

Neural Configuration-Space Barriers for Manipulation Planning and Control

Kehan Long¹, Ki Myung Brian Lee¹, Nikola Raicevic¹, Niyas Attasseril¹, Melvin Leok¹, and Nikolay Atanasov¹

Abstract—Planning and control for high-dimensional robot manipulators in cluttered dynamic environments require computational efficiency and robust safety guarantees. Inspired by recent advances in learning configuration-space distance functions (CDFs) as representations of robot bodies, we propose a unified approach for motion planning and control that formulates safety constraints as CDF barriers. A CDF barrier approximates the local free configuration space, substantially reducing the number of collision-checking operations during motion planning. However, learning a CDF barrier with a neural network and relying on online sensor observations introduces uncertainties that must be considered during control synthesis. To address this, we develop a distributionally robust CDF barrier formulation for control that accounts for modeling errors and sensor noise without assuming a known underlying distribution. Simulations and hardware experiments on a UFactory xArm6 manipulator show that our neural CDF barrier formulation enables efficient planning and robust safe control in cluttered and dynamic environments, relying only on onboard point-cloud observations.

Note to Practitioners—Robotic manipulators operating in cluttered dynamic environments, such as warehouses or assembly lines, face challenges in ensuring safe and efficient real-time motion. Traditional methods often rely on detailed models of the environment and frequent collision checks, which are impractical when using onboard sensors and when obstacles move unpredictably. This paper introduces a new framework that integrates motion planning and control using learned configuration-space distance functions (CDFs) to identify local safe regions, or “bubbles.” These bubbles substantially reduce the number of collision checks during planning, reducing computation time. For real-time control, the method employs a robust safety filter that explicitly accounts for errors in the learned CDF model and sensor measurements, maintaining safe operation even with dynamic obstacles. Our approach is demonstrated on a UFactory xArm6 manipulator and can be extended to other robot manipulators operating in dynamic environments. We hope this approach improves productivity and safety in industrial settings where accurate environment models are unavailable.

I. INTRODUCTION

Efficient planning and safe control for high-degree-of-freedom (DoF) robotic manipulators are a central challenge in robotics, particularly when operating in cluttered and dynamic environments with noisy sensor measurements. Sampling-based motion planners, such as rapidly-exploring random trees (RRT) [1, 2] and probabilistic roadmaps (PRM) [3], have gained widespread adoption for manipulator motion planning because they effectively explore high-dimensional spaces without exhaustive configuration-space discretization. Despite their

We gratefully acknowledge support from NSF CCF-2112665 (TILOS) and from the Ministry of Trade, Industry and Energy (MOTIE), Korea, under the Strategic Technology Development Program supervised by the Korea Institute for Advancement of Technology (KIAT) [Grant No. P0026052].

¹The authors are with Contextual Robotics Institute, University of California San Diego, La Jolla, CA 92093, USA.

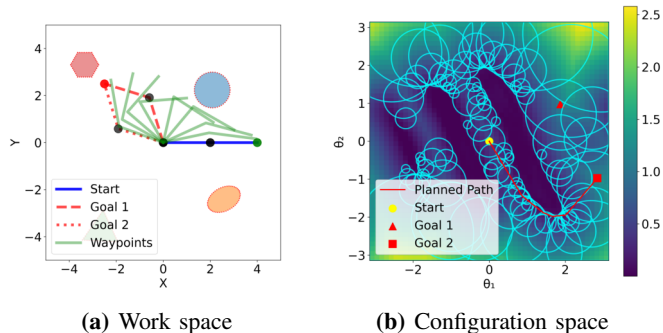


Fig. 1: Illustration of sampling-based planning using configuration-space spheres to represent collision-free space for a planar 2-link robot arm. (a) The robot’s initial configuration (solid blue lines) and two potential goal configurations (dashed red). Four obstacles are present, represented as point-cloud data. Intermediate waypoints from the planned trajectory appear in light green. (b) The 2D configuration space (θ_1 vs. θ_2) of the 2-link robot, where the color map indicates the learned configuration-space distance function (CDF) value (darker regions denote proximity to collision). Cyan circles represent safe bubbles derived from the CDF, forming a graph of collision-free regions. A smooth, optimized trajectory (red) connects the start configuration (yellow circle) to the closest goal (red square).

strengths, these methods rely on frequent collision checks, making the efficiency and accuracy of collision checking a critical bottleneck. Moreover, in highly dynamic settings with sensor noise and rapidly changing obstacles, planning alone is insufficient to ensure safety during execution, necessitating real-time safety-aware control strategies.

To address these challenges, signed distance functions (SDFs) have gained popularity for continuously encoding the distance from any point in the workspace to the boundary of obstacles, thereby enabling precise collision checks and facilitating gradient-based optimization. Recently, neural networks have been leveraged to learn SDF representations of both environments [4–6] and robot bodies [7–9], providing rich shape modeling along with efficient distance and gradient queries. Building on these advancements, neural SDFs have been adopted in motion planning [9–11] and optimization-based safe control [6, 12] for mobile robots, yielding promising results in complex, dynamic scenarios.

To extend the benefits of SDF to high-DoF manipulators, recent work [13] introduced a *configuration-space distance function* (CDF) that directly represents distances in configuration space. In a CDF, the distance is measured in joint space between a given configuration and a workspace point, eliminating the need for repeated transformations between task and configuration spaces. Like neural SDFs, neural CDFs offer a compact and differentiable representation, supporting efficient gradient-based motion planning [13] and reactive control

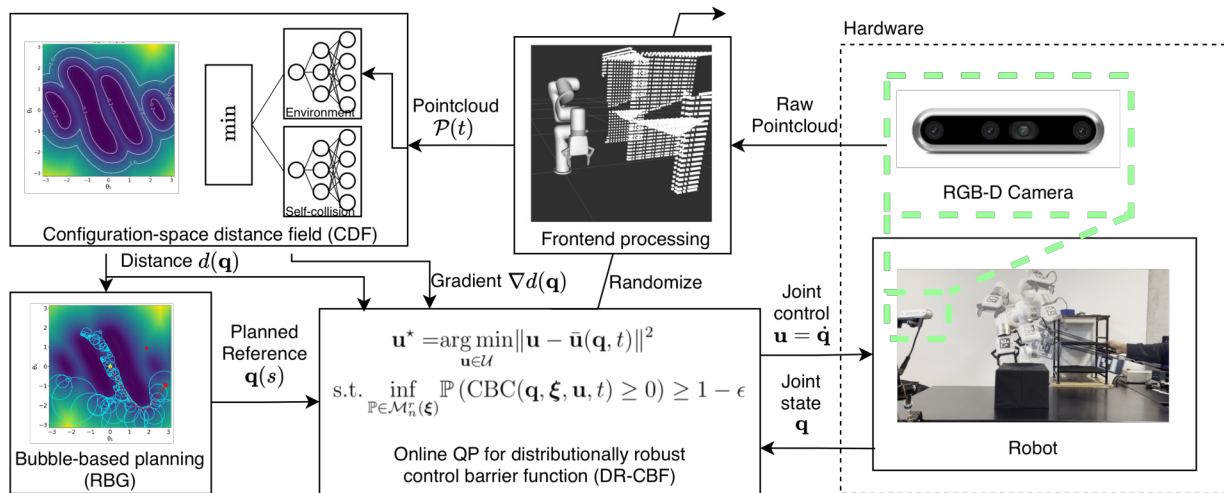


Fig. 2: Overview: Two multi-layer perceptron networks, corresponding to the environment and robot self collisions, are trained to approximate the configuration-space distance function (CDF) of a robot manipulator. A sampling-based planner uses CDF values to form collision-free spheres (bubbles) organized in a Rapidly-exploring Bubble Graph (RBG). A controller imposes safety constraints via a distributionally-robust control barrier function obtained from the CDF and solves quadratic programs online to track the planned trajectory safely. The controller considers all distributions within a Wasserstein-1 ball of sampled environment and robot states obtained by randomizing the front-end CDF estimation model.

[14]. However, as learning-based representations inevitably introduce modeling errors, real-world operations must also contend with sensor noise and other uncertainties.

Motivated by these challenges, we propose an approach that integrates a neural CDF barrier into both motion planning and control, addressing the twin requirements of computational efficiency and safety under uncertainty. The **contributions** of this work are summarized as follows.

- We propose a unified neural configuration-space distance function that estimates distances to workspace obstacles and to self-collision, providing fast distance and gradient queries for manipulation from point-cloud observations.
- Using the neural CDF, we formulate a distributionally robust control barrier function (CBF) for safe manipulator control that accommodates multiple sources of uncertainty, including neural CDF modeling errors, point-cloud observation noise, and dynamic obstacle motion.
- We develop a sampling-based planner that builds a graph of safe configuration-space bubbles using neural CDF queries, reducing explicit collision checking and improving planning efficiency.
- We validate our integrated planning-and-control approach in both simulation and real-robot experiments with a 6-DoF UFactory xArm manipulator operating in dynamic and cluttered environments.

II. RELATED WORK

This section reviews prior work on *motion planning* and *safe control* for robotic manipulators.

Manipulator planning. Sampling-based motion planning algorithms, such as Rapidly-exploring Random Trees (RRT) [1], Probabilistic Roadmaps (PRM) [3], and their variants [15, 16], are widely used for generating global trajectories in high-dimensional configuration spaces. These methods have been

successful in cluttered or unknown environments and are often paired with low-level feedback controllers, such as PID [17], to track the computed paths. However, these planners suffer from high collision-checking overhead and lack real-time adaptability, making them less effective in dynamic or uncertain environments where frequent replanning is needed.

A long-standing challenge in motion planning is how to represent obstacles in the configuration space (*C-space*), which is crucial for generating safe and efficient trajectories. One class of approaches focuses on explicitly approximating the collision-free subset of C-space. Decomposition-based methods such as IRIS [18], and more recent works [19, 20], construct a union of convex polyhedra to approximate the free space by solving a series of optimization problems, typically assuming that workspace obstacles are simple geometric primitives (e.g., spheres and cylinders). [21] proposed D-Fastron, which uses k-means clustering in forward kinematics space to decompose the configuration space and trains local kernel perceptron models for fast proxy collision checking. Alternative approaches, such as [22, 23], utilize workspace distances to construct safe regions; however, these methods are restricted to simple manipulator geometries, such as infinitely thin lines. Additionally, safe spherical regions have been explored for efficient collision checking [24, 25], biased sampling in free workspace representations [26], and navigation function design [27]. Another relevant line of work composes invariant funnels of low-level controllers [28, 29] to plan a safe sequence of controller switches to a desired goal. Our approach differs by using spherical regions to construct a rapidly exploring random graph in configuration space, extending the idea introduced by [30] to high-dimensional robot configurations and more complex environments. Compared to algorithms that rely on safe regions with explicit collision checking or iterative inflation, our bubbles are constructed using fast batched neural CDF distance queries, reducing

expensive collision-check calls. Compared to trajectory optimization over general convex sets that often requires mixed-integer global planning [19, 20], our approach provides free-space approximation via a union of spheres constraints that support sampling-based planning with fewer collision checks or convex optimization of feasible global trajectories. The planned trajectory is executed by an optimization-based safe tracking controller that enforces the safety constraint online. In addition to simplicity, this decoupling allows accounting for uncertainties (e.g., in obstacle sensing and neural models) online during execution, avoiding the conservatism required to account for all uncertainties offline during planning.

