

Learning Safety for Obstacle Avoidance via Control Barrier Functions

Shuo Liu¹, Zhe Huang¹ and Calin A. Belta²

Abstract—Obstacle avoidance is central to safe navigation, especially for robots with arbitrary and nonconvex geometries operating in cluttered environments. Existing Control Barrier Function (CBF) approaches often rely on analytic clearance computations, which are infeasible for complex geometries, or on polytopic approximations, which become intractable when robot configurations are unknown. To address these limitations, this paper trains a residual neural network on a large dataset of robot–obstacle configurations to enable fast and tractable clearance prediction, even at unseen configurations. The predicted clearance defines the radius of a Local Safety Ball (LSB), which ensures continuous-time collision-free navigation. The LSB boundary is encoded as a Discrete-Time High-Order CBF (DHOCBF), whose constraints are incorporated into a nonlinear optimization framework. To improve feasibility, a novel relaxation technique is applied. The resulting framework ensure that the robot’s rigid-body motion between consecutive time steps remains collision-free, effectively bridging discrete-time control and continuous-time safety. We show that the proposed method handles arbitrary, including nonconvex, robot geometries and generates collision-free, dynamically feasible trajectories in cluttered environments. Experiments demonstrate millisecond-level solve times and high prediction accuracy, highlighting both safety and efficiency beyond existing CBF-based methods.

I. INTRODUCTION

Ensuring safety in autonomous robotic navigation is a fundamental yet challenging problem, especially when robots and obstacles possess complex (e.g., nonconvex, nonsmooth) geometries. A widely used formal tool for safety-critical control is the Control Barrier Function (CBF), which enforces safety by rendering a designated set forward invariant for a dynamical system. Recent studies have shown that CBFs can be combined with Control Lyapunov Functions (CLFs) to achieve both stabilization and safety while respecting input constraints. This integration yields real-time implementable optimization problems, typically formulated as Quadratic Programs (QPs) when the cost is quadratic [1], [2].

Beyond relative-degree one constraints, several extensions of CBFs have been proposed. Exponential CBFs [3] and High-Order CBFs (HOCBFs) [4], [5] address safety functions with higher relative degrees. These methods have been applied to robotic obstacle avoidance (e.g., [6]–[9]). However, due to the complexity of obstacle shapes, the authors of [9] employed logistic regression to construct polynomial barrier functions that approximate obstacle boundaries,

enabling analytical enforcement of nonnegative clearance. Nevertheless, this method could not handle robots with complex geometries. In [6], [7], the authors focused on polytopic robots and obstacles, where the CBFs were defined through polytope-to-polytope distances. Such functions were not globally differentiable, and hence safety could not be guaranteed at configurations where the CBF derivative was undefined. Although the authors of [8] proposed a smooth approximation to unify multiple CBFs into a single differentiable function, this led to overly conservative control, since smoothing enlarged the safe set margin and restricted feasible control actions more than necessary.

To avoid reliance on differentiable CBFs and address discrete-time systems, Discrete-Time CBFs (DCBFs) were introduced in [10], and later extended to Discrete-Time High-Order CBFs (DHOCBFs) [11], [12]. These methods have been applied to collision avoidance between polytopic robots and polytopic obstacles [13], and even extended to handle robots navigating around nonconvex obstacles [14], [15]. However, [13] remains limited to polytopic robots. Accurately approximating complex geometries with polytopes incurs significant computational overhead at each robot configuration. Moreover, constructing such approximations at unknown future configurations is intractable, fundamentally limiting the applicability of this approach in predictive control settings. While the methods proposed in [14], [15] effectively address nonconvex obstacle avoidance, they do not extend to handling robots with complex shapes.

More recently, learning-based methods that directly regress a Signed Distance Function (SDF) have been explored (e.g., [16]–[19]). While such approaches can predict the clearance between the robot and obstacles, the methods in [16], [17] enforce safety only at discrete configurations, without guaranteeing collision avoidance along the continuous trajectory between these configurations. In contrast, [18], [19] achieve continuous-time obstacle avoidance by incorporating SDF-based CBF constraints, but their frameworks are limited to point robots or robots with circular shapes, and are not applicable to robots with complex geometries.

These limitations motivate our proposed approach, which combines clearance prediction, local safety region construction, and continuous-time safety enforcement between discrete-time steps into a unified framework. A residual neural network is trained on a large dataset of robot–obstacle configurations to predict the clearance. The predicted clearance defines a Local Safety Ball (LSB), which is anchored to the robot’s boundary to construct a local safety region and ensure continuous-time safety. The LSB is then encoded as DHOCBF constraints, which are incorporated into a non-

This work was supported in part by the NSF under grant IIS-2024606 at Boston University.

¹S. Liu and Z. Huang are with Boston University, Brookline, MA, USA {liushuo, huangz7}@bu.edu

²C. Belta is with the Department of Electrical and Computer Engineering and with the Department of Computer Science, University of Maryland, College Park, MD, USA cbelta@umd.edu

TABLE I: List of acronyms.

Acronyms	Meaning
CBF	Control Barrier Function
DCBF	Discrete-Time Control Barrier Function
DHOCBF	Discrete-Time High-Order Control Barrier Function
SDF	Signed Distance Function
LSB	Local Safety Ball
MLP	Multilayer Perceptron
EPA	Extreme Point Anchor
NLP	Nonlinear Programming
MSE	Mean Squared Error

linear optimization problem. To improve feasibility, a novel relaxation technique is applied. This overall framework guarantees that the robot's rigid-body motion between discrete time steps remains collision-free, thereby bridging discrete-time control with continuous-time collision avoidance.

We demonstrate through numerical results that the proposed framework can generate dynamically feasible, collision-free trajectories for robots with various shapes navigating cluttered, maze-like environments. The results further show that each optimization problem can be solved within milliseconds, confirming the approach's efficiency and scalability.

II. PRELIMINARIES

In this section, we provide mathematical preliminaries on safety-critical control. We consider a discrete-time control system in the form:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t), \quad (1)$$

where $\mathbf{x}_t \in \mathcal{X} \subset \mathbb{R}^n$ is the state at time step $t \in \mathbb{N}$, $\mathbf{u}_t \in \mathcal{U} \subset \mathbb{R}^q$ is the control input, and the function $f : \mathbb{R}^n \times \mathbb{R}^q \rightarrow \mathbb{R}^n$ is assumed to be locally Lipschitz continuous. We define safety as the forward invariance of a set $\mathcal{C}$. Specifically, system (1) is considered safe if, once initialized in $\mathcal{C}$, its state remains in $\mathcal{C}$ for all future time steps. We define $\mathcal{C}$ as the superlevel set of a continuous function $h : \mathbb{R}^n \rightarrow \mathbb{R}$:

$$\mathcal{C} := \{\mathbf{x} \in \mathbb{R}^n : h(\mathbf{x}) \geq 0\}. \quad (2)$$

Definition 1 (Relative degree [12]). The output $y_t = h(\mathbf{x}_t)$ of system (1) is said to have relative degree m if

$$\begin{aligned} y_{t+i} &= h(\bar{f}_{i-1}(f(\mathbf{x}_t, \mathbf{u}_t))), \quad i \in \{1, 2, \dots, m\}, \\ \text{s.t. } \frac{\partial y_{t+m}}{\partial \mathbf{u}_t} &\neq \mathbf{0}_q, \quad \frac{\partial y_{t+i}}{\partial \mathbf{u}_t} = \mathbf{0}_q, \quad i \in \{1, 2, \dots, m-1\}, \end{aligned} \quad (3)$$

i.e., m is the number of steps (delay) in the output y_t in order for any component of the control input $\mathbf{u}_t$ to explicitly appear ($\mathbf{0}_q$ is the zero vector of dimension q).

