# Safe Manipulation in Dynamic Environments via Online Perception-Based Neural Control Barrier Functions

***Abstract*— In this paper, we investigate the safe execution of robotic manipulators operating in dynamic environments. Safety in such scenarios requries enforcing collision-avoidance constraints under rapidly changing observations of moving obstacles. We present an online, perception-based neural control barrier function (Neural CBF) framework for real-time safety enforcement during manipulation. Our key contribution is a CBF formulation and learning objective that explicitly accounts for obstacle motion through the time variation of perception. We consider dynamic environments with uncontrolled but observable obstacle agents whose states and velocities can be accessed but cannot be actuated. In this setting, the safety condition must account for how observations evolve as obstacle agents move. We address this by introducing a differentiable observation model that maps obstacle states to observations, enabling stable estimation of observation dynamics and incorporating them into the neural CBF learning objective. Experiments in dynamic manipulation scenarios show that our approach achieves higher safety and better constraint satisfaction compared to competitive baselines. The codes and videos are available at our project page: `https://neural-CBF.github.io/NCBF/`.**

## I. Introduction

Robotic manipulators are increasingly deployed in dynamic environments such as collaborative workspaces, where robots must safely execute manipulation tasks while interacting with moving obstacles, including humans, other robots, and mobile objects. Ensuring safety is fundamentally challenging because collision-avoidance constraints must be enforced in real time under continuously changing observations. Failure to properly account for the motion of surrounding agents can lead to unsafe behaviors or overly conservative control strategies that degrade task performance. Therefore, developing principled methods for safety enforcement in such environments remains a challenging problem.

Control Barrier Functions (CBFs) have emerged as a powerful framework for enforcing safety constraints in dynamic systems [1], [2]. However, constructing accurate barrier functions often requires explicit knowledge of system geometry, environment models, and safety boundaries [3], [4], which can be difficult to specify in complex or perception-driven environments. Therefore, learning-based extensions such as neural CBFs [5], [6], [7] have been proposed to handle complex safety constraints that are difficult to model analytically. These approaches use neural networks to represent barrier functions and learn safety certificates from data while retaining the theoretical structure of the CBF condition.

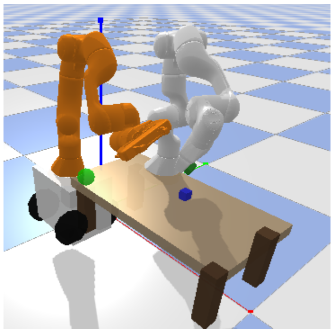


Fig. 1: **Motivating example scenario.** A flexible mobile manipulator (orange) operates in a shared workspace to assist task execution while avoiding collisions with a second manipulator (gray) that follows its own autonomous motion and acts as a dynamic obstacle.

Despite these advances, most existing CBF and neural CBF methods assume that the environment is either static [1], [5] or that obstacle dynamics are explicitly incorporated into the system state [8], [9], [10]. While such formulations can handle moving obstacles, they typically require explicit dynamic models of the environment, which may not be available in many practical scenarios. A promising approach to address this challenge is to incorporate perception directly into CBF formulations. However, this introduces additional difficulties, including sensor noise, state-estimation errors [11], [12], and non-smooth observation models that may violate the differentiability assumptions required by standard CBF conditions [13]. In particular, existing perception-based approaches typically focus on static environments or bounded observation errors and do not explicitly account for how safety constraints evolve as observations change over time in dynamic environments.

Let us consider a motivating example to further illustrate the challenge, shown in Figure 1. In this scenario, a mobile and flexible manipulator is dispatched to execute a task in a shared workspace where another manipulator is simultaneously performing its own autonomous task and thus acts as a dynamic obstacle. In this setting, the obstacle manipulator is articulated, high-dimensional, and may move rapidly with coupled joint dynamics, producing complex time-varying collision configurations. Moreover, its geometry is far from a simple primitive shape, making it difficult to derive reliable distance-based safety constraints without accurate models and careful geometric processing. As a result, safety constraints must be enforced under continuously changing, perception-driven information rather than a static map, and the controller must react online to the evolving observations induced by the dynamic obstacle's motion.

In this paper, we address these limitations by introducing an online perception-based neural control barrier function framework for safe manipulation in dynamic environments. Our key idea is to explicitly account for the time variation of observations induced by moving obstacles when formulating the CBF condition. To achieve this, we introduce a differentiable observation model that maps obstacle states to perception observations. This model enables the estimation of observation dynamics and allows the time derivative of the barrier function to incorporate the effect of obstacle motion. By integrating this observation-aware formulation into the neural CBF learning objective, the learned barrier function can properly capture how safety constraints evolve as the environment changes.