More recently, neural representations have emerged to directly model the geometry of the manipulator and its environment. Signed distance fields (SDFs) are used to encode collision information in the workspace [7–9, 31], and have been integrated into reactive planning frameworks [32] and receding-horizon planners [33]. While SDFs offer differentiability, they operate in the workspace and often require projection to C-space for planning and control. To overcome this limitation, [13] introduced the concept of CDF, which estimates the shortest joint-space distance from a configuration to workspace obstacles using a learned model, which we build on. A recent concurrent paper [34] also uses a neural CDF to obtain safe regions in configuration space from a workspace represented as a collection of spheres. We consider a more practical setting where the workspace is partially observed with an RGB-D sensor, and augment the planning module with a distributionally-robust controller that avoids collisions using online sensor measurements. These aspects enable the demonstration of our method on a real robot.

Manipulator safe control. Safe and efficient execution of motion plans requires controllers capable of handling high-dimensional dynamics, enforcing safety constraints, and adapting to changes in the environment. Model predictive control [35] has been widely applied to robotic manipulators due to its ability to incorporate constraints and optimize control actions over a receding horizon [36, 37]. Sampling-based variants such as model predictive path integral control [38] have demonstrated effectiveness in handling non-convex costs and complex dynamics, including for robotic manipulators [32, 39]. In parallel, control barrier functions (CBFs) have emerged as a popular framework for enforcing safety constraints across a range of robotic systems [40–43]. CBFs enable the construction of safety filters that guarantee forward invariance of safe sets by solving a real-time quadratic program (QP), making them computationally tractable and easy to integrate with existing nominal control policies.

CBF-based approaches have increasingly been applied to robotic manipulation tasks. [12] proposed a CBF-based trajectory filtering strategy that dynamically adjusts pre-planned trajectories to account for updated collision constraints. [42] developed a CBF-induced neural controller (CBF-INC) that integrates with RRT for motion planning, utilizing state-based or point-cloud inputs to enhance success rates and reduce exploration nodes. [44] introduced integral feedback into high-gain task-space QPs to ensure provable closed-loop stability under unmodeled dynamics. [14] leveraged neural CDFs as control

barrier functions (CBFs) for safe control, demonstrating that CDF-based methods outperform SDF-based approaches due to their more reliable gradient estimates. [45] proposed a semantic safety filter that integrates scene semantics with CBFs to enforce “common-sense” safety. [46] introduced an operational-space CBF formulation that enforces hundreds of simultaneous safety constraints (e.g., collision, singularity, workspace limits) at kilohertz control rates while preserving task consistency via operational-space hierarchy.

III. PROBLEM FORMULATION

Notation: We use \mathbb{R} , $\mathbb{R}_{\geq 0}$, and \mathbb{N} for the real, nonnegative real, and natural numbers, and $[N] = \{1, \dots, N\}$. For a set $\mathcal{C} \subset \mathbb{R}^n$, $\text{Int}(\mathcal{C})$ and $\partial\mathcal{C}$ denote its interior and boundary. The gradient of a differentiable function V is ∇V . A function α is of class \mathcal{K}_∞ if it is continuous, strictly increasing, and satisfies $\alpha(0) = 0$ and $\alpha(r) \rightarrow \infty$ as $r \rightarrow \infty$. The sets $\text{SO}(p) = \{\mathbf{R} \in \mathbb{R}^{p \times p} \mid \mathbf{R}^\top \mathbf{R} = \mathbf{I}_p, \det(\mathbf{R}) = 1\}$ and $\text{SE}(p)$ with elements $\begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix} \in \mathbb{R}^{(p+1) \times (p+1)}$ where $\mathbf{R} \in \text{SO}(p)$, $\mathbf{t} \in \mathbb{R}^p$ denote the special orthogonal and special Euclidean groups, respectively.

Problem Statement: We consider an m -DoF revolute manipulator with joint configuration $\mathbf{q} = [q_1, \dots, q_m]^\top \in \mathcal{Q}$, where $\mathcal{Q} := \{\mathbf{q} \in \mathbb{R}^m \mid \mathbf{q}_{\min} \leq \mathbf{q} \leq \mathbf{q}_{\max}\}$ denotes the joint-limit-bounded configuration space. The end-effector pose is given by the forward-kinematics map $\mathbf{T}_{ee}(\mathbf{q}) \in \text{SE}(3)$, and the occupied workspace volume of the robot is represented by a set-valued shape function $\mathcal{S}(\mathbf{q}) \subset \mathbb{R}^3$ with boundary $\partial\mathcal{S}(\mathbf{q})$. The robot kinematics are:

$$\dot{\mathbf{q}}(t) = \mathbf{u}(t), \quad \mathbf{q}(0) = \mathbf{q}_0, \quad (1)$$

where $\mathbf{u}(t) \in \mathcal{U} \subset \mathbb{R}^m$ is the joint-velocity input and \mathbf{q}_0 is the initial configuration. The robot operates in a dynamic environment with obstacles $\mathcal{O}(t) \subset \mathbb{R}^3$ and free space $\mathcal{F}(t) = \mathbb{R}^3 \setminus \mathcal{O}(t)$. A sensor (e.g., depth camera or LiDAR) provides point-cloud measurements $\mathcal{P}(t) \subset \mathbb{R}^3$ approximating $\partial\mathcal{O}(t)$, and the sensor-to-base extrinsic calibration is assumed known.

Problem 1. Consider a robot manipulator with initial configuration \mathbf{q}_0 and dynamics in (1). Given point-cloud observations $\mathcal{P}(t)$ of the obstacles $\mathcal{O}(t)$ in the environment, design a control policy that drives the robot’s end-effector $\mathbf{T}_{ee}(\mathbf{q}(t))$ to a desired goal pose $\mathbf{T}_G \in \text{SE}(3)$, while ensuring the robot body remains in free space, i.e., $\mathcal{S}(\mathbf{q}(t)) \subset \mathcal{F}(t)$, $\forall t \geq 0$.

IV. SAFE BARRIER FROM CONFIGURATION-SPACE DISTANCE FUNCTION

Efficient motion planning and control for robotic manipulators require scalable and precise safety representations in configuration space. This section reviews signed distance functions (SDFs), introduces environment and self-collision CDFs and their key properties, and concludes with the definition of a neural CDF barrier for planning and control.

A. Environment Configuration Space Distance Function

Motivated by successful applications in safe mobile robot navigation [6, 41, 47], SDFs have recently been used for motion planning and control of robot manipulators [7, 8, 42, 48]. An SDF f_s measures the distance from a workspace point $\mathbf{p} \in \mathbb{R}^3$ to the robot surface $\partial\mathcal{S}(\mathbf{q})$:

$$f_s(\mathbf{p}, \mathbf{q}) = \begin{cases} -\min_{\mathbf{p}^* \in \partial\mathcal{S}(\mathbf{q})} \|\mathbf{p} - \mathbf{p}^*\|, & \text{if } \mathbf{p} \in \mathcal{S}(\mathbf{q}), \\ \min_{\mathbf{p}^* \in \partial\mathcal{S}(\mathbf{q})} \|\mathbf{p} - \mathbf{p}^*\|, & \text{if } \mathbf{p} \notin \mathcal{S}(\mathbf{q}). \end{cases} \quad (2)$$

When represented using a differentiable model (e.g., neural network [4, 7]), this function enables efficient motion planning and collision avoidance by facilitating rapid computation of distances and gradients. While SDFs are widely applied to mobile robots to enforce workspace safety constraints, manipulators with high-dimensional joint spaces benefit from a configuration-centric perspective. [13] introduced a configuration distance function (CDF), which encodes the minimum joint-space distance (in radians) needed for a robot at configuration \mathbf{q} to make contact with a point \mathbf{p} . Formally, given a robot SDF f_s , the robot CDF can be computed as

$$f_c(\mathbf{p}, \mathbf{q}) = \min_{\mathbf{q}'} \|\mathbf{q} - \mathbf{q}'\|, \quad \text{subject to } f_s(\mathbf{p}, \mathbf{q}') = 0. \quad (3)$$

Similar to an SDF, this CDF representation satisfies an Eikonal equation with respect to \mathbf{q} :

$$\|\nabla_{\mathbf{q}} f_c(\mathbf{p}, \mathbf{q})\| = 1, \quad (4)$$

whenever it is differentiable in \mathbf{q} .

In practice, computing (3) requires identifying a set of zero-level configurations \mathbf{q}' satisfying $f_s(\mathbf{p}, \mathbf{q}') = 0$. These configurations are typically obtained using numerical optimization methods such as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [49]. However, due to the high dimensionality of the configuration space and the sparsity of \mathbf{q}' samples, this direct formulation can lead to an overly smooth or inaccurate approximation of the true CDF. Therefore, [13] leverages the fact that the contact at a point \mathbf{p} is primarily determined by a subset of joints preceding the contact link. Let k denote the robot link that comes into contact with \mathbf{p} , and let $\mathbf{q}_{:k}$ represent the joint angles influencing the motion of link k . The CDF is then refined by restricting the distance computation to these relevant joints:

$$f_c(\mathbf{p}, \mathbf{q}) = \min_{k=1, \dots, m} \min_{\mathbf{q}'} \|\mathbf{q}_{:k} - \mathbf{q}'_{:k}\|, \quad \text{s.t. } f_s(\mathbf{p}, \mathbf{q}') = 0, \quad (5)$$

where m is the total number of robot joints.

B. Self-Collision Configuration Space Distance Function

While environment CDFs measure proximity to external obstacles, self-collisions between different parts of the robot pose a distinct challenge, especially for high-DoF manipulators with complex self-collisions. To address this, we define a *Self-Collision Configuration Distance Function* (SCDF), which quantifies the minimum joint-space distance from a configuration \mathbf{q} to the set of self-colliding configurations.

Let $\mathcal{C}_{sc} \subset \mathcal{Q}$ denote the closed set of joint configurations that result in self-collision. Then, for a given configuration $\mathbf{q} \in \mathcal{Q}$, we define the SCDF as:

$$f_{sc}(\mathbf{q}) := \min_{\mathbf{q}' \in \mathcal{C}_{sc}} \|\mathbf{q} - \mathbf{q}'\|. \quad (6)$$

Similar to Section IV-A, the SCDF in (6) represents the Euclidean distance from \mathbf{q} to the closed set of self-colliding configurations $\mathcal{C}_{sc} \subset \mathbb{R}^m$. Note that \mathcal{C}_{sc} may consist of multiple disconnected components, as different joint subsets can lead to distinct types of self-collision. Nevertheless, the SCDF inherits the Eikonal property and satisfies

$$\|\nabla_{\mathbf{q}} f_{sc}(\mathbf{q})\| = 1. \quad (7)$$

In practice, \mathcal{C}_{sc} is often only sparsely sampled, and self-collisions typically involve only a subset of joints. As such, computing the full-joint distance in (6) can lead to overly conservative and inaccurate approximations. To improve accuracy and better exploit the robot's kinematic structure, we refine the SCDF by computing distance only over the subset of joints responsible for the self-collision.

Specifically, if a self-collision at \mathbf{q}' occurs between Link i and Link j , we define the joint subvector $\mathbf{q}_{a:b}$, where $a = \min(i, j)$ and $b = \max(i, j)$, to include only joints that affect the relative motion of these links. The refined SCDF becomes:

$$f_{sc}(\mathbf{q}) := \min_{\mathbf{q}' \in \mathcal{C}_{sc}} \|\mathbf{q}_{a:b} - \mathbf{q}'_{a:b}\|. \quad (8)$$

For example, if the collision occurs between Link 3 and Link 5, then joints 3 through 5 may primarily determine their relative positions, so the distance is computed using $\mathbf{q}_{3:5}$.

