Enforcing safety for vision-based controllers via Control Barrier Functions and Neural Radiance Fields

Mukun Tong
Dept. of Automation
Tsinghua University
Beijing, China
tmk19@mails.tsinghua.edu.cn

Charles Dawson
Dept. of Aeronautics and Astronautics
MIT
Cambridge, USA
cbd@mit.edu

Chuchu Fan
Dept. of Aeronautics and Astronautics
MIT
Cambridge, USA
cbd@mit.edu

***Abstract*—To navigate complex environments, robots must increasingly use high-dimensional visual feedback (e.g. images) for control. However, relying on high-dimensional image data to make control decisions raises important questions; particularly, how might we prove the safety of a visual-feedback controller? Control barrier functions (CBFs) are powerful tools for certifying the safety of feedback controllers in the state-feedback setting, but CBFs have traditionally been poorly-suited to visual feedback control due to the need to predict future observations in order to evaluate the barrier function. In this work, we solve this issue by leveraging recent advances in neural radiance fields (NeRFs), which learn implicit representations of 3D scenes and can render images from previously-unseen camera perspectives, to provide single-step visual foresight for a CBF-based controller. This novel combination is able to filter out unsafe actions and intervene to preserve safety. We demonstrate the effect of our controller in real-time simulation experiments where it successfully prevents the robot from taking dangerous actions.**

***Index Terms*—safe vision-based control, control barrier functions, neural radiance fields**

I. Introduction & Related Work

Robots are increasingly making use of visual feedback to explore novel, complex environments, particularly in GPS-denied environments, and recent works have demonstrated the potential for using this visual feedback directly for control [1], [2]. However, before we can use visual data to make control decisions in safety-critical robotic systems, we must be able to certify the safety of vision-based controllers.

Unfortunately, ensuring safety for visual-feedback controllers is a challenging problem, since this issue lies in the intersection of control theory and computer vision. To solve this problem, we bring together two powerful concepts from each of these disciplines. Control barrier functions (CBFs) are powerful tools from control theory for proving the safety of a feedback controller [3], [4]; in contrast with reachability-based approaches to safety verification [5], CBFs require us to consider only a single-step horizon, reducing computation time and allowing them to be used for real-time control. Unfortunately, CBFs typically require a known model for how control actions affect future measurements of the CBF. For systems with state feedback, such a model is often available, and approximate models are available for some types of high-dimensional observations such as Lidar [6], but the lack of a predictive model for visual feedback has so far prevented the use of CBFs in the visual-feedback setting. To close this gap, we turn to neural radiance fields (NeRFs), a recently developed technique from the computer vision community where a neural network is used to learn an implicit representation of a 3D scene [7]. NeRFs can be used to re-render a scene from a previously-unseen perspective, allowing a robot to predict the image it would observe if it were to move from its current viewpoint. Recent work on NeRF-enabled SLAM [8], [9] has shown that neural implicit scene representations can be trained online for novel environments, opening up a range of robotics applications for this technology.

Our key insight in this paper is that implicit neural scene representations (like NeRFs) can provide *foresight* for vision-based controllers; they allow a robot to predict the effect of its actions on future image observations, allowing it to filter out actions that produce unsafe future observations. This insight leads us to the following contributions:

1) We present a novel CBF-based controller that ensures safety for a controller operating solely on high-dimensional visual feedbacks.
2) We combine CBFs with NeRFs to enable single-step visual foresight, allowing our controller to determine whether an action is safe or unsafe using only a single-step horizon.
3) We demonstrate that this controller can run in simulation at real-time rates (improving on previous NeRF-based motion controllers by several orders of magnitude).

C. Dawson is supported by the NSF GRFP under Grant No. 1745302. The Defense Science and Technology Agency in Singapore and IBM provided funds to assist the authors with their research, but this article solely reflects the opinions and conclusions of its authors and not DSTA Singapore, the Singapore Government, or IBM.

II. Background & Preliminaries

In this section we will briefly introduce NeRFs and CBFs, upon which our visual-foresight CBF controller builds.

