Safety-Critical Manipulation for Collision-Free Food Preparation

Andrew Singletary, William Guffey, Tamas G. Molnar, *Member, IEEE*, Ryan Sinnet, and Aaron D. Ames

***Abstract*—Recent advances allow for the automation of food preparation in high-throughput environments, yet the successful deployment of these robots requires the planning and execution of quick, robust, and ultimately collision-free behaviors. In this work, we showcase a novel framework for modifying previously generated trajectories of robotic manipulators in highly detailed and dynamic collision environments using Control Barrier Functions (CBFs). This method dynamically re-plans previously validated behaviors in the presence of changing environments—and does so in a computationally efficient manner. Moreover, the approach provides rigorous safety guarantees of the resulting trajectories, factoring in the true underlying dynamics of the manipulator. This methodology is extensively validated on a full-scale robotic manipulator in a real-world cooking environment, and has resulted in substantial improvements in computation time and robustness over re-planning.**

***Index Terms*—Industrial robots, manipulation planning, robot safety.**

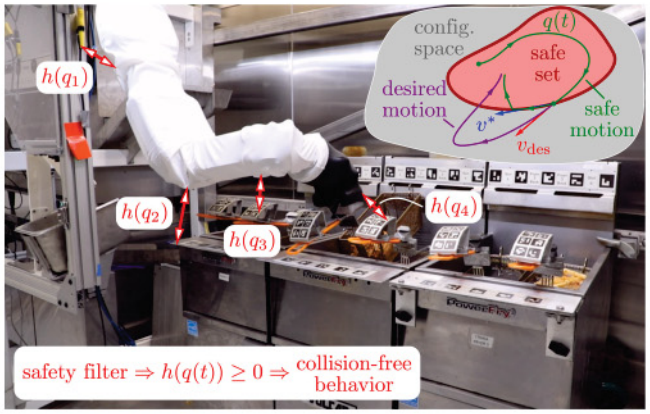


Fig. 1. Miso Robotics "Flippy2" robot frying food using our proposed safety-critical framework for food preparation.

I. Introduction

ROBOTICS and automation have great potential to transform the food industry. In the domain of autonomous cooking, robotic manipulators like those in Fig. 1 are used to pick up, deep fry, and dispense the food in the dynamic environment of the kitchen. This requires motion plans that are constantly computed, hundreds or thousands of times per day, subject to different environmental factors and initial conditions. Due to the extremely complex collision environments and non-trivial kinematics, highly non-linear planning algorithms such as TrajOpt [1], CHOMP [2], and several in the OMPL library [3] are used to plan joint trajectories offline, which the manipulator then executes. The vast majority of plans, however, deviates only slightly from previously computed trajectories: food baskets may move and deform slightly, workers may push the equipment, or the robot may have slightly different configuration initially. In these situations, rather than re-planning a trajectory with the existing motion planner, we propose a safety filtering method that produces collision-free trajectories from existing reference trajectories in minimal computation time, and with formal safety guarantees.

Minimally modifying existing trajectories is possible by optimization solvers that have warm-start or hot-start options for resolving problems with slightly modified initial conditions. In [4], the authors introduced a method for building a dataset of motion plans that were used to warm-start the trajectory generator to boost the success-rate of trajectories. Similarly, in [5], the authors proposed a dataset of expert trajectories to warm-start a Sequential Convex Programming (SCP) problem for solving locally optimal trajectories rapidly. In [6], the authors used incremental solvers to update trajectories via Gaussian processes and factor graphs.

More generally, local planners have been used for decades to modify rough, global trajectories under new collision constraints [7] or dynamic environments [8]. While many of these works could certainly be modified to tackle the robotic cooking problem, we believe that our approach's balance of simplicity, computational speed, and formality of resulting safety guarantees makes it the best fit for the problem at hand. Moreover, this algorithm can be run in real-time as a feedback controller with dynamically updating environments, offering a great deal of flexibility in implementation.

Our approach relies on control barrier functions (CBFs) [9], that have been proven to provide an effective means of enforcing safety on a wide variety of robotic systems [10], including robotic manipulators [11]–[13]. In prior works, CBFs were used as safety filters on desired velocity commands, and obstacle

Manuscript received 24 February 2022; accepted 27 June 2022. Date of publication 20 July 2022; date of current version 24 August 2022. This letter was recommended for publication by Associate Editor F. Pierri and Editor C. Gosselin upon evaluation of the Reviewers' comments. This work was supported by Miso Robotics and NSF CPS under Award 1932091. *(Corresponding author: Andrew Singletary.)*

Andrew Singletary, Tamas G. Molnar, and Aaron D. Ames are with the Department of Mechanical and Civil Engineering, California Institute of Technology, Pasadena, CA 91125 USA (e-mail: asinglet@caltech.edu; tmolnar@caltech.edu; ames@caltech.edu).

William Guffey and Ryan Sinnet are with the Miso Robotics, Pasadena, CA 91101 USA (e-mail: wguffey@misorobotics.com; rsinnet@alumni.caltech.edu).