C. Neural CDF Barrier

Building on the environment and self-collision CDF, we define a CDF barrier to ensure the robot remains in a collision-free region of the configuration space. Formally, a CDF barrier associated with environment CDF f_c , self-collision CDF f_{sc} , and obstacle set $\mathcal{O}(t)$ is defined as:

$$h(\mathbf{q}, t) := \min\left\{\inf_{\mathbf{p} \in \partial\mathcal{O}(t)} f_c(\mathbf{p}, \mathbf{q}), f_{sc}(\mathbf{q})\right\} \quad (9)$$

where $\partial\mathcal{O}(t)$ is the obstacle set boundary. Then, the time-varying *safe set* in configuration space induced by $h(\mathbf{q}, t)$ is

$$\mathcal{C}_{safe}(t) = \{\mathbf{q} \in \mathcal{Q} \mid h(\mathbf{q}, t) \geq 0\}. \quad (10)$$

In real-world settings, $\partial\mathcal{O}(t)$ must be inferred from point-cloud measurements $\mathcal{P}(t)$ from a depth camera or LiDAR. Because the points in $\mathcal{P}(t)$ are noisy samples from the boundary of $\mathcal{O}(t)$, we approximate the CDF barrier as

$$h(\mathbf{q}, t) \approx \min\left\{\min_{\mathbf{p} \in \mathcal{P}(t)} f_c(\mathbf{p}, \mathbf{q}), f_{sc}(\mathbf{q})\right\}. \quad (11)$$

Moreover, computing $f_c(\mathbf{p}, \mathbf{q})$ and $f_{sc}(\mathbf{q})$ exactly is difficult for high-DoF manipulators because it involves (infinitely) many potential contact configurations \mathbf{q}' in (3) and self-collision configurations in (8). Similar to [13], we model f_c and f_{sc} by multi-layer perceptrons $\hat{f}_c(\mathbf{p}, \mathbf{q}; \boldsymbol{\theta}_1)$ and $\hat{f}_{sc}(\mathbf{p}, \mathbf{q}; \boldsymbol{\theta}_2)$ with learnable parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, respectively. This representation reduces storage requirements

compared to volumetric or tabular encodings and supports parallel distance or gradient queries for a set of points \mathbf{p} or configurations \mathbf{q} . Combining the point-cloud measurements with the neural CDF model yields a practical CDF barrier:

$$\hat{h}(\mathbf{q}, t; \boldsymbol{\theta}) = \min\left\{\min_{\mathbf{p} \in \mathcal{P}(t)} \hat{f}_c(\mathbf{p}, \mathbf{q}; \boldsymbol{\theta}_1), \hat{f}_{sc}(\mathbf{q}; \boldsymbol{\theta}_2)\right\}. \quad (12)$$

However, learning-based representations inevitably introduce modeling errors. The noise in $\mathcal{P}(t)$, its lack of complete coverage of the obstacle boundary, and the approximation error in \hat{f}_c and \hat{f}_{sc} constitute multiple sources of uncertainty.

In what follows, we leverage the fast parallel query capabilities of the neural CDF barrier to enable efficient sampling-based planning in configuration space (Sec. V), and propose a distributionally robust control strategy that tracks the planned trajectory while ensuring real-time safety despite the aforementioned uncertainties and motion in obstacles (Sec. VI).

V. CONFIGURATION SPACE BUBBLE-BASED PLANNING

Motion planning for high-DoF manipulators typically involves numerous collision checks [16], making global motion planning computationally expensive. To address this, we propose the **bubble-CDF planner**, a sampling-based approach that efficiently explores the configuration space while ensuring safety via neural CDF barriers. Unlike conventional RRT-based planners that rely on dense edge validation, our approach constructs *configuration space bubbles* as local safe regions, enabling rapid exploration and collision-free path generation.

In Problem 1, the robot must move from an initial configuration \mathbf{q}_0 to any valid joint configuration \mathbf{q}_G that achieves a desired end-effector pose $\mathbf{T}_{ee}(\mathbf{q}_G) = \mathbf{T}_G \in \text{SE}(3)$. Inverse kinematics (IK) provides K goal configurations $\{\mathbf{q}_G^i\}_{i \in [K]}$, each of which achieves the desired end-effector pose.

Our approach builds on the *Rapidly-exploring Bubble Graph (RBG)* algorithm [30], which constructs a configuration-space roadmap where each node represents a locally safe region (bubble) instead of a single configuration. The learned CDF barrier $\hat{h}(\mathbf{q}, t)$ is used to define the bubble radius dynamically, ensuring rapid, collision-free exploration without incremental edge validation. This substantially reduces the collision-checking overhead in high-dimensional spaces while preserving the efficiency of sampling-based planners.

A. Rapidly-exploring Bubble Graph (RBG)

Suppose $\mathbf{q} \in \mathcal{Q}$ is a safe configuration. We aim to identify a radius $r(\mathbf{q}) \geq 0$ such that any configuration \mathbf{q}' satisfying $\|\mathbf{q} - \mathbf{q}'\| \leq r(\mathbf{q})$ is safe. We call this spherical region a *configuration-space bubble* and denote it as $\mathcal{B}(\mathbf{q}, r(\mathbf{q}))$. Given the neural CDF barrier $\hat{h}(\mathbf{q}, t)$ for fixed time t , we construct the bubble radius at configuration \mathbf{q} as:

$$r(\mathbf{q}) = \hat{h}(\mathbf{q}, t) - \eta, \quad (13)$$

where $\eta > 0$ is a safety margin. All configurations \mathbf{q}' within this radius satisfy the neural CDF barrier with margin $h(\mathbf{q}', t) \geq \eta$ [30].

To explore a high-dimensional configuration space efficiently using these local safety certificates, RBG adapts the

Algorithm 1 Rapidly Exploring Bubble Graph

Require: Start configuration \mathbf{q}_0 , goal configurations $\{\mathbf{q}_G^i\}_{i=1}^K$, Neural CDF barrier $\hat{h}(\mathbf{q}, t)$, safety margin η , Max. no. of bubbles N_{\max} , Min. radius r_{\min} .

Ensure: Graph \mathcal{G} containing bubbles from start to goals

- 1: **function** BUILDBUBBLEGRAPH($\mathbf{q}_0, \{\mathbf{q}_G^i\}$)
- 2: $\mathcal{G}.V \leftarrow \{\mathcal{B}(\mathbf{q}_0, \hat{h}(\mathbf{q}_0, t) - \eta)\}, \quad \mathcal{G}.E \leftarrow \emptyset$
- 3: **while** $|\mathcal{G}.V| < N_{\max}$ **do**
- 4: $\mathbf{q}_{\text{rand}} \leftarrow \text{SAMPLERANDOM}()$ ▷ With goal bias
- 5: $\mathcal{B}_{\text{near}} \leftarrow \text{NEARESTBUBBLE}(\mathcal{G}, \mathbf{q}_{\text{rand}})$
- 6: **if** \mathbf{q}_{rand} **outside** $\mathcal{B}_{\text{near}}$ **then**
- 7: $\mathbf{q}_{\text{new}} \leftarrow \text{EXTENDTOWARD}(\mathcal{B}_{\text{near}}, \mathbf{q}_{\text{rand}})$
- 8: $r_{\text{new}} \leftarrow \hat{h}(\mathbf{q}_{\text{new}}, t) - \eta$
- 9: **if** $r_{\text{new}} > r_{\min}$ **then**
- 10: $\mathcal{B}_{\text{new}} \leftarrow \mathcal{B}(\mathbf{q}_{\text{new}}, r_{\text{new}})$
- 11: UPDATECONNECTIONS($\mathcal{G}, \mathcal{B}_{\text{new}}$)
- 12: **return** \mathcal{G}
- 13: **function** UPDATECONNECTIONS($\mathcal{G}, \mathcal{B}_{\text{new}}$)
- 14: **for** $\mathcal{B}_i \in \mathcal{G}.V$ **do**
- 15: **if** $\|\mathcal{B}_{\text{new}} \cdot \mathbf{q} - \mathcal{B}_i \cdot \mathbf{q}\| \leq r(\mathbf{q}_{\text{new}}) + r(\mathbf{q}_i)$ **then**
- 16: $\mathcal{G}.E \leftarrow \mathcal{G}.E \cup (\mathcal{B}_{\text{new}}, \mathcal{B}_i)$
- 17: $\mathcal{G}.V \leftarrow \mathcal{G}.V \cup \{\mathcal{B}_{\text{new}}\}$

RRT algorithm [1] so that each vertex in the graph corresponds to a bubble rather than a single configuration. In RBG, the step size, corresponding to the distance the planned tree advances toward a sampled point, is dynamically set by the radius of each bubble. This obviates manual tuning of a fixed step size and reduces collision-checking overhead.

Algorithm 1 details how RBG constructs a bubble-based graph. RBG iteratively samples random configurations (biased toward goal configurations), identifies the nearest bubble in configuration space, and if the sample lies outside that bubble, creates a new bubble by querying the neural CDF barrier to determine its radius. The newly created bubble is then connected to any existing bubbles $\mathcal{B}(\mathbf{q}_i, r(\mathbf{q}_i))$ with overlapping safe regions, i.e., if

$$\|\mathbf{q}_{\text{new}} - \mathbf{q}_i\| \leq r(\mathbf{q}_{\text{new}}) + r(\mathbf{q}_i). \quad (14)$$

This process continues until the maximum number of bubbles N_{\max} or a goal configuration is reached.

B. Path Selection and Trajectory Optimization

Once the bubble graph \mathcal{G} is constructed, we must extract and refine a collision-free path connecting the start configuration \mathbf{q}_0 to one of the K goal configurations $\{\mathbf{q}_G^i\}$. Since \mathcal{G} may contain multiple feasible routes, we first select an optimal path according to a desired criterion (e.g., minimum total edge distance). We then solve a local trajectory optimization problem, fitting a continuous curve through the sequence of bubbles to ensure a smooth, dynamically feasible trajectory that remains collision-free.

Path Selection. We assign a cost to each edge in \mathcal{G} , for instance, the single-sided Hausdorff distance between two overlapping bubbles [30]. A shortest-path search (e.g., Dijkstra [50] or A* [51]) is run from the start bubble to every goal bubble. The path with the lowest overall cost is then passed to the optimization step, as summarized in Algorithm 2.

Trajectory Optimization. To obtain a smooth trajectory, we optimize a sequence of Bézier curves connecting the start configuration \mathbf{q}_0 to the selected goal configuration \mathbf{q}_G^* through the path of bubbles $\{\mathcal{B}_1, \dots, \mathcal{B}_n\}$. Each Bézier curve segment of degree d is defined by its control points:

$$\gamma_j(t) = \sum_{l=0}^d \binom{d}{l} t^l (1-t)^{d-l} \mathbf{c}_{j,l}, \quad (15)$$

where $\mathbf{c}_{j,l} \in \mathbb{R}^m$ is the l -th control point of the j -th curve segment. The k -th derivative of a Bézier curve is also a Bézier curve, and its squared norm integral can be expressed as:

$$\int_0^1 \|\gamma_j^{(k)}(t)\|^2 dt = \mathbf{c}_j^\top \mathbf{Q}_k \mathbf{c}_j, \quad (16)$$

where \mathbf{c}_j is the vectorized form of all control points in segment j , and \mathbf{Q}_k is a positive semidefinite matrix. This leads to the following convex quadratically constrained quadratic program:

$$\begin{aligned} \min_{\{\mathbf{c}_{j,i}\}} \quad & \sum_{j=1}^n \sum_{k=1}^K w_k \int_0^1 \|\gamma_j^{(k)}(t)\|^2 dt \\ \text{subject to} \quad & \|\mathbf{c}_{j,l} - \mathbf{q}_j\|^2 \leq r(\mathbf{q}_j)^2, \quad \forall j, l \\ & \gamma_j(1) = \gamma_{j+1}(0), \quad j = 1, \dots, n-1 \\ & \gamma_j^{(k)}(1) = \gamma_{j+1}^{(k)}(0), \quad j = 1, \dots, n-1, k = 1, 2 \\ & \gamma_1(0) = \mathbf{q}_0, \quad \gamma_n(1) = \mathbf{q}_G^*, \\ & \gamma_1^{(i)}(0) = \mathbf{0}, \quad \gamma_n^{(i)}(1) = \mathbf{0}, \quad i = 1, \dots, d \end{aligned} \quad (17)$$

where $\mathbf{c}_{j,i}$ are the control points of the j -th Bézier curve segment γ_j of degree d , and $\gamma_j^{(k)}$ denotes its k -th derivative. The objective minimizes a weighted sum of integrated squared derivatives, promoting smoothness. The first constraint ensures safety by keeping control points within their respective bubbles, which, by the convex hull property of Bézier curves, guarantees the entire trajectory remains collision-free. The remaining constraints enforce continuity of position and derivatives between segments, as well as boundary conditions including zero velocity at endpoints. Unlike traditional sampling-based planners that may not have safety guarantees or require additional collision checks for a post-optimized trajectory, our formulation provides a convex optimization problem that simultaneously guarantees smoothness and safety without requiring additional collision checking.