A. Neural Radiance Fields

Neural Radiance Fields (NeRFs) are a recently-developed method to synthesize views of 3D scenes by learning a latent, continuous representation of a 3D volume. NeRFs represent a scene with a fully connected deep network that predicts the density and color of points in 3D space as a function of location and viewing direction. A NeRF is typically trained via a reconstruction loss on a series of example images, and the scene can be re-rendered from a new perspective using differentiable ray-tracing [7].

Recent works such as iMAP [9] and NICE-SLAM [8] have applied NeRFs to SLAM problems in robotics. By relying on a pre-trained network to warm-start the learning process, NeRFs can be fine-tuned online to represent novel environments, while simultaneously optimizing the parameters of the scene representation and the estimated camera pose allows a robot to simultaneously estimate its own position. Other works have attempted to use NeRFs as the basis for a trajectory optimization pipeline [10], but since trajectory optimization over a long horizon requires hundreds of calls to the underlying NeRF, these techniques are extremely computationally expensive.

B. Control Barrier Functions

Safety is a recurring problem in robotics: given a potentially nonlinear dynamical system, how can we choose control inputs to ensure that this system will remain safe? Control barrier functions (CBFs) have emerged to handle this problem analogously to how Control Lyapunov functions (CLFs) may be used to prove the stability of a control system [3], [4]. We will briefly review the theory of CBFs here, but for a more thorough survey the reader is referred to [4].

Consider a discrete-time dynamical system with state $x \in \mathcal{X} \subseteq \mathbb{R}^n$ and control $u \in \mathcal{U} \subseteq \mathbb{R}^m$, dynamics $x_{t+1} = f(x, u)$, and observations in the form of $w \times h$-size RGBd images of the robot's environment $y = o(x) \in \mathbb{R}^{w \times h \times 4}$. CBFs have been applied previously to Lidar feedback [6], but we are not aware of any other work considering the case of visual feedback without assuming access to a bounded-error state-estimation oracle [11]. Let us denote as $\mathcal{X}_u \subset \mathcal{X}$ a set of *unsafe* states that is disjoint from the set of admissible initial conditions $\mathcal{X}_0 \subseteq \mathcal{X} \backslash \mathcal{X}_u$, and let us assume that we have some function h that can distinguish between safe and unsafe states based on the observations y; i.e. $h(o(x)) > 0 \iff x \in \mathcal{X}_u$ and $h(o(x)) \leq 0$ otherwise. Further, let us assume that we have access to some *nominal policy* $\pi_0(y)$. This nominal policy may be wildly unsafe, and so the goal of a CBF is to find some small control intervention that ensures the system remains safe.

The key result from CBF theory [12] is that this safety-preserving control action can be found by solving the optimization problem:

$$u = \arg\min_{u} \quad ||u - \pi_0(y_t)||^2 \tag{1a}$$

$$\text{s.t.} \quad h \circ o \circ f(x_t, u) \leq \alpha h \circ o(x) \tag{1b}$$

where $\circ$ denotes function composition and $\alpha \in (0, 1)$ is a rate-determining constant. If a feasible u can always be found, then h is a valid CBF and acts as a certificate to prove the safety of the control system. In this case, h can be used to "filter" any potentially unsafe nominal policy, as might arise through the use of reinforcement learning or human teleoperation.

Of course, since the constraint (1b) requires predicting the image $o \circ f(x_t, u)$ that would be observed if the robot were to take action u, CBFs have not previously been used in the direct visual-feedback setting. Some papers make promising steps towards understanding the case where visual feedback is used to for state estimation alongside a traditional state-feedback controller [11], but in this paper we attempt to solve the more challenging problem where the CBF and controller are defined directly in the space of visual observations. The main issue that arises when defining the CBF as a function of state is poor generalization. Previous works have shown that CBFs defined as functions of state do not generalize well to new environments (e.g. when obstacles have moved), whereas CBFs defined as functions of observations are naturally conditioned on the observed state of the environment [6].

III. Problem Statement & Assumptions

Our goal in this paper is to adapt the CBF safety filter framework described in Eq (1a) and [3], [4], [12] for use in the visual-feedback setting. In this section, we state the control objectives and assumptions used in the rest of this paper.