Digital Object Identifier 10.1109/LRA.2022.3192634

representations were simplified. In this work, safe velocity commands synthesized based on kinematics are tracked by low-level controllers, and, unlike [11], a formal proof is given that this method preserves safety for the full dynamics of the robot. Moreover, it is achieved without requiring any knowledge of the dynamics, unlike [12], [13]. Furthermore, our work utilizes significantly more complex obstacle representations and environments than previous works involving CBFs, which facilitates practical implementation. The primary contribution of this work is a rigorously tested CBF-based filtering strategy that modifies previously generated trajectories to account for new collision constraints in a provably safe manner. This strategy often eliminates the need for re-planning in updated environments, saving computation time and providing robust safety guarantees for the resulting trajectory. We formally prove that these trajectories are not only valid for the kinematic model of the manipulator, but also for the underlying full-order dynamical system. The proposed novel control algorithm is implemented in the MoveIt framework [14], and applied to full-scale autonomous food-frying in collaboration with Miso Robotics. The speed and efficacy of this method are extensively explored in real-world cooking environments, and the method has been shown to dramatically increase planning speed and reliability.

The layout of this paper is as follows. In Section II, CBFs are used to enforce safety on both the kinematic model of the manipulator and the full dynamics. Section III formulates distance functions in complex, real-world environments, which are used in the context of CBFs for collision avoidance. Section IV outlines the software implementation of the proposed algorithm and the simulation environment. Lastly, Section V shows the details and results of the extensive, real-world hardware tests in the application of robotic cooking.

II. Control Barrier Functions for Safety

A. Background: Control Barrier Functions

Consider a nonlinear system in control-affine form:

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

with state $x \in \mathbb{R}^k$ and control input $u \in U \subset \mathbb{R}^m$ to be chosen from an admissible input set $U \subseteq \mathbb{R}^m$. The functions $f : \mathbb{R}^k \to \mathbb{R}^k$ and $g : \mathbb{R}^k \to \mathbb{R}^{k\times m}$ describe the dynamics of the system and are assumed to be Lipschitz continuous. Given a Lipschitz continuous control law $k : \mathbb{R}^k \to \mathbb{R}^m$, $u = k(x)$ we obtain the closed-loop dynamics:

$$\dot{x} = f_{\text{cl}}(x) := f(x) + g(x)k(x). \tag{2}$$

For the initial condition $x(t_0) = x_0 \in \mathbb{R}^k$, this system has a unique solution $x(t)$ which we assume to exist for all $t \geq t_0$.

Consider a safe subset of the state-space $S \subset \mathbb{R}^k$ which may represent, for example, the collision-free states of a manipulator. To guarantee safety, we must ensure that the state of the closed-loop system is kept within in S for all time. This is formalized through the notion of set invariance.

Definition 1: The set S is *forward invariant* if the solution $x(t)$ of system (2) satisfies $x(t) \in S, \forall t \geq t_0$ for any $x_0 \in S$.

Control barrier functions are a common tool to synthesize controllers that enforce forward invariance for a given set S.

Definition 2 ([9]): Let $S \subset \mathbb{R}^k$ be defined as the 0-superlevel set of a continuously differentiable function $h : \mathbb{R}^k \to \mathbb{R}$:

$$S = \{x \in \mathbb{R}^k \ : \ h(x) \geq 0\}. \tag{3}$$

Function h is a *control barrier function (CBF)* for (1) on S if there exists an *extended class K_∞ function*[1] α such that for all $x \in S$:

$$\sup_{u\in U} \underbrace{\left[\frac{\partial h}{\partial x} f(x) + \frac{\partial h}{\partial x} g(x)u\right]}_{\dot{h}(x,u)} \geq -\alpha(h(x)), \tag{4}$$

where $\dot{h}(x, u)$ is the derivative of $h(x)$ along system (1).

This definition yields the following key result for CBFs.

Theorem 1 ([9]): If h is a CBF for (1), then any locally Lipschitz continuous controller $k : \mathbb{R}^k \to \mathbb{R}^m$, $u = k(x)$ satisfying

$$\dot{h}(x, k(x)) \geq -\alpha(h(x))$$

renders the set S in (3) forward invariant for the system (2).

This condition can be incorporated into a quadratic program (QP) to synthesize pointwise optimal and safe controllers, by minimally modifying a desired but not necessarily safe input $u_{\text{des}}(x, t) \in U$ to a safe input $u^*(x, t) \in U$:

$$\begin{aligned} u^*(x,t) = \underset{u\in U}{\text{argmin}} \quad & \|u - u_{\text{des}}(x,t)\|_2^2 \\ \text{s.t.} \quad & \dot{h}(x,u) \geq -\alpha(h(x)). \end{aligned} \tag{5}$$

B. Application to Robotic Manipulators

Now let us use CBFs for controlling robotic manipulators whose state $x = (q, \dot{q})$ consists of the configuration $q \in \mathbb{R}^n$ and the joint velocities $\dot{q} \in \mathbb{R}^n$. For obstacle avoidance, we define the safe set over the configuration space:

$$S = \{q \in \mathbb{R}^n \ : \ h(q) \geq 0\}, \tag{6}$$

where $h : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable. First, we consider the kinematics of robotic manipulators with state q. In particular, we consider the system:

$$\dot{q} = v, \tag{7}$$

wherein we assume direct control over the joint velocities via the commanded velocity $v \in \mathbb{R}^n$. We design a velocity v by considering it as input to system (7) and guaranteeing safety by CBFs. In Section II-C, it will be verified that safety guarantees extend to the full dynamics when the commanded velocity is tracked by a low-level controller. Because each joint's velocity is directly controlled according to (7), we can simplify the QP shown in (5) to:

$$\begin{aligned} v^*(q,t) = \underset{v\in\mathbb{R}^n}{\text{argmin}} \quad & \|v - v_{\text{des}}(q,t)\|_2^2 \\ \text{s.t.} \quad & \frac{\partial h}{\partial q} v \geq -\alpha h(q), \end{aligned} \tag{8}$$

[1] $\alpha : \mathbb{R} \to \mathbb{R}$ is an extended class K_∞ function if it is continuous, strictly monotonically increasing, and satisfies $\alpha(0) = 0$, $\lim_{r\to\pm\infty} \alpha(r) = \pm\infty$.