In Def. 1, $\bar{f}(\mathbf{x}) := f(\mathbf{x}, 0)$ denotes the uncontrolled state dynamics. The subscript of $\bar{f}$ indicates recursive composition: $\bar{f}_0(\mathbf{x}) = \mathbf{x}$ and $\bar{f}_i(\mathbf{x}) = \bar{f}(\bar{f}_{i-1}(\mathbf{x}))$ for $i \geq 1$. We assume that $h(\mathbf{x})$ has relative degree m with respect to system (1) based on Def. 1. Starting with $\psi_0(\mathbf{x}_t) := h(\mathbf{x}_t)$, we define a sequence of discrete-time functions $\psi_i : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, \dots, m$ as:

$$\psi_i(\mathbf{x}_t) := \Delta\psi_{i-1}(\mathbf{x}_t) + \alpha_i(\psi_{i-1}(\mathbf{x}_t)), \quad (4)$$

where $\Delta\psi_{i-1}(\mathbf{x}_t) := \psi_{i-1}(\mathbf{x}_{t+1}) - \psi_{i-1}(\mathbf{x}_t)$, and $\alpha_i(\cdot)$ denotes the i^{th} class κ function which satisfies $\alpha_i(\psi_{i-1}(\mathbf{x}_t)) \leq \psi_{i-1}(\mathbf{x}_t)$ for $i = 1, \dots, m$. A sequence of sets $\mathcal{C}_i$ is defined based on (4) as

$$\mathcal{C}_i := \{\mathbf{x} \in \mathbb{R}^n : \psi_i(\mathbf{x}) \geq 0\}, \quad i \in \{0, \dots, m-1\}. \quad (5)$$

Definition 2 (DHOCBF [11]). Let $\psi_i(\mathbf{x})$, $i \in \{1, \dots, m\}$ be defined by (4) and $\mathcal{C}_i$, $i \in \{0, \dots, m-1\}$ be defined by (5). A function $h : \mathbb{R}^n \rightarrow \mathbb{R}$ is a Discrete-time High-Order Control Barrier Function (DHOCBF) with relative degree m for system (1) if there exist $\psi_m(\mathbf{x})$ and $\mathcal{C}_i$ such that

$$\psi_m(\mathbf{x}_t) \geq 0, \quad \forall \mathbf{x}_t \in \mathcal{C}_0 \cap \dots \cap \mathcal{C}_{m-1}, t \in \mathbb{N}. \quad (6)$$

Theorem 1 (Safety Guarantee [11]). Given a DHOCBF $h(\mathbf{x})$ from Def. 2 with corresponding sets $\mathcal{C}_0, \dots, \mathcal{C}_{m-1}$ defined by (5), if $\mathbf{x}_0 \in \mathcal{C}_0 \cap \dots \cap \mathcal{C}_{m-1}$, then any Lipschitz controller $\mathbf{u}_t$ that satisfies the constraint in (6), $\forall t \geq 0$ renders $\mathcal{C}_0 \cap \dots \cap \mathcal{C}_{m-1}$ forward invariant for system (1), i.e., $\mathbf{x}_t \in \mathcal{C}_0 \cap \dots \cap \mathcal{C}_{m-1}, \forall t \geq 0$.

We can simply define an i^{th} order DCBF $\psi_i(\mathbf{x})$ in (4) as

$$\psi_i(\mathbf{x}_t) := \Delta\psi_{i-1}(\mathbf{x}_t) + \gamma_i\psi_{i-1}(\mathbf{x}_t), \quad (7)$$

where $\alpha(\cdot)$ is defined linear and $0 < \gamma_i \leq 1, i \in \{1, \dots, m\}$. The above DCBF is typically incorporated as a safety constraint into the following safety-critical optimal control problem:

$$J(\mathbf{u}_t, \mathbf{x}_{t+1}) = \min_{\mathbf{u}_t} \mathbf{u}_t^T \mathbf{u}_t + \sum_{k=t}^{t+1} (\mathbf{x}_k - \mathbf{x}_T)^T (\mathbf{x}_k - \mathbf{x}_T) \quad (8a)$$

$$\text{s.t. } \psi_{m-1}(\mathbf{x}_{t+1}) \geq (1 - \gamma_m)\psi_{m-1}(\mathbf{x}_t), \quad 0 < \gamma_m \leq 1, \quad (8b)$$

$$\mathbf{u}_t \in \mathcal{U} \subset \mathbb{R}^q, \quad \mathbf{x}_{t+1} \in \mathcal{X} \subset \mathbb{R}^n, \quad (8c)$$

where (8a) specifies the objective function of the optimization problem, $\mathbf{x}_T$ denotes the target state, and $t \in \mathbb{N}$. At each time step t , the current state $\mathbf{x}_t$ is given. Constraint (8b) enforces that the system state $\mathbf{x}_{t+1}$ in (1) remains within the safe set $\mathcal{C}$ defined in (2) (Thm. 1). Constraint (8c) imposes bounds on both the control input and the state; however, these bounds may conflict with the safety constraint (8b), potentially leading to infeasibility.

III. PROBLEM FORMULATION AND APPROACH

We consider a 2D car-like robot with arbitrary, possibly nonconvex geometry navigating through a cluttered environment. The robot must reach a specified target while ensuring safety and adhering to its dynamics and input constraints. The environment contains K static obstacles $\mathcal{O}_1, \dots, \mathcal{O}_K \subset \mathbb{R}^2$, each with known, fixed boundaries. Let $\mathcal{O} := \bigcup_{i=1}^K \mathcal{O}_i$ denote the union of all obstacles. Let $\mathcal{R}(x_t, y_t, \theta_t) \subset \mathbb{R}^2$ denote the robot at time t , where (x_t, y_t) is the position of a reference point of the robot and θ_t is its orientation with respect to a reference frame (see Fig. 1). The (safety)

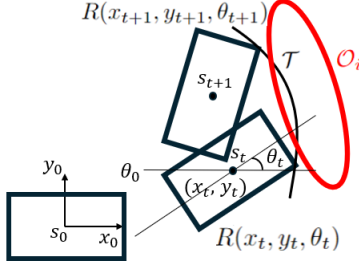


Fig. 1: Rigid-body transition of the robot (black) between two discrete time steps with respect to an obstacle (red). (x_t, y_t) denotes the location, and θ_t denotes the orientation of the robot at t . $\mathcal{T}$ denotes the continuous trajectory of a boundary point on the robot between time steps t and $t + 1$.

requirement that the robot avoids all obstacles at all times can be written as:

$$\mathcal{R}(x_t, y_t, \theta_t) \cap \mathcal{O} = \emptyset, \quad \forall t \geq 0. \quad (9)$$

We assume that the dynamics of the robot are described by (1), where the system state $\mathbf{x}_t$ includes the robot's location (x_t, y_t) , orientation θ_t , and potentially additional variables such as speed. The control input is denoted by $\mathbf{u}_t$. The obstacles correspond to configuration-space (C-space) obstacles, i.e., the images of workspace obstacles mapped into the robot's configuration space.¹

In this paper, we consider the following problem:

Problem 1. For a robot with given geometry $\mathcal{R}$ and dynamics (1) moving in an environment with obstacles $\mathcal{O}_i$, $i = 1, \dots, K$, design a control policy that steers the reference point (x_t, y_t) from an initial location (x_0, y_0) to a desired region centered at the target location (x_T, y_T) in finite time, while avoiding collisions, satisfying the control and state constraints in (8c), and minimizing its control effort.