As the robot navigates its environment, it builds an implicit scene representation in the form of a NeRF (which is partly pre-trained to enable fast adaptation). As the robot explores, the controller must ensure safety, i.e. avoiding any collisions with the environment. To achieve this goal using a safety filter like Eq. (1a), the filter must be able to tell whether an intended control action will bring the robot dangerously close to any nearby obstacle. If the intended action is found to be unsafe, the filter must find the smallest control intervention that still preserves safety. We can formalize these requirements as:

1) *Safety:* the robot must remain a sufficient distance from any obstacles visible nearby; i.e. given an RGBd image $y \in \mathbb{R}^{w \times h \times 4}$, $\min_{0 \leq i \leq w;\ 0 \leq j \leq h} y[i, j]_{depth} > d_c$, where $d_c > 0$ is the minimum safe clearance.
2) *Minimal intervention:* the safety filter should minimize any deviation from the nominal control policy, insofar as is possible without violating the safety requirement. This requirement is naturally captured by the CBF-QP with objective (1a) and constraint (1b).

We note that our definition of safety is limited to what can be observed by the robot via its cameras, so we do not consider the case when a robot can collide with an obstacle in its blind spot. This is an important case that deserves additional study, but it is outside the scope of this paper; we focus only on the case when safety can be defined purely as a function of the robot's observations. We also note that our approach can be easily extended to the case when the robot has multiple cameras, so covering these blind spots would be as simple as adding additional cameras in practice.

Finally, we would like to design our safety filter so that it can be easily adapted to robots with different dynamics. To do this, we will make use of a *two-level control architecture*, similar to that used in other works on formal methods for robot safety [13], [14]. In this architecture, we apply our visual-feedback safety filter at the top level with an abstracted model of the robot's dynamics (e.g. a single- or double-integrator or a Dubins car), then use a low-level controller to track the top-level state using the full nonlinear dynamics of the system. A number of techniques exist for deriving lower-level controllers with guaranteed tracking bounds [13], [15], and so we focus on deriving a CBF visual-feedback safety filter for the high level controller with simplified plant dynamics (the error bound of the tracking controller drives the choice of the safety margin d_c in our framework, ensuring that safety is still maintained).

IV. Approach

We will first provide an overview of our control architecture, then provide more detail on the two main subsystems: the observation-space CBF used to maintain safety and the NeRF used to predict future values of that CBF.

A. Overview

The algorithm for our visual CBF safety filter is provided in Algorithm 1, and the interaction between the neural-implicit scene representation (for which we use the NICE-SLAM implementation [8]) and the CBF is illustrated in Fig. 1. Algorithm 1 proceeds as follows: first, the most recent observation is used to update both the state estimate and neural scene representation using the NICE-SLAM subsystem [8]. Then, starting with the nominal control input $u_t = \pi_0(y_t)$, we predict the next state and observation given this control input using the dynamics and neural scene representation, respectively. We then use the CBF to decide whether this control action is safe; if it is, we accept this control action, but if it is not we re-sample randomly in the vicinity of u_0 to find new candidate control actions. In practice, instead of re-sampling control actions one-at-a-time if the nominal policy is unsafe, we evaluate multiple candidate interventions in parallel to speed execution time and find a safe candidate close to the nominal control action.

B. Visual Control Barrier Functions

The CBF $h(y) : \mathbb{R}^{w \times h \times 4} \mapsto \mathbb{R}$ is used to ensure that the system remains safe. In order to be useful, a CBF must not only distinguish safe ($h < 0$) and unsafe ($h > 0$) situations,

Algorithm 1 The visual-foresight CBF safety filter. θ_t denotes the model parameters of the neural scene representation, which are fine-tuned at each step.

Require: Image y_t, nominal policy π_0, NeRF parameters θ_{t-1}
  $u_0, u \leftarrow \pi_0(y_t)$, *safe* $\leftarrow$ FALSE
  $\theta_t, \hat{x}_t \leftarrow \text{NSUPDATE}(y_t, \theta_{t-1})$ ▷ update model & state
  while not *Safe* **do**
    $\hat{x}_{t+1} \leftarrow f(\hat{x}_t, u)$ ▷ predict state
    $\hat{y}_{t+1} \leftarrow \text{NSPREDICT}(\hat{x}_{t+1}, \theta_t)$ ▷ predict observation
    $h_t \leftarrow h(y_t),\ h_{t+1} \leftarrow h(\hat{y}_{t+1})$ ▷ evaluate CBF
    if $h_{t+1} \leq \alpha h_t$ **then**
      safe $\leftarrow$ TRUE, **return** u, *safe*
    else
      safe $\leftarrow$ FALSE, $u \leftarrow \text{RE-SAMPLE}(u_0)$
    end if
  end while