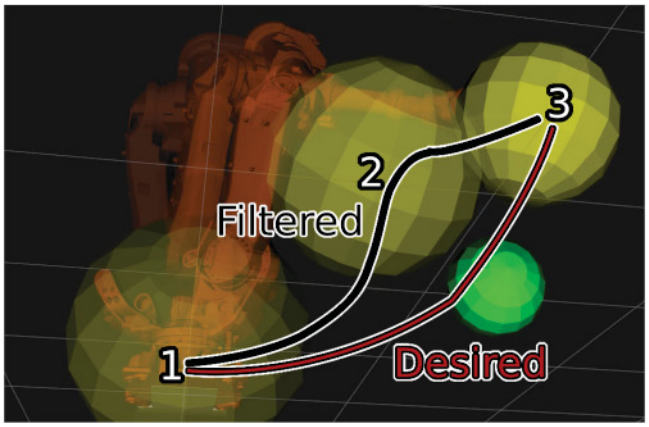


Fig. 2. Manipulator trajectory resulting from the control barrier function detailed in Example 1. The tool is marked in yellow, whereas the obstacle is shown in green.

where a desired velocity $v_{\text{des}}(q,t) \in \mathbb{R}^n$ is modified to a safe velocity $v^*(q,t) \in \mathbb{R}^n$. Note that we chose the extended class K_∞ function to be linear with constant gradient $\alpha > 0$ for the sake of simpler exposition of the upcoming formulas.

Example 1: Consider a 6-degrees-of-freedom manipulator ($n = 6$) with a spherical tool attachment of radius r_1. The manipulator is intended to track a desired joint velocity $v_{\text{des}}(q,t)$ and we wish to avoid a spherical region centered at $O \in \mathbb{R}^3$ of radius r_2. The CBF can be written as the distance from the spherical tool to the sphere in the surroundings:

$$\begin{aligned} h(q) &= \|F(q) - O\|_2 - (r_1 + r_2) \\ &= \sqrt{(F_x - O_x)^2 + (F_y - O_y)^2 + (F_z - O_z)^2} - (r_1 + r_2) \end{aligned} \tag{9}$$

where $F : \mathbb{R}^6 \to \mathbb{R}^3$ are the forward kinematics that give the position of the end-effector in space, $(F_x, F_y, F_z) = F(q)$. The gradient of the CBF can be computed as:

$$\frac{\partial h}{\partial q} = \frac{\partial h}{\partial F}\frac{\partial F}{\partial q} = \frac{1}{\|F(q) - O\|_2}\begin{bmatrix} F_x - O_x \\ F_y - O_y \\ F_z - O_z \end{bmatrix}^T J(q), \tag{10}$$

where $J : \mathbb{R}^6 \to \mathbb{R}^3 \times \mathbb{R}^6$, $J(q) = \frac{\partial F}{\partial q}$ is the top three rows of the manipulator Jacobian. By enforcing the CBF-QP (8), we obtain the path illustrated in Fig. 2.

C. Safety Guarantees: From Kinematics to Dynamics

We now establish the first theoretic contribution of the paper: we leverage the kinematics of the manipulator to guarantee safe behavior on the full-order dynamics. We establish that tracking the safe velocity obtained from (8) results in safety under reasonable conditions on the tracking controller.

Specifically, consider the full-order dynamics associated with a robotic manipulator [15]:

$$D(q)\dot{q} + C(q,\dot{q})\dot{q} + G(q) = Bu, \tag{11}$$

with $q, \dot{q} \in \mathbb{R}^n$, $D(q) \in \mathbb{R}^{n\times n}$ the inertia matrix, $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$ the Coriolis matrix, and $G(q) \in \mathbb{R}^n$ the gravity vector. Here we assume full actuation: the actuation matrix $B \in \mathbb{R}^{n\times n}$ is invertible and $u \in \mathbb{R}^n$. Associated with these dynamics is a control system of the form (1) with $x = (q,\dot{q})$ (hence $k = 2n$).

Motivated by the approach in [16], we assume the existence of a "good" low-level velocity tracking controller on the manipulator (as is common on industrial robots). [16] only considered smooth velocity reference signals and smooth CBFs to prove the safety of the full system, whereas in this paper we address nonsmoothness in both of these aspects. This is essential for operation in complicated collision environments where nonsmoothness naturally arises.

Concretely, for a velocity command $v^*(q,t)$ consider the corresponding error in tracking this velocity:

$$\dot{e} = \dot{q} - v^*, \tag{12}$$

and assume exponentially stable tracking.

Assumption 1: There exists a low-level controller $u = k(x,t)$ for the control system (1) obtained from (11) such that

$$\|\dot{e}(t)\|_2 \le Me^{-\lambda t}\|\dot{e}_0\|_2 \tag{13}$$

holds for some $M, \lambda > 0$ along the solution $x(t)$ of the closed-loop system (2) with $q(t_0) = q_0$, $\dot{q}(t_0) = \dot{q}_0$ and $\dot{e}(t_0) = \dot{e}_0$.