We evaluate the proposed approach in dynamic manipulation scenarios involving moving obstacles and rapidly changing observations. Experimental results demonstrate that the proposed online perception-based neural CBF framework achieves improved safety and constraint satisfaction compared to existing neural CBF baselines. By explicitly modeling observation dynamics induced by moving obstacles, our method enables reliable safety enforcement for robotic manipulation in dynamic environments, where safety constraints are defined directly over perception observations.

The remaining part of this paper is organized as follows. Section II formulates the safe manipulation problem in dynamic environments. Section III presents the proposed perception-based neural CBF framework, including differentiable observation modeling, neural CBF learning, and online safe controller synthesis. Section IV reports experimental validations in simulation. Finally, Section V concludes the paper and discusses future directions.

## II. Problem Formulation

We consider a manipulator whose configuration evolves according to a control-affine model

$$\dot{q} = f(q) + g(q)u, \tag{1}$$

where $q \in \mathcal{Q} \subseteq \mathbb{R}^n$ denotes the robot configuration and $u \in \mathcal{U} \subseteq \mathbb{R}^m$ is the control input. The robot operates in an environment $\mathcal{E}$. The robot perceives the environment $\mathcal{E}$ through a observation

$$o = o(q, \mathcal{E}) \in \mathcal{O} \subset \mathbb{R}^k. \tag{2}$$

The state-observation space is defined as $\mathcal{X} := \mathcal{Q} \times \mathcal{O}$, which can be partitioned into three subspaces according to collision status:

- *Unsafe set* $\mathcal{X}_u$: states demonstrating the robot collides with environmental obstacles or self-collides.
- *Safe set* $\mathcal{X}_s$: states that are separated from the unsafe set by a safety margin $r_s \geq 0$,

$$\mathcal{X}_s = \Big\{ x \in \mathcal{X} : \|x - x_u\|_2 \geq r_s, \ \forall x_u \in \mathcal{X}_u \Big\}. \tag{3}$$

- *Boundary set* $\partial\mathcal{X} = \mathcal{X} \backslash (\mathcal{X}_u \cup \mathcal{X}_s)$, capturing near-contact or margin-violating states.

To enforce safety, we adopt a control barrier function defined on the state-observation space.

**Definition 1** (**Control Barrier Function**). *Consider system* (1) *with observation model* (2)*. A continuously differentiable function* $h : \mathcal{X} \to \mathbb{R}$ *is a control barrier function if there exist constants* $\alpha > 0$ *and margins* $\gamma, \epsilon > 0$ *such that:*

i) *for any* $(q, o) \in \mathcal{X}_s$*, we have* $h(q, o) \leq -\gamma$*;*
ii) *for any* $(q, o) \in \mathcal{X}_u$*, we have* $h(q, o) > \gamma$*;*
iii) *for any* $(q, o) \in \mathcal{X}$*, we have*

$$\inf_{u \in \mathcal{U}} \frac{\partial h(q,o)}{\partial q} \left( f(q) + g(q)u \right) + \frac{\partial h(q,o)}{\partial o} \dot{o} \leq -\alpha h(q,o) - \epsilon \tag{4}$$

*where* $\dot{o}$ *is the time derivative of the environment observation induced by the evolution of* $(q, \mathcal{E})$*. The margin* $\epsilon$ *encourages strict satisfaction in practice.*

By Definition 1, the induced safe set is the superlevel set

$$\mathcal{C} = \{(q, o) \in \mathcal{X} : \ h(q, o) \geq 0\}. \tag{5}$$

It has been proven in [2] that by enforcing condition (4), $\mathcal{C}$ is forward invariant. Therefore, if $(q(0), o(0)) \in \mathcal{C} \subseteq \mathcal{X}_s$, then $(q(t), o(t)) \in \mathcal{C} \subseteq \mathcal{X}_s$ for all $t \geq 0$.

In this paper, we suppose that internal states of the obstacle agents are available to the robot, summarized as follows.

**Assumption 1.** *In dynamic environments containing moving obstacle agents, the robot has access to agents' state and velocity, while these agents remain uncontrolled by the robot.*

Now, we state our problem under investigation as follows.