Throughout this section, we have introduced the **bubble-CDF planner**, which leverages neural CDF barriers to efficiently explore the configuration space while ensuring safety. By constructing *configuration-space bubbles* around safe configurations and dynamically adjusting step sizes based on bubble radii, our approach significantly reduces collision-checking overhead. Additionally, the trajectory optimization formulation in (17) guarantees smooth, dynamically feasible paths without requiring post-processing for collision checks.

Thus far, we have described the bubble-CDF planner in a nominal setting, where the neural CDF barrier $\hat{h}(\mathbf{q}, t)$ is treated as time-invariant and accurate. In real deployments, however, dynamic obstacles, partial/noisy point-cloud observations, CDF approximation errors, and sensor noise may cause the bubble radius to be locally overestimated, leading

Algorithm 2 Path Selection and Trajectory Optimization

Require:

- 1: Graph \mathcal{G} with start bubble $\mathcal{B}_{\text{start}}$, goal bubbles $\{\mathcal{B}_{\text{goal}}^i\}_i$
- 2: Edge cost function $\text{EDGE_COST}(\mathcal{B}_i, \mathcal{B}_j)$
- 3: Weights $\{w_k\}$ for trajectory cost

Ensure: Smooth collision-free Bézier trajectory from \mathbf{q}_0 to some \mathbf{q}_G^i

```

4: function PATHSELECTANDOPTIMIZE( $\mathcal{G}, \mathcal{B}_{\text{start}}, \{\mathcal{B}_{\text{goal}}^i\}$ )
5:    $bestCost \leftarrow \infty, \quad bestPath \leftarrow None, \quad i_{\text{best}} \leftarrow -1$ 
6:   for  $i = 1 \rightarrow K$  do
7:      $path_i \leftarrow \text{SHORTESTPATH}(\mathcal{G}, \mathcal{B}_{\text{start}}, \mathcal{B}_{\text{goal}}^i, \text{EDGE\_COST})$ 
8:     if  $path_i$  exists then
9:        $cost_i \leftarrow \text{PATHCOST}(path_i, \text{EDGE\_COST})$ 
10:      if  $cost_i < bestCost$  then
11:         $bestCost \leftarrow cost_i, \quad bestPath \leftarrow path_i, \quad i_{\text{best}} \leftarrow i$ 
12:   if  $bestPath = None$  then
13:     return failure ▷ No feasible path found
14:   curves  $\leftarrow \text{BEZIEROPT1}(bestPath, \{w_k\})$  ▷ Eq. (17)
15:   return (curves,  $i_{\text{best}}$ )

```

to a nominally collision-free plan under \hat{h} that is unsafe in the true environment. To address these uncertainties, Sec. VI introduces a distributionally robust control barrier function that accounts for potential CDF errors and planning over-approximations, and enforces safety at execution time by adjusting the nominal trajectory when needed.

VI. DISTRIBUTIONALLY ROBUST MANIPULATION CONTROL

In this section, we present a control synthesis approach that guarantees safe real-time configuration trajectory tracking using neural CDF barrier estimates subject to sensing and modeling errors, and even dynamic obstacles. We first describe our approach to track the reference trajectory (Sec. VI-A), and incorporate a distributionally robust CBF (DR-CBF) [40] to ensure safe control synthesis despite dynamic obstacles and neural CDF barrier uncertainty (Sec. VI-B).

A. Governor-based Trajectory Tracking

We first present our approach for tracking the planned piecewise Bézier configuration trajectory $\gamma : [0, 1] \rightarrow \mathcal{Q}$ from Sec. V-B, in the absence of safety constraints. Inspired by [52, 53], we design a reference governor to determine a time-varying reference configuration $\gamma(s)$ along the trajectory, where $s \in [0, 1]$ is a progress variable. In other words, $\gamma(0) = \mathbf{q}_0$ is the initial robot configuration, while $\gamma(1) = \mathbf{q}_G$ is one of the goal configurations. The governor system adapts the value of the progress variable s based on the manipulator's current configuration to ensure smooth progress along the trajectory. We then stabilize the manipulator configuration \mathbf{q} to $\gamma(s)$ using a proportional-derivative (PD) controller.

The dynamics of s are designed to balance the robot's tracking capabilities with smooth progression along the trajectory:

$$\dot{s} = \frac{k}{1 + \|\mathbf{q} - \gamma(s)\|} (1 - s^\zeta), \quad (18)$$

where $k > 0$ is a gain parameter, $\zeta \geq 1$ controls the rate of saturation as s approaches 1, and $\|\mathbf{q} - \gamma(s)\|$ is the Euclidean distance between the current and reference configuration. This

adaptive mechanism accelerates s when the manipulator is near the reference and decelerates it when farther away, ensuring smooth progression without abrupt jumps.

To track the reference trajectory γ , we use a PD controller as the nominal controller to reach the local goal $\gamma(s(t))$:

$$\bar{\mathbf{u}}(\mathbf{q}, t) = -K_P(\mathbf{q} - \gamma(s(t))) - K_D\dot{\mathbf{q}}, \quad (19)$$

where K_P and K_D are positive definite matrices controlling the proportional and derivative contributions, respectively. The first term ensures that the robot is drawn towards $\gamma(s)$, while the second term provides damping to minimize oscillations.

B. Distributionally Robust Neural CDF Barrier

To ensure safe operation while tracking the planned trajectory, the robot's configuration \mathbf{q} must remain within the safe set $\mathcal{C}_{\text{safe}}(t)$ as defined in (10). We utilize the CDF barrier as control barrier functions (CBFs) [54] by considering its time derivative. This leads to the following time-varying control barrier condition (CBC),

$$\begin{aligned} \text{CBC}(\mathbf{q}, \mathbf{u}, t) &:= \nabla_{\mathbf{q}} h(\mathbf{q}, t)^\top \mathbf{u} + \frac{\partial h(\mathbf{q}, t)}{\partial t} + \alpha_h(h(\mathbf{q}, t)) \quad (20) \\ &= \underbrace{[\nabla_{\mathbf{q}} h(\mathbf{q}, t)^\top, \alpha_h(h(\mathbf{q}, t)), \frac{\partial h(\mathbf{q}, t)}{\partial t}]^\top}_{\boldsymbol{\xi}^\top(\mathbf{q}, t)} \begin{bmatrix} \mathbf{u} \\ 1 \\ 1 \end{bmatrix}, \end{aligned}$$

where α_h is a class \mathcal{K}_∞ function.

With the nominal PD control law $\bar{\mathbf{u}}(\mathbf{q}, t)$ in (19), one can enforce safety by solving CBF quadratic program (QP) [55] that generates the safe manipulator velocity control:

$$\begin{aligned} \mathbf{u}^* &= \arg \min_{\mathbf{u} \in \mathcal{U}} \|\mathbf{u} - \bar{\mathbf{u}}(\mathbf{q}, t)\|^2 \\ \text{s.t. } &\text{CBC}(\mathbf{q}, \mathbf{u}, t) \geq 0, \end{aligned} \quad (21)$$

where $\mathcal{U} := \{\mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max}\}$ defines the admissible control space, ensuring the commanded manipulator velocities remain within feasible operational limits. However, the CDF barrier $h(\mathbf{q}, t)$ and its gradients are not known exactly but rather are approximated through the neural CDF barrier $\hat{h}(\mathbf{q}, t)$ in (12). Due to uncertainties introduced by the neural network model and the point-cloud data, \hat{h} cannot be treated as an exact representation of h . Instead, we work with samples of the associated uncertainty vector $\{\boldsymbol{\xi}_i\}_{i \in [N]}$, defined in (20), which capture the variations in \hat{h} , its gradient, and its time derivative. The practical approach of obtaining such samples is discussed in Sec. VI-C.

Drawing inspiration from recent advances in distributionally robust optimization (DRO) [40, 56], we reformulate the CBC in a distributionally robust manner:

$$\inf_{\mathbb{P} \in \mathcal{M}_N^r} \mathbb{P}(\text{CBC}(\mathbf{q}, \mathbf{u}, \boldsymbol{\xi}) \geq 0) \geq 1 - \epsilon, \quad (22)$$

where \mathcal{M}_N^r is a Wasserstein ambiguity set constructed from $\{\boldsymbol{\xi}_i\}_{i=1}^N$ and ϵ is a user-specified risk tolerance. To formally define \mathcal{M}_N^r , let $\Xi \subseteq \mathbb{R}^{m+2}$ denote the space of uncertainty vectors $\boldsymbol{\xi}$, and let $\mathcal{P}_p(\Xi)$ be the space of all Borel probability measures over Ξ with finite p -th moment, by utilizing the p -Wasserstein distance W_p [56], the ambiguity set

$$\mathcal{M}_N^r := \{\mu \in \mathcal{P}_p(\Xi) \mid W_p(\mu, \mathbb{P}_N) \leq r\} \quad (23)$$

defines a ball of distributions around the discrete empirical distribution $\mathbb{P}_N := \frac{1}{N} \sum_{i=1}^N \delta_{\boldsymbol{\xi}_i}$ of the samples. Intuitively, the p -Wasserstein distance measures the minimum amount of probability mass transport needed to transform one distribution into another. Thus, the ambiguity set \mathcal{M}_N^r encodes all distributions that are within a radius r of the empirical distribution, and offers a principled way to model uncertainty in the neural CDF and point-cloud observations through its parameters (radius r and risk level ϵ), without requiring explicit knowledge of the true distribution of $\boldsymbol{\xi}$.

Next, by utilizing the 1-Wasserstein distance and following a convex reformulation of the DR-CBF in [40, Proposition 6.5], lead to the following optimization problem:

$$\begin{aligned} \mathbf{u}^* &= \arg \min_{\mathbf{u} \in \mathcal{U}, s \in \mathbb{R}, \beta_i \in \mathbb{R}} \|\mathbf{u} - \bar{\mathbf{u}}(\mathbf{q}, t)\|^2, \quad (24) \\ \text{s.t. } &r \|\mathbf{u}\|_\infty \leq s\epsilon - \frac{1}{N} \sum_{i=1}^N \beta_i, \\ &\beta_i \geq \max\{s - [\mathbf{u} \ 1 \ 1]^\top \boldsymbol{\xi}_i(\mathbf{q}, t), 0\}, \forall i \in [N]. \end{aligned}$$

The reformulation maintains the linear dependence of the constraints on \mathbf{u} , ensuring that the optimization remains a QP and can be solved efficiently in real time.

As with any CBF QP, feasibility is not guaranteed in all situations. Prior work on distributionally robust safety filtering [57] and on compositions of CBFs under input constraints [58] analyzes conditions under which infeasibility can arise, such as restrictive control limits, adversarial obstacle motion, or overly conservative uncertainty sets. In practice, we choose the Wasserstein radius and the number of uncertainty samples so that the DR-CBF constraint remains compatible with the admissible control bounds, and we did not observe infeasibility in any of our experiments.

Our formulation assumes access to a velocity-control interface, which is standard for many robotic manipulators where a low-level torque controller tracks commanded joint velocities. Although the full manipulator dynamics are second-order, the closed-loop behavior at this interface can be written as:

$$\dot{\mathbf{q}} = \mathbf{u} + \mathbf{d}(\mathbf{q}, \dot{\mathbf{q}}), \quad (25)$$

where $\mathbf{d}(\mathbf{q}, \dot{\mathbf{q}})$ represents bounded discrepancies arising from inertia, Coriolis and centrifugal effects, gravity, friction, and other unmodeled dynamics. These discrepancies enter the barrier derivative as additional uncertain terms and can be incorporated into the uncertainty vector $\boldsymbol{\xi}(\mathbf{q}, t)$ in (20).

Our DR-CBF formulation also extends directly to torque-level control of second-order manipulator dynamics. Defining the state as $\mathbf{x} = (\mathbf{q}, \dot{\mathbf{q}})$ and the input as the joint torque $\mathbf{u} = \boldsymbol{\tau}$ yields the standard control-affine representation:

$$\dot{\mathbf{x}} = f(\mathbf{x}) + G(\mathbf{x}) \mathbf{u}, \quad (26)$$

$$f(\mathbf{x}) = \begin{bmatrix} \dot{\mathbf{q}} \\ -\mathbf{M}(\mathbf{q})^{-1}(\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q})) \end{bmatrix}, \quad G(\mathbf{x}) = \begin{bmatrix} \mathbf{0} \\ \mathbf{M}(\mathbf{q})^{-1} \end{bmatrix}.$$

Our DR-CBF approach can be applied to this second-order model to enforce safety at the torque level.

C. Uncertainty Sample Selection

This section details our approach for obtaining samples $\{\xi_i\}_{i=1}^N$ of the uncertainty vector ξ in the CBC in (20). The definition of the neural CDF barrier $\hat{h}(\mathbf{q}, t)$ in (12) depends on the point-cloud data $\mathcal{P}(t)$, which is represented as $\mathcal{P}(t) = \{\mathbf{p}_j\}_{j=1}^{M_1}$. Each $\mathbf{p}_j \in \mathbb{R}^3$ corresponds to a workspace point captured at time t . The velocity of each point, denoted as $\frac{\partial \mathbf{p}_j}{\partial t}$, is also considered to account for dynamic obstacles.

To account for uncertainty in the neural CDF model $\hat{f}(\mathbf{p}, \mathbf{q})$, we employ Monte Carlo (MC) dropout in the network architecture [59]. MC dropout provides an efficient approximate Bayesian inference method by treating dropout as sampling from an implicit distribution over the network weights. This allows us to perform M_2 stochastic forward passes through the robot CDF neural network, collecting multiple realizations of $\hat{f}_c(\mathbf{p}, \mathbf{q}; \theta)$, along with its gradients. Specifically, for each forward pass of $f_c^{(i)}$, we evaluate $f_c^{(i)}$ across all M_1 points in $\mathcal{P}(t)$. Consequently, we obtain $M_1 \times M_2$ realizations of:

$$\left\{ f_c^{(i)}(\mathbf{p}_j, \mathbf{q}), \nabla_{\mathbf{q}} f_c^{(i)}(\mathbf{p}_j, \mathbf{q}), \nabla_{\mathbf{p}} f_c^{(i)}(\mathbf{p}, \mathbf{q})^\top \frac{\partial \mathbf{p}_j}{\partial t} \right\}, \quad (27)$$

where $i \in [M_2]$ corresponds to the MC dropout realization and $j \in [M_1]$ corresponds to the point in $\mathcal{P}(t)$.

From the $M_1 \times M_2$ realizations, we select $N \ll M_1 \times M_2$ samples in the set (27) based on the N minimal values of

$$\alpha_h(f_c^{(i)}(\mathbf{p}_j, \mathbf{q})) + \nabla_{\mathbf{p}} f_c^{(i)}(\mathbf{p}, \mathbf{q})^\top \frac{\partial \mathbf{p}_j}{\partial t}, \quad (28)$$

which effectively identifies samples $\{\xi_i\}_{i \in [N]}$.

This section introduced a DR-CBF formulation that ensures safe manipulation under uncertainty. Unlike naive safety margin adjustments, our approach explicitly accounts for uncertainties in the neural CDF model, its gradient estimates, and point-cloud velocity observations. By leveraging a Wasserstein ambiguity set, we optimize for the worst-case uncertainty within a data-driven confidence region, providing stronger safety guarantees than standard CBF formulations. This enables adaptive and robust control in dynamic environments without requiring explicit probabilistic models of uncertainty.

VII. EVALUATION

This section evaluates our planning and control techniques, which utilize neural CDF barriers, in simulations on a 2-DoF planar arm and the 6-DoF xArm, as well as in real-world experiments on the 6-DoF xArm. We evaluate the computational efficiency and solution quality of the bubble-CDF planner and the robustness of the DR-CBF controller in ensuring safety under dynamic obstacles and uncertainty.

A. Implementation Details

Training Dataset and Loss Function: We discretize the robot's workspace into a $40 \times 40 \times 40$ grid. For each workspace point, we generate contact configurations using BFGS optimization, and then select 200 diverse configurations via farthest-point sampling in configuration space [13]. During training, we employ an online batching strategy where each batch randomly samples 10 configurations and 50 workspace

TABLE I: Inference time (ms) of Neural CDF barrier on xArm for varying batch sizes of configurations (\mathbf{q}) and workspace points (\mathbf{p}).

$\mathbf{q} \backslash \mathbf{p}$	1	10	100	1000	10000
1	0.272	0.281	0.295	0.291	0.742
10	0.281	0.307	0.297	0.758	6.818
100	0.301	0.301	0.763	7.060	70.215

points. For each configuration–point pair $(\mathbf{p}_i, \mathbf{q}_i)$, the ground-truth environment CDF value $f_c^*(\mathbf{p}_i, \mathbf{q}_i)$ is computed as the minimum joint-space distance to the set of contact configurations associated with \mathbf{p}_i .

The network is trained using the following loss function:

$$\mathcal{L}_{\text{env}} = \frac{1}{N_1 N_2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \left(\hat{f}_c(\mathbf{p}_i, \mathbf{q}_j; \theta_1) - f_c^*(\mathbf{p}_i, \mathbf{q}_j) \right)^2 + \lambda \left(\|\nabla_{\mathbf{q}} \hat{f}_c(\mathbf{p}_i, \mathbf{q}_j; \theta_1)\| - 1 \right)^2, \quad (29)$$

where $\lambda = 0.05$ is a user-specified weight.

For the self-collision CDF (SCDF), we generate training data by uniformly sampling joint configurations and detecting self-collisions using the PyBullet simulation engine [60]. Each self-colliding configuration $\mathbf{q}' \in \mathcal{C}_{\text{sc}}$ is stored along with the minimal joint index range $[a : b]$ responsible for the collision. During training, each batch consists of N randomly sampled configurations $\{\mathbf{q}_j\}_{j=1}^N$. For each configuration, the ground-truth SCDF value $f_{\text{sc}}^*(\mathbf{q}_j)$ is computed according (8).

The SCDF network is trained with the following loss:

$$\mathcal{L}_{\text{sc}} = \frac{1}{N_3} \sum_{j=1}^{N_3} \left(\hat{f}_{\text{sc}}(\mathbf{q}_j; \theta_2) - f_{\text{sc}}^*(\mathbf{q}_j) \right)^2 + \lambda \left(\|\nabla_{\mathbf{q}} \hat{f}_{\text{sc}}(\mathbf{q}_j; \theta_2)\| - 1 \right)^2, \quad (30)$$

where $\lambda = 0.05$ is user-specified weight.

Network Architecture and Training: The neural CDF and SCDF are both implemented as 5-layer fully connected multilayer perceptrons (MLPs) with GELU activation. Skip connections are included at the second and fourth layers to improve learning stability. Both networks are trained using PyTorch on an NVIDIA RTX 4090 GPU for 50,000 epochs with the Adam optimizer and a learning rate of 2×10^{-4} .

Training Results: The environment CDF model is evaluated on a held-out test set consisting of 50,000 triplets of configurations (\mathbf{q}), workspace points (\mathbf{p}), and corresponding CDF values. It achieves a Mean Absolute Error (MAE) of 0.041 radians and a Root Mean Square Error (RMSE) of 0.073 radians. Similarly, the SCDF model is evaluated on 10,000 held-out configurations with known self-collision distances, achieving MAE of 0.029 radians and RMSE of 0.054 radians. As shown in Table I, the combined inference time for evaluating both the environment and the self-collision CDFs, which together define the neural CDF barrier in (12), remains low across different batch sizes. This scalability enables effective integration with sampling-based motion planning algorithms and real-time control using point cloud observations.

B. Baselines and Parameters

We compare our approach against the following baselines.

• Planners:

- CDF-RRT: RRT with collision checks based on the neural CDF barrier.
- SDF-RRT [9]: RRT with collision checks using a neural SDF barrier.
- SDF-RRT-Connect [15]: A variant of SDF-RRT with a bi-directional search strategy.
- SDF-Lazy-RRT [16]: A variant of SDF-RRT with reduced collision-checking overhead.
- MPPI [38]: A sampling-based MPC method. We compare against the initial “bootstrapping” iterations used for planning.

• Controllers:

- PD Controller: A proportional-derivative controller $\bar{u}(\mathbf{q}, t)$ focused solely on trajectory tracking.
- PD + CBF-QP [14]: A PD controller $\bar{u}(\mathbf{q}, t)$ augmented with a CBF-QP safety filter, where the CBF is constructed using the neural CDF barrier. However, this approach disregards uncertainty in the CDF model or sensor measurements, unlike ours.
- MPPI: [38]: A real-time control baseline that applies MPPI updates at every timestep.

The baseline planners are implemented using the Open Motion Planning Library (OMPL) [61], ensuring consistency across all algorithms. All controllers are implemented using Casadi [62], which solves the CBF-QP and our DR-CBF-QP using an interior point optimizer at a rate of 50 Hz.

Evaluation Metrics: For planners, we evaluate the number of collision checks and path length. For controllers, the evaluation metrics include success rate (percentage of trials where the robot safely reaches the goal) and tracking error (measured as the Fréchet distance between planned and executed paths).

Parameters: Tables II and III summarize the parameters used for the planners and controllers. For the DR-CBF-QP controller, the Wasserstein radius r and the number of uncertainty samples N are chosen to balance robustness and real-time performance. Following standard practice in data-driven DRO control, we treat r as a tunable robustness parameter. Empirically, values in the range $r \in [0.01, 0.04]$ yield reliable performance: larger radii can lead to overly conservative behavior, whereas very small radii may reduce robustness to perception noise and dynamic obstacles. The sample size is limited by the 50 Hz real-time control requirement; using $N = 10$ Monte Carlo dropout samples provides a good tradeoff between robustness and computational efficiency.

C. 2-DoF Planar Robot Simulation

We evaluate our bubble-CDF planner on a 2-DoF planar manipulator with two 2 m links and a fixed base at the origin. Joint angles lie in $[-\pi, \pi)$, and the initial configuration $(0, 0)$ places the end-effector at $(4, 0)$ (Fig. 1). All planners operate from this fixed start toward one of two inverse-kinematically valid goal configurations.

Setup: Each of the 500 environments contains four static obstacles of randomized size and position. The end-effector

TABLE II: Planner parameters used for evaluation. **Goal bias** specifies the probability of sampling configurations directly toward the goal. **Step size** defines the maximum extension distance during tree expansion. **Safety margin** indicates the minimum clearance required for collision checking, and **collision check resolution** determines the granularity of edge validity checks, expressed as a fraction of the configuration space diagonal. For the 2-DoF planar robot, a resolution of 0.01 corresponds to 0.089 radians, while for the 6-DoF xArm, a resolution of 0.002 corresponds to 0.04 radians.

Planner	Goal Bias	Step Size	Safety Margin	Col. Check Resolution
Baselines	0.1	0.1	0.05	0.01 (2-DoF) / 0.002 (6-DoF)
Bubble-CDF	0.1	N/A	0.05	N/A

TABLE III: Control parameters used for evaluation. The PD control gains K_P and K_D are positive definite matrices, assumed to be diagonal with identical values along the diagonal. The function α_h is a linear class \mathcal{K}_∞ function used in the CBF formulation. Parameters k and ζ define the governor dynamics in (18). The risk tolerance ϵ and sample size N correspond to the DR-CBF in (22), while r represents the Wasserstein radius.

Parameter	K_P	K_D	α_h	k	ζ	ϵ	N	r
Value	0.8	0.1	1.0	0.2	12	0.1	10	0.02

goal is sampled at least 4 m from the start and guaranteed to be reachable. The robot receives point-cloud observations of obstacle surfaces. All planners must find a collision-free path from the initial configuration to either a valid goal configuration.

Results: Table IV reports performance across all trials. All planners succeed in finding a feasible solution. The bubble-CDF planner achieves the largest reduction in collision checks, requiring only 153.8 ± 62.2 evaluations—over an order of magnitude fewer than the best baseline (Lazy-RRT), which requires at least $10\times$ more collision checks—while maintaining comparable path length. We report both the planning time (tree expansion/search) and the post-processing time in Table IV. For RRT-based planners, post-processing time refers to shortcutting simplification [63] and B-spline smoothing, both of which are necessary for practical applications. For the bubble-CDF planner, the post-processing time refers to the convex QCQP solution for trajectory optimization. Bubble-CDF achieves the lowest overall runtime, with a planning time of 0.04 ± 0.02 s and a trajectory-optimization time of 0.08 ± 0.02 s. In contrast, the RRT-based baselines require thousands of collision checks and incur similar post-processing costs (approximately 0.23 s), even when their planning times are small. In contrast, MPPI requires far more collision evaluations because it checks collisions along every state of every sampled trajectory in each planning iteration, which results in noticeably slower planning times than graph-based planners. These results highlight the efficiency of bubble-based configuration-space exploration.

Next, we present our evaluation of the DR-CBF-QP formulation for safe control synthesis on a planar robot.

Setup: Controllers are tested in both static (Fig. 1) and dynamic environments (Fig. 3) using point-cloud observations. For dynamic settings, obstacle velocities are sampled as $v \sim \mathcal{N}(0.5, 0.1)$ m/s, while the robot assumes a nominal velocity of 0.5 m/s for control synthesis. Each controller is evaluated over 500 random trials for both environment types. The task is

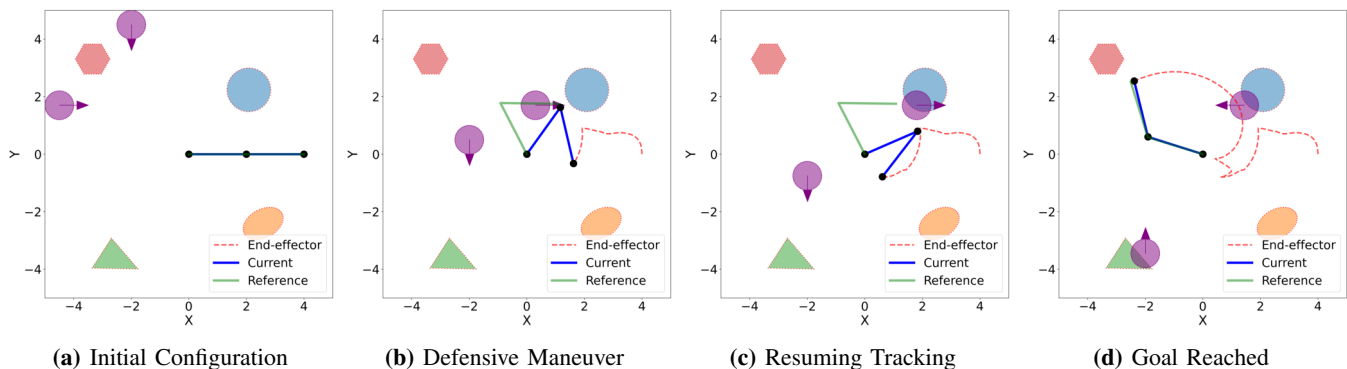


Fig. 3: Snapshots of a 2-link arm navigating a dynamic environment with purple obstacles (velocity directions shown by arrows). The arm is tasked to follow the planned path in Fig. 1. The arm is shown in blue, and the trajectory of its end-effector is shown in red, and the local reference configuration goal $\gamma(s)$ is shown in green.

TABLE IV: Planning performance comparison on a 2-link arm.

Planner	Col. Checks	Path Length (rad)	Plan Time (s)	Postproc. Time (s)
CDF-RRT	2108.9 ± 438.7	3.64 ± 0.69	0.08 ± 0.05	0.22 ± 0.07
SDF-RRT [9]	2118.3 ± 388.9	3.61 ± 0.69	0.08 ± 0.04	0.23 ± 0.07
SDF-RRT-Connect	2144.6 ± 459.7	3.63 ± 0.74	0.03 ± 0.01	0.22 ± 0.08
SDF-Lazy-RRT	1759.2 ± 299.3	3.75 ± 0.83	0.02 ± 0.01	0.23 ± 0.08
MPPI [38]	366012.0 ± 51563.0	5.76 ± 1.37	0.87 ± 0.15	-
Bubble-CDF	153.8 ± 62.2	3.72 ± 0.81	0.04 ± 0.02	0.08 ± 0.02

TABLE V: Control performance comparison on a 2-link arm.

Controller	Static		Dynamic	
	Success Rate	Tra. Error	Success Rate	Tra. Error
PD	0.874	0.068 ± 0.017	0.112	0.062 ± 0.027
PD + CBF-QP [14]	0.982	0.131 ± 0.049	0.638	0.217 ± 0.106
MPPI [38]	1.0	N/A	0.684	N/A
PD + DR-CBF-QP	1.0	0.173 ± 0.074	0.992	0.394 ± 0.172

to track the bubble-CDF trajectory γ while avoiding collisions.

Results: Table V summarizes the control performance. DR-CBF-QP attains perfect success in static environments and 99.2% success in dynamic settings, outperforming PD, standard CBF-QP, and MPPI.

The PD controller, designed only for tracking, performs poorly in both scenarios. While CBF-QP performs well in static environments, its success rate drops sharply under dynamic conditions (63.8%), as it does not account for obstacle-velocity uncertainty. MPPI achieves slightly higher success rates compared to CBF QP, as it optimizes over a receding horizon and can anticipate future states. However, it still does not account for uncertainty, which limits its performance in dynamic environments.

DR-CBF-QP exhibits higher tracking error than the baselines, reflecting a deliberate prioritization of safety: the controller modifies the trajectory as necessary to preserve collision margins, particularly under dynamic uncertainty. Qualitative results in Fig. 3 show the robot executing defensive maneuvers (Fig. 3b) to avoid moving obstacles before rejoining the nominal plan (Fig. 3c) and reaching the goal (Fig. 3d).

D. 6-DoF xArm Robot Simulation

We evaluate our planning and control methods on a 6-DoF xArm manipulator simulated in PyBullet [60], considering both static and dynamic obstacles.

Setup: The environment consists of a 6-DoF xArm mounted on a table adjacent to a shelf (Fig. 4). A depth camera provides point-cloud observations of surrounding obstacles. We perform

TABLE VI: Planning performance comparison on a 6-DoF xArm robot in PyBullet Simulation.

Planner	Col. Checks	Path Length	Plan Time (s)	Postproc. Time (s)
CDF-RRT	3415.4 ± 803.6	3.11 ± 0.46	0.32 ± 0.05	0.47 ± 0.13
SDF-RRT [9]	3251.8 ± 782.4	3.07 ± 0.49	0.33 ± 0.08	0.50 ± 0.11
SDF-RRT-Connect	3437.4 ± 723.2	3.11 ± 0.43	0.04 ± 0.01	0.55 ± 0.13
SDF-Lazy-RRT	2741.1 ± 737.9	3.15 ± 0.53	0.04 ± 0.01	0.48 ± 0.13
MPPI [38]	1331893.2 ± 500936.1	6.83 ± 1.22	4.97 ± 0.83	-
Bubble-CDF	278.5 ± 80.7	3.04 ± 0.46	0.11 ± 0.03	0.08 ± 0.03

50 randomized trials by varying the shelf position and sampling different end-effector goals. For each trial, five feasible goal configurations are generated using inverse kinematics, and motion planning is executed in the static environment. To evaluate controller performance, the best bubble-CDF planned path for each trial is selected for tracking. The dynamic environment includes three moving obstacles with known positions and velocities, sampled from $\mathcal{N}(0.2, 0.05)$ m/s.

Results and Discussion: Table VI reports the planning performance across all 50 trials. As in the 2-DoF benchmark, the bubble-CDF planner achieves a dramatic reduction in collision checks (278.5 ± 80.7) while maintaining path lengths comparable to all baselines. We report both the planning time and the trajectory post-processing time. The Bubble-CDF planner achieves the lowest overall runtime. Notably, the trajectory optimization step in Sec. V-B is a convex QCQP and does not require additional collision checks, since safety is enforced by constraining Bézier control points to lie inside the bubbles. This highlights the efficiency of the bubble-CDF planner in higher-dimensional configuration spaces. Figure 4 shows a path planned with bubble-CDF for a representative static-environment trial.

Table VII summarizes controller performance. The baseline PD controller achieves 68% success in static environments but only 18% in dynamic scenarios, reflecting its lack of collision-awareness. Adding a CBF-QP safety filter increases static-environment success to 88% but remains limited in dynamic settings (70%), as standard CBF-QP does not account for uncertainty in obstacle motion or CDF estimation.

MPPI is run with 300 samples and a horizon of 50. Thanks to its receding horizon formulation, MPPI achieves a higher success rate than the standard CBF QP controller, especially in dynamic environments.

Our PD + DR-CBF-QP controller achieves 100% success

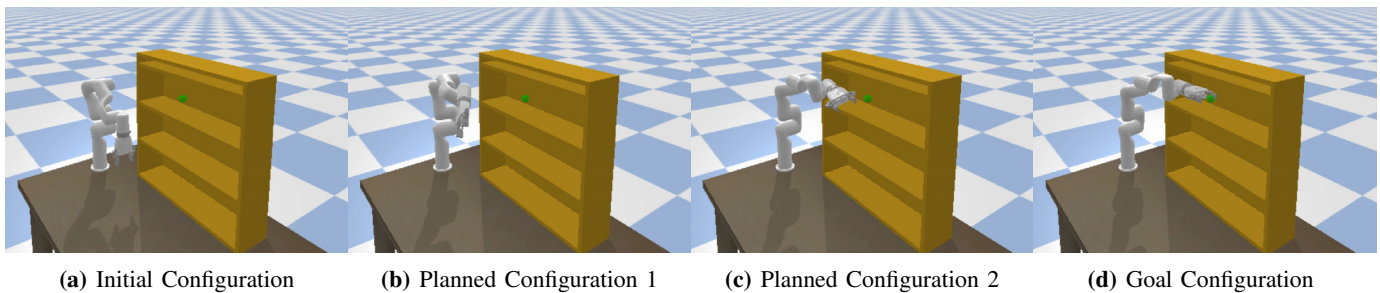


Fig. 4: Bubble-CDF planning for a 6-DoF xArm robot in a static environment, targeting an end-effector goal represented by a green sphere. (a) The initial configuration of the xArm. (b, c) Intermediate configurations illustrating the planned path as the robot avoids obstacles while progressing toward the goal. (d) The final goal configuration reached by the robot.

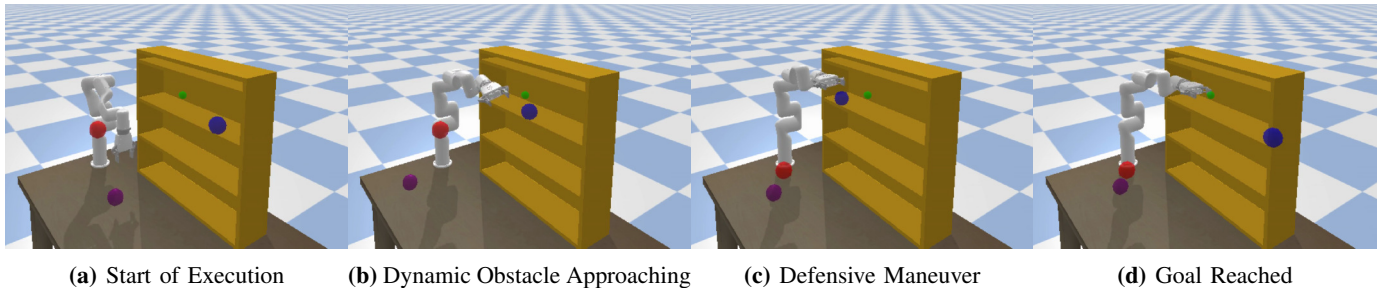


Fig. 5: Snapshots of safe control execution on a 6-DoF xArm robot in an environment with dynamic obstacles. (a) The control execution begins at the initial configuration. (b) A dynamic obstacle (blue) approaches the robot from the right. (c) The robot executes a defensive maneuver, moving upward to avoid the obstacle. (d) The robot successfully resumes tracking and reaches the goal configuration.

TABLE VII: Control performance comparison on a 6-DoF xArm robot in PyBullet simulation.

Controller	Static		Dynamic	
	Success Rate	Tra. Error	Success Rate	Tra. Error
PD	0.68	0.053 ± 0.014	0.18	0.051 ± 0.013
PD + CBF-QP [14]	0.88	0.115 ± 0.046	0.70	0.327 ± 0.104
MPPi [38]	0.9	N/A	0.86	N/A
PD + DR-CBF-QP	1.0	0.143 ± 0.055	1.0	0.516 ± 0.221

TABLE VIII: Planning performance comparison on a real 6-DoF xArm robot.

Planner	Col. Checks	Path Length	Time (s)
CDF-RRT	5617.4 ± 1333.6	3.71 ± 0.71	1.02 ± 0.33
SDF-RRT [9]	5911.9 ± 1487.7	3.72 ± 0.68	1.04 ± 0.33
SDF-RRT-Connect	5801.5 ± 1265.4	3.75 ± 0.63	1.05 ± 0.36
SDF-Lazy-RRT	4765.8 ± 1331.4	3.84 ± 0.69	0.96 ± 0.32
Bubble-CDF	634.8 ± 225.1	3.81 ± 0.66	0.22 ± 0.08

in both static and dynamic environments. While tracking error is higher than the baselines, this reflects its safety-first behavior: the controller adaptively modifies the trajectory to maintain collision margins under dynamic uncertainty, even when deviating from the nominal plan.

Fig. 5 illustrates the execution of a planned path by the 6-DoF xArm robot in a dynamic environment. The robot begins at the initial configuration (Fig. 5a) and dynamically reacts to an approaching obstacle (blue) as shown in Fig. 5b. By executing a defensive maneuver (Fig. 5c), the robot moves upward to maintain safety and avoid a collision. After the obstacle passes, the robot resumes trajectory tracking and successfully reaches the goal configuration (Fig. 5d).

E. 6-DoF xArm Robot Experiments

We validate the efficiency of the bubble-CDF planner and the robustness of the DR-CBF controller in real-world experiments conducted on an xArm6 manipulator operating in cluttered and dynamic environments.

Setup: A 6-DoF xArm is mounted on a table within a workspace containing multiple static obstacles (Fig. 6). A depth camera provides point-cloud observations of the scene. We perform 20 randomized trials by varying obstacle configurations and selecting different goal poses. Motion planning is conducted assuming a static environment. During execution, the robot tracks the planned trajectory while reacting to dynamic obstacles. The velocities of the moving objects are estimated in real time using ArUco markers.

Results and Discussion: Table VIII reports planning performance across all evaluated methods. We report the end-to-end runtime (including both planning and post-processing time) for all methods in this real-robot experiment. The bubble-CDF planner achieves a substantial reduction in collision checks (nearly an order of magnitude fewer than baseline approaches) due to its use of configuration-space bubbles that certify local collision-free regions and reduce reliance on edge-based collision checking. Path lengths remain comparable across all planners, indicating that efficiency gains do not compromise solution quality, consistent with our simulation findings.

Fig. 6 shows snapshots of the bubble-CDF planner and DR-CBF controller in two different workspace arrangements. The robot exhibits similar behavior to the simulation, following the nominal planned trajectory (Figs. 6a, 6e) but avoiding dynamic obstacles as they are presented. Video recordings of

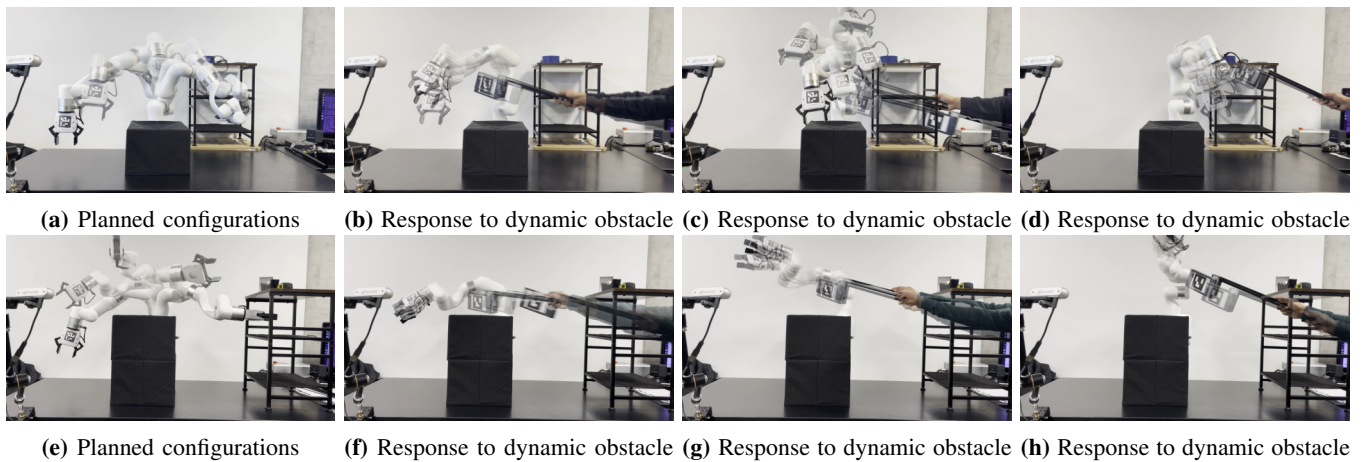


Fig. 6: Bubble-CDF planner and DR-CBF control applied to two real-world setups for a 6-DoF xArm robot. The top row (a-d) represents Setup 1, with the robot navigating a cluttered environment featuring a combination of static and dynamic obstacles. Similarly, the bottom row (e-h) depicts Setup 2, showcasing the planner’s adaptability in a different obstacle layout. For both setups: (a, e) illustrate the bubble-CDF planned configuration in static environments, while (b-d, f-h) demonstrate the robot’s real-time adaptive responses to dynamic obstacles.

these experiments are shown in the supplementary material.

VIII. CONCLUSION

In this work, we proposed an integrated framework for safe motion planning and control of robotic manipulators in dynamic and cluttered environments. Our approach combines a novel bubble-CDF planner for efficient trajectory generation and a distributionally robust control barrier function (DR-CBF) formulation for safe navigation. Comprehensive evaluations, conducted in both simulation and real-world settings, validate the effectiveness of our approach, demonstrating high success rates and significant reductions in computational overhead.

The bubble-CDF planner leverages the neural configuration-space distance function (CDF) to construct local safety regions, termed *configuration-space bubbles*, enabling efficient exploration of the configuration space. By certifying entire regions as collision-free with a single query, the planner significantly reduces the number of collision checks compared to conventional sampling-based methods. This efficiency is achieved without compromising the quality of the planned trajectories, as evidenced by comparable path lengths to baseline approaches. Additionally, the planner allows trajectory optimization to be formulated as a convex program, enabling smooth and safe trajectory refinement without requiring additional collision checks.

For real-time trajectory execution, we introduced a DR-CBF-QP safety filter that ensures the robot remains within a safe set throughout its motion. This formulation explicitly incorporates uncertainties in the neural CDF model, gradient estimates, and sensor measurements, leveraging a distributionally robust optimization framework. The DR-CBF-QP controller dynamically adapts to obstacles and sensor inaccuracies, maintaining safety while ensuring smooth trajectory tracking in both static and dynamic environments. An important direction for future work is the development of explicit feasibility recovery strategies for the DR-CBF-QP when uncertainty sets, control limits, or environment dynamics render the optimization infeasible.

REFERENCES

- [1] S. M. LaValle and J. J. Kuffner, “Rapidly-exploring random trees: Progress and prospects,” *Algorithmic and Computational Rob.*, pp. 303–307, 2001.
- [2] K. Wei and B. Ren, “A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved RRT algorithm,” *Sensors*, vol. 18, no. 2, p. 571, 2018.
- [3] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE Trans. on Rob. (T-RO)*, vol. 12, no. 4, pp. 566–580, 1996.
- [4] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, “DeepSDF: Learning continuous signed distance functions for shape representation,” in *Proc. of IEEE/CVF Conf. on Comput. Vis. and Pattern Recognition*, 2019, pp. 165–174.
- [5] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, “Implicit neural representations with periodic activation functions,” in *Proc. of Neural Inf. Process. Sys. (NeurIPS)*, vol. 33, 2020, pp. 7462–7473.
- [6] K. Long, C. Qian, J. Cortés, and N. Atanasov, “Learning barrier functions with memory for robust safe navigation,” *IEEE Rob. and Automat. Lett. (RA-L)*, vol. 6, no. 3, pp. 4931–4938, 2021.
- [7] M. Koptev, N. Figueroa, and A. Billard, “Neural joint space implicit signed distance functions for reactive robot manipulator control,” *IEEE Rob. and Automat. Lett.*, vol. 8, no. 2, pp. 480–487, 2023.
- [8] Y. Li, Y. Zhang, A. Razmjoo, and S. Calinon, “Representing robot geometry as distance fields: Applications to whole-body manipulation,” in *IEEE Int. Conf. on Rob. Automat. (ICRA)*, 2024, pp. 15351–15357.
- [9] X. Zhu, Y. Xin, S. Li, H. Liu, C. Xia, and B. Liang, “Efficient collision detection framework for enhancing collision-free robot motion,” in *IEEE Int. Conf. on Rob. Automat. (ICRA)*, 2025, pp. 16162–16168.
- [10] C. Quintero-Peña, W. Thomason, Z. Kingston, A. Kyrillidis, and L. E. Kavraki, “Stochastic implicit neural signed distance functions for safe motion planning under sensing uncertainty,” in *Proc. of IEEE Int. Conf. on Robot. and Automat. (ICRA)*, 2024, pp. 2360–2367.
- [11] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, “Voxblox: Incremental 3d Euclidean signed distance fields for on-board MAV planning,” in *IEEE/RSJ Int. Conf. on Intel. Rob. and Sys. (IROS)*, 2017, pp. 1366–1373.
- [12] A. Singletary, W. Guffey, T. G. Molnar, R. Sinnet, and A. D. Ames, “Safety-critical manipulation for collision-free food preparation,” *IEEE Rob. and Automat. Lett.*, vol. 7, no. 4, pp. 10954–10961, 2022.
- [13] Y. Li, X. Chi, A. Razmjoo, and S. Calinon, “Configuration space distance fields for manipulation planning,” in *Proc. of Rob. Sci. Sys.*, 2024.
- [14] X. Chi, Y. Li, J. Huang, B. Dai, Z. Liu, and S. Calinon, “Safe dynamic motion generation in configuration space using differentiable distance fields,” *arXiv preprint arXiv:2412.16456*, 2024.
- [15] J. Kuffner and S. LaValle, “Rrt-connect: An efficient approach to single-query path planning,” in *Proc. of IEEE Int. Conf. on Rob. Automat.*, vol. 2, 2000, pp. 995–1001 vol.2.

- [16] K. Hauser, "Lazy collision checking in asymptotically-optimal motion planning," in *Proc. of IEEE Int. Conf. on Robot. and Automat. (ICRA)*, 2015, pp. 2951–2957.
- [17] M. A. Johnson and M. H. Moradi, *PID control*. Springer, 2005.
- [18] R. Deits and R. Tedrake, *Computing Large Convex Regions of Obstacle-Free Space Through Semidefinite Programming*. Cham: Springer International Publishing, 2015, pp. 109–124.
- [19] H. Dai, A. Amice, P. Werner, A. Zhang, and R. Tedrake, "Certified polyhedral decompositions of collision-free configuration space," *Int. J. of Rob. Res.*, vol. 43, no. 9, pp. 1322–1341, 2024.
- [20] P. Werner, T. Cohn, R. H. Jiang, T. Seyde, M. Simchowitz, R. Tedrake, and D. Rus, "Faster algorithms for growing collision-free convex polytopes in robot configuration space," *arXiv preprint arXiv:2410.12649*, 2024.
- [21] M. Verghese, N. Das, Y. Zhi, and M. Yip, "Configuration space decomposition for scalable proxy collision checking in robot planning and control," *IEEE Rob. and Automat. Lett.*, vol. 7, no. 2, pp. 3811–3818, 2022.
- [22] A. Ademovic and B. Lacevic, "Path planning for robotic manipulators using expanded bubbles of free c-space," in *Proc. of IEEE Int. Conf. on Rob. and Automat. (ICRA)*, 2016, pp. 77–82.
- [23] C. Danielson, "Invariant configuration-space bubbles for revolute serial-chain robots," *IEEE Contr. Sys. Lett.*, vol. 7, pp. 745–750, 2023.
- [24] J. Bialkowski, M. Otte, S. Karaman, and E. Frazzoli, "Efficient collision checking in sampling-based motion planning via safety certificates," *Int. J. of Rob. Res.*, vol. 35, no. 7, pp. 767–796, 2016.
- [25] A. Shkolnik and R. Tedrake, "Sample-based planning with volumes in configuration space," *arXiv preprint arXiv:1109.3145*, 2011.
- [26] O. Brock and L. Kavraki, "Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional configuration spaces," in *Proc. of IEEE Int. Conf. on Rob. Automat.*, vol. 2, 2001, pp. 1469–1474.
- [27] L. Yang and S. LaValle, "The sampling-based neighborhood graph: an approach to computing and executing feedback motion strategies," *IEEE Trans. on Rob.*, vol. 20, no. 3, pp. 419–432, 2004.
- [28] C. K. Verginis, D. V. Dimarogonas, and L. E. Kavraki, "Kdf: Kinodynamic motion planning via geometric sampling-based algorithms and funnel control," *IEEE Trans. on Rob.*, vol. 39, no. 2, pp. 978–997, 2023.
- [29] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "Lqr-trees: Feedback motion planning via sums-of-squares verification," *Int. J. of Rob. Res.*, vol. 29, no. 8, pp. 1038–1052, 2010.
- [30] K. M. B. Lee, Z. Dai, C. L. Gentil, L. Wu, N. Atanasov, and T. Vidal-Calleja, "Safe bubble cover for motion planning on distance fields," *arXiv preprint arXiv:2408.13377*, 2024.
- [31] D. Driess, J.-S. Ha, M. Toussaint, and R. Tedrake, "Learning models as functionals of signed-distance fields for manipulation planning," in *Conf. on Rob. Learn.* PMLR, 2022, pp. 245–255.
- [32] V. Vasilopoulos, S. Garg, P. Piacenza, J. Huh, and V. Isler, "Ramp: Hierarchical reactive motion planning for manipulation tasks using implicit signed distance functions," in *Proc. of IEEE/RSJ Int. Conf. on Intel. Rob. and Sys. (IROS)*, 2023, pp. 10551–10558.
- [33] 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," in *Proc. of Rob. Sci. Sys. (RSS)*, 2024.
- [34] B. Wulft, M. Norrlof, P. Mattsson, and T. Schön, "Probabilistic bubble roadmap," *arXiv preprint arXiv:2502.16205*, 02 2025.
- [35] J. Richalet, A. Rault, J. Testud, and J. Papon, "Model predictive heuristic control," *Automatica*, vol. 14, no. 5, pp. 413–428, 1978.
- [36] S. Kleff, A. Meduri, R. Budhiraja, N. Mansard, and L. Righetti, "High-frequency nonlinear model predictive control of a manipulator," in *2021 Proc. of IEEE Int. Conf. on Robot. and Automat. (ICRA)*. IEEE, 2021, pp. 7330–7336.
- [37] L. Dai, Y. Yu, D.-H. Zhai, T. Huang, and Y. Xia, "Robust model predictive tracking control for robot manipulators with disturbances," *IEEE Trans. on Ind. Elec.*, vol. 68, no. 5, pp. 4288–4297, 2020.
- [38] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *Proc. of IEEE Int. Conf. on Rob. Automat.*, 2016, pp. 1433–1440.
- [39] Y. Zhang, C. Pezzato, E. Trevisan, C. Salmi, C. H. Corbato, and J. Alonso-Mora, "Multi-modal mppi and active inference for reactive task and motion planning," *IEEE Rob. and Automat. Lett.*, vol. 9, no. 9, pp. 7461–7468, 2024.
- [40] K. Long, Y. Yi, Z. Dai, S. Herbert, J. Cortés, and N. Atanasov, "Sensor-based distributionally robust control for safe robot navigation in dynamic environments," *Int. J. of Rob. Res.*, vol. 45, pp. 328–351, 2026.
- [41] S. Keyumarsi, M. W. S. Atman, and A. Gusrialdi, "Lidar-based online control barrier function synthesis for safe navigation in unknown environments," *IEEE Rob. and Automat. Lett.*, vol. 9, no. 2, pp. 1043–1050, 2024.
- [42] M. Yu, C. Yu, M.-M. Naddaf-Sh, D. Upadhyay, S. Gao, and C. Fan, "Efficient motion planning for manipulators with control barrier function-induced neural controller," in *Proc. of IEEE Int. Conf. on Robot. and Automat. (ICRA)*, 2024, pp. 14348–14355.
- [43] S. Zhang, O. So, K. Garg, and C. Fan, "Gcbf+: A neural graph control barrier function framework for distributed safe multi-agent control," *IEEE Trans. on Rob.*, 2025.
- [44] M. Djeha, P. Gergondet, and A. Kheddar, "Robust task-space quadratic programming for kinematic-controlled robots," *IEEE Trans. on Rob.*, vol. 39, no. 5, pp. 3857–3874, 2023.
- [45] L. Brunke, Y. Zhang, R. Römer, J. Naimer, N. Staykov, S. Zhou, and A. P. Schoellig, "Semantically safe robot manipulation: From semantic scene understanding to motion safeguards," *IEEE Rob. and Automat. Lett.*, 2025.
- [46] D. Morton and M. Pavone, "Safe, task-consistent manipulation with operational space control barrier functions," *arXiv preprint arXiv:2503.06736*, 2025.
- [47] L. Han, F. Gao, B. Zhou, and S. Shen, "Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots," in *Proc. of IEEE/RSJ Int. Conf. on Intel. Rob. and Sys. (IROS)*, 2019, pp. 4423–4430.
- [48] K. Long, H. Parwana, G. Fainekos, B. Hoxha, H. Okamoto, and N. Atanasov, "Neural configuration distance function for continuum robot control," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025, pp. 358–364.
- [49] J. D. Head and M. C. Zerner, "A Broyden—Fletcher—Goldfarb—Shanno optimization procedure for molecular geometries," *Chemical Phys. Lett.*, vol. 122, no. 3, pp. 264–270, 1985.
- [50] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.
- [51] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. on Sys. Sci. and Cybern.*, vol. 4, no. 2, pp. 100–107, 1968.
- [52] E. Garone and M. M. Nicotra, "Explicit reference governor for constrained nonlinear systems," *IEEE Trans. on Automat. Contr.*, vol. 61, no. 5, pp. 1379–1384, 2015.
- [53] Z. Li, Ö. Arslan, and N. Atanasov, "Fast and safe path-following control using a state-dependent directional metric," in *Proc. of IEEE Int. Conf. on Robot. and Automat. (ICRA)*, 2020, pp. 6176–6182.
- [54] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *Proc. of Euro. Contr. Conf. (ECC)*, 2019, pp. 3420–3431.
- [55] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Trans. on Automat. Contr.*, vol. 62, no. 8, pp. 3861–3876, 2017.
- [56] P. M. Esfahani and D. Kuhn, "Data-driven distributionally robust optimization using the Wasserstein metric: performance guarantees and tractable reformulations," *Mathematical Programming*, vol. 171, pp. 115–166, 2018.
- [57] P. Mestres, K. Long, N. Atanasov, and J. Cortés, "Feasibility analysis and regularity characterization of distributionally robust safe stabilizing controllers," *IEEE Contr. Sys. Lett.*, vol. 8, pp. 91–96, 2024.
- [58] J. Breeden and D. Panagou, "Robust control barrier functions under high relative degree and input constraints for satellite trajectories," *Automatica*, vol. 155, p. 111109, 2023.
- [59] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proc. of Int. Conf. on Mach. Learn.*, vol. 48, 2016, pp. 1050–1059.
- [60] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," <http://pybullet.org>, 2016–2021.
- [61] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Rob. and Automat. Mag.*, vol. 19, no. 4, pp. 72–82, December 2012, <https://ompl.kavrakilab.org>.
- [62] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.
- [63] L. Petit and A. L. Desbiens, "RRT-Rope: A deterministic shortening approach for fast near-optimal path planning in large-scale uncluttered 3D environments," in *Proc. of IEEE Int. Conf. on Sys., Man, and Cybern. (SMC)*, 2021, pp. 1111–1118.