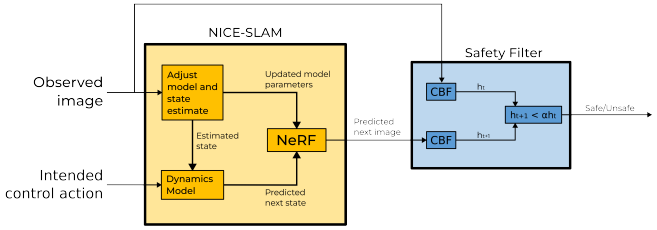


Fig. 1: The NeRF-enabled safety filter used in our control architecture. The NICE-SLAM subsystem [8] is used to predict future RGBd observations using a neural implicit representation of the scene, and a CBF uses those predictions to decide if a control action is safe. If the intended control action is unsafe, our algorithm searches for nearby control actions that are safe (not shown, but described in Algorithm 1).

but it must also generalize well to new environments. This need for generalization is our key motivation for defining the CBF as a function of observations. Previous work [6] has shown that observation-space CBFs generalize much better than state-based CBFs; the intuition is that if an obstacle moves in the scene, then previously-safe states may become unsafe, but an image of the obstacle will always be a reliable indicator of how far away it is.

In keeping with our use of a two-layer control architecture, as discussed in Section III, we will define a CBF for two common simplified dynamics models, the single- and double-integrators, which we can then combine with a lower-level tracking controller to ensure the safety of robots with a wide range of possible dynamics [13], [14]. When using first-order dynamics, we use the CBF $h(y) = d_c - \min_{i,j} y[i,j]_{depth}$. When using second-order dynamics, we can augment the CBF with an estimate of the robot's velocity: $h(y, \hat{v}) = d_c - \min_{i,j} y[i,j]_{depth} + \beta||\hat{v}||$ where $\beta > 0$ is an empirically-tuned hyperparameter.

C. Neural Implicit Scene Representation and State Estimation

Our safety filter in Algorithm 1 relies on the ability not only estimate the robot's position and build a representation of the scene (NSUPDATE) but also predict what observations the robot *could* receive if it were to take a particular action (NSPREDICT). We rely on recently-developed NeRF-based tools for both of these functions. In particular, we integrate the NICE-SLAM (Neural Implicit Scalable Encoding for SLAM) framework presented in [8] into our safety filter. The core of NICE-SLAM is a set of pre-trained NeRFs that are retrained online to build a representation of the scene as the robot explores. Since NeRF rendering is differentiable, NICE-SLAM is able to optimize both the parameters of its NeRFs (the mapping step) and an estimate of the robot's pose (tracking); NSUPDATE refers to both the mapping and tracking steps.

NICE-SLAM is also able to predict the RGBd image the robot would likely observe if it were to move to a new pose in the environment by simply re-rendering the NeRF from the new viewpoint (NSPREDICT). This capability allows us the robot to imagine what observations it would receive as a result of taking different actions, so the robot can proactively determine whether an action would be unsafe and avoid it. Calls to NSPREDICT can be easily batched and parallelized on GPU to reduce the time needed to execute Algorithm 1.

V. EXPERIMENTS

In this section, we present experimental results to characterize our proposed safety filter, justify its key design decisions, and understand the effect of hyperparameters on controller performance. We compare our controller with the NERF-NAVIGATION framework, which maintains safety by solving a trajectory optimization problem using the pointwise density of the NeRF scene representation as a proxy for collision cost. To our knowledge, NERF-NAVIGATION is the only other published work using NeRFs for robot motion control, but we find that it has several performance and reliability issues.

To evaluate our safety filter, we use a simple *teleoperation* setting where a human is remotely piloting the robot through an environment. In this setting, the human provides a convenient source of nominal policy $\pi_0(y)$: the human observes the feed from the robot's camera and makes control decisions. Since these actions may be unsafe, we must use a safety filter to prevent the human from causing a collision. To create a fair comparison, we adapt NERF-NAVIGATION for this use case by attempting to find a trajectory that stays close to the human's intended motion over a short horizon.