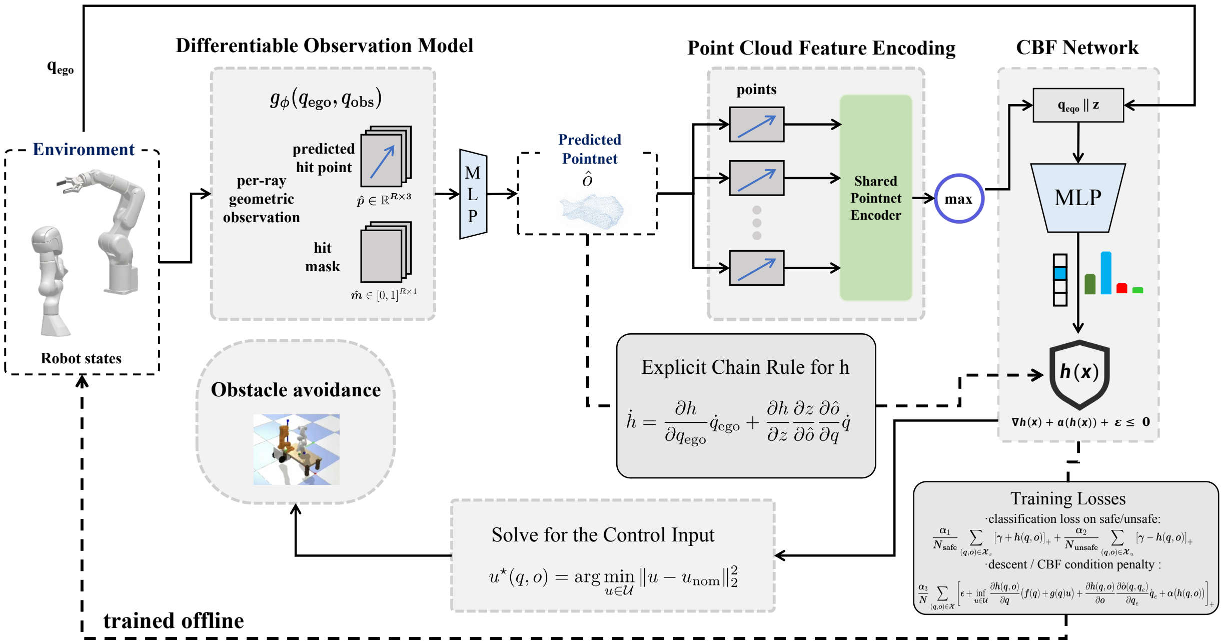
**Problem 1.** *Given a robot modeled by system* (1) *with the observation model* (2) *in a dynamic environment* $\mathcal{E}$*, under Assumption 1, synthesize an online perception-based safe controller* $u = u(q, o)$ *such that, for any initial condition* $(q(0), o(0)) \in \mathcal{C}$*, the resulting closed-loop trajectory satisfies*

$$(q(t), o(t)) \in \mathcal{C} \subseteq \mathcal{X}_s, \quad \forall\, t \geq 0, \tag{6}$$

*i.e., the induced safe set* $\mathcal{C} = \{(q, o) \in \mathcal{X} : h(q, o) \geq 0\}$ *is forward invariant and the robot remains collision-free.*

In order to concretize the above problem setting, we provide the following illustrative example, which will serve as a case study throughout the paper.

**Example 1.** *Consider a mobile flexible manipulator with configuration q operating in a shared workspace with a second manipulator that is executing its own task autonomously and thus acts as a dynamic obstacle, illustrated in Figure 1. This setting is motivated by tightly integrated multi-robot operation: the mobile manipulator is deployed as an agile platform to provide temporary assistance to the second manipulator, while only being able to command its own control input u and not the motion of the other manipulator. Since both manipulators are instrumented and monitored by the same platform, the second manipulator's internal states, including its joint configuration and joint velocity, are accessible. Under this setting, the mobile manipulator is required to accomplish its assigned task while guaranteeing collision avoidance with the second manipulator as it follows its own trajectory.*



$\mathbf{q}_{\text{ego}}$
Differentiable Observation Model
$g_\phi(q_{\text{ego}}, q_{\text{obs}})$
predicted hit point
per-ray geometric observation
$\hat{p} \in \mathbb{R}^{R\times 3}$
hit mask
$\hat{m} \in [0,1]^{R\times 1}$
M L P
Predicted Pointnet
$\hat{o}$
Environment
Robot states
Point Cloud Feature Encoding
points
Shared Pointnet Encoder
max
CBF Network
$q_{\text{ego}} \,\|\, z$
MLP
$h(x)$
$\nabla h(x) + \alpha(h(x)) + \varepsilon \leq 0$
Explicit Chain Rule for h
$\dot{h} = \frac{\partial h}{\partial q_{\text{ego}}}\dot{q}_{\text{ego}} + \frac{\partial h}{\partial z}\frac{\partial z}{\partial \hat{o}}\frac{\partial \hat{o}}{\partial q}\dot{q}$
Obstacle avoidance
Training Losses
·classification loss on safe/unsafe:
$\frac{\alpha_1}{N_{\text{safe}}}\sum_{(q,o)\in\mathcal{X}_s}[\gamma + h(q,o)]_+ + \frac{\alpha_2}{N_{\text{unsafe}}}\sum_{(q,o)\in\mathcal{X}_u}[\gamma - h(q,o)]_+$
·descent / CBF condition penalty :
$\frac{\alpha_3}{N}\sum_{(q,o)\in\mathcal{X}}\left[\epsilon + \inf_{u\in\mathcal{U}}\frac{\partial h(q,o)}{\partial q}(f(q)+g(q)u) + \frac{\partial h(q,o)}{\partial o}\frac{\partial \hat{o}(q,q_e)}{\partial q_e}\dot{q}_e + \alpha(h(q,o))\right]_+$
Solve for the Control Input
$u^\star(q,o) = \arg\min_{u\in\mathcal{U}} \|u - u_{\text{nom}}\|_2^2$
trained offline

Fig. 2: **Overall pipeline of the proposed framework.** Offline, we construct a ray-based observation representation and train a differentiable surrogate observation model $\hat{o}(q, q_e)$, which is then used to learn a perception-based neural control barrier function $h(q, o)$ with an observation-dynamics-aware CBF objective. Online, the current robot state and observation are fed into the learned CBF, and a CBF-constrained QP filters the nominal control input to produce a safe action in dynamic environments.

Although this scenario contains only one single obstacle manipulator, it is still sufficiently general for two reasons. First, compared with typical obstacles like rigid bodies with simple kinematics, a manipulator features high-dimensional nonlinear dynamics and complex geometry. Second, extending from a single obstacle agent to multiple moving agents is straightforward. To this end, in what follows, we first present a general solution framework, and then instantiate and evaluate our method in the example scenario above.

## III. Methodology

Our goal is to learn a perception-based neural CBF $h(q, o)$ that satisfies the requirements i)–iii) in Definition 1, based on which we synthesize an online safe controller. A central challenge in dynamic environments is to account for the time variation of perception $\dot{o}$. Under Assumption 1, we denote by $q_e$ the internal state of a moving obstacle agent and by $\dot{q}_e$ its velocity, which leads to

$$\frac{\partial o}{\partial t} = \frac{\partial o(q, q_e)}{\partial q_e}\,\dot{q}_e. \tag{7}$$

To address this, we first specify a ray-based point-cloud observation representation $o$, then train a differentiable surrogate observation model $\hat{o}(q, q_e)$ to enable stable evaluation of observation sensitivities. Based on $\hat{o}$, we develop a training procedure for the neural CBF that enforces the dynamic CBF condition, and finally present an online controller synthesis via a quadratic program (QP) using the learned barrier. An overview of the proposed framework is shown in Fig. 2.

### A. Observation Representation for Online Perception

To enable online synthesis in dynamic environments with complex obstacle geometry, we build safety directly from perception, enabling an end-to-end pipeline from sensing to control. Although Assumption 1 provides access to internal states of obstacles, converting high-dimensional states into collision-relevant quantities still requires accurate geometry and nontrivial modeling for nonconvex shapes.

In this paper, we represent perception as a ray-based point cloud (e.g., obtained from LiDAR or a depth camera). At each control step, a set of $R$ rays is emitted from a sensor mounted on the robot, producing an observation $o = o(q, \mathcal{E}) \in \mathcal{O} \subset \mathbb{R}^k$ as in (2). For each ray $i \in \{1, \ldots, R\}$, we record the first intersection point $p_i \in \mathbb{R}^3$ and a binary hit indicator $m_i \in \{0, 1\}$. If no intersection occurs within range, $m_i = 0$ and $p_i$ is set to a fixed placeholder. We then represent the point-cloud observation in a fixed-size tensorized form:

$$o = \{(p_i, m_i)\}_{i=1}^{R} \in \mathbb{R}^{R\times 4},$$

where the valid surface points correspond to $\{p_i \mid m_i = 1\}$. The resulting point cloud is illustrated in Figure 3.

### B. Learning a Differentiable Observation Function

Although the above ray-based sensing provides geometric information, but such observation is not differentiable with

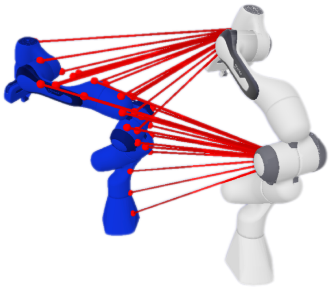


Fig. 3: Ray-based point-cloud observation. A set of rays is emitted from sensors mounted on the robot manipulator (white) toward the workspace. The first intersection points on the obstacle agent (blue) together with hit indicators form the tensorized observation $o = \{(p_i, m_i)\}_{i=1}^R$.

respect to the environmental states due to discontinuities. To address this, we introduce a differentiable surrogate observation model that approximates the point-cloud observation as a smooth function of configurations.

Specifically, we learn a parametric mapping

$$\hat{o} \;=\; \varphi(q, q_e),$$

Following the ray-based representation in (8), the predicted observation is $\hat{o} = \{(\hat{p}_i, \hat{m}_i)\}_{i=1}^R$, where $\hat{p}_i \in \mathbb{R}^3$ and $\hat{m}_i \in (0, 1)$ denote the predicted intersection point and hit probability, respectively. Training data are generated in simulation by sampling robot and obstacle configurations and running ray-based sensing to obtain the corresponding point-cloud observations:

$$\mathcal{D} = \{(q^{(k)}, q_e^{(k)}, o^{(k)})\}_{k=1}^N.$$

The surrogate is trained by minimizing a loss that combines per-ray hit classification and hit-masked point regression:

$$\mathcal{L}_{obs} = \frac{1}{R}\sum_{i=1}^{R} \mathrm{BCE}(\hat{m}_i, m_i) + \frac{\sum_{i=1}^R m_i\, \rho_\delta(\|\hat{p}_i - p_i\|_2)}{\sum_{i=1}^R m_i + \epsilon}, \tag{8}$$

where $\rho_\delta(\cdot)$ is the Huber penalty and $\epsilon > 0$ prevents division by zero. Importantly, the mask $m_i$ ensures that point regression is applied only to valid rays, while the BCE term encourages accurate prediction of hit events.

In practice, $\varphi$ is implemented as a fully-connected multilayer perceptron (MLP). The network takes the concatenated states $(q, q_e)$ as input and outputs per-ray geometric quantities for a fixed set of $R$ rays, including hit-point positions $\hat{p}$ and hit probabilities $\hat{m}_i$. Several hidden layers with SiLU activations are used, followed by a linear output layer producing concatenated ray-wise predictions.

### C. Learning Neural Control Barrier Function

Inspired by the offline training strategy for observation-based neural CBFs in [14], we train a neural CBF $h(q, o)$ from a pre-collected dataset. Following Definition 1, the objective is to learn $h$ such that it (i) separates safe and unsafe regions with a margin and (ii) admits a feasible control input that satisfies the CBF condition.

The training set is generated from two sources, similar in spirit to [14]: (i) we collect rollout trajectories produced by classical LQR controllers from randomly sampled initial and goal configurations; and (ii) to improve coverage beyond visited rollouts, we additionally sample robot configurations uniformly in $\mathcal{Q}$ and record the associated observations.

In the dynamic environment, condition iii) in Definition 1 depends on the observation variation rate $\dot{o}$. Based on the differentiable surrogate $\hat{o}(q, q_e)$, we have $\dot{o} \approx \frac{\partial \hat{o}(q,q_e)}{\partial q_e}\dot{q}_e$. Let $N_{\text{safe}}$, $N_{\text{unsafe}}$, and $N$ denote the number of training samples in $\mathcal{X}_s$, $\mathcal{X}_u$, and $\mathcal{X}$, respectively, and $[\cdot]_+ := \max(0, \cdot)$. We train the neural CBF by minimizing the empirical loss (9), where $\alpha_1, \alpha_2, \alpha_3 > 0$ are weighting coefficients and $\gamma, \epsilon > 0$ are margins. The first two terms enforce separation between $\mathcal{X}_s$ and $\mathcal{X}_u$, while the third term penalizes violations of the CBF condition using the observation-dynamics approximation. The feasibility of the CBF constraint is encouraged by driving the third hinge term to zero. Moreover, the inner infimum over $u$ is tractable since it is affine in $u$ and can be minimized over a bounded action set.

**Remark 1.** *In [14], the Lie-derivative computation assumes obstacle motion is sufficiently slow and thus ignores observation changes induced by the environment when evaluating $\dot{o}$; moreover, $\partial h/\partial q$ is obtained via numerical differentiation. These approximations may degrade safety when obstacle agents exhibit non-negligible motion, particularly under complex, high-dimensional dynamics. In contrast, our formulation retains the observation-dynamics term in the CBF residual and estimates it through the differentiable surrogate $\hat{o}(q, q_e)$, thereby explicitly incorporating obstacle motion into both training and online safety synthesis.*

### D. Online Safe Controller Synthesis

Given the learned neural CBF $h(q, o)$, we synthesize a safe control input online by filtering a nominal controller through a CBF-constrained quadratic program (QP). Let $u_{\text{norm}}$ denote a nominal controller that achieves the reference manipulation objective. At each time step, we solve a QP optimization problem that finds the control input closest to $u_{\text{norm}}$ while satisfying the CBF condition.

Specifically, the safe controller is obtained by

$$\begin{aligned} u^\star(q, o) &= \arg\min_{u\in\mathcal{U}} \; \|u - u_{\text{norm}}\|_2^2 \\ \text{s.t.}\;\; &\frac{\partial h}{\partial q}\big(f(q) + g(q)u\big) + \frac{\partial h}{\partial o}\frac{\partial \hat{o}(q, q_e)}{\partial q_e}\dot{q}_e \geq -\alpha\, h(q, o) + \epsilon. \end{aligned} \tag{10}$$

When $\mathcal{U}$ is a box constraint, (10) is a standard convex QP that can be solved efficiently at control frequency. Moreover, all partial derivatives are evaluated online via reverse-mode automatic differentiation (i.e., backpropagation) through the learned networks, and the required Jacobian-vector products are computed without explicitly forming full Jacobian matrices, enabling real-time deployment.

Here we remark that the above controller is *online and perception-based* due to the following reasons. First, the optimization in (10) is solved at every time step using the current robot configuration and the latest observation and

$$
\begin{aligned}
\mathcal{L} = \frac{\alpha_1}{N_{\text{safe}}} \sum_{(q,o)\in\mathcal{X}_s} [\gamma + h(q,o)]_+ + \frac{\alpha_2}{N_{\text{unsafe}}} \sum_{(q,o)\in\mathcal{X}_u} [\gamma - h(q,o)]_+ \\
+ \frac{\alpha_3}{N} \sum_{(q,o)\in\mathcal{X}} \left[ \epsilon + \inf_{u\in\mathcal{U}} \frac{\partial h(q,o)}{\partial q}(f(q)+g(q)u) + \frac{\partial h(q,o)}{\partial o}\frac{\partial \hat{o}(q,q_e)}{\partial q_e}\dot{q}_e + \alpha(h(q,o)) \right]_+
\end{aligned} \tag{9}
$$

no precomputed trajectory or global planner is required. Second, the safety certificate $h(q, o)$ and its constraint are evaluated directly from the sensed point-cloud observation $o = o(q, q_e)$, rather than from explicit geometric distances or full environment reconstruction. Moreover, obstacle motion is also incorporated in real time through the observation dynamics term, using the differentiable surrogate $\hat{o}(q, q_e)$. Therefore, the QP filter adapts immediately to changes in the dynamic environment and produces control actions that remain consistent with the learned safety condition.

## IV. Experimental Validation

In this section, we evaluate the proposed framework in a physics-based simulation with 7-DoF Franka Panda manipulators. We consider kinematic-level control and use joint velocities as the control input. The dynamic environment follows the setting of Example 1, that is, in addition to the robot manipulator, a second Franka Panda operates in the same workspace and executes its own predefined task autonomously, acting as a moving obstacle agent. The robot must accomplish its task while guaranteeing collision avoidance with the obstacle manipulator throughout its motion.

### A. Experimental Setting

**Simulation environment and tasks.** All experiments are conducted in a physics-based simulator in `PyBullet` with two 7-DoF Franka Panda manipulators sharing the same workspace. The evaluated robot is required to execute a grasp manipulation task, while maintaining collision avoidance throughout execution. The obstacle manipulator follows its own predefined task autonomously; across trials, we randomize its task execution by randomizing initial configurations and target choices to induce diverse, time-varying interaction patterns. This setup yields a challenging dynamic environment with frequent changes in relative geometry.

**Baseline.** We compare against the baseline adapted from the observation-based neural CBF method of [14]. We focus on this work for two reasons: (i) it provides a strong and representative reference for perception-based neural CBF learning, and (ii) it has demonstrated competitive performance in static-obstacle settings. Our goal is therefore to isolate the impact of explicitly modeling observation dynamics and to stress-test both methods in the more challenging regime where obstacle agents move with nontrivial, high-dimensional dynamics.

**Evaluation metrics.** We report (i) *task success rate*, defined as the fraction of trials that reach the goal within a fixed time horizon, and (ii) *safety rate*, defined as the fraction of trials that remain collision-free for the entire episode.

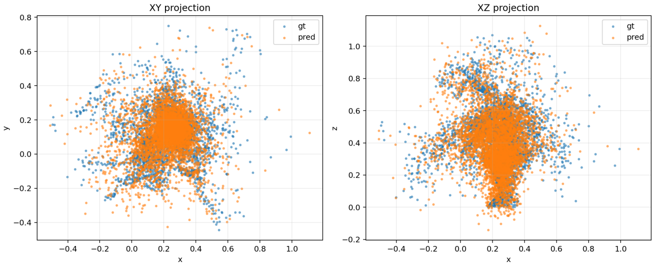
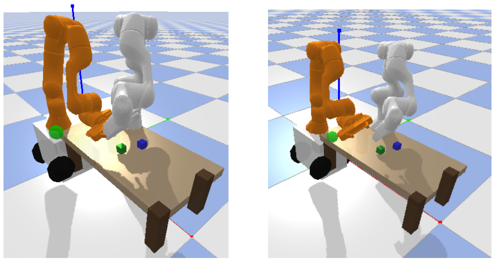


Fig. 4: Ground truth hit-point positions vs. predictions from the learned differentiable observation function $\hat{o}(q, q_e)$.



(a) baseline

(b) ours

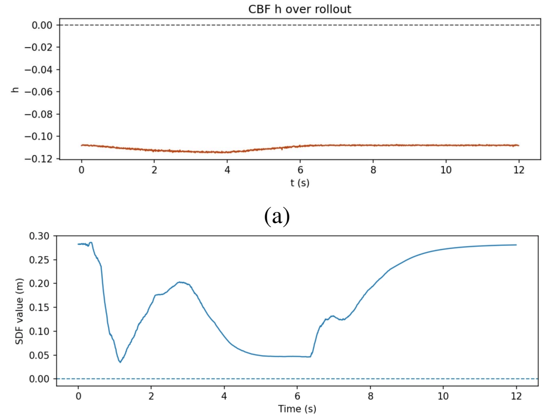
Fig. 5: Snapshot comparison of the baseline and our approach during closed-loop execution. The baseline collides with the obstacle manipulator, whereas our approach maintains safe separation throughout the task.

### B. Performance of Obervsation Approximation

Before the comparison test, we first evaluate the learned differentiable observation surrogate $\hat{o}(q, q_e)$ and its ability to approximate ray-based point-cloud measurements in dynamic scenes. Figure 4 compares the predicted hit-point positions from $\hat{o}(q, q_e)$ with the corresponding ground-truth raycast intersections. As shown, the predictions closely match the ground truth for the vast majority of points, indicating that the surrogate provides an accurate and smooth approximation of the observation mapping that is suitable for downstream computation of observation sensitivities.

### C. Ablation Study in Simulation

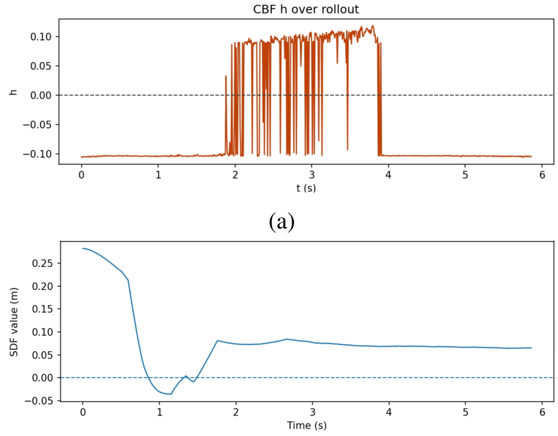
We conduct ablation studies to isolate the effect of explicitly modeling observation dynamics in the neural CBF condition. In these experiments, the robot executes the task under the same nominal controller, while a moving obstacle manipulator follows its own prescribed trajectory, inducing time-varying collision risks and rapidly changing observations. We evaluate both task-level outcomes (success/failure) and trajectory-level safety signals, including the evolution of



(a)

(b)

Fig. 6: Comparison of the minimum separation distance between the two manipulators. The baseline violates safety, i.e., distance drops below zero, whereas our approach maintains a positive separation throughout the rollout.



(a)

(b)

Fig. 7: Comparison of neural control barrier function values during a representative rollout. The baseline becomes unsafe as the CBF value rises above zero and exhibits large oscillations near constraint violation, while our approach keeps the CBF value strictly negative throughout execution.

the barrier value $h(t)$ and the signed distance function (SDF), i.e., the minimum inter-manipulator separation distance.

Figure 5 provides a qualitative snapshot comparison during closed-loop execution: the baseline fails by colliding with the obstacle manipulator, whereas our approach maintains a clear safety margin throughout the task. To quantify safety behavior over time, Figure 6 plots the minimum separation distance. The baseline violates safety as the distance drops below zero (indicating penetration), while our approach keeps the distance positive for the entire rollout. Figure 7 further compares the barrier values: the baseline becomes unsafe when $h(t)$ rises above zero and exhibits large oscillations near the constraint boundary, whereas our approach keeps $h(t)$ strictly negative, consistent with a stablecertificate.

Finally, Table I summarizes results over 1000 trials. Overall, explicitly incorporating observation dynamics improves closed-loop safety behavior and reduces constraint violations under dynamic obstacle motion.

TABLE I: Statistical Results of 1000 Trials

| Method | Success Rate | Safety Rate |
|---|---|---|
| baseline | 0.868 | 0.64 |
| ours | 0.968 | 0.82 |

## V. Conclusion

This paper presented an online, perception-based neural control barrier function framework for safe manipulation in dynamic environments. We develop a learning-and-control pipeline that makes the CBF constraint responsive to obstacle motion by introducing a differentiable observation surrogate and using it to compute the additional terms required in the barrier derivative. With the learned neural CBF, we synthesize safe controls through a QP filter that minimally modifies a nominal command while enforcing safety directly from point-cloud observations. Experimental results demonstrate improved safety and constraint satisfaction over a competitive perception-based neural CBF baseline.

## References

[1] Aaron D Ames, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs with application to adaptive cruise control. In *53rd IEEE conference on decision and control*, pages 6271–6278. IEEE, 2014.

[2] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2016.

[3] Xiangru Xu, Paulo Tabuada, Jessy W Grizzle, and Aaron D Ames. Robustness of control barrier functions for safety critical control. *IFAC-PapersOnLine*, 48(27):54–61, 2015.

[4] Quan Nguyen and Koushil Sreenath. Exponential control barrier functions for enforcing high relative-degree safety-critical constraints. In *2016 American Control Conference (ACC)*, pages 322–328. IEEE, 2016.

[5] Charles Dawson, Zengyi Qin, Sicun Gao, and Chuchu Fan. Safe nonlinear control using robust neural lyapunov-barrier functions. In *Conference on Robot Learning*, pages 1724–1735. PMLR, 2022.

[6] Wei Xiao, Ross Allen, and Daniela Rus. Safe neural control for non-affine control systems with differentiable control barrier functions. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 3366–3371. IEEE, 2023.

[7] Wei Xiao, Tsun-Hsuan Wang, Ramin Hasani, Makram Chahine, Alexander Amini, Xiao Li, and Daniela Rus. Barriernet: Differentiable control barrier functions for learning of safe robot control. *IEEE Transactions on Robotics*, 39(3):2289–2307, 2023.

[8] Bolun Dai, Rooholla Khorrambakht, Prashanth Krishnamurthy, and Farshad Khorrami. Differentiable optimization based time-varying control barrier functions for dynamic obstacle avoidance. *Robotics and Autonomous Systems*, page 105182, 2025.

[9] Zhuozhu Jian, Zihong Yan, Xuanang Lei, Zihong Lu, Bin Lan, Xueqian Wang, and Bin Liang. Dynamic control barrier function-based model predictive control to safety-critical obstacle-avoidance of mobile robot. In *2023 IEEE international conference on robotics and automation (ICRA)*, pages 3679–3685. Ieee, 2023.

[10] Manan Tayal, Rajpal Singh, Jishnu Keshavan, and Shishir Kolathaya. Control barrier functions in dynamic uavs for kinematic obstacle avoidance: A collision cone approach. In *2024 American Control Conference (ACC)*, pages 3722–3727. IEEE, 2024.

[11] Golnaz Raja, Teemu Mökkönen, and Reza Ghabcheloo. Safe robot control using occupancy grid map-based control barrier function (ogm-cbf). *arXiv preprint arXiv:2405.10703*, 2024.

[12] Yuepeng Zhang, Yu Chen, Yuda Li, Shaoyuan Li, and Xiang Yin. Online synthesis of control barrier functions with local occupancy grid maps for safe navigation in unknown environments. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 14884–14891. IEEE, 2025.

[13] Ryan K Cosner, Ivan D Jimenez Rodriguez, Tamas G Molnar, Wyatt Ubellacker, Yisong Yue, Aaron D Ames, and Katherine L Bouman. Self-supervised online learning for safety-critical control using stereo vision. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 11487–11493. IEEE, 2022.

[14] Mingxin Yu, Chenning Yu, M-Mahdi Naddaf-Sh, Devesh Upadhyay, Sicun Gao, and Chuchu Fan. Efficient motion planning for manipulators with control barrier function-induced neural controller. In *2024 IEEE international conference on robotics and automation (ICRA)*, pages 14348–14355. IEEE, 2024.