When considering Problem 1, we typically need to compute the clearance between the robot and obstacles to ensure it remains nonnegative. However, obtaining an analytical expression for the clearance is generally infeasible for robots and obstacles with complex geometries. To enable numerical computation, their continuous boundaries are approximated by sampled points, and the clearance is typically computed via exhaustive search over these samples. This process becomes computationally expensive when the boundary samples are overly dense. Furthermore, estimating the clearance for future configurations—such as $(x_{t+1}, y_{t+1}, \theta_{t+1})$ —which are unknown at time step t , is intractable through brute-force search. In addition, enforcing nonnegative clearance only at discrete time steps, e.g., $\mathcal{R}(x_t, y_t, \theta_t) \cap \mathcal{O}_i = \emptyset, \mathcal{R}(x_{t+1}, y_{t+1}, \theta_{t+1}) \cap \mathcal{O}_i = \emptyset$ as illustrated in Fig. 1, does not prevent the transitional trajectory $\mathcal{T}$ of the robot between t and $t+1$ from intersecting

¹Although we focus on a 2D setting in this work, the proposed method is general and can be extended to higher-dimensional systems such as aerial drones or robotic manipulators.

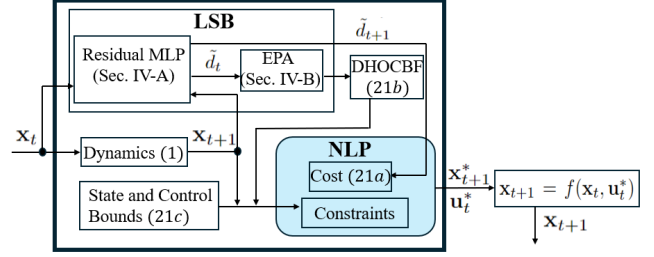


Fig. 2: Schematic of the control computation at time t . The MLP predicts the clearance, which is passed to the EPA to define the radius of the LSB. The LSB's boundary is then converted into DHOCBF constraints. At each time step, an NLP with a quadratic cost and nonlinear constraints is solved.

the obstacle—especially in the presence of complex shapes. To overcome these challenges, our approach is as follows.

Approach: We sample the boundaries of the robot and the union of obstacles using sufficiently dense boundary points. Based on these samples, we collect a large and diverse set of robot configurations along with their corresponding ground-truth clearances—computed via exhaustive search—to train a residual neural network that accurately and efficiently predicts the robot–obstacle clearance. Even when the robot's configuration is represented as an optimization variable at a future time step, the network can output a clearance prediction that serves as a decision variable to the optimization problem. This predicted clearance defines the radius of a reference ball, which is anchored to an extreme boundary point of the robot using the Extreme Point Anchor (EPA) method, forming the Local Safety Ball (LSB). The LSB ensures continuous-time collision avoidance. Its boundary is encoded as a Discrete-Time High-Order Control Barrier Function (DHOCBF) constraint with respect to the robot's reference point. Combined with state and input constraints, these DHOCBFs define a nonlinear optimization problem with the cost defined by Eq. (8a), illustrated in Fig. 2, whose feasibility is enhanced by a relaxation technique. The target location is included in the cost function as a reference state to minimize the deviation from the current location. This framework guarantees that the robot's entire rigid-body motion between consecutive time steps remains collision-free. The resulting controller balances target-reaching performance with obstacle avoidance and is applied iteratively at each time step until the desired region is reached.

IV. FROM LEARNED CLEARANCE TO SAFE CONTROL: CONSTRUCTING AND INTEGRATING LOCAL SAFETY BALLS

A. Learning Clearance: Residual MLP Design

1) *Geometric Modeling:* Let $\{\mathcal{O}_k\}_{k=1}^K$ denote the workspace obstacles. Each obstacle boundary is discretized into samples $\partial\mathcal{O}_k = \{o_{k,m} \in \mathbb{R}^2\}_{m=1}^{M_k}$ and we define the global boundary-sample set $\partial\mathcal{O} := \bigcup_{k=1}^K \partial\mathcal{O}_k$. The robot has a nontrivial shape with boundary $\partial\mathcal{R}$. We choose a reference point $s = [x, y]^\top \in \mathbb{R}^2$ whose motion is governed

by the dynamic model, and an orientation $\theta \in (-\pi, \pi]$. For configuration (x, y, θ) , the robot's workspace boundary is

$$\partial\mathcal{R}(x, y, \theta) = s \oplus \{\mathbf{R}(\theta) r_0 \mid r_0 \in \partial\mathcal{R}_0\}, \quad (10)$$

where $\oplus$ denotes the Minkowski sum, $\mathbf{R}(\theta)$ is a 2D rotation matrix, and $\partial\mathcal{R}_0$ denotes the robot boundary defined in the reference frame located at the origin with zero orientation. The robot-obstacle clearance d is calculated as:

$$d(x, y, \theta) = \min_{r \in \partial\mathcal{R}(x, y, \theta)} \min_{o \in \partial\mathcal{O}} \|r - o\|_2, \quad (11)$$

where r and o are samples from the boundaries of the robot and obstacle, respectively. If $\partial\mathcal{R}(x, y, \theta)$ intersects $\mathcal{O}$, the clearance is defined as negative, equal to the maximum Euclidean distance from the robot boundary samples inside $\mathcal{O}$ to its boundary.

2) *Ground-Truth Clearance:* Given (x, y, θ) , the clearance is defined by (11). To populate the dataset, we uniformly (or according to a coverage strategy) sample locations $s_i = [x_i, y_i]^\top$ across the map and discretize orientations $\theta_j \in \Theta = \{\theta_1, \dots, \theta_{N_\theta}\}$. For each (x_i, y_i, θ_j) we compute d_{ij} from (11). The resulting dataset is $\mathcal{D} = \{(\mathbf{x}_{ij}, d_{ij})\}$ with $\mathbf{x}_{ij} = [x_i, y_i, \theta_j]^\top$.

3) *Network and Target Normalization:* We train a fully connected residual network [20] $f_\phi : \mathbb{R}^3 \rightarrow \mathbb{R}$ as an MLP to regress the clearance. Inputs are standardized feature-wise to $\bar{\mathbf{x}} = (\mathbf{x} - \mu_x) \oslash \sigma_x$ ($\oslash$ denotes elementwise division, μ_x, σ_x are the mean and standard deviation of the input features). To reduce target scale imbalance, we apply a log transform:

$$\bar{d} = \frac{\log(d + \varepsilon) - \mu_{\log}}{\sigma_{\log}}, \quad \varepsilon > |d_{\min}|, \quad (12)$$

where $\mu_{\log}, \sigma_{\log}$ denote the mean and standard deviation of the log-transformed distances. The training objective is mean squared error in the normalized space:

$$\mathcal{L}(\phi) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x}, d) \in \mathcal{D}} \|f_\phi(\bar{\mathbf{x}}) - \bar{d}\|_2^2. \quad (13)$$

At inference, we recover metric distances via

$$\tilde{d} = \exp(f_\phi(\bar{\mathbf{x}}) \sigma_{\log} + \mu_{\log}) - \varepsilon. \quad (14)$$

4) *Architecture:* The network comprises three input-processing layers expanding $3 \rightarrow h/4 \rightarrow h/2 \rightarrow h$ (with BatchNorm [21] and GELU [22]), a stack of $L_b = 6$ residual blocks of width $h = 2048$, and a three-layer head reducing $h \rightarrow h/2 \rightarrow h/4 \rightarrow 1$ as shown in Fig. 3. Besides the standard identity skip inside each block, we add non-adjacent skips every two blocks to promote long-range gradient flow: if $\mathbf{z}_i$ is the output of block i ,

$$\mathbf{z}_{i+2} \leftarrow \mathbf{z}_{i+2} + \mathbf{A}_i \mathbf{z}_i, \quad \mathbf{A}_i \in \mathbb{R}^{h \times h}, \quad i \in \{1, 3\}. \quad (15)$$

5) *Training Details:* Linear layers use Kaiming normal initialization [23]. BatchNorm scale/bias are initialized to 1/0. We optimize with AdamW [24] (base learning rate 10^{-4}) and a cyclic scheduler between 10^{-4} and 10^{-3} (halving amplitude each cycle). We apply gradient clipping at 1.0 and early stopping on validation loss.