Such exponentially stable tracking controller can be designed, for example, by means of feedback linearization or by using control Lyapunov functions. Under this assumption, we have the first theoretic result of the paper.

Theorem 2: Consider the full-order dynamics of a robot manipulator (11) expressed as the control system (1), and the safe set S in (6). Assume that h has bounded gradient, i.e., there exists $C_h > 0$ s.t. $\|\frac{\partial h}{\partial q}\|_2 \le C_h$ for all $q \in S$. Let $v^*(q,t)$ be the safe velocity given by the QP (8), with corresponding error in (12). If Assumption 1 holds with $\lambda > \alpha$, safety is achieved for the full-order dynamics (11) in that:

$$(q_0, \dot{e}_0) \in S_M \Rightarrow q(t) \in S, \quad \forall t \ge t_0, \tag{14}$$

where:

$$S_M = \left\{(q,\dot{e}) \in \mathbb{R}^{2n} \;:\; h(q) - \frac{C_h M}{\lambda - \alpha}\|\dot{e}\|_2 \ge 0\right\}. \tag{15}$$

Proof: First, we lower-bound $\dot{h}(q,\dot{q})$ as follows:

$$\begin{aligned} \dot{h}(q,\dot{q}) &= \frac{\partial h}{\partial q}v^* + \frac{\partial h}{\partial q}\dot{e} \\ &\ge -\alpha h(q) - \left\|\frac{\partial h}{\partial q}\right\|_2 \|\dot{e}\|_2 \\ &\ge -\alpha h(q) - C_h M\|\dot{e}_0\|_2 e^{-\lambda t}, \end{aligned} \tag{16}$$

where we used (i) the definition (12) of the tracking error; (ii) the constraint on the safe velocity in (8) and the Cauchy-Schwartz inequality; and (iii) the upper bound C_h on $\|\frac{\partial h}{\partial q}\|_2$ and the upper bound (13) on the tracking error. Then, consider the following continuous function $y : \mathbb{R} \to \mathbb{R}$:

$$y(t) = \left(h(q_0) - \frac{C_h M\|\dot{e}_0\|_2}{\lambda - \alpha}\right)e^{-\alpha t} + \frac{C_h M\|\dot{e}_0\|_2}{\lambda - \alpha}e^{-\lambda t}, \tag{17}$$

which satisfies:

$$\dot{y}(t) = -\alpha y(t) - C_h M \|\dot{e}_0\|_2 \mathrm{e}^{-\lambda t}$$
$$y(t_0) = h(q_0). \tag{18}$$

For $(q_0, \dot{e}_0) \in S_M$, we have $y(t) \geq 0, \forall t \geq t_0$, and by the comparison lemma we get:

$$h(q(t)) \geq y(t) \geq 0, \quad \forall t \geq t_0, \tag{19}$$

that implies $q(t) \in S, \forall t \geq t_0$. This completes the proof. ■

III. Distance Functions and Safety Filtering

A. Collisions With Environment

In order to prevent collisions with the environment, we must ensure that any point on the robot does not come into contact with any point in the environment. However, unlike the simple example before, we cannot rely on the robot and environment being represented by simple spheres.

Let us denote the set of all points on the robot as $A \subset \mathbb{R}^3$, and the set of all points in the collision environment as $B \subset \mathbb{R}^3$. To guarantee safety, we require that $A \cap B = \emptyset$, thus distance$(A, B) > 0$. More formally, *distance* is defined as:

$$\text{distance}(A, B) = \inf_{\substack{p_A \in A \\ p_B \in B}} \|p_A - p_B\|_2, \tag{20}$$

which can be computed in $\mathbb{R}^3$ using the GJK algorithm [17].

This notion gives a nonnegative distance, which could be used as CBF. However, it is advantageous to define a CBF that is negative in the event of collision, since CBFs may also ensure that the boundary of the set S is re-approached if $h(x) < 0$ [9]. In collision, *penetration* is defined as:

$$\text{penetration}(A, B) = \inf_{\substack{p_A \in A \\ p_B \in \overline{B}}} \|p_A - p_B\|_2, \tag{21}$$

where $\overline{B}$ is the complement of B, or the set of points outside the collision scene. Penetration is often computed using the EPA algorithm [18].

These two functions can be combined to form the notion of *signed distance*. Signed distance is typically written as

$$\text{sd}(A, B) = \text{distance}(A, B) - \text{penetration}(A, B). \tag{22}$$

When the points p_A and p_B of the robot and the environment are given in local coordinates, the following expression from [1] can be utilized to compute the signed distance:

$$\text{sd}_{AB}(q) = \max_{\substack{\tilde{n} \in \mathbb{R}^3 \\ \|\tilde{n}\|_2 = 1}} \min_{\substack{p_A \in A \\ p_B \in B}} \tilde{n} \cdot \left(F_A^{\mathrm{W}}(q) p_A - F_B^{\mathrm{W}} p_B\right), \tag{23}$$

where $F_A^{\mathrm{W}}(q) \in \mathbb{R}^{3\times 3}$ gives the pose of the robot in the world frame that depends on the configuration q, and $F_B^{\mathrm{W}} \in \mathbb{R}^{3\times 3}$ gives the pose of the collision environment, i.e., $F_A^{\mathrm{W}}(q) p_A$ and $F_B^{\mathrm{W}} p_B$ indicate points in the world frame.

B. Controller Synthesis With Control Barrier Functions

Given the signed distance, we propose the CBF candidate:

$$h(q) = \text{sd}_{AB}(q), \tag{24}$$

which defines the corresponding safe set of the system:

$$S = \{q \in \mathbb{R}^n \ : \ h(q) = \text{sd}_{AB}(q) \geq 0\}. \tag{25}$$

We remark that based on (23) h can be written as:

$$h(q) = \hat{n}(q)^\top \left(F_A^{\mathrm{W}}(q) \hat{p}_A(q) - F_B^{\mathrm{W}} \hat{p}_B(q)\right). \tag{26}$$

Here $\hat{n}(q)$ and $\hat{p}_A(q)$, $\hat{p}_B(q)$ denote the direction and points that maximize and minimize the expression in (23), respectively, which depend on the configuration q.

It is important to note that in Euclidean space, signed distance, h, is differentiable almost everywhere, and satisfies $\|\frac{\partial h}{\partial p_A}\|_2 = 1$ [19]. There exists, however, a set of measure zero where $\frac{\partial h}{\partial q}$ is discontinuous, since functions $\hat{n}$ and $\hat{p}_A, \hat{p}_B$ are nonsmooth due to the max and min operators in (23). Since the above framework requires continuously differentiable h, we take special care in applying the theory, and we handle nonsmoothness under the following construction.

First, we express the gradient of h as follows:

$$\frac{\partial h}{\partial q} = \hat{n}(q)^\top J_A(q) + \delta(q), \tag{27}$$

where $J_A(q) = \frac{\partial F_A^{\mathrm{W}}}{\partial q} \hat{p}_A(q)$ and $\delta(q)$ is the remainder term associated with the derivatives of $\hat{n}, \hat{p}_A$, and $\hat{p}_B$. Importantly, note that $\hat{n}(q)^\top J_A(q)$ is continuous, while $\delta(q)$ is discontinuous on a set of measure zero. The term $\hat{n}(q)^\top J_A(q)$ can be interpreted as a continuous approximation of $\frac{\partial h}{\partial q}$, while the approximation error $\delta(q)$ acts as disturbance. The size of the disturbance is characterized by its essential supremum[2]:

$$\|\delta\|_\infty := \operatorname*{ess\,sup}_{t \geq t_0} \|\delta(q(t))\|_2.$$

The points where h is not differentiable and δ is discontinuous occur on a set of measure zero, and therefore do not impact the essential supremum. Now we incorporate the continuous approximation $\hat{n}(q)^\top J_A(q)$ in (27) into the control design. The following result demonstrates that this approximation is sufficient to maintain safety if the disturbance $\delta(q)$ is properly accounted for (in an input-to-state safety (ISSf) context [20], [21]).

Proposition 1: Consider the kinematic model of a robotic manipulator (7). Then, the controller expressed as the QP:

$$v^*(q,t) = \operatorname*{argmin}_{v \in \mathbb{R}^n} \ \|v - v_{\text{des}}(q,t)\|_2^2$$
$$\text{s.t.} \ \ \hat{n}(q)^\top J_A(q) v \geq -\alpha h(q) + 2 J_{\max} \dot{q}_{\max}, \tag{28}$$

with $\dot{q}_{\max} = \|\dot{q}\|_\infty$ and $J_{\max} = \max_{q \in \mathbb{R}^n} \|J_A(q)\|_2$, renders the set S in (25) forward invariant for the resulting closed-loop system. That is, the controller (28) keeps system (7) safe.

As such, collision-free behavior is enforced for the kinematic model of the manipulator, since the disturbance in (27) is handled by the last term of (28). The feasibility of (28) in singular

[2]The function δ is *essentially bounded* if $\|\delta(t)\|_2$ is bounded by a finite number for almost all $t \geq t_0$ (i.e., $\|\delta(t)\|_2$ is bounded except on a set of measure zero). The quantity $\|\delta\|_\infty$ is then defined as the least such bound.

configurations can be guaranteed by increasing α or decreasing $\dot{q}_{\max}$ by reducing the desired speed.

Proof: First, we bound the essential supremum $\|\delta\|_\infty$ of the disturbance. Recall that the points where h is not differentiable are on a set of measure zero and do not impact the essential supremum, thus we construct the bound on $\|\delta\|_\infty$ by picking generic points where the h is differentiable. For an arbitrary point on the robot $p_A \in A$ where h is differentiable:

$$\left\|\frac{\partial h}{\partial q}\right\|_2 \leq \left\|\frac{\partial h}{\partial p_A}\right\|_2 \left\|\frac{\partial p_A}{\partial q}\right\|_2 \leq 1 \cdot J_{\max}. \tag{29}$$

This leads to the bound:

$$\begin{aligned}\|\delta\|_\infty &= \left\|\frac{\partial h}{\partial q} - \hat{n}(q)^\top J_A(q)\right\|_\infty \\ &\leq \left\|\frac{\partial h}{\partial q} - \hat{n}(q)^\top J_A(q)\right\|_2 \\ &\leq \left\|\frac{\partial h}{\partial q}\right\|_2 + \left\|\hat{n}(q)^\top J_A(q)\right\|_2 \\ &\leq J_{\max} + \|J_A(q)\|_2 \leq 2J_{\max}.\end{aligned} \tag{30}$$

Then, we differentiate the CBF h in (24) and use (27):

$$\begin{aligned}\dot{h}(q,\dot{q}) &= \frac{\partial h}{\partial q}\dot{q} = \hat{n}(q)^\top J_A(q)\dot{q} + \delta(q)\dot{q} \\ &\geq \hat{n}(q)^\top J_A(q)\dot{q} - \|\delta\|_\infty \dot{q}_{\max}.\end{aligned} \tag{31}$$

Substituting $\dot{q}$ with the solution $v^*(q,t)$ to (28) and incorporating the bound on $\|\delta\|_\infty$, the result is:

$$\begin{aligned}\dot{h}(q, v^*(q,t)) &\geq \hat{n}(q)^\top J_A(q) v^*(q,t) - \|\delta\|_\infty \dot{q}_{\max} \\ &\geq -\alpha h(q) + 2J_{\max}\dot{q}_{\max} - \|\delta\|_\infty \dot{q}_{\max} \\ &\geq -\alpha h(q).\end{aligned} \tag{32}$$

Thus, the condition in Theorem 1 holds almost everywhere except on a set of measure zero, which yields that set S is forward invariant based on Lemma 2 of [22]. ■

C. Self-Collisions

Self-collisions are defined as collisions between any two links of the robot that are not explicitly allowed to collide. For these types of collisions, we still use the signed distance function, but now F_B^{W} also depends on the configuration q:

$$\mathrm{sd}_{AB}(q) = \max_{\substack{\tilde{n}\in\mathbb{R}^3 \\ \|\tilde{n}\|_2=1}} \min_{\substack{p_A\in A \\ p_B\in B}} \tilde{n}\cdot\left(F_A^{\mathrm{W}}(q)p_A - F_B^{\mathrm{W}}(q)p_B\right). \tag{33}$$

Thus, the gradient of $h(q) = \mathrm{sd}_{AB}(q)$ becomes:

$$\frac{\partial h}{\partial q} = \hat{n}(q)^\top \left(J_A(q) - J_B(q)\right) + \delta(q). \tag{34}$$

Proposition 1 can again be applied to self-collisions, with slight modifications. The analysis results in the QP:

$$\begin{aligned} v^*(q,t) = \underset{v\in\mathbb{R}^n}{\operatorname{argmin}} \quad & \|v - v_{\mathrm{des}}(x,t)\|_2^2 \\ \text{s.t.} \quad & \hat{n}(q)^\top \left(J_A(q) - J_B(q)\right) v \geq -\alpha h(q) + 4J_{\max}\dot{q}_{\max}. \end{aligned} \tag{35}$$

D. Safety Guarantees for the Full-Order Dynamics

The safety guarantees of Proposition 1 are valid for the kinematic model (7). However, like in Theorem 2, the controllers (28) and (35) lead to collision-free motion also on the full-order dynamics—assuming good velocity tracking.

Theorem 3: Consider the full-order dynamics of a robot manipulator (11) expressed as the control system (1), and the safe set S in (25) associated with the signed distance $\mathrm{sd}_{AB}(q)$ between the robot and the environment in (23). Let $v^*(q,t)$ be the safe velocity given by the QP (28), with corresponding error in (12). If Assumption 1 holds with $\lambda > \alpha$, safety is achieved for the full-order dynamics (11) in that:

$$(q_0, \dot{e}_0) \in S_M \Rightarrow q(t) \in S, \quad \forall t \geq t_0, \tag{36}$$

where:

$$S_M = \left\{(q,\dot{e}) \in \mathbb{R}^{2n} \;:\; \mathrm{sd}_{AB}(q) - \frac{J_{\max}M}{\lambda - \alpha}\|\dot{e}\|_2 \geq 0\right\}. \tag{37}$$

Note that the selection of α must satisfy $\lambda > \alpha$. The same safety guarantees can be stated for self-collision avoidance with the QP (35), and changing environments can be treated similarly if the resulting safe velocity is tracked well. Moreover, a practical advantage of this approach is that robust tracking yields safety robust to those disturbances.

Proof: The proof follows the same steps as in the Proof of Theorem 2 with the substitution $C_h = J_{\max}$, which is justified by $\|\frac{\partial h}{\partial q}\|_2 \leq J_{\max}$ based on (29). Furthermore, note that $\frac{\partial h}{\partial q}v^* \geq -\alpha h(q)$ still holds due to (32). ■

With this result, we achieve guarantees of safety that could not be achieved with traditional methods utilizing the kinematics and/or signed-distance approximations only.

IV. Software Implementation and Simulation

A. CBF Implementation on Precomputed Trajectories

Assuming the knowledge of a reference trajectory, we now detail the trajectory *safety filter* algorithm. The most straightforward implementation of the QPs (28) and (35) is to run them in real-time paired with a desired joint velocity controller that tracks the reference. This can be achieved with a P controller to the next waypoint i:

$$v_{\mathrm{des}}(q,t) = K_P\left(q_{\mathrm{des}}^i - q\right). \tag{38}$$

For the best results, the error $q_{\mathrm{des}}^i - q$ is heavily saturated to avoid large values of $v_{\mathrm{des}}(q,t)$ far from the goal. The tracked waypoint is iterated forwards when the robot gets sufficiently close ($\|q_{\mathrm{des}}^i - q\|_2 < \epsilon_q$) or stuck ($\|v_{\mathrm{des}}(q,t) - \dot{q}\|_2 > \epsilon_v$ for a certain amount of time).

However, due to the large (∼200 ms) time delay of many industrial manipulators, it is often desired to instead send precomputed time-stamped trajectories, rather than attempting to track a trajectory online with feedback. The basic algorithm for generating these safe trajectories, given a cache of previously

Algorithm 1: Trajectory Generation in Modified Collision Environments With Safety Filters.

Require: C, the cache that contains behaviors C_B^i, planning scenes C_P^i, and trajectories C_X^i
for each C^i s.t. $B == C_B^i$ **do** ▷Search through cache
  $T^i = f(C_P^i, C_{X_0}^i, P, q)$ ▷Compute suitability metric
  if $T^i < T_1$ **then** ▷Reference is extremely similar
    $X \leftarrow \text{CBF}(C_X^i, P, q)$
    return
  end if
end for
$[T_{\min}, \text{idx}] \leftarrow \min(T^i)$ ▷Find best reference
if $T_{\min} < T_2$ **then** ▷Close match
  $X \leftarrow \text{CBF}(C_X^{\text{idx}}, P, q)$ ▷Safety filter
  return
else if $T_{\min} < T_3$ **then** ▷Suitable match
  $X \leftarrow \text{CBF}(C_X^{\text{idx}}, P, q)$
  $C \leftarrow X$
  return
else ▷Best reference is very dissimilar
  $X \leftarrow$ Re-plan from scratch
  $C \leftarrow X$ ▷X gets added to cache
end if

computed reference trajectories, is detailed in Algorithm 1. The cache is filled with hand-picked trajectories that reach the goal, avoid obstacles, and are visually pleasing, as the public perception of this robot matters.

There are three fields of interest in the cached trajectories: the desired behavior B, the manipulator's trajectory T, and the collision environment used by the original planner, referred to as the planning scene P.

The algorithm first assesses the suitability of previously computed trajectories in the cache. There are two major considerations: the difference in initial conditions and the similarity of the planning scene. The suitability of the i^{th} member of the cache C^i is evaluated by the function:

$$T^i = f\left(C_P^i, C_{X_0}^i, P, q\right) = \delta_q^i + \delta_P^i, \tag{39}$$

where

$$\delta_q^i = \left\| C_{X_0}^i - q \right\|_2, \quad \delta_P^i = \sum_{o \in O} \left\| C_{P_o}^i - P_o \right\| \tag{40}$$

assess the differences in the initial conditions of the robot and the collision objects $o \in O$ making up the planning scene. There are three threshold values (T_1, T_2 and T_3) for this suitability metric. If $T^i < T_1$, then the search stops, as the trajectory in the cache is so close that it is not worth searching, and the CBF filter is applied. After searching through all cache members, if $T^i < T_2$, then the filter is applied, but the trajectory is not added to the cache to prevent it from growing unnecessarily large. If $T_2 < T^i < T_3$, then the filter is applied and the resulting trajectory is added to the cache. Finally, if $T^i > T_3$, then the original motion planning algorithm is used, and the result is added to the cache.

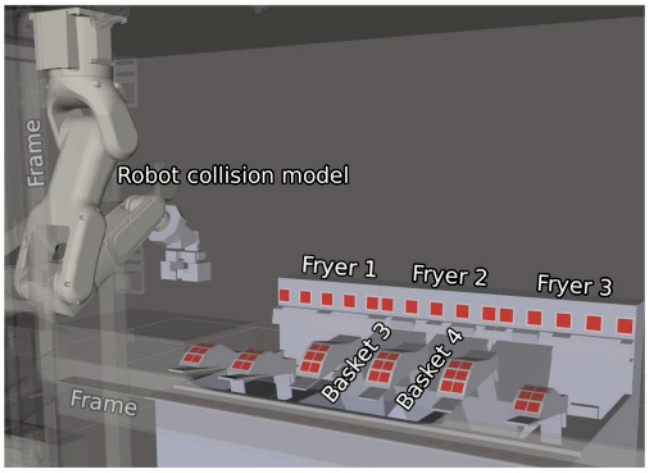


Fig. 3. The simulation environment, which shows the collision objects and their representations as mesh files. The same mesh representations are used on the hardware system.

To obtain the joint trajectory X via the CBF, we simply utilize a trajectory tracking controller like (38) along with the CBF-QP, and integrate its solution throughout the behavior.

B. Software Implementation and Simulation

Fig. 3 shows the simulated cooking environment. The robot and obstacle representations are a series of meshes described by URDF and SRDF files. The position and orientation of objects are updated before each planning attempt.we

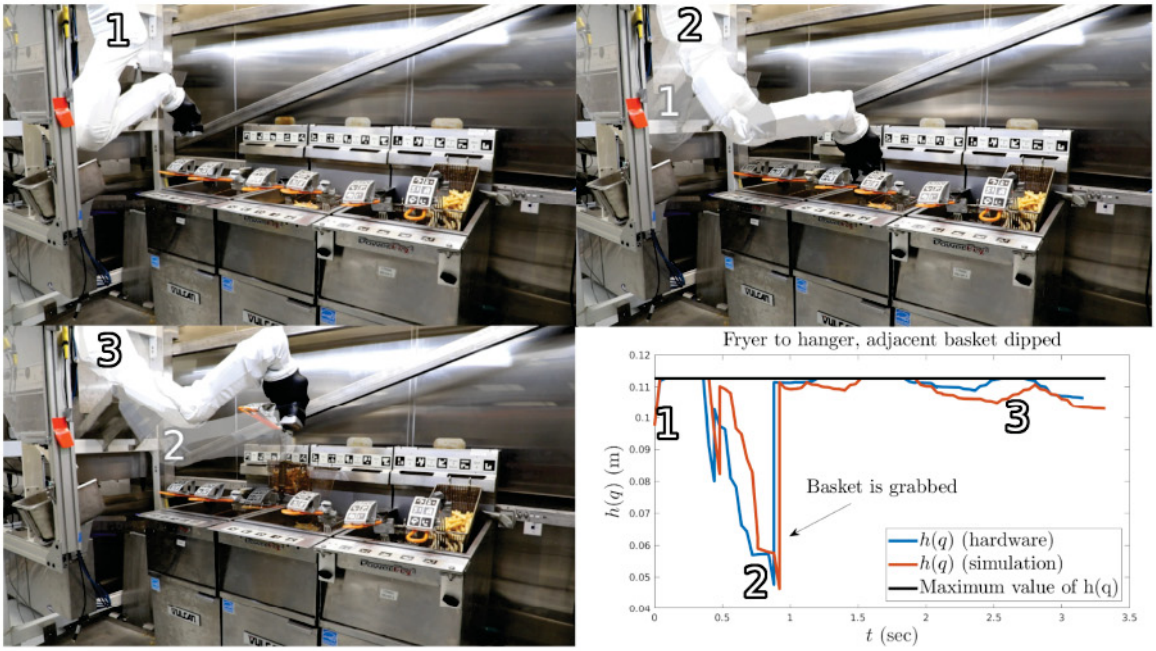
To implement the CBF filter, we require three values to be computed: the signed distance to the obstacles and other links $\text{sd}(q)$, the normal vectors corresponding to the points with minimal signed distance $\hat{n}(q)$, and the manipulator Jacobian at these points $J(q)$. The MoveIt framework [14], an open-source robotics software package for motion planning, is able to compute all three of these values. Specifically, the `distanceRobot()` and `distanceSelf()` functions of the `CollisionEnv` class provide the signed distances and normal vectors needed for environmental and self-collisions. Moreover, the `getJacobian()` function in the `RobotState` class returns the manipulator Jacobian. Thus, no other external libraries are required to implement this algorithm. Once these three values are computed, the OSQP quadratic program solver [23] is used to calculate the safe velocity commands, and integration is done manually.

Before hardware implementation, the algorithm was tested in simulation. The resulting behaviors are described in the next section, and the simulation results are shown along with the hardware trajectories in Fig. 4.

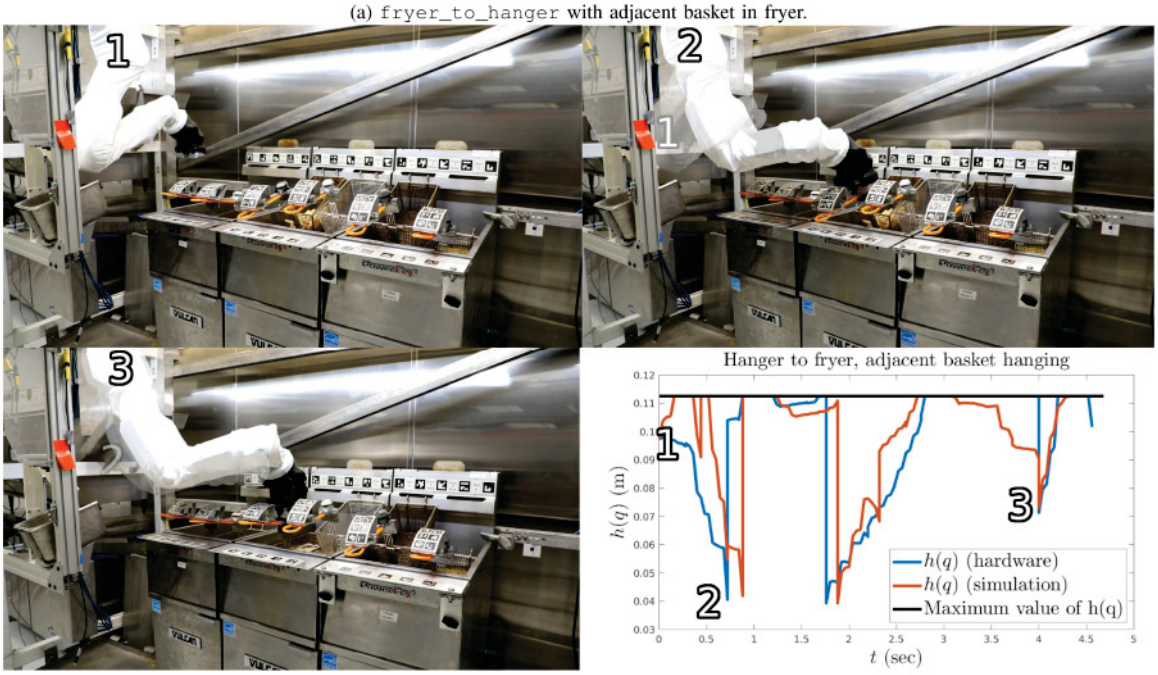
V. Hardware Results

A. Experimental Testing Environment

We apply the approach described in this paper to one of the Miso Robotics robotic cooking environments. Specifically, we utilize a FANUC LR Mate 200iD/7LC robotic manipulator



(a) `fryer_to_hanger` with adjacent basket in fryer.



(b) `hanger_to_fryer` with adjacent basket hanging.

Fig. 4. Two examples behaviors implemented on the Flippy2 robot. See https://youtu.be/nmkbya8XBmw for video. The large spikes in signed distