Literature Review on Control Barrier Functions

Kaustubh Kanagalekar
Department of Mechanical and Aerospace Engineering, University of California San Diego
Email: kkanagalekar@ucsd.edu

***Abstract*—As aerial drones and autonomous vehicles are gaining popularity, there has been a greater emphasis on ensuring safety for these safety-critical systems. One approach of ensuring safety is implementing a control barrier function (CBF), which acts as a filter to keep a system safe within a safe operating region. This literature review introduces the fundamentals of CBFs, and explores some common types, such as Zeroing Control Barrier Functions (ZCBFs) and Reciprocal Control Barrier Functions (RCBF). These CBFs take different approaches for enforcing safety as the system approaches the safety boundary. The review also discusses some of the potential challenges faced with implementing CBFs, such as incorporating multiple CBFs, and current approaches that address these challenges.**

***Index Terms*—CLF, CBF, QP, RCBF, ZCBF**

I. Introduction

Safety is a critical component of multi-disciplinary systems in various application domains due to the possibility of costly failures occurring. In real world environments, even small control errors could potentially result in damages and loss of life. For example, a failure in safety enforcement in autonomous vehicles could potentially lead to collisions, and insufficient safety checks in medical robotics could result in harm for patients and doctors alike. Therefore, ensuring safety in safety-critical domains, such as autonomous vehicles, industrial automation, and medical robotics, is extremely important in order to prevent catastrophic consequences.

One approach to guarantee safety is to implement a safety filter that prevents the system from taking unsafe actions. Control barrier functions (CBFs) are a type of safety filter designed to keep the system within safe boundaries at all times by implementing state-dependent constraints on control input [1]. CBFs are built upon the idea of Control Lyapunov functions (CLF), which are used for stabilising a system by bringing it towards a particular point or target. CBFs modify the implementation of a CLF by incorporating state-dependent constraints to keep the system in a safe region [2]. Incorporating constraints makes CBFs advantageous to use in various safety-critical applications where only implementing stability is inadequate. There are multiple types of CBFs with their own advantages and drawbacks, including Zeroing CBFs and Reciprocal CBFs. Zeroing CBFs define a barrier function $h(x)$ that gradually reduces as the system approaches the boundary of the safe set, whereas Reciprocal CBFs define a barrier function $B(x)$ that tends to infinity near the boundary. Both types of CBFs enforce safety by ensuring that the system remains within a safe set, although using different formulations [3].

There are some challenges associated with implementing CBFs, including handling multiple CBFs and non-smooth conditions. Conditions involving multiple CBFs could lead to conflicts in safety constraints, potentially leading to a case where the viability domain can get too restrictive or situations where a valid control input cannot be applied to the system if there are multiple conflicting control inputs [4]. When there are non-smooth conditions, it is possible that for certain points, the CBF cannot be differentiable, resulting in complications of control input and safety guarantees [5].

This paper aims to provide an in-depth review of CBFs, including their different formulations and types, and discuss the challenges associated with their implementation.

II. Related Work

This section reviews key studies and advancements in CBFs, focusing on brief formulation, types, and challenges.

A. Introduction to CBFs

Ames et al. explain the formulation of CBFs and current developments in CBFs in the paper *Control Barrier Functions: Theory and Applications*. Their work introduces the role of CBFs as a safety filter that enforces constraints on control inputs. They also discuss current applications of CBFs in various industries.

B. Different Types of CBFs

In the paper *Control Barrier Function Based Quadratic Programs for Safety Critical Systems*, Ames et al. further explore the formulation of Zeroing CBFs and Reciprocal CBFs. They discuss the differences between these CBFs.

C. Challenges of Implementing CBFs

In *Compositions of Multiple Control Barrier Functions Under Input Constraints*, Breeden et al. discuss the difficulties that arise when multiple CBFs are used simultaneously, potentially leading to conflicts in control constraints. They propose some solutions to resolve such conflicts. In *Safety-Critical Control of Discontinuous Systems with Nonsmooth Safe Sets* by Alyaseen et al., the authors discuss the complications introduced by non-smooth safe sets. Their study explores techniques to handle such cases.

III. Control Barrier Functions (CBF)

In order to understand Control Barrier Functions (CBFs), it is imperative to first understand Control Lyapunov Functions (CLFs), which can be seen as a foundation to CBFs. CLFs are used to design controllers that help stabilise the system to a desired state. The formulation of CLFs is well documented in [2], and the following discussion follows the presentation of that work.