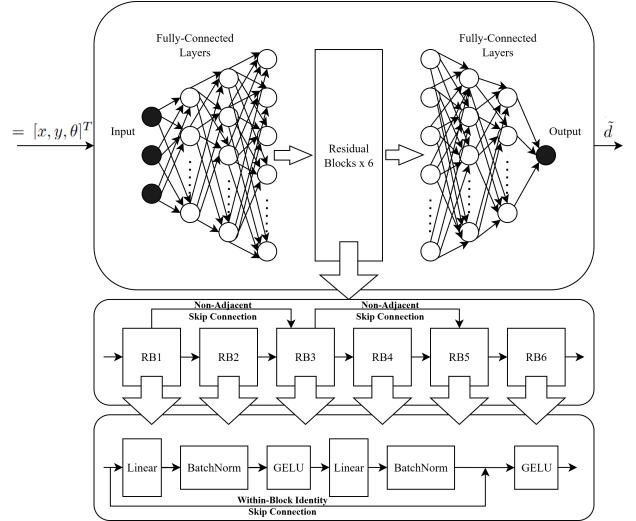


Fig. 3: System identification of the residual MLP, a deep neural network $f_\phi(\mathbf{x})$ trained on map data. Given the robot's state (x, y, θ) , the MLP outputs the predicted clearance $\tilde{d}$ between the boundaries of the robot and the obstacle.

6) *Evaluation:* On a held-out test set, we report mean squared error in the original distance scale:

$$\text{MSE}_{\text{test}} = \frac{1}{N_\theta} \sum_{n=1}^{N_\theta} (\tilde{d}(\mathbf{x}_n^{\text{test}}) - d_n^{\text{test}})^2, \quad (16)$$

where N_θ denotes the number of orientation samples for each location (x, y) (e.g., $N_\theta = 360$ when one sample is taken per degree).

Remark 1 (DNN Prediction Errors and Certified Bounds). Our MLP does not provide strict a priori error bounds. For low-dimensional inputs (e.g., 2D states), techniques such as Lipschitz certificates [25] could, in principle, yield worst-case guarantees. Incorporating such bounds will be pursued as future work.

B. EPA: Shape-Aware Local Safety Ball

Definition 3 (Extreme Point Anchor). Given a robot configuration (x, y, θ) , let $\partial\mathcal{R}(x, y, \theta)$ denote the robot boundary and $\partial\mathcal{O}$ the obstacle boundary. If the clearance d is defined by (11) and let (r_p, o_p) denote a pair of boundary points attaining the clearance in (11). The *reference ball* is the closed ball $\mathbb{B}(r_p, d)$ centered at r_p with radius d . Let s be the reference point of the robot and define

$$r_c \in \arg \max_{r \in \partial\mathcal{R}(x, y, \theta)} \|r - s\|_2, \quad (17)$$

i.e., the boundary point farthest from s (ties are broken arbitrarily). The *translated ball* is then given by $\mathbb{B}(r_c, d)$, i.e., the ball of radius d centered at r_c . This construction is referred to as the Extreme Point Anchor (EPA), as illustrated in Fig. 4.

Lemma 1 (Ball Containment under Bounded Rigid Motion). *Let d be the clearance in (11). Let $\mathcal{T} \in SE(2)$ be the one-step rigid motion of the robot boundary, and let $\{\mathcal{T}_\tau\}_{\tau \in [0,1]}$ denote a continuous transition from the identity motion ($\tau = 0$) to $\mathcal{T}$ ($\tau = 1$). For each boundary point $r \in \partial\mathcal{R}$ define $r(\tau) := \mathcal{T}_\tau(r)$. Assume the step is d -bounded that*

$$\sup_{\tau \in [0,1]} \sup_{r \in \partial\mathcal{R}} \|r(\tau) - r\|_2 \leq d. \quad (18)$$

Then, for every $r \in \partial\mathcal{R}$ and every $\tau \in [0,1]$, we have $r(\tau) \in \mathbb{B}(r, d)$ and, in particular, $r' = \mathcal{T}(r) \in \mathbb{B}(r, d)$. Consequently, since each ball $\mathbb{B}(r, d)$ contains no obstacle boundary points by (11), the step is continuous-time collision-free.

Proof. Fix $r \in \partial\mathcal{R}$. By (18), for all $\tau \in [0,1]$, $\|r(\tau) - r\| \leq d$, which is exactly the statement that $r(\tau) \in \mathbb{B}(r, d)$. Because d is the clearance to the obstacle boundary, $\mathbb{B}(r, d)$ contains no obstacle boundary points for any r . Hence no boundary trajectory $r(\tau)$ can intersect $\partial\mathcal{O}$ during the step, and the motion is safe. $\square$

In Lemma. 1, the double supremum in (18) corresponds to the maximum location change among all boundary points over the step, which in practice is attained by the boundary point r_c that is farthest from the reference point s as defined in Def. 3. i.e., $\|r_c(\tau) - r_c\|_2 = \sup_{r \in \partial\mathcal{R}} \|r(\tau) - r\|_2$. An advantage of using the EPA is that r_c can be directly determined from the reference point s and the robot shape, without requiring the closest boundary pair (r_p, o_p) . Another advantage, formalized in Lemma 1 (e.g., Eq. (18)), is that by constraining $r_c(\tau)$ to remain within its translated ball, all other boundary points are guaranteed to remain within their corresponding balls throughout the entire rigid-body transition, thereby reducing the number of constraints.

C. DHOCBF-Constrained NLP Controller

When we obtain the translated ball, we need to consider how to use it to restrict the movement of the robot's reference point s , so that the robot's trajectory remains collision-free. According to Lemma 1, we must constrain $r(\tau)$ to lie within the ball centered at r_c with radius d . We define a DHOCBF

$$\tilde{\psi}_0(\mathbf{x}_t | \tilde{d}_t, r_{c,t}) := \tilde{h}(\mathbf{x}_t | \tilde{d}_t, r_{c,t}) = \tilde{d}_t - \|r_c(\mathbf{x}_t) - r_{c,t}\|_2, \quad (19)$$

where $\tilde{d}_t = f_\phi(x_t, y_t, \theta_t)$ is obtained at t from the MLP and $r_c(\mathbf{x}_t) = r_{c,t}$ denotes the center of the translated ball at t computed according to (17). Since the LSB constructed based on $\tilde{d}_t$ and $r_{c,t}$ often defines a local safety region that is smaller than the actual global safety region, strictly enforcing the DHOCBF based on local safety region tends to shrink the feasible set and increases the risk of infeasibility. To address this, we introduce slack variables to relax the DHOCBF constraints while still ensuring obstacle avoidance.

Theorem 2. *Given a DHOCBF $\tilde{\psi}_0(\mathbf{x}, \Omega | \tilde{d}_t, r_{c,t}) = \tilde{h}(\mathbf{x} | \tilde{d}_t, r_{c,t})$ for system (1) with corresponding relaxed func-*

tions $\tilde{\psi}_i(\mathbf{x}, \Omega | \tilde{d}_t, r_{c,t})$, $i \in \{1, \dots, m\}$ defined by

$$\begin{aligned} \tilde{\psi}_i(\mathbf{x}_t, \Omega_t | \tilde{d}_t, r_{c,t}) &:= \tilde{\psi}_{i-1}(\mathbf{x}_{t+1}, \Omega_t | \tilde{d}_t, r_{c,t}) + \\ &\omega_{i,t}(\gamma_i - 1)\tilde{\psi}_{i-1}(\mathbf{x}_t, \Omega_t | \tilde{d}_t, r_{c,t}), \end{aligned} \quad (20)$$

where $0 < \gamma_i \leq 1$ and $\Omega = [\omega_1, \dots, \omega_m]^\top$ is a slack variable vector. If $\tilde{\psi}_0(\mathbf{x}_0, \Omega_0 | \tilde{d}_0, r_{c,0}) \geq 0$ and $\omega_i \geq 0$, then any Lipschitz controller $\mathbf{u}_t$ that satisfies the constraints $\tilde{\psi}_i(\mathbf{x}_t, \Omega_t | \tilde{d}_t, r_{c,t}) \geq 0$, $i \in \{1, \dots, m\}$ renders $\tilde{h}(\mathbf{x}_{t+m} | \tilde{d}_t, r_{c,t}) \geq 0, \forall t \geq 0$ and obstacle avoidance is guaranteed.

Proof. At $t = 0$, since $\tilde{\psi}_0(\mathbf{x}_0, \Omega_0 | \tilde{d}_0, r_{c,0}) \geq 0$, based on (20) and Def. 1, we have $\tilde{\psi}_0(\mathbf{x}_1, \Omega_0 | \tilde{d}_0, r_{c,0}) \geq \omega_{1,0}(1 - \gamma_1)\tilde{\psi}_0(\mathbf{x}_0 | \tilde{d}_0, r_{c,0}) \geq 0, \dots, \tilde{\psi}_{m-1}(\mathbf{x}_1, \Omega_0 | \tilde{d}_0, r_{c,0}) \geq \omega_{m,0}(1 - \gamma_m)\tilde{\psi}_{m-1}(\mathbf{x}_0 | \tilde{d}_0, r_{c,0}) \geq 0$. Since $\tilde{\psi}_1(\mathbf{x}_1, \Omega_0 | \tilde{d}_0, r_{c,0}) \geq 0 \Leftrightarrow \tilde{\psi}_0(\mathbf{x}_2, \Omega_0 | \tilde{d}_0, r_{c,0}) \geq \omega_{1,0}(1 - \gamma_1)\tilde{\psi}_0(\mathbf{x}_1, \Omega_0 | \tilde{d}_0, r_{c,0}) \geq 0$, repeatedly we have $\tilde{\psi}_2(\mathbf{x}_1, \Omega_0 | \tilde{d}_0, r_{c,0}) \geq 0 \Leftrightarrow \tilde{\psi}_0(\mathbf{x}_3, \Omega_0 | \tilde{d}_0, r_{c,0}) \geq 0$, and so on up to $\tilde{\psi}_{m-1}(\mathbf{x}_1, \Omega_0 | \tilde{d}_0, r_{c,0}) \geq 0 \Leftrightarrow \tilde{\psi}_0(\mathbf{x}_m, \Omega_0 | \tilde{d}_0, r_{c,0}) \geq 0$. At $t = 1$, since $\tilde{\psi}_0(\mathbf{x}_1, \Omega_0 | \tilde{d}_0, r_{c,0}) \geq 0$, we have $\tilde{\psi}_0(\mathbf{x}_1, \Omega_1 | \tilde{d}_1, r_{c,1}) \geq 0$, based on (20) and Def. 1, we have $\tilde{\psi}_0(\mathbf{x}_2, \Omega_1 | \tilde{d}_1, r_{c,1}) \geq \omega_{1,1}(1 - \gamma_1)\tilde{\psi}_0(\mathbf{x}_1 | \tilde{d}_1, r_{c,1}) \geq 0$, and so on up to $\tilde{\psi}_{m-1}(\mathbf{x}_2, \Omega_1 | \tilde{d}_1, r_{c,1}) \geq \omega_{m,1}(1 - \gamma_m)\tilde{\psi}_{m-1}(\mathbf{x}_1, \Omega_1 | \tilde{d}_1, r_{c,1}) \geq 0$. Since $\tilde{\psi}_1(\mathbf{x}_2, \Omega_1 | \tilde{d}_1, r_{c,1}) \geq 0 \Leftrightarrow \tilde{\psi}_0(\mathbf{x}_3, \Omega_1 | \tilde{d}_1, r_{c,1}) \geq \omega_{1,1}(1 - \gamma_1)\tilde{\psi}_0(\mathbf{x}_2 | \tilde{d}_1, r_{c,1}) \geq 0$, repeatedly we have $\tilde{\psi}_2(\mathbf{x}_2, \Omega_1 | \tilde{d}_1, r_{c,1}) \geq 0 \Leftrightarrow \tilde{\psi}_0(\mathbf{x}_4, \Omega_1 | \tilde{d}_1, r_{c,1}) \geq 0$, and so on up to $\tilde{\psi}_{m-1}(\mathbf{x}_2, \Omega_1 | \tilde{d}_1, r_{c,1}) \geq 0 \Leftrightarrow \tilde{\psi}_0(\mathbf{x}_{1+m}, \Omega_1 | \tilde{d}_1, r_{c,1}) \geq 0$. Repeatedly do above process we have $\tilde{\psi}_i(\mathbf{x}_{t+1}, \Omega_t | \tilde{d}_t, r_{c,t}) \geq 0, i \in \{0, \dots, m-1\}$ renders $\tilde{\psi}_0(\mathbf{x}_{t+m}, \Omega_t | \tilde{d}_t, r_{c,t}) = \tilde{h}(\mathbf{x}_{t+m} | \tilde{d}_t, r_{c,t}) \geq 0$, therefore, for all $t \geq 0$, the clearance between the robot boundary and the obstacle boundary over the horizon $[0, t+m]$ remains nonnegative, and thus obstacle avoidance is guaranteed. $\square$

In Thm. 2, the slack variable $\omega_i \in [0, 1]$ relaxes the DHOCBF constraint to improve feasibility, scaling with $1 - \gamma_i$ to adjust the class κ function and ensure obstacle avoidance for time-varying safety sets. Under high-order constraints, it also extends safety guarantees forward by m steps, enabling an m -step lookahead. While DHOCBFs provide step-wise discrete-time safety, Lemma 1 ensures that the geometry of the LSB yields collision-free rigid-body motion over the continuous interval $\tau \in [t\Delta t, (t+m)\Delta t]$, where $\Delta t > 0$ is the sampling period. Thus, the LSB construction bridges discrete-time control and continuous-time safety. This enables us to incorporate relaxed DHOCBF constraints in a nonlinear programming (NLP) problem:

$$\min_{\mathbf{u}_t, \Omega_t} p(\mathbf{x}_{t+1}) + q(\mathbf{x}_t, \mathbf{u}_t, \Omega_t) - \lambda \tilde{d}_{t+1} \quad (21a)$$

$$\begin{aligned} \text{s.t. } & \tilde{\psi}_{i-1}(\mathbf{x}_{t+1}, \Omega_t | \tilde{d}_t, r_{c,t}) + \\ & \omega_{i,t}(\gamma_i - 1)\tilde{\psi}_{i-1}(\mathbf{x}_t, \Omega_t | \tilde{d}_t, r_{c,t}) \geq 0, \end{aligned} \quad (21b)$$

$$\mathbf{u}_t \in \mathcal{U} \subset \mathbb{R}^q, \mathbf{x}_{t+1} \in \mathcal{X} \subset \mathbb{R}^n, \quad (21c)$$

where $0 < \gamma_i \leq 1$, $\Omega = [\omega_1, \dots, \omega_m]^\top$, and the current state $\mathbf{x}_t$ is given. In (21a), the cost function consists of the current

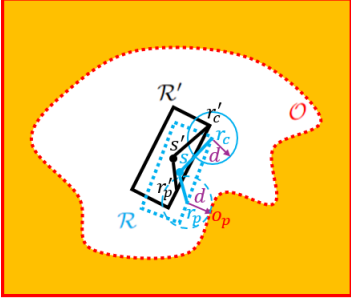


Fig. 4: Illustration of the Extreme Point Anchor. The reference ball (dashed blue) is centered at boundary point r_p with clearance d to o_p , and translated to the farthest boundary point r_c from the reference point s , forming the translated ball (solid blue).

cost $q(\mathbf{x}_t, \mathbf{u}_t, \Omega_t)$, the next-step cost $p(\mathbf{x}_{t+1})$, and a clearance penalty term $-\lambda \tilde{d}_{t+1}$, where $\lambda > 0$ is a weight factor. The penalty $-\lambda \tilde{d}_{t+1} = -\lambda f_\phi(x_{t+1}, y_{t+1}, \theta_{t+1})$ discourages negative values of $\tilde{d}_{t+1}$ and thereby encourages the robot to maintain a greater clearance from obstacles at the next time step. The predictive MLP enables tractable clearance prediction by providing outputs that serve as decision variables in the NLP when the input configuration is defined as variables. The discrete-time dynamics in (1) are incorporated into the relaxed DHOCBF constraints in (21b) to guarantee safety, e.g., $\tilde{\psi}_0(\mathbf{x}_{t+i}, \Omega_t | \tilde{d}_t, r_{c,t}) = \tilde{\psi}_0(\tilde{f}_{i-1}(f(\mathbf{x}_t, \mathbf{u}_t)), \Omega_t | \tilde{d}_t, r_{c,t})$ based on Def. 1 and $\omega_i \geq 0, i \in \{1, \dots, m\}$. The state and input constraints are considered in (21c). This NLP problem is solved at each time step, and the resulting optimal input $\mathbf{u}_t^*$ is applied to system (1) to generate the state $\mathbf{x}_{t+1}$ at the next time step.

Remark 2. In Thm. 20, the relaxation technique places the slack variable ω_i in front of $(1 - \gamma_i)$, yielding

$$\tilde{\psi}_{i-1}(\mathbf{x}_{t+1}, \Omega_t | \tilde{d}_t, r_{c,t}) \geq \omega_{i,t}(1 - \gamma_i)\tilde{\psi}_{i-1}(\mathbf{x}_t, \Omega_t | \tilde{d}_t, r_{c,t}),$$

which is stronger than treating ω_i as a additive term,

$$\tilde{\psi}_{i-1}(\mathbf{x}_{t+1}, \Omega_t | \tilde{d}_t, r_{c,t}) \geq (1 - \gamma_i)\tilde{\psi}_{i-1}(\mathbf{x}_t, \Omega_t | \tilde{d}_t, r_{c,t}) + \omega_{i,t}.$$

Both improve feasibility, but only the multiplicative form preserves safety guarantees when $\omega_{i,t} \in [0, 1)$ and $\tilde{\psi}_{i-1}(\mathbf{x}_t) \geq 0$. In contrast, the additive form may violate safety if $\omega_{i,t} < 0$. Prior works [12], [26] use similar strategies; however, [26] is limited to relative-degree-one DCBFs, and [12] partially relaxes the decay rate via linearization. In contrast, our method fully relaxes the decay rate without linearization, enabling general nonlinear DHOCBFs.

D. Complexity Scaling with State and Input Dimensions and Constraint Numbers

The NLP problem in (21) is solved at each time step, and its complexity depends on the number of state and input variables as well as the order m of the DHOCBF. For a system with state dimension n and input dimension q , the decision vector scales as $\mathcal{O}(n + q + m)$, while the constraints include bounds and m DHOCBF terms. Due to the nonlinear nature of the learned LSBs and slack relaxation, the NLP

remains nonconvex, and solver time grows superlinearly with the number of variables and constraints. While the formulation can be extended to an N -step nonlinear model predictive control for improved safety and convergence, we focus on the single-step case to limit computation time.

V. NUMERICAL RESULTS

In this section, we address an autonomous navigation problem, modeling the controlled robot with various shapes, including a rectangle, triangle and cross-shape (nonconvex). The proposed optimization-based control algorithm is able to generate dynamically feasible, collision-free trajectories even in tight maze environments, as illustrated in Fig. 6. Animations of these navigation scenarios are provided at <https://youtu.be/dqay1R2u5-I>.

A. Numerical Setup

1) *System Dynamics*: Consider a discrete-time unicycle model in the form

$$\begin{bmatrix} x_{t+1} - x_t \\ y_{t+1} - y_t \\ \theta_{t+1} - \theta_t \\ v_{t+1} - v_t \end{bmatrix} = \begin{bmatrix} v_t \cos(\theta_t) \Delta t \\ v_t \sin(\theta_t) \Delta t \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} u_{1,t} \\ u_{2,t} \end{bmatrix}, \quad (22)$$

where $\mathbf{x}_t = [x_t, y_t, \theta_t, v_t]^\top$ captures the 2-D location, heading angle, and linear speed; $\mathbf{u}_t = [u_{1,t}, u_{2,t}]^\top$ represents angular velocity (u_1) and linear acceleration (u_2), respectively. The system is discretized with $\Delta t = 0.05$. System (22) is subject to the following state and input constraints:

$$\begin{aligned} \mathcal{X} &= \{\mathbf{x}_t \in \mathbb{R}^4 : -5 \cdot \mathcal{I}_{4 \times 1} \leq \mathbf{x}_t \leq 5 \cdot \mathcal{I}_{4 \times 1}\}, \\ \mathcal{U} &= \{\mathbf{u}_t \in \mathbb{R}^2 : -10 \cdot \mathcal{I}_{2 \times 1} \leq \mathbf{u}_t \leq 10 \cdot \mathcal{I}_{2 \times 1}\}. \end{aligned} \quad (23)$$

2) *System Configuration*: The rectangle, triangle, and cross-shape robots have dimensions (0.35×0.2) , (0.4×0.3) , and (0.35×0.35) , respectively, with reference points located at their center, right-angle vertex, and arm intersection. To account for reference point offsets, the initial positions are set to $[0.5, 4.7]^\top$ (rectangle and cross-shape) and $[0.5, 4.6]^\top$ (triangle), targeting $\mathbf{x}_T = [4.7, 0.5]^\top$ or $[4.7, 0.4]^\top$ accordingly. All robots start with an initial speed of 0.5, but the target speed is 0 for the rectangle robot and 0.1 for the triangle and cross-shape robots. Heading angles follow $\theta_0 = \theta_T = \text{atan2}\left(\frac{y_T - y_t}{x_T - x_t}\right)$. The robot switches trajectory once its reference point enters a waypoint-centered circle of radius 0.1. The other reference vectors are $\mathbf{u}_r = [0, 0]^\top$ and $\Omega_r = [1, 1]^\top$.