In the following experiments, we are interested in three key metrics: the magnitude of the action intervention $\Delta u = ||u - \pi_0(y)||$, the minimum clearance $d_{min} = \min_{i,j} y[i,j]_{depth}$, and the time required to evaluate the control policy.

A. Experimental Setup

a) Environments: We conduct experiments in the `Replica Room1` indoor environment included in the Replica dataset [16].

b) Test Procedure: We initialize the robot facing towards a wall at a distance of approximately $2.5\,\mathrm{m}$. We then provide a worst-case nominal control input driving the robot directly towards the wall and observe its behavior (this nominal control input is $1\,\mathrm{m/s}$ towards the wall for the single integrator and $1\,\mathrm{m/s^2}$ towards the wall for the double integrator).

c) Implementation Details: We conduct our experiments on a desktop PC with a 2200 MHz Intel i7-10700K CPU and an NVIDIA RTX 3090 GPU. For the NICE-SLAM and NERF-NAVIGATION implementations, we use the hyperparameters provided in their respective source repositories. We run all controllers at a rate of $10\,\mathrm{Hz}$ for $10\,\mathrm{s}$, slowing the simulation clock if necessary (our control method is capable of running in real-time, but NERF-NAVIGATION is not). We sample candidate actions from a Gaussian distribution $\mathcal{N}(0, I)$, and evaluate in batches of 10. We use parameters $d_c = 0.1, \alpha = 0.5$ for the single-integrator CBF and $\beta = 1$ for the double-integrator CBF. We run all the experiments 5 times and report the average and range for all measurements.

B. Experimental Results

We study the performance of our CBF safety filter with single- and double-integrator dynamics, comparing the performance of our method with that of NERF-NAVIGATION [10].

a) Single-integrator System: Fig. 2 shows a representative series of rendered RGBd images demonstrating how our CBF controller detects when the robot approaches the wall and gradually intervenes to avoid collision. We measure the minimum distance to the nearest obstacle, the CBF value, and the intervention for each time step and plot the results in Fig. 3a. We see that our CBF safety filter does not intervene until the robot gets within 0.25 m of the wall, at which point it intervenes to bring the robot to a smooth stop. In contrast, the NERF-NAVIGATION controller (Fig. 3b) is much more conservative, applying a non-zero intervention even when the robot is more than 1 m from the wall. Moreover, NERF-NAVIGATION is not able to prevent the robot from colliding with the wall (0 m minimum separation), which may be due to the fact that the NeRF density (upon which NERF-NAVIGATION relies) is not a reliable indicator of safety; we explore this point further in Section V-C.

b) Double-integrator System: In these experiments, we observe the velocity of the robot in addition to the quantities discussed above. Fig. 4a shows the performance of our controller: we see that the controller does not intervene until the CBF value approaches 0, at which point the controller begins to decelerate, bringing the robot to a stop just before the wall. In contrast, the performance of NERF-NAVIGATION is shown in Fig. 4b; this controller is much more conservative and does not allow the robot to approach the wall at all.

c) Runtime: We compare time required to evaluate our controller and NERF-NAVIGATION at a single step. We found that NERF-NAVIGATION required more than 80 seconds to find a trajectory, which is not suitable for online