Mathematically, let there be a nonlinear control affine system:

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

where $g(x)$ and $f(x)$ are the system dynamics, $u \in \mathbb{R}^m$ is the control input, and $x \in \mathbb{R}^n$ is the state. Note, $f(x)$ and $g(x)$ are locally Lipschitz, meaning that they have bounded derivatives.

If the desired goal is to stabilize the nonlinear control to $x^* = 0$, a feedback control law needs to be determined that makes a positive definite function V approach zero. Mathematically, this leads to:

$$\exists u = k(x) \text{ such that } \dot{V}(x, k(x)) \leq -\gamma(V(x)), \tag{2}$$

where

$$\dot{V}(x, k(x)) = L_f V(x) + L_g V(x) k(x). \tag{3}$$

The symbol $\exists$ is a mathematical symbol for "there exists," and it signifies the presence of a control law $k(x)$ that ensures the system remains stable. $L_f V(x)$ and $L_g V(x)$ represent the Lie derivatives of the Lyapunov function. γ is a class $\mathcal{K}$ function, which is strictly monotonic and maps zero to zero. The process of stabilising the system can be reduced to finding a control law that will result in the Lyapunov function decreasing over time, thus leading to a stable equilibrium.

This concept forms the basis of a CLF, a positive definite $V(x)$ that ensures system stability by showing that a control law exists. The mathematical definition of a CLF is:

$$\inf_{u \in U} \left[L_f V(x) + L_g V(x) u \right] \leq -\gamma(V(x)), \tag{4}$$

where $\inf$ is the infimum (lowest value approachable). CLFs assist in finding stabilising controllers without needing to explicitly construct one in every case.

While CLFs are designed to stabilise the state by bringing it to a particular point, CBFs are designed to enforce safety by ensuring that the system remains within a specified safe region. The formulation of CBFs is also well documented in [2], and the following work also follows the presentation of that work. A primary difference between CLF and CBF is that CLFs are overly restrictive by making every sublevel set of the Lyapunov function invariant. CBFs, on the other hand, make the set $\mathcal{C}$ invariant but not the sublevel sets, allowing flexibility in system dynamics. Superlevel set is a set of all points of a function where the function value is equal to or greater than a certain threshold, and invariance means a system will always remain in a set once entered. Mathematically, $h(x)$ can be defined as a CBF which describes the boundary of the safe region or the safety condition. A CBF constraint is defined as

$$\sup_{u \in U} \left[L_f h(x) + L_g h(x) u \right] \geq -\alpha(h(x)), \quad \forall x \in D \tag{5}$$

where $h(x)$ is the barrier function (usually positive), $L_f h(x)$ and $L_g h(x)$ represent the Lie derivative of the barrier function, α is a class $\mathcal{K}$ function, and $\sup$ stands for the supremum (least upper bound) of the set.

Given a CBF, the set of all possible control inputs for ensuring safety can be written as

$$K_{\text{cbf}}(x) = \{ u \in U \mid L_f h(x) + L_g h(x) u + \alpha(h(x)) \geq 0 \} \tag{6}$$

$K_{\text{cbf}}(x)$ can help in understanding all the possible sets of control inputs that can be chosen to keep the system safe. However, it does not specify the optimal control input that needs to be selected. A valid control input u needs to be obtained that will optimally satisfy all the CBF constraints whilst also optimising for performance. One way to achieve this is to implement a quadratic program (QP). A QP can be defined as

$$u(x) = \arg\min_{u \in \mathbb{R}^m} \frac{1}{2} \| u - u_{\text{nom}} \|^2 \tag{7}$$

subject to:

$$L_f h(x) + L_g h(x) u \geq -\alpha(h(x)) \tag{8}$$

u_{nom} is the nominal control input that represents the controller without any safety constraints. In case the nominal controller u_{nom} satisfies CBF constraints and is optimal, the quadratic cost term will be minimised, and the QP will select $u = u_{\text{nom}}$. If the nominal controller is not the most optimal controller, then the QP will choose the most optimal controller from $K_{\text{cbf}}(x)$ to minimise the quadratic cost term whilst also enforcing the CBF constraints.

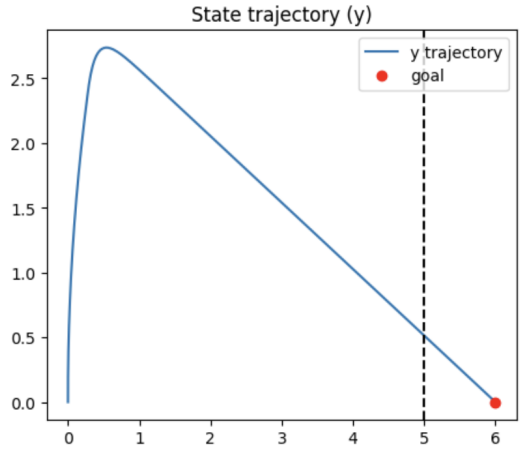

Fig. 1. This image shows a quadcopter modelled with simple dynamics. With using just the nominal controller and with a safe set of [0,5] units, it can be seen that safety is violated when the quadcopter model exits the safe set and reaches the goal of 6 units (which is outside the safe set).

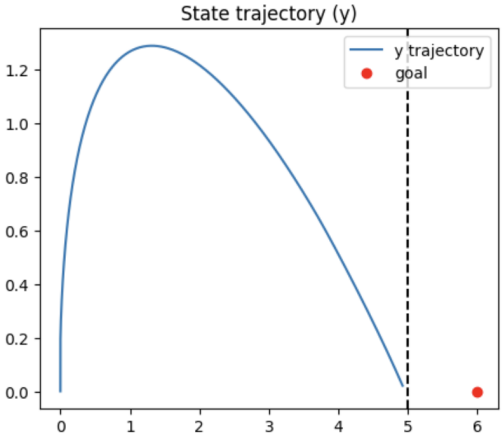

Fig. 2. This image shows a quadcopter modelled with simple dynamics. Using a CBF-QP, it can be seen that the quadcopter model remains within the safe set of [0,5] units at all times, without exiting it even if the goal is located outside the safe set.

IV. Different Types of CBFs

There are various types of CBF formulations present in the literature, including backup barrier functions, reduced order control barrier functions, zeroing control barrier functions, reciprocal control barrier functions, and exponential control barrier functions. While all these CBFs ensure system safety, they exhibit different behaviours. In this review, zeroing control barrier functions (ZCBFs) and reciprocal control barrier functions (RCBFs) will be discussed, which is well documented in [3] and will be the basis of this review.

A. Zeroing Control Barrier Functions (ZCBFs)

It is imperative to understand a zeroing barrier function first to fully understand a zeroing control barrier function. A zeroing barrier function (ZBF) is a continuously differentiable function $h(x)$ that is zero at the boundary of the safe set C. This means that for a continuously differentiable function $h(x)$, if the condition

$$L_f h(x) \geq -\alpha(h(x)), \quad \forall x \in D \tag{9}$$

holds true for $x \in D$, where α is an extended class $\mathcal{K}$ function and D is a set such that $C \subseteq D \subset \mathbb{R}^n$, then $h(x)$ is a zeroing barrier function. This function satisfies

$$h(x) = 0, \quad \text{for } x \text{ on the boundary of } C \tag{10}$$

and

$$h(x) > 0, \quad \text{for } x \text{ inside } C. \tag{11}$$

Forward invariance is enforced as the Lie derivative of $h(x)$, along with the system dynamics, prevents $h(x)$ from rapidly decreasing. A zeroing barrier function works on a non-controlled (i.e., $\dot{x} = f(x)$) system (where there is no control input u).

A zeroing control barrier function (ZCBF), on the other hand, is implemented for control affine systems ($\dot{x} = f(x) + g(x)u$). Mathematically, it is defined when there exists an extended class $\mathcal{K}$ function α where

$$\sup_{u \in U}[L_f h(x) + L_g h(x)u + \alpha(h(x))] \geq 0. \tag{12}$$

This condition indicates that there will always exist a control input u that prevents $h(x)$ from decreasing too quickly, ensuring feasibility whilst maintaining safety. To guarantee forward invariance, there needs to be a control input $u(x)$ in $K_{zcbf}(x)$ where

$$K_{zcbf}(x) = \{u \in U : L_f h(x) + L_g h(x)u + \alpha(h(x)) \geq 0\}. \tag{13}$$

$K_{zcbf}(x)$ defines all the possible safe sets of controls rendering the system safe. ZCBFs can be enforced using a QP that will keep the system safe and find the most optimal control input.

As seen in [6], some types of ZCBFs include:

- **Input-to-State Safe CBF:** Uses the concept of input-to-state stability to ensure safety.
- **Adaptive CBF:** Modifies the barrier condition based on changing dynamics.
- **Integral CBF:** Uses the integral of the barrier condition to allow for smoother control.

B. Reciprocal Control Barrier Functions (RCBFs)

The motivation for reciprocal control barrier functions (RCBF) comes from reciprocal barrier functions, which are applied to systems without control inputs $\dot{x} = f(x)$. A reciprocal barrier function (RBF) $B(x)$ is a continuously differentiable function that approaches infinity as the system approaches the boundary of the safe set while remaining positive whenever the system is in the interior of the safe set. For $B(x)$ to be valid, the interior of the safe set must be forward invariant; this condition can be met using the inequality

$$\dot{B} \leq \frac{\gamma}{B}, \tag{14}$$

where γ is a positive constant. This inequality allows $\dot{B}$ to expand when the system is away from the boundary of the safe set and slow down near the boundary. Some examples of this type of function are

$$B(x) = -\log\left(\frac{h(x)}{1+h(x)}\right) \quad \text{and} \quad B(x) = \frac{1}{h(x)}. \tag{15}$$

For these choices, $B(x)$ approaches infinity near the boundary and remains positive when in the safe set.

Similar to a ZCBF, an RCBF is implemented on control affine systems, where the system dynamics are given as

$$\dot{x} = f(x) + g(x)u. \tag{16}$$

A continuously differentiable function $B(x)$, where $B(x) : \text{Int}(C) \to \mathbb{R}$, is an RCBF if there exist class $\mathcal{K}$ functions $\alpha_1, \alpha_2, \alpha_3$ such that

$$\frac{1}{\alpha_1(h(x))} \leq B(x) \leq \frac{1}{\alpha_2(h(x))}. \tag{17}$$

Forward invariance is defined by

$$\inf_{u \in U}[L_f B(x) + L_g B(x)u - \alpha_3(h(x))] \leq 0. \tag{18}$$

A possible control input from the set

$$K_{\text{rcbf}}(x) = \{u \in U : L_f B(x) + L_g B(x)u - \alpha_3(h(x)) \leq 0\} \tag{19}$$

can be selected to ensure that the system remains within the safe set while maintaining smooth dynamics.

C. Comparison Between ZCBFs and RCBFs

ZCBFs enforce boundary conditions such that the barrier function approaches zero at the boundary of the safe set, whereas RCBFs tend to infinity at the boundary. ZCBFs can be more aggressive than RCBFs near the boundary as the controller opposes any decrease in the barrier function to prevent it from exiting the safe set. However, they can provide better robustness to the system due to direct enforcement. RCBFs ensure the barrier function increases near the boundary, resulting in passive safety, less aggressive control, and better handling of model uncertainty.

V. Challenges Faced with CBFs

Implementing CBFs into systems can present some challenges, such as handling multiple safety constraints and designing controllers for systems with discontinuous dynamics and non-smooth safe sets. This section will explore these issues in depth and discuss current solutions to these challenges.

A. Handling Multiple Safety Constraints

The formulation of handling multiple safety constraints along with potential solutions is well documented in [4], and the following text follows the presentation of that work. The domain for safety for a single CBF and its constraints is manageable to determine. However, the introduction of multiple CBFs can create conflicts between CBF constraints, resulting in challenges in determining the viability domain. A viability domain is a set of states from which there exists a control input that can keep the system within the safe set and can satisfy all CBF constraints. With multiple CBFs, the viability domain can be very restrictive or may not even exist, which can make the solution to a set of CBF constraints infeasible.

For example, assume a robot navigating an environment with two obstacles: one on the east side and one on the west side. If one CBF requires the robot to move left to avoid the east side obstacle, while another CBF simultaneously requires the robot to move right to avoid the west side obstacle, the control inputs conflict, making the solution infeasible. The authors of [4] propose two solutions to address this issue. The most common strategy is to solve the CBF constraints using geometric approaches by decoupling them. If this still doesn't yield a strong viability domain, then the authors introduce an algorithm that finds and removes conflicting states from the domain.

The authors introduce a simple control barrier function (SCBF) to simplify viability domain computations to

show the geometric approaches. The SCBF is defined for a function h if the constraint

$$h(q,v,u) \leq \alpha(-h(q,v)) \quad (20)$$
$$\text{can be satisfied for a control input}$$
$$u \text{ anti-parallel to the vector}$$
$$\nabla_v h(q,v) g_2(q).$$

Here, q is the coordinates, v is the velocity of a second-order system, and α is a class $\mathcal{K}$ function. They also mention that safe sets (denoted as S) do not directly form viability domains. Instead, SCBFs can be used to describe subsets of S that form viability domains. However, even with SCBFs, defining a CBF for each constraint does not ensure joint feasibility: due to conflicts between CBFs, a set of valid control inputs $\mu_{all}(q,v)$ could be empty at certain points, leading to a loss of safety guarantees.

As such, the authors try to address this problem by designing SCBFs within a restricted set of control inputs. If control inputs are restricted to a smaller set when defining the CBFs, there is a possibility that the CBF constraints can remain feasible for the entire control set. For this to occur, all the CBFs have to be non-interfering. This means that for any two CBFs h_i and h_j,

$$(\nabla_v h_i(q,v) g_2(q)) \cdot (\nabla_v h_j(q,v) g_2(q)) \geq 0 \quad \forall (q,v) \in X. \quad (21)$$

This is essential to ensure safety conditions since it proves that multiple CBFs can exist without any conflicts.

With a set $U_{\text{prime}} \subset U$, if CBFs are designed one-at-a-time using the orthogonal extension property (OEP) or quadrant extension property (QEP), then jointly feasible conditions can be guaranteed for U. OEP involves adding elements to a system whilst ensuring their behaviour is not impacted, whereas QEP ensures that new elements maintain their behaviour across all areas when the set is expanding. OEP is less conservative (meaning it can allow for a larger U_{prime}), however it is only applicable for stricter SCBF conditions.

If the CBFs are not non-interfering or still cannot be rendered feasible after geometric approaches, then the authors propose an algorithm to find the viability domain. The algorithm begins with a set of CBFs or SCBFs and a safe set S.

1) It computes the intersection of S and the CBFs to get a domain X.
2) It creates a set D that contains boundaries between CBFs.
3) It finds the infeasible sets E from D.
4) E is divided into clusters using a `getCluster()` function described by the authors.
5) For each cluster, a new CBF is generated using a `getCBF()` function described by the authors such that the cluster lies outside the set defined by the new CBF.
6) X is updated when the new CBF is added to the working set of CBFs.

This is conducted until all CBFs are jointly feasible or X becomes empty (in which case there is no feasibility between all CBFs).

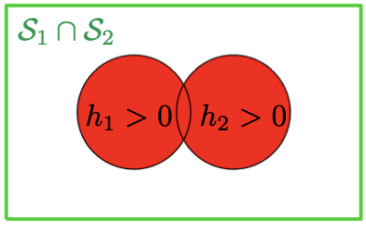

Fig. 3. This image shows the motivation behind the authors' work. Both CBF conditions must be simultaneously satisfied, which can potentially result in conflicts. Adapted from [4]

B. Designing Controllers for Discontinuous Dynamics and Non-Smooth Sets

The formulation of this problem along with potential solutions is well documented in [5], and the following text follows the presentation of that work. For complex systems, discontinuous dynamics and non-smooth sets can introduce additional challenges. Discontinuous dynamics involve sudden changes in the system's behaviour, and non-smooth sets contain boundaries or surfaces that are not continuously differentiable.

A common technique for controller design is an implementation of a CBF with a QP; however, this is insufficient for cases involving non-smooth sets and discontinuous dynamics as this type of controller only satisfies point-wise safety constraints (indicating instantaneous safety condition satisfaction) and because QPs rely on continuous gradients, which are undefined or inconsistent at non-smooth points. One such scenario involves overlapping CBFs where there can be non-smooth points at the intersection of the CBFs. When this occurs, a system may struggle to find a feasible control input at the intersection of the CBFs due to contradictory or restrictive constraints, leading to infeasibilities.

To tackle this issue, the authors of [5] introduce a function $\beta(x)$ within each CBF constraint. $\beta(x)$ is designed to modify the CBF conditions near the inter-

sections, thus enabling smoother transitions. They define $\beta(x)$ as

$$\beta(x) = \begin{cases} 0 & \text{when the system is inside the first CBF} \\ > 0 & \text{when the system is outside the first CBF} \end{cases} \tag{22}$$

Thus, when the system is near or at the intersection, the addition of $\beta(x)$ will allow the system to move outside the first CBF or remain within the CBF, instead of getting "stuck" at the intersection. The system will not always transition, however $\beta(x)$ enables the possibility of transition without leading to infeasibilities. This function is applied to every CBF constraint to prevent infeasibilities for all CBFs.

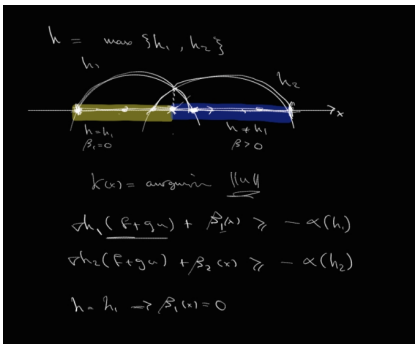

Fig. 4. This image shows a simplified version of the authors' work. In this image, $\beta(x)$ is introduced to both CBF constraints and is defined to be 0 and greater than 0, depending on where the system is present. Image credit: Mohammed Alyaseen

APPENDIX

For figures 1 and 2, the dynamics of the simple quadcopter model can be described by the state vector x and the state derivatives $\dot{x}$. The state vector is given by:

$$x = \begin{bmatrix} y \\ z \\ \dot{y} \\ \dot{z} \end{bmatrix}$$

where y and z represent the position in the vertical and horizontal directions, respectively, and $\dot{y}$ and $\dot{z}$ are the velocities in these directions. The system's state evolution is given by:

$$\dot{x} = \begin{bmatrix} \dot{y} \\ \dot{z} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} x_3 \\ x_4 \\ g u_1 \\ u_2 - g \end{bmatrix}$$

where $x_3 = \dot{y}$ and $x_4 = \dot{z}$ are the velocities in the y and z directions, respectively. The control input u is given by:

$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

where $u_1 = \phi$ represents the control input for the angle and $u_2 = T$ represents the thrust applied to the system. The term g is a constant, where g represents gravitational acceleration.

The constraint used for calculating the trajectories in figure 2 is:

$$\text{circle\_constraint} = 5 - \sqrt{y^2 + z^2}$$

The nominal controller used in figure 1 is:

$$\begin{bmatrix} -1.0 \cdot (x_1 - 6.0) - 2.0 \cdot x_3 \\ 9.81 - 0.75 \cdot (x_2 - 6.0) - 1.5 \cdot x_4 \end{bmatrix}$$

where $x_1 = y$ and $x_2 = z$ are the positions in the y and z directions. The figures provided in the report are only shown for the y direction trajectory.

REFERENCES

[1] J. J. Choi, D. Lee, K. Sreenath, C. J. Tomlin, and S. L. Herbert, "Robust Control Barrier–Value Functions for Safety-Critical Control," in *Proc. 60th IEEE Conference on Decision and Control (CDC)*, Austin, TX, USA, 2021, pp. 6814-6821, doi: 10.1109/CDC45484.2021.9683085.

[2] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control Barrier Functions: Theory and Applications," in *Proc. 18th European Control Conference (ECC)*, Naples, Italy, 2019, pp. 3420–3431.

[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, Aug. 2017, doi: 10.1109/TAC.2016.2638961.

[4] J. Breeden and D. Panagou, "Compositions of Multiple Control Barrier Functions Under Input Constraints," in *Proc. 2023 American Control Conference (ACC)*, San Diego, CA, USA, 2023, pp. 3688-3695, doi: 10.23919/ACC55779.2023.10156625.

[5] M. Alyaseen, N. Atanasov, and J. Cortes, "Safety-Critical Control of Discontinuous Systems with Nonsmooth Safe Sets," *arXiv preprint arXiv:2412.15437*, 2024. [Online]. Available: https://arxiv.org/abs/2412.15437.

[6] I. Tezuka and H. Nakamura, "Strict Zeroing Control Barrier Function for Continuous Safety Assist Control," in IEEE Control Systems Letters, vol. 6, pp. 2108-2113, 2022, doi: 10.1109/LCSYS.2021.3138526. keywords: Control systems;Time-varying systems;Safety;Computational modeling;Task analysis;Sufficient conditions;Nonlinear systems;Time-varying systems;human-in-the-loop control,