3) *Map Processing*: The scope of the map is $x \in [0, 5]$ and $y \in [0, 5]$. The boundary samples of the robots and obstacles are manually selected. Our neural network follows the structure in IV-A.4. For training, we generated 3,396,600 samples by sampling 9,345 random locations across the map and rotating each over 360° (one degree per sample). An additional 686,880 samples were collected for testing using the same procedure. The prediction accuracy of the MLP is evaluated using the MSE in (16).

4) *DHOCBF*: The candidate DHOCBF function $\tilde{\psi}_0(\mathbf{x}_t|\hat{d}_t, r_{c,t})$ is defined by Eq. (19) where the predicted clearance $\hat{d}_t$ is defined by MLP in Sec. IV-A and the center of translated ball $r_{c,t}$ is defined by Eq. (17). The hyperparameters γ_1 and γ_2 are set to 0.1.

5) *NLP Design*: The cost function of the NLP problem (21) consists of current cost $q(\mathbf{x}_t, \mathbf{u}_t, \Omega_t) = \|\mathbf{x}_t - \mathbf{x}_T\|_Q^2 + \|\mathbf{u}_t - \mathbf{u}_r\|_R^2 + \|\Omega_t - \Omega_r\|_S^2$, next-step cost $p(\mathbf{x}_t) = \|\mathbf{x}_{t+1} - \mathbf{x}_T\|_P^2$ and a clearance penalty term $-\lambda\hat{d}_{t+1}$, where $Q = P = 100 \cdot \mathcal{I}_4$, $R = \mathcal{I}_2$, $S = 100 \cdot \mathcal{I}_2$ and $\lambda = 1000$.

6) *Solver Configurations and CPU Specs*: The NLP problem is formulated in Python using CasADi [27] and solved with IPOPT [28] on Ubuntu 20.04, running on an AMD Ryzen 7 5800U CPU, with all inference performed on the CPU. Model training and inference are implemented in PyTorch, with training carried out on a Linux desktop equipped with an NVIDIA RTX 4090 GPU to leverage hardware acceleration.

7) *MLP Prediction Accuracy*: The prediction accuracy of MLP is demonstrated in Fig. 5. We selected seven different locations from the testing data on the map, where each location corresponds to a specific configuration (x, y, θ) for each robot shape. These selected configurations are illustrated by the blue boundary samples of the robots. For each case, the ground-truth clearance was computed as the minimum distance from each robot boundary sample to the obstacle boundary samples. Using this clearance as the radius, we generated red circles centered at each sample of the robot as translated circles. Similarly, the MLP-predicted clearance was used as the radius to generate blue circles centered at each pixel. As shown in Figs. 5a, 5b and 5c, the red and blue circles nearly coincide, indicating high prediction accuracy. Figs. 5d, 5e, 5f and 5g show the relationship between the MLP's prediction error (MSE) and the rectangle robot's positions in 3D and projected views. The results show that the MSE dips (darker colors) near obstacle boundaries but rises (brighter colors) in other regions, implying that prediction errors are relatively larger away from obstacle boundaries. Overall, statistical analysis indicates that the prediction error at most testing locations is below 0.01, demonstrating that the trained MLP can predict clearance with high accuracy.

8) *Safe Trajectory Generation*: Eight closed-loop trajectories connecting the waypoints are illustrated in Fig. 6. For clarity, the robot locations are plotted at fixed time intervals to visualize the overall obstacle-avoidance behavior. Even within narrow passages requiring sharp turns, robots of different shapes are able to navigate safely through the free space, regardless of whether the robots or obstacles are convex or nonconvex. This demonstrates the adaptability of the proposed controller to complex environments with narrow passages and nonconvex geometries.

In addition, we record the computation time per NLP solve for each robot shape. The average single-step computation times are 14.16 ± 1.26 ms for the rectangle robot, 13.96 ± 1.94 ms for the triangle robot, and 15.82 ± 2.12 ms for the cross-shape robot, demonstrating the high efficiency of the

optimization algorithm.

VI. CONCLUSION AND FUTURE WORK

In this paper, we develop a novel framework that combines learning-based clearance prediction with DHOCBF-constrained control through the construction of Local Safety Balls. By leveraging a residual MLP to predict clearances and anchoring them via the EPA, we provide a tractable way to guarantee transitional safety for robots with arbitrary shapes navigating in cluttered environments. The proposed relaxation technique further enhance feasibility for high-order constraints, and the closed-loop simulations validated both the safety and efficiency of our approach. Future research will extend this framework in several directions, such as integrating the approach into a model predictive control framework to improve long-horizon safety and performance, and considering dynamic obstacles to handle more realistic and time-varying environments.

REFERENCES

- [1] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 6271–6278.
- [2] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.
- [3] Q. Nguyen and K. Sreenath, "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints," in *2016 American Control Conference (ACC)*, 2016, pp. 322–328.
- [4] W. Xiao and C. Belta, "High-order control barrier functions," *IEEE Transactions on Automatic Control*, vol. 67, no. 7, pp. 3655–3662, 2021.
- [5] X. Tan, W. S. Cortez, and D. V. Dimarogonas, "High-order barrier functions: Robustness, safety, and performance-critical control," *IEEE Transactions on Automatic Control*, vol. 67, no. 6, pp. 3021–3028, 2021.
- [6] A. Thirugnanam, J. Zeng, and K. Sreenath, "Duality-based convex optimization for real-time obstacle avoidance between polytopes with control barrier functions," in *2022 American Control Conference (ACC)*, 2022, pp. 2239–2246.
- [7] Y.-H. Chen, S. Liu, W. Xiao, C. Belta, and M. Otte, "Control barrier functions via minkowski operations for safe navigation among polytopic sets," *arXiv preprint arXiv:2504.00364*, 2025.
- [8] S. Wu, Y. Fang, N. Sun, B. Lu, X. Liang, and Y. Zhao, "Optimization-free smooth control barrier function for polygonal collision avoidance," *IEEE Transactions on Cybernetics*, 2025.
- [9] C. Peng, O. Donca, G. Castillo, and A. Hereid, "Safe bipedal path planning via control barrier functions for polynomial shape obstacles estimated using logistic regression," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 3649–3655.
- [10] A. Agrawal and K. Sreenath, "Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation," in *Robotics: Science and Systems*, vol. 13. Cambridge, MA, USA, 2017.
- [11] Y. Xiong, D.-H. Zhai, M. Tavakoli, and Y. Xia, "Discrete-time control barrier function: High-order case and adaptive case," *IEEE Transactions on Cybernetics*, pp. 1–9, 2022.
- [12] S. Liu, J. Zeng, K. Sreenath, and C. A. Belta, "Iterative convex optimization for model predictive control with discrete-time high-order control barrier functions," in *2023 American Control Conference (ACC)*, 2023, pp. 3368–3375.
- [13] A. Thirugnanam, J. Zeng, and K. Sreenath, "Safety-critical control and planning for obstacle avoidance between polytopes with control barrier functions," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 286–292.
- [14] S. Liu, Y. Mao, and C. A. Belta, "Safety-critical planning and control for dynamic obstacle avoidance using control barrier functions," in *2025 American Control Conference (ACC)*, 2025, pp. 348–354.