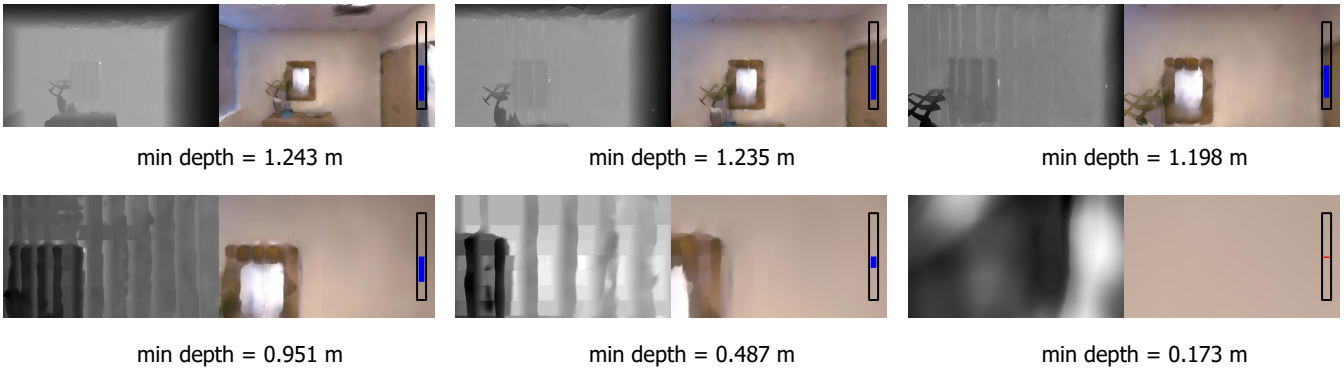
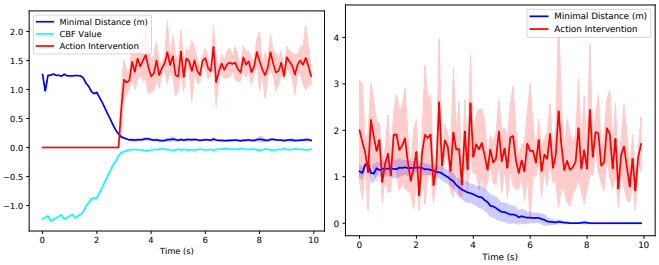
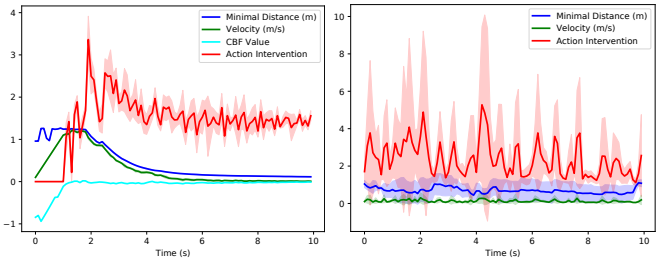


Fig. 2: Depth (left, greyscale) and RGB (right) images rendered by the NeRF showing the effect or our visual CBF safety filter in simulation with abstracted single-integrator dynamics. Images are shown $0.5\,\mathrm{s}$ apart. The bar on the right side of each image shows the current CBF value (the center of the range is $h = 0$); the bar is blue when the safety filter does not modify the intended action and red when it intervenes. The CBF value decrease as the robot approaches to the wall, and the controller successfully avoids collision.



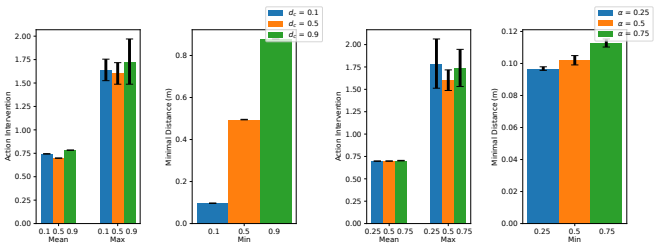
(a) CBF (b) NERF-NAVIGATION

Fig. 3: **(a)** the CBF value and action intervention over time for our CBF controller with single integrator dynamics. **(b)** a similar plot for NERF-NAVIGATION. While NeRF-Navigation fails to accurately control the robot and finally makes it crash into the wall, our method achieves the goal of constraining unsafe actions when the robot gets quite close to the wall and avoiding unnecessary intervention when it is still at a safe position.



(a) CBF (b) NERF-NAVIGATION

Fig. 4: **(a)** the CBF value, action intervention, velocity, and minimal distance over time for our CBF controller with double integrator dynamics. **(b)** a similar plot for NERF-NAVIGATION. NERF-NAVIGATION is able to stay safe only by applying large interventions and acting conservatively. Our CBF controller is able to gradually increase its intervention as the robot moves faster towards the wall, enabling it to be both safe and not conservative.

