, while in reality there can be considerable model uncertainty. Unfortunately, both of these factors may increase the potential for the robot to collide with obstacles.

To address this challenge, this paper proposes Conformalized Reachable Sets for Obstacle Avoidance With Spheres (CROWS), a neural network-based safety representation that can be efficiently integrated into a mid-level trajectory optimization algorithm (Fig. 2). CROWS extends [6] by learning an overapproximation of the swept volume (*i.e.* reachable set) of a serial robot manipulator that is composed entirely of spheres. Prior to planning, a neural network is trained to approximate the sphere-based reachable set (Sec. 3.1). Then, CROWS applies conformal prediction to compute a confidence bound that provides a probabilistic safety guarantee (Sec. 3.2). Finally, CROWS uses the conformalized reachable set and its learned gradient to solve an optimization problem to generate probabilistically-safe trajectories online (Sec. 3.3). In Sec. 4.1, we demonstrate that this novel reachable set formulation enables planning in extremely cluttered environments.

1.1 Related Work

Safe motion planning algorithms ensure that a robot can move from one configuration to another in its environment while avoiding collisions for all time. Ideally, one would compute the robot's swept volume [7, 8, 9, 10, 11, 12] and ensure that it does not intersect with any obstacles nor enter any unwanted regions of its workspace. Alternatively, many algorithms approximate the swept volume using occupancy grids, convex polyhedra, or CAD models [13, 14, 15]. However, both approaches are often intractable or suffer from high computational costs while approximate swept volume algorithms may be overly conservative [15, 16]. This may limit their utility to offline motion planning [17].

An alternative approach to robot collision avoidance involves modeling the robot or the environment with simple collision primitives such as spheres [18, 19], ellipsoids [20], capsules [21, 22], and then performing collision-checking along a given trajectory at discrete time instances. This is common for state-of-the-art trajectory optimization-based approaches [23, 3, 24, 25]. However, the resulting trajectories cannot be considered safe as collision avoidance is not enforced for all time.

Reachability-based Trajectory Design (RTD) [5], a recent approach to real-time motion planning, uses (polynomial) zonotopes [26] to construct reachable sets that overapproximate all possible robot positions corresponding to a pre-specified family of parameterized trajectories. Notably, RTD's reachable sets are constructed to ensure that its obstacle-avoidance constraints are satisfied in continuous-time. SPARROWS [6], a recent extension to RTD, utilizes sphere-based reachable sets to generate certifiably-safe motion plans that are considerably less conservative than previous approaches [1, 27, 2, 28]. The purpose of this paper is to introduce a neural representation of SPARROWS with uncertainty quantification for real-time *probabilistically-safe* motion planning.

Several methods have been developed to quantify the uncertainty of neural network models such as Monte Carlo dropout [29, 30, 31], Laplace approximation [32, 33, 34], and deep ensembles [35, 36]. More recently, conformal prediction [37, 38] has gained popularity due to its ability to provide probabilistic guarantees on coverage. Conformal prediction has been employed in various robotics applications including: ensuring the safety of learning-based object detection systems [39]; quantifying the uncertainty of human inputs during teleoperation [40]; ensuring safety in environments with dynamic obstacles [41, 42]; and applying conformal prediction to align uncertainty in large language models, enabling them to decide when to seek human assistance. Notably, [43] is closely related to our work, as it enhances the efficiency of reachable set computations for high-dimensional systems with neural network while verifying it with conformal prediction.

2 Background

This section summarizes the background necessary for the development of CROWS in Sec. 3.

2.1 Mathematical Preliminaries

Sets, subspaces, and matrices are typeset using capital letters. Let $\mathbb{R}$ and $\mathbb{N}$ denote the spaces of real numbers and natural numbers, respectively. Subscripts are primarily used as an index or as a label for relevant countable sets. For example, if $n_\alpha \in \mathbb{N}$, then we denote the set $N_\alpha = \{1, \cdots, n_\alpha\}$. Given a set $\mathcal{A}$, denote its power set as $P(\mathcal{A})$. Given a set $\Omega \subset \mathbb{R}^{n_d}$, $co(\Omega)$ denote its convex hull.

2.2 Arm Occupancy

This subsection describes how to overapproximate the forward occupancy, or swept volume, of a moving robot arm. Consider a compact time interval $T \subset \mathbb{R}$. We define a trajectory for the robot's configuration as $q : T \to Q \subset \mathbb{R}^{n_q}$ and a trajectory for the velocity as $\dot{q} : T \to \mathbb{R}^{n_q}$.

To facilitate the later discussion of the robot's occupancy, we begin by restating an assumption [6, Ass. 4] about the robot model:

Assumption 1. *The robot operates in an three-dimensional workspace, denoted $W_s \subset \mathbb{R}^3$. There exists a reference frame called the base frame, denoted the 0^{th} frame, that indicates the origin of the robot's kinematic chain within the workspace. The robot is fully actuated and 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 $q_j(t) \in [q^-_{j,\text{lim}}, q^+_{j,\text{lim}}]$ and $\dot{q}_j(t) \in [\dot{q}^-_{j,\text{lim}}, \dot{q}^+_{j,\text{lim}}]$ for all $t \in T$, respectively.*

Let $L_j \subset W_s \subset \mathbb{R}^3$ denote the volume occupied by the robot's j^{th} link with respect to the j^{th} reference frame. Then the *forward occupancy* of the j^{th} link is the map $\text{FO}_j : Q \to \mathcal{P}(W_s)$ defined as

$$\text{FO}_j(q(t)) = p_j(q(t)) \oplus R_j(q(t))L_j, \tag{1}$$

where $p_j(q(t))$ and $R_j(q(t))$ are computed by the forward kinematics [44] and specify the pose of the j^{th} joint, and $R_j(q(t))L_j$ is the rotated volume of link j. The volume occupied by the entire arm in the workspace is the union of the link volumes defined by the map $\text{FO} : Q \to W_s$:

$$\text{FO}(q(t)) = \bigcup_{j=1}^{n_q} \text{FO}_j(q(t)) \subset W_s. \tag{2}$$

Due to the potentially complex geometry of a robot's links, we reiterate an assumption [6, Ass. 5] that facilitates the construction of an overapproximation of the forward occupancy:

Assumption 2. *Given a robot configuration $q(t)$ and any $j \in \{1, \dots, n_q\}$, there exists a ball with center $p_j(q(t))$ and radius r_j that overapproximates the volume occupied by the j^{th} joint in W_s. We further assume that link volume L_j is a subset of the tapered capsule formed by the convex hull of the balls overapproximating the j^{th} and $j+1^{th}$ joints.*

Following Assum. 2, we now define the ball $S_j(q(t))$ overapproximating the volume occupied by the j^{th} joint as

$$S_j(q(t)) = \mathcal{B}(p_j(q(t)), r_j) \tag{3}$$

and the tapered capsule $TC_j(q(t))$ overapproximating the j^{th} link as

$$TC_j(q(t)) = co\Big(S_j(q(t)) \cup S_{j+1}(q(t))\Big). \tag{4}$$

Then, the volumes occupied by the j^{th} link (1) and the entire arm (2) can be overapproximated by

$$\text{FO}_j(q(t)) \subset TC_j(q(t)) \tag{5}$$

and

$$\text{FO}(q(t)) \subset \bigcup_{j=1}^{n_q} TC_j(q(t)) \subset W_s, \tag{6}$$

respectively.

Note that if the forward occupancy (or *reachable set*) $\text{FO}(q(T))$ does not intersect with the environment, then the arm is *collision-free* over the time interval T. To facilitate the exposition of our approach, we summarize the construction of the robot's safety representation by restating [6][Thm. 10]:

Theorem 3. *Given a serial manipulator with $n_q \in \mathbb{N}$ revolute joints, a time partition T of a finite set of intervals, T_i (i.e., $T = \cup_{i=1}^{n_t} T_i$), the swept volume corresponding to the robot's motion over T is overapproximated by a collection of L_2 balls in $\mathbb{R}^3$, which we call the Spherical Forward Occupancy (SFO [6] defined as*

$$\mathcal{SFO} = \cup_{j=1}^{n_q} \cup_{i=1}^{n_t} \cup_{m=1}^{n_S} S_{j,i,m}(q(T_i; k)), \tag{7}$$

where each $S_{j,i,m}(q(T_i; k))$ is an L_2 ball in $\mathbb{R}^3$, $n_S \in \mathbb{N}$ is a parameter that specifies the number of closed balls overapproximating each of the robot's links, and k is a trajectory parameter that characterizes the motion of the robot over T.

Note that one can explicitly construct an $\mathcal{SFO}$ that satisfies this assumption using the approach described in [6].

2.3 Environment Modeling

The $\mathcal{SFO}$ constructed in Thm. 3 is composed entirely of spheres, where each sphere is a collision primitive consisting of a point with a safety margin corresponding to its radius. For simplicity, we assume that each obstacle $\mathcal{O}$ is a polytope. Therefore, to make use of this sphere-based representation for trajectory planning, we state the result of [6] [Lem. 11] as an assumption describing the existence of an obstacle exact signed distance:

Assumption 4 (Environment Signed Distance Function)**.** *Given the $\mathcal{SFO}$ and a convex obstacle polytope $\mathcal{O} \in \mathbb{R}^3$, there exists a known function, $\mathbf{s_d}(\mathcal{SFO}, \mathcal{O})$, that computes the exact signed distance between $\mathcal{SFO}$ and $\mathcal{O}$.*

2.4 Conformal Prediction

Consider N_{cal} i.i.d. samples $\{\delta_d\}_{d=1}^{N_{cal}}$ of a random variable. For a new i.i.d. sample $\delta_{d'}$, conformal prediction provides the following guarantee:

$$\mathbb{P}(\delta_{d'} \leq \Delta) \geq 1 - \epsilon \tag{8}$$

where Δ is the $\lceil (N_{cal}+1)(1-\epsilon) \rceil$-th quantile of $\{\delta_d\}_{d=1}^{N_{cal}}$. To extend this guarantee to a new sample $\bar{\delta}_{d'}$, one would typically need to refresh the calibration data $\{\bar{\delta}_d\}_{d=1}^{N_{cal}}$. In this work, however, we employ a guarantee conditioned on the calibration dataset $\mathcal{D}_{cal} = \{\delta_d\}_{d=1}^{N_{cal}}$, with a probability of $1 - \rho$ [45, 39]:

$$\mathbb{P}(\delta_{d'} \leq \Delta_{1-\hat{\epsilon}} | \mathcal{D}_{cal}) \geq \text{Beta}_{N_{cal}+1-\nu,\nu}(\rho) \tag{9}$$

where $\nu := \lfloor (N_{cal}+1)\hat{\epsilon} \rfloor$, $\text{Beta}_{N_{cal}+1-\nu,\nu}(\rho)$ is the ρ-quantile of the Beta distribution, $\hat{\epsilon}$ is a user-defined coverage parameter, and $\Delta_{1-\hat{\epsilon}}$ is the $\lceil (N_{cal}+1)(1-\hat{\epsilon}) \rceil$-th quantile of the nonconformity scores in the calibration set $\mathcal{D}_{cal}$. To facilitate the following exposition, we will refer to the notation in (9) using the same form as in (8).

3 Trajectory Optimization Formulation

The method described in [6] computes provably-safe motion plans by solving a nonlinear optimization program in a receding-horizon manner:

$$\min_{k \in K} \quad \texttt{cost}(q_{\text{goal}}, k) \qquad \texttt{(Opt)} \tag{10}$$

$$q_j(T_i; k) \subseteq [q^-_{j,\text{lim}}, q^+_{j,\text{lim}}] \qquad \forall (i,j) \in N_t \times N_q \tag{11}$$

$$\dot{q}_j(T_i; k) \subseteq [\dot{q}^-_{j,\text{lim}}, \dot{q}^+_{j,\text{lim}}] \qquad \forall (i,j) \in N_t \times N_q \tag{12}$$

$$\mathbf{s_d}(\mathcal{SFO}, \mathcal{O}_n) > 0 \qquad \forall n \in N_{\mathcal{O}}, \tag{13}$$

where $\mathcal{O}_n$ is the n^{th} obstacle for $n \in N_{\mathcal{O}}$. Offline, we pre-specify a continuum of trajectories over a compact set $K \subset \mathbb{R}^{n_k}$ such that $n_k \in \mathbb{N}$. Then each trajectory, $q(t; k)$, is defined over a compact time interval T and is uniquely determined by a *trajectory parameter* $k \in K$. The cost function (10) ensures the robot moves towards a user- or task-defined goal q_{goal}. The constraints (11)–(12) ensure that the trajectory remains feasible and does not violate the robot's joint position and velocity limits, respectively. The last constraint (13) guarantees safety by ensuring that the robot's forward occupancy does not collide with any obstacles in the environment.

In the remainder of this section, we describe how CROWS replaces (13) with a fast, neural representation with probabilistic safety-guarantees.

3.1 Neural Network Models

In this subsection, we describe how CROWS learns neural network representations of the centers and radii of $\mathcal{SFO}$ in (7) and their gradients with respect to the trajectory parameter.

3.1.1 Neural SFO

Given a family of parameterized trajectories $q(t; k)$ for $k \in K$, we train a neural network to predict the centers $c_{j,i}(q_j(T_i; k))$ and radii $r_{j,i}(q_j(T_i; k))$ for all $(i,j) \in N_t \times N_q$ of the Spherical Forward Occupancy. For simplicity, we have dropped the subscript m to indicate that the center and radii networks only predict the spheres that overapproximate the joint volumes.

Following training, we use [6][Lem. 9, Thm. 10] to construct the remaining spheres that overapproximate the links. The network takes as input $x = (q_0, \dot{q}_0, k, i)$, which consists of a concatenation of vectors corresponding to the initial joint positions q_0, initial joint velocities $\dot{q}_0$, trajectory parameter k, and the i^{th} time index. The output of the network is $y = (\hat{c}_{j,i}, \hat{r}_{j,i}) \in \mathbb{R}^{4(n_q+1)}$. We discuss specific training details in Sec. 7.2.

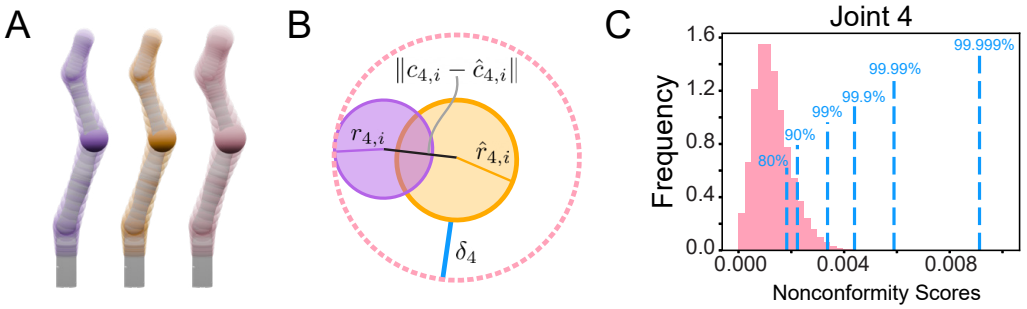

Figure 2: A visual illustration of the construction of CROWS's conformalized reachable sets. Panel A compares the ground truth (left, purple), predicted (middle, orange), and conformalized (right, pink) reachable sets. CROWS first defines the nonconformity score in (14), which is the minimum buffer (blue) to ensure that the predicted sphere (orange) encloses the ground truth sphere (purple) (Panel B). The distribution of the nonconformity scores for joint 4 over the interval T_i with the values defining the quantiles indicated in blue (Panel V). Next, conformal prediction computes a confidence bound that upper bounds the nonconformity scores with a probability of $1-\epsilon$. The predicted joint sphere (A, middle) is then expanded by the size of this confidence bound to give the conformalized joint sphere (A, right). Finally, applying this procedure for all of the joint spheres gives the conformalized neural (A, right) reachable set that is guaranteed to cover the ground truth reachable set with probability greater than $(1-\epsilon)^{n_q+1}$ (Thm. 5).

3.1.2 Neural SFO Gradient

To facilitate real-time solutions of `(Opt)`, we also train a neural network to output $\widehat{\frac{\partial c_{j,i}}{\partial k}}$, which is a prediction of the gradient of each joint center with respect to the trajectory parameter k. Because the radii of $\mathcal{SFO}$ do not depend on k, we do not train its corresponding gradient network.

3.2 Conformal Prediction

Next, we describe the use of conformal prediction to formulate probabilistic guarantees for collision avoidance.

3.2.1 Calibration

We first construct a calibration set $\mathcal{D}_{cal} = \{(x_d, y_d)\}_{d=1}^{N_{cal}}$ with i.i.d. samples, where $N_{cal} \in \mathbb{N}$. Let $\mathcal{B}_{j,i} = \mathcal{B}(c_{j,i}, r_{j,i})$ be a ground truth sphere and $\hat{\mathcal{B}}_{j,i}(\delta_j) = \mathcal{B}(\hat{c}_{j,i}, \hat{r}_{j,i} + \delta_j)$ be the corresponding prediction. Note that $\mathcal{B}_{j,i}$ corresponds to $S_{j,i}(q(T_i; k))$ in Thm. 3. Then for the calibration step, we define a nonconformity score δ_j for each joint sphere such that

$$\delta_j = \max(||c_{j,i} - \hat{c}_{j,i}||_2 + r_{j,i} - \hat{r}_{j,i},\ 0) \tag{14}$$

is minimum buffer required for $\mathcal{B}_{j,i} \subseteq \hat{\mathcal{B}}_{j,i}(\delta_j)$ as illustrated in Fig. 2 (B).

Given the nonconformity scores over the calibration set $\mathcal{D}_{cal}$, the nonconformity score $\delta_{j,d'}$ on a new sample $(x_{d'}, y_{d'})$ follows a gaurantee by conformal prediction:

$$\mathbb{P}(\delta_j \leq \Delta_j) \geq 1 - \epsilon \tag{15}$$

For notational simplicity, we have dropped the subscript d' here.

3.2.2 Conformalized Reachable Sets

Given (15), the conformalized sphere $\hat{\mathcal{B}}_{j,i}(\Delta_j)$ is guaranteed to enclose the ground truth sphere $\mathcal{B}_{j,i}$ with the following probability:

$$\mathbb{P}(\mathcal{B}_{j,i} \subseteq \hat{\mathcal{B}}_{j,i}(\Delta_j)) \tag{16}$$

$$\geq \mathbb{P}(\mathcal{B}_{j,i} \subseteq \hat{\mathcal{B}}_{j,i}(\delta_j)) \cdot \mathbb{P}(\hat{\mathcal{B}}_{j,i}(\delta_j) \subseteq \hat{\mathcal{B}}_{j,i}(\Delta_j)) \tag{17}$$

$$= \mathbb{P}(\hat{\mathcal{B}}_{j,i}(\delta_j) \subseteq \hat{\mathcal{B}}_{j,i}(\Delta_j)) \tag{18}$$

$$= \mathbb{P}(\delta_j \leq \Delta_j) \geq 1 - \epsilon. \tag{19}$$

Methods	# Successes			
Scenario	10 Rand. Obs.	20 Rand. Obs.	40 Rand. Obs.	Realistic
CROWS	**87**	58	37	13
CROWS(-)	83	60	30	12
SPARROWS	**87**	**62**	**40**	**14**
ARMTD	56	17	0	8
CHOMP [23]	76	40	15	10
TrajOpt [3]	33 (67)	9 (91)	6 (94)	5 (9)
MPOT [24]	58 (42)	23 (77)	9 (91)	6 (8)
cuRobo [25]	59 (41)	45 (55)	22 (78)	5 (9)

Table 1: Number of successes out of 100 trials on **Random Obstacle Scenarios** (the first three columns) and out of 14 trials on the **Realistic Scenarios** (the last column). Red indicates the number of failures due to collision.

This approach can be extended to all joint occupancy spheres to construct a probabilistic safety guarantee to ensure that the ground truth reachable set is collision-free with probability greater than $(1-\epsilon)^{n_q+1}$ over the i^{th} time interval T_i. We summarize this result in the following theorem whose proof can be found in Sec. 6:

Theorem 5. *Given $\mathbb{P}(\mathcal{B}_{j,i} \subseteq \hat{\mathcal{B}}_{j,i}(\Delta_j)) \geq 1-\epsilon$ for $j \in N_q$, the following probability holds under the condition $\mathbf{s_d}(\mathcal{SF}\hat{\mathcal{O}}_i, \mathscr{O}) > 0$:*

$$\mathbb{P}(\mathbf{s_d}(\boldsymbol{FO}_i, \mathscr{O}) > 0) \geq (1-\epsilon)^{n_q+1} \tag{20}$$

where $\boldsymbol{FO}_i$ denotes the forward occupancy over a single time interval T_i ((2)), $\mathcal{SF}\hat{\mathcal{O}}_i$ is the Spherical Forward Occupancy constructed from $\hat{\mathcal{B}}_{j,i}(\Delta_j)$ for all $j \in N_q$ over the same time interval T_i by Theorem 3, $\mathscr{O}$ represents a set of obstacles defined as $\{\mathcal{O}_n \mid n \in N_{\mathcal{O}}\}$, and $\mathbf{s_d}(\cdot, \mathscr{O}) = \min_{n \in N_{\mathcal{O}}} \mathbf{s_d}(\cdot, \mathcal{O}_n)$.

3.3 Online Trajectory Optimization

After training, we generate models to predict the spheres of the $\mathcal{SFO}$. Using this representation, we can reformulate the optimization problem described by (10)–(13) into:

$$\min_{k \in K} \quad \texttt{cost}(q_{\text{goal}}, k) \qquad \texttt{(CROWS-Opt)} \tag{21}$$

$$q_j(T_i; k) \subseteq [q^-_{j,\text{lim}}, q^+_{j,\text{lim}}] \qquad \forall (i,j) \in N_t \times N_q \tag{22}$$

$$\dot{q}_j(T_i; k) \subseteq [\dot{q}^-_{j,\text{lim}}, \dot{q}^+_{j,\text{lim}}] \qquad \forall (i,j) \in N_t \times N_q \tag{23}$$

$$\mathbf{s_d}(\mathcal{SF}\hat{\mathcal{O}}_i, \mathscr{O}) > 0 \qquad \forall i \in N_t \tag{24}$$

where $\mathcal{SF}\hat{\mathcal{O}}$ is the conformalized reachable set in Thm. 5.

4 Results

We demonstrate the performance of CROWS in both simulation and hardware experiments.

4.1 Simulation Experiments

We compare the performance of CROWS to SPARROWS [6], ARMTD [1], CHOMP [23], TrajOpt [3], MPOT [24], and cuRobo [25] on two sets of planning tasks. We also include a comparison to CROWS(-), which uses PyTorch's built-in `autograd` function to compute the gradient rather than the learned model introduced in Sec. 3.1.2. The first set of tasks, **Random Obstacle Scenarios**, contains $n_{\mathcal{O}} = 10$, 20, or 40 axis-aligned boxes that are 20cm on each side. For each number of obstacles, we generate 100 scenes with obstacles placed randomly such that the start and goal configurations are collision-free. The second set of tasks, **Realistic Scenarios**, contains 14 scenes with geometric features similar to those found in a household setting. For both planning tasks, a

Figure 3: Sequential images showing CROWS demonstrating safe trajectory planning in real-world experiments, from the initial configuration (left) to the goal configuration (right).

failure occurs if CROWS, SPARROWS, or ARMTD fails to find a plan for two consecutive planning iterations or if a collision occurs. CROWS, SPARROWS, and ARMTD are given a planning time limit of 0.5s and a maximum of 150 planning iterations to reach the goal. CHOMP, TrajOpt, and cuRobo are given planning time limited of 100s, 30s, and 2s, respectively.

Tables 1 compare the success rate of CROWS to the baselines on the **Random Obstacle Scenarios** and **Realistic Scenarios**, respectively. SPARROWS achieves the highest success rate on both sets of tasks followed by CROWS or CROWS(-). Note that both CROWS and CROWS(-) are competitive with SPARROWS while also remaining collision-free. In contrast, the baselines TrajOpt, MPOT, and cuRobo all experience collisions. Sec. 8.2 presents a runtime comparison between CROWS, SPARROWS, and ARMTD, highlighting the computational advantages of using learned representation of SFO and its gradient.

4.2 Hardware Experiments

We demonstrate the motion planning task using various initial and goal configurations (Fig. 3). The hardware setup follows a similar architecture to [46], where a high-level planner and mid-level trajectory planner run in parallel, and a low-level controller tracks trajectories using the Recursive Newton-Euler Algorithm (RNEA) [47] alongside a PD controller. In the real-world experiments, CROWS successfully performs real-time trajectory planning, reaching the goal configuration without colliding with the environment.

5 Conclusion and Future Directions

We present CROWS, a method for generating real-time motion plans for robot manipulators in cluttered environments. We show CROWS remains safe across many motion planning trials and has performance comparable to SPARROWS, a state-of-the-art, model-based trajectory optimizer. We also demonstrate that learning a representation of $\mathcal{SFO}$'s gradient leads to improvements in both planning time and success rate. CROWS shows promise, as planning time with the learned representation of reachable sets in [27] scales linearly with problem dimension, whereas planning time with closed-form methods like those in [6] and [1] scales at a rate exceeding linear growth with dimension. Future directions will extend CROWS to higher-dimensional systems such as humanoid robots as well as learning conformalized representations of the robot's dynamics. Furthermore, we anticipate that embedding CROWS within end-to-end architectures for planning in more general scene representations is also a promising direction for future research.

Acknowledgments

This work is supported by the National Science Foundation Career Award #1751093 and by the Air Force Office of Scientific Research under award 23-S15.

References

[1] P. Holmes, S. Kousik, B. Zhang, D. Raz, C. Barbalata, M. J. Roberson, and R. Vasudevan. Reachable Sets for Safe, Real-Time Manipulator Trajectory Design. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, 7 2020. doi:10.15607/RSS.2020.XVI.100.

[2] J. Michaux, P. Holmes, B. Zhang, C. Chen, B. Wang, S. Sahgal, T. Zhang, S. Dey, S. Kousik, and R. Vasudevan. Can't touch this: Real-time, safe motion planning and control for manipulators under uncertainty, 2023.

[3] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014. doi:10.1177/0278364914528132. URL https://doi.org/10.1177/0278364914528132.

[4] F. Caccavale. *Inverse Dynamics Control*, pages 1–5. Springer Berlin Heidelberg, Berlin, Heidelberg, 2020. ISBN 978-3-642-41610-1. doi:10.1007/978-3-642-41610-1_95-1. URL https://doi.org/10.1007/978-3-642-41610-1_95-1.

[5] S. Kousik, S. Vaskov, M. Johnson-Roberson, and R. Vasudevan. Safe trajectory synthesis for autonomous driving in unforeseen environments. In *ASME 2017 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers Digital Collection, 2017.

[6] J. Michaux, A. Li, Q. Chen, C. Chen, B. Zhang, and R. Vasudevan. Safe planning for articulated robots using reachability-based obstacle avoidance with spheres. *ArXiv*, abs/2402.08857, 2024. URL https://arxiv.org/abs/2402.08857.

[7] D. Blackmore and M. Leu. A differential equation approach to swept volumes. In *[1990] Proceedings. Rensselaer's Second International Conference on Computer Integrated Manufacturing*, pages 143–149, May 1990. doi:10.1109/CIM.1990.128088.

[8] D. Blackmore and M. Leu. Analysis of Swept Volume via Lie Groups and Differential Equations. *The International Journal of Robotics Research*, 11(6):516–537, Dec. 1992. ISSN 0278-3649. doi:10.1177/027836499201100602. URL https://doi.org/10.1177/027836499201100602. Publisher: SAGE Publications Ltd STM.

[9] D. Blackmore, M. C. Leu, and F. Shih. Analysis and modelling of deformed swept volumes. *Computer-Aided Design*, 26(4):315–326, Apr. 1994. ISSN 0010-4485. doi:10.1016/0010-4485(94)90077-9. URL https://www.sciencedirect.com/science/article/pii/0010448594900779.

[10] D. Blackmore, M. Leu, and L. P. Wang. The sweep-envelope differential equation algorithm and its application to NC machining verification. *Computer-Aided Design*, 29(9):629–637, Sept. 1997. ISSN 0010-4485. doi:10.1016/S0010-4485(96)00101-7. URL https://www.sciencedirect.com/science/article/pii/S0010448596001017.

[11] D. Blackmore, R. Samulyak, and M. C. Leu. Trimming swept volumes. *Computer-Aided Design*, 31(3):215–223, Mar. 1999. ISSN 0010-4485. doi:10.1016/S0010-4485(99)00017-2. URL https://www.sciencedirect.com/science/article/pii/S0010448599000172.

[12] Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C-32(2):108–120, 1983. doi:10.1109/TC.1983.1676196.

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

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

[15] 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.

[16] C. Ekenna, D. Uwacu, S. Thomas, and N. M. Amato. Improved roadmap connection via local learning for sampling based planners. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3227–3234, 2015. doi:10.1109/IROS.2015.7353825.

[17] 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*, 28(2):427–439, 2012. doi:10.1109/TRO.2011.2172152.

[18] S. Duenser, J. M. Bern, R. Poranne, and S. Coros. Interactive robotic manipulation of elastic objects. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3476–3481, 2018. doi:10.1109/IROS.2018.8594291.

[19] M. Gaertner, M. Bjelonic, F. Farshidian, and M. Hutter. Collision-free mpc for legged robots in static and dynamic scenes, 2021.

[20] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora. Model predictive contouring control for collision avoidance in unstructured dynamic environments, 2020.

[21] C. Dube. Self collision avoidance for humanoids using circular and elliptical capsule bounding volumes. In *2013 Africon*, pages 1–6, 2013. doi:10.1109/AFRCON.2013.6757663.

[22] A. El Khoury, F. Lamiraux, and M. Taïx. Optimal motion planning for humanoid robots. In *2013 IEEE International Conference on Robotics and Automation*, pages 3136–3141, 2013. doi:10.1109/ICRA.2013.6631013.

[23] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa. Chomp: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193, 2013. doi:10.1177/0278364913488805. URL `https://doi.org/10.1177/0278364913488805`.

[24] A. T. Le, G. Chalvatzaki, A. Biess, and J. Peters. Accelerating motion planning via optimal transport, 2023.

[25] B. Sundaralingam, S. K. S. Hari, A. Fishman, C. Garrett, K. V. Wyk, V. Blukis, A. Millane, H. Oleynikova, A. Handa, F. Ramos, N. Ratliff, and D. Fox. curobo: Parallelized collision-free minimum-jerk robot motion generation, 2023.

[26] N. Kochdumper and M. Althoff. Sparse polynomial zonotopes: A novel set representation for reachability analysis. *IEEE Transactions on Automatic Control*, 66(9):4043–4058, 2020.

[27] J. B. Michaux, Y. S. Kwon, Q. Chen, and R. Vasudevan. Reachability-based Trajectory Design with Neural Implicit Safety Constraints. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi:10.15607/RSS.2023.XIX.062.

[28] Z. Brei, J. Michaux, B. Zhang, P. Holmes, and R. Vasudevan. Serving time: Real-time, safe motion planning and control for manipulation of unsecured objects. *IEEE Robotics and Automation Letters*, 9(3):2383–2390, 2024. doi:10.1109/LRA.2024.3355731.

[29] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[30] A. Shamsi, H. Asgharnezhad, M. Abdar, A. Tajally, A. Khosravi, S. Nahavandi, and H. Leung. Improving mc-dropout uncertainty estimates with calibration error-based optimization. *arXiv preprint arXiv:2110.03260*, 2021.

[31] B. Lütjens, M. Everett, and J. P. How. Safe reinforcement learning with model uncertainty estimates. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8662–8668. IEEE, 2019.

[32] E. Daxberger, A. Kristiadi, A. Immer, R. Eschenhagen, M. Bauer, and P. Hennig. Laplace redux-effortless bayesian deep learning. *Advances in Neural Information Processing Systems*, 34:20089–20103, 2021.

[33] H. Ritter, A. Botev, and D. Barber. A scalable laplace approximation for neural networks. In *6th international conference on learning representations, ICLR 2018-conference track proceedings*, volume 6. International Conference on Representation Learning, 2018.

[34] L. Goli, C. Reading, S. Sellán, A. Jacobson, and A. Tagliasacchi. Bayes' rays: Uncertainty quantification for neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20061–20070, 2024.

[35] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

[36] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

[37] A. N. Angelopoulos and S. Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511*, 2021.

[38] G. Shafer and V. Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(3), 2008.

[39] A. Dixit, Z. Mei, M. Booker, M. Storey-Matsutani, A. Z. Ren, and A. Majumdar. Perceive with confidence: Statistical safety assurances for navigation with learning-based perception. In *8th Annual Conference on Robot Learning*, 2024.

[40] M. Zhao, R. Simmons, H. Admoni, and A. Bajcsy. Conformalized teleoperation: Confidently mapping human inputs to high-dimensional robot actions. *arXiv preprint arXiv:2406.07767*, 2024.

[41] L. Lindemann, M. Cleaveland, G. Shim, and G. J. Pappas. Safe planning in dynamic environments using conformal prediction. *IEEE Robotics and Automation Letters*, 2023.

[42] J. Lekeufack, A. N. Angelopoulos, A. Bajcsy, M. I. Jordan, and J. Malik. Conformal decision theory: Safe autonomous decisions from imperfect predictions. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11668–11675. IEEE, 2024.

[43] A. Lin and S. Bansal. Verification of neural reachable tubes via scenario optimization and conformal prediction. In *6th Annual Learning for Dynamics & Control Conference*, pages 719–731. PMLR, 2024.

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

[45] V. Vovk. Conditional validity of inductive conformal predictors. In *Asian conference on machine learning*, pages 475–490. PMLR, 2012.

[46] J. Michaux, S. Isaacson, C. E. Adu, A. Li, R. K. Swayampakula, P. Ewen, S. Rice, K. A. Skinner, and R. Vasudevan. Let's make a splan: Risk-aware trajectory optimization in a normalized gaussian splat. *arXiv preprint arXiv:2409.16915*, 2024.

[47] J. Y. Luh, M. W. Walker, and R. P. Paul. On-line computational scheme for mechanical manipulators. 1980.

[48] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[49] A. Wächter and L. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106:25–57, 03 2006. doi:10.1007/s10107-004-0559-y.

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

[51] Dawson-Haggerty et al. trimesh, 2019. URL https://trimesh.org/.

[52] D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus), 2023. URL https://arxiv.org/abs/1606.08415.

[53] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

6 Proof of Theorem 5

Proof. The convex hull of two conformalized spheres encloses the convex hull of the corresponding ground truth spheres with probability:

$$\mathbb{P}(\text{TC}_{i,j} \subseteq \hat{\text{TC}}_{i,j}) \geq \prod_{j'=j}^{j+1} \mathbb{P}(\mathcal{B}_{j',i} \subseteq \hat{\mathcal{B}}_{j',i}(\Delta_{j'})) \geq (1-\epsilon)^2 \tag{25}$$

where $\text{TC}_{i,j} = co(\bigcup_{j'=j}^{j+1}(\mathcal{B}_{j',i}))$ and $\hat{\text{TC}}_{i,j} = co(\bigcup_{j'=j}^{j+1}(\hat{\mathcal{B}}_{j',i}(\Delta_{j'}))$. This probability extends across all the joints as follows:

$$\mathbb{P}(\bigcup_{j=1}^{n_q} \text{TC}_{i,j} \subseteq \bigcup_{j=1}^{n_q} \hat{\text{TC}}_{i,j}) \geq (1-\epsilon)^{n_q+1} \tag{26}$$

Because $\text{FO}_i \subseteq \bigcup_{j=1}^{n_q} \text{TC}_{i,j}$ and $\bigcup_{j=1}^{n_q} \hat{\text{TC}}_{i,j} \subseteq \mathcal{S}\hat{\mathcal{F}}\mathcal{O}_i$, the following guarantee holds:

$$\mathbb{P}(\text{FO}_i \subseteq \mathcal{S}\hat{\mathcal{F}}\mathcal{O}_i) \geq (1-\epsilon)^{n_q+1} \tag{27}$$

Therefore, the probability that the signed distance between the ground truth forward occupancy FO_i and obstacle set $\mathscr{O}$ remains positive is bounded by:

$$\mathbb{P}(\mathbf{s_d}(\text{FO}_i, \mathscr{O}) > 0) \tag{28}$$

$$= \mathbb{P}(\text{FO}_i \subseteq \mathcal{S}\hat{\mathcal{F}}\mathcal{O}_i) \cdot \mathbb{P}(\mathbf{s_d}(\mathcal{S}\hat{\mathcal{F}}\mathcal{O}_j, \mathscr{O}) > 0) \tag{29}$$

$$\geq (1-\epsilon)^{n_q+1} \cdot \mathbb{P}(s(\mathcal{S}\hat{\mathcal{F}}\mathcal{O}_j, \mathscr{O}) > 0) \tag{30}$$

If we enforce $\mathbb{P}(\mathbf{s_d}(\mathcal{S}\hat{\mathcal{F}}\mathcal{O}_j, \mathscr{O}) > 0) = 1$, then the following probability holds:

$$\mathbb{P}(\mathbf{s_d}(\text{FO}_i, \mathscr{O}) > 0) \geq (1-\epsilon)^{n_q+1}. \tag{31}$$

□

7 Implementation Details

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. 4.1. The CROWS model is built and trained with Pytorch [48]. The trajectory optimization `(CROWS-Opt)` was solved with IPOPT [49].

7.1 Simulation and Simulation Environment

We simulate the Kinova Gen3 7-DOF serial manipulator [50]. The robot's collision geometry is provided as a mesh, which we utilize with the trimesh library [51] to check for collisions with obstacles. For simplicity, we do not check self-collisions of the robot and all obstacles are static, axis-aligned cubes. We assume that the start and goal configurations of the robot are collision-free.

7.2 Training Details

Fully-connected networks were trained to predict the centers, radii, and gradients of the centers with respect to the trajectory parameter. These networks have a width 1024 with 3, 9, and 12 hidden layers, respectively. The radii network uses the ReLU activation function while the others use GELU [52]. As discussed in 3.1.1, the networks only predict the centers and radii that overapproximate the joint volumes. The networks also do not predict the center and radius of the base joint sphere as it remains constant for all time. For the loss function, we used the mean squared error between the target and the prediction. Each network was trained using the AdamW [53] optimizer with learning rate 0.0003, beta (0.9,0.999), and weight decay 0.0001. The training, validation, and calibration datasets consisted of 8e5, 1e5, and 1e5 samples, respectively.

Methods	mean planning time [s]		
# Obstacles	10	20	40
CROWS	**0.14 ± 0.29**	**0.16 ± 0.10**	**0.21 ± 0.10**
CROWS(-)	0.17 ± 0.10	0.20 ± 0.10	0.24 ± 0.11
SPARROWS	0.16 ± 0.08	0.18 ± 0.08	0.24 ± 0.09
ARMTD	0.24 ± 0.09	0.38 ± 0.10	0.51 ± 0.04

Table 2: Mean per-step planning time across 100 **Random Obstacle Scenarios** under a 0.5s planning time limit ↓. Red indicates that the average planning time limit has been exceeded.

Methods	mean constraint eval. time [ms]		
# Obstacles	10	20	40
CROWS	3.5 ± 0.2	4.1 ± 0.1	5.5 ± 0.1
CROWS(-)	6.3 ± 1.5	6.9 ± 1.2	8.3 ± 1.2
SPARROWS	**3.1± 0.1**	**3.8 ± 0.1**	**5.2 ± 0.1**
ARMTD	4.1 ± 0.3	5.4 ± 0.5	8.1 ± 0.7

Table 3: Mean runtime for constraint and constraint gradient evaluation across 100 **Random Obstacle Scenarios** under a 0.5s planning time limit ↓.

8 Additional Experimental Results

This section evaluates the performance of CROWS by measuring its prediction accuracy and its runtime.

8.1 Model Prediction Accuracy

The mean and maximum errors of the center predictions across all joints are 0.87cm and 2.10cm, respectively. For the radius predictions, the mean and maximum errors are 0.12 cm and 0.59cm. The median relative error of the neural SFO gradient is 1.25%. The nonconformity scores (14) are computed using the calibration set. The 99.9$^{\text{th}}$ percentile of the nonconformity scores for joints 2 through 8 are (0.05, 0.18, 0.35, 0.44, 0.56, 0.64, 0.75) cm, measured from the proximal to distal joint. In the trajectory optimization `(CROWS-Opt)`, an uncertainty bound of $\hat{\epsilon} = 0.001$ was used.

8.2 Runtime Comparisons

We compare the runtime performance of CROWS to SPARROWS and ARMTD under a planning time limit while varying the number of obstacles. We measure the mean constraint evaluation time as well as the mean planning time. Constraint evaluation includes the time to compute the constraints and the constraint gradients. Planning time includes the time required to construct the reachable sets and the total time required to solve `(Opt)` including constraint gradients evaluations. For each case, the results are averaged over 100 trials.

Tables 2 and 3 summarize the runtime comparisons under a planning time limit of $0.5s$. SPARROWS has the lowest mean constraint evaluation time followed by CROWS, CROWS(-), and ARMTD. CROWS, on the other hand, has the lowest per-step planning time followed by SPARROWS, CROWS(-), and ARMTD. These results indicate that learned gradient improves the solve time of `(CROWS-Opt)` when using the learned safety representation.

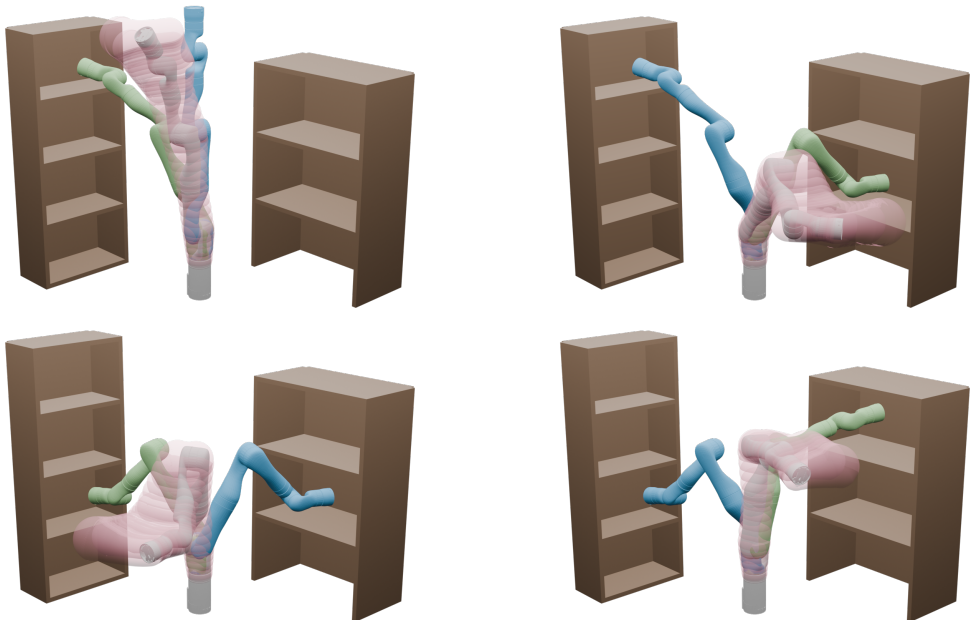

Figure 4: A demonstration of CROWS generating safe motion plans in a scene with bookshelves. Each start and goal configuration is shown in blue and green, respectively. The conformalized reachable set for an intermediate trajectory is shown in pink (transparent).

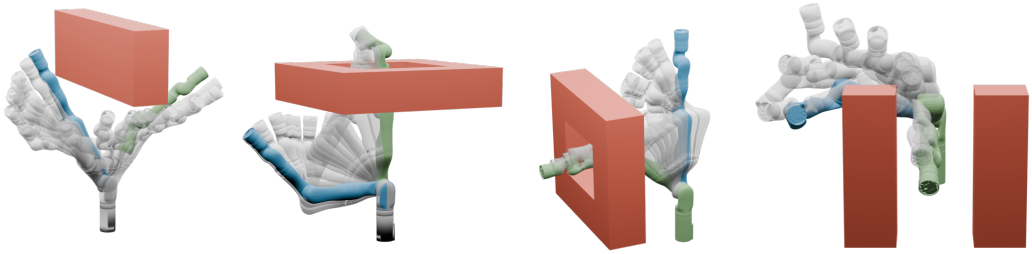

Figure 5: A subset of **Realistic Scenarios** where CROWS succeeds. The start, goal, and intermediate poses are shown in blue, green, and grey (transparent), respectively. Obstacles are shown in red (transparent).