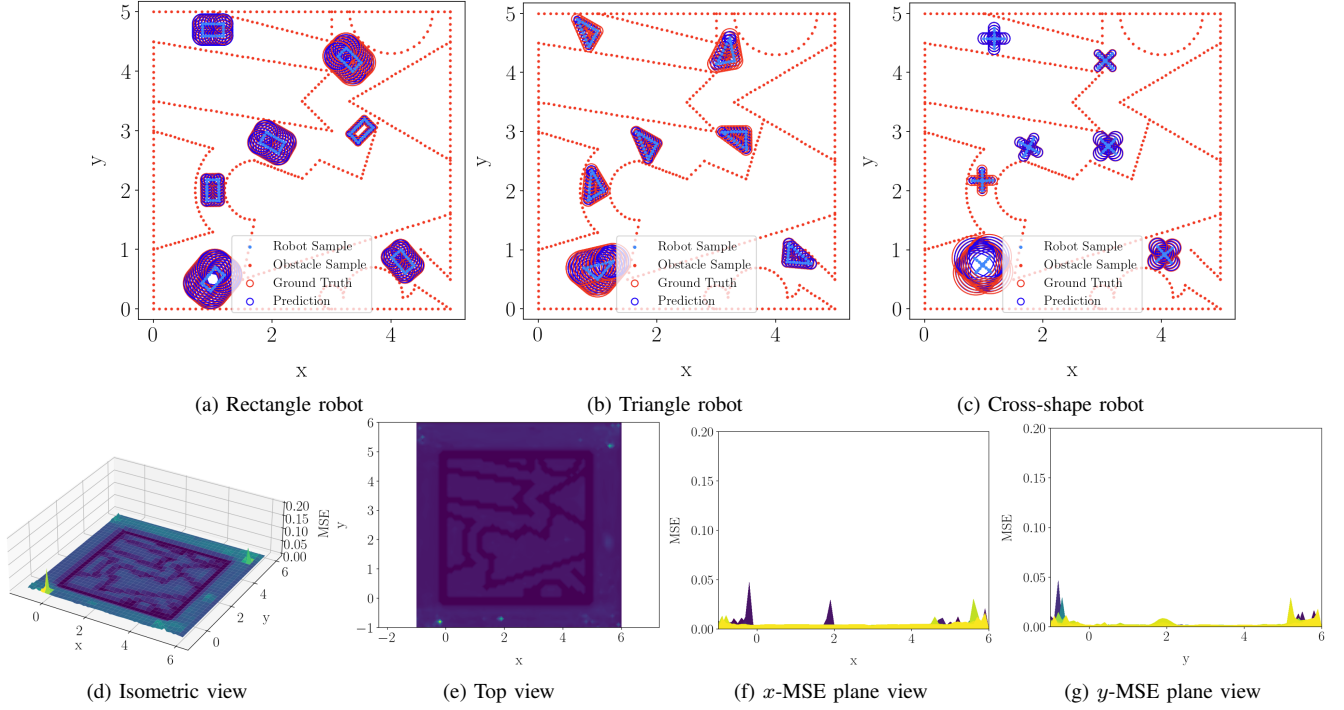


Fig. 5: Visualization of MLP prediction accuracy. (a–c) Comparison between ground-truth clearances (red circles) and MLP-predicted clearances (blue circles) for rectangle, triangle, and cross-shape robots, showing near coincidence of the two. (d–g) 3D visualization of the MSE of the testing data for the rectangle robot, where errors are smaller near obstacle boundaries and larger elsewhere, with most prediction errors below 0.01.

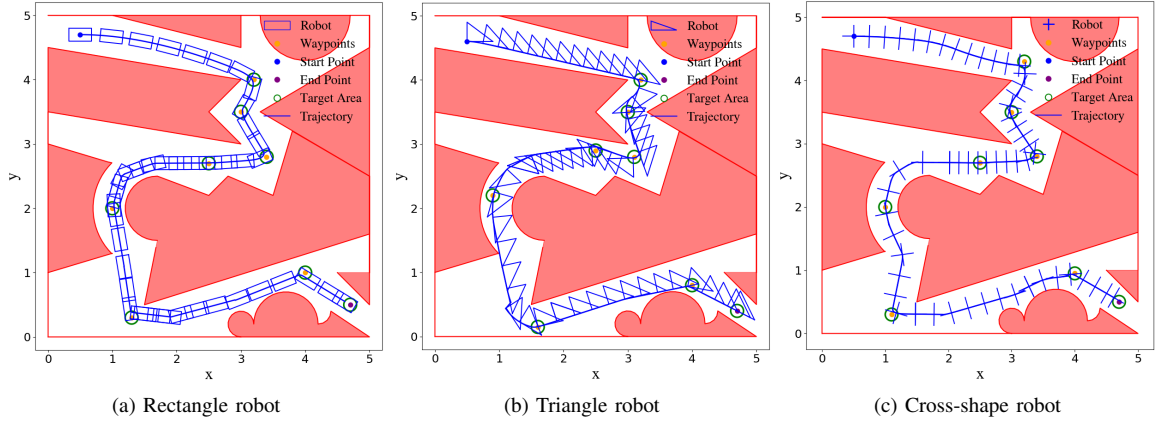


Fig. 6: Closed-loop trajectories of rectangle, triangle, and cross-shape robots in a maze-like environment. Robots successfully navigate narrow passages and sharp turns, demonstrating that the proposed DHOCBF-based controller ensures safe obstacle avoidance for both convex and nonconvex robots and obstacles.

- [15] S. Liu, Z. Huang, J. Zeng, K. Sreenath, and C. A. Belta, “Learning-enabled iterative convex optimization for safety-critical model predictive control,” *arXiv preprint arXiv:2409.08300*, 2024.
- [16] B. Liu, G. Jiang, F. Zhao, and X. Mei, “Collision-free motion generation based on stochastic optimization and composite signed distance field networks of articulated robot,” *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 7082–7089, 2023.
- [17] Y. Li, Y. Zhang, A. Razmjoo, and S. Calinon, “Representing robot geometry as distance fields: Applications to whole-body manipulation,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 15 351–15 357.
- [18] S. Das, Y. Xue, H. Li, and N. Figueroa, “Rnbf: Real-time rgb-d based neural barrier functions for safe robotic navigation,” *arXiv preprint arXiv:2505.02294*, 2025.
- [19] K. Long, C. Qian, J. Cortés, and N. Atanasov, “Learning barrier functions with memory for robust safe navigation,” *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4931–4938, 2021.
- [20] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [21] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International conference on machine learning*. pmlr, 2015, pp. 448–456.
- [22] D. Hendrycks and K. Gimpel, “Gaussian error linear units (gelus),” *arXiv preprint arXiv:1606.08415*, 2016.
- [23] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

- [24] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.
- [25] Y. Tsuzuku, I. Sato, and M. Sugiyama, "Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks," *Advances in neural information processing systems*, vol. 31, 2018.
- [26] J. Zeng, Z. Li, and K. Sreenath, "Enhancing feasibility and safety of nonlinear model predictive control with discrete-time control barrier functions," in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 6137–6144.
- [27] J. A. 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.
- [28] L. T. Biegler and V. M. Zavala, "Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization," *Computers & Chemical Engineering*, vol. 33, no. 3, pp. 575–582, 2009.