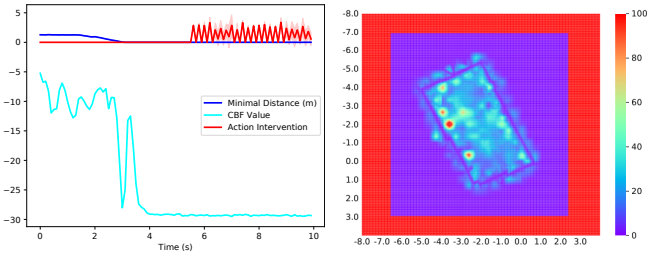


(a) d_c sweep (b) α sweep

Fig. 5: **(Left)** Mean and maximum intervention and minimum distance to collision for $d_c = \{0.1, 0.5, 0.9\}, \alpha = 0.5$, showing that the minimum distance to collision corresponds almost exactly with d_c, validating the effectiveness of our CBF method. **(Right)** Mean and maximum intervention and minimum distance to collision for $\alpha = \{0.25, 0.5, 0.75\}, d_c = 0.1$. We see that smaller values of α require larger maximum interventions, since the system must slow down sharply at the last minute when α is small. Both plots use single-integrator dynamics.

use. In contrast, our CBF safety filter can be evaluated in 0.01 s when the nominal control action is safe, and if an intervention is needed then we can evaluate a batch of 10 candidate actions in 0.095 s, suitable for online use at 10 Hz. This significant advantage in runtime stems from our use of CBFs; while NERF-NAVIGATION must consider a multi-step trajectory optimization problem (ultimately requiring more than a thousand NeRF queries), our CBF-based system uses a one-step horizon and requires only one NeRF query for each candidate action.

d) Parameter Sweep: We study the effect of varying d_c (the required safety margin) and α (governing the rate at which the CBF is allowed to approach 0). We test the performance of $d_c = \{0.1, 0.5, 0.9\}$ with constant $\alpha = 0.5$ and $\alpha = \{0.25, 0.5, 0.75\}$ with constant $d_c = 0.5$ and compare the average intervention, maximum intervention,



(a) Performance of density CBF

(b) Density map

Fig. 6: **(Left)** CBF value, intervention, and distance to collision over time for a density-based CBF controller with single integrator dynamics. We see that the CBF remains negative even when the robot collides with the wall, suggesting that NeRF density is not an appropriate indicator of safety. **(Right)** the density map of the environment at height $z = 0.5m$ helps explain this failure: the density map does not accurately capture the boundaries of the walls (density peaks at the edges of the room but then decays at the walls).

and the minimum distance to the wall observed during the experiment. Fig. 5a shows that the minimal distance in each trial matches d_c very closely, indicating that the CBF safety filter is working as intended. Fig. 5b shows that decreasing α (which allows a faster approach to the unsafe region) increases the maximum intervention required to preserve safety, as the control filter only intervenes at the last minute to preserve safety.

C. Ablation Study

Our proposed CBF safety filter relies on the rendered RGB and depth images from the NeRF, rather than using the underlying NeRF density field as a proxy for collision likelihood (as NeRF-Navigation does). In this section we justify this decision by showing that a CBF based on density alone is not sufficient to preserve safety. This failure is due to the fact that the underlying density field is not guaranteed to be a reliable indicator of safety.

To conduct this comparison, we define a density-based CBF $h(\hat{x}) = d_c + \sigma(\hat{x})$, where $\sigma(\hat{x})$ is the NeRF density at the location implied by state x. Density is positive and tends to increase in occupied regions of space, so we set d_c to a negative value determined by measuring the density near the walls; note that d_c no longer directly refers to a physical safety margin in this setting. Fig. 6a shows the performance of a density-based CBF controller with $d_c = -30$; we see that this CBF does not intervene even as the distance between the robot and the wall drops to 0. This behavior appears to be caused by a lack of a clear dependence between NeRF density and distance to the walls. We also test $d_c = -40$ (finding a similar result) and $d_c = -20$ (unable to find a safe control action by solving Eq. (1a)).

To gain a better understanding of the NeRF density field, we plot a 2D slide of the density field at a height $z = 0.5$ m in Fig. 6b. We see that certain obstacles in the environment cause high density, but the NeRF density near the walls is not particularly high. As a result, we conclude that without further development of reliable methods for training NeRFs, NeRF density is not a reliable indicator of safety.

VI. Discussion & Limitations

In this paper, we present a novel CBF-based controller that ensures safety for a controller operating solely on high-dimensional visual feedback. In particular, we combine CBFs with NeRFs to enable single-step visual foresight, allowing our controller to determine whether an action is safe or unsafe using only a single-step horizon. In comparison with previously-published NeRF-based motion controllers, which cannot run in real-time and can be overly conservative, our controller can run in simulation at real-time rates and maintains safety while limiting the magnitude of its control intervention.

Our work in this paper also presents a number of drawbacks that point to interesting directions of future work. On a practical level, the most crucial point for future work is the acceleration in search of safe actions. Our method can evaluate at most 10 control actions during each step (in order to meet a 10 Hz control frequency requirement). This time is driven largely by the time needed to render depth images from the NeRF; improvements in the ability to optimize NeRFs (even at the expense of resolution) would help our method consider more candidate actions at each step. A second important step for future research is the development of vision-based CBFs for more complex dynamics, for instance by using neural networks to adapt NeRF-based CBFs for new dynamics (as [6] explores in the context of Lidar-based CBFs). On a theoretical level, our approach in this paper ignores any error in the NeRF scene representation, which may negatively impact safety; future work should explore combining NeRF error bounds (potentially derived from statistical learning theory) with error-robust CBFs [11] to provide more solid safety guarantees.

References

[1] G. B. Margolis, T. Chen, K. Paigwar, X. Fu, D. Kim, S. b. Kim, and P. Agrawal, "Learning to jump from pixels," in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164. PMLR, 08–11 Nov 2022, pp. 1025–1034. [Online]. Available: https://proceedings.mlr.press/v164/margolis22a.html

[2] E. Malis, "Survey of vision-based robot control," *ENSIETA European Naval Ship Design Short Course, Brest, France*, vol. 41, p. 46, 2002.

[3] 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, 2017.

[4] C. Dawson, S. Gao, and C. Fan, "Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods," *arXiv preprint arXiv:2202.11762*, 2022.

[5] S. M. Katz, A. L. Corso, C. A. Strong, and M. J. Kochenderfer, "Verification of image-based neural network controllers using generative models," *Journal of Aerospace Information Systems*, vol. 19, no. 9, pp. 574–584, 2022. [Online]. Available: https://doi.org/10.2514/1.I011071

[6] C. Dawson, B. Lowenkamp, D. Goff, and C. Fan, "Learning safe, generalizable perception-based hybrid control with certificates," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1904–1911, 2022.

[7] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *European conference on computer vision*. Springer, 2020, pp. 405–421.

[8] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, "Nice-slam: Neural implicit scalable encoding for slam," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 786–12 796.

[9] E. Sucar, S. Liu, J. Ortiz, and A. Davison, "iMAP: Implicit mapping and positioning in real-time," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[10] M. Adamkiewicz, T. Chen, A. Caccavale, R. Gardner, P. Culbertson, J. Bohg, and M. Schwager, "Vision-only robot navigation in a neural radiance world," *IEEE Robotics and Automation Letters*, vol. 7, pp. 4606–4613, 2022.

[11] S. Dean, N. Matni, B. Recht, and V. Ye, "Robust Guarantees for Perception-Based Control," in *Proceedings of Machine Learning Research*, vol. 120. PMLR, jul 2020, pp. 1–11. [Online]. Available: http://proceedings.mlr.press/v120/dean20a.html

[12] A. Agrawal and K. Sreenath, "Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation," in *Proceedings of Robotics: Science and Systems*, Cambridge, Massachusetts, July 2017.

[13] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, "Fastrack: A modular framework for fast and guaranteed safe motion planning," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, pp. 1517–1522.

[14] J. Chen, J. Li, C. Fan, and B. C. Williams, "Scalable and safe multi-agent motion planning with nonlinear dynamics and bounded disturbances," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 13, pp. 11 237–11 245, May 2021. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/17340

[15] D. Sun, S. Jha, and C. Fan, "Learning certified control using contraction metric," in *Proceedings of the 2020 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. Tomlin, Eds., vol. 155. PMLR, 16–18 Nov 2021, pp. 1519–1539. [Online]. Available: https://proceedings.mlr.press/v155/sun21b.html

[16] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma *et al.*, "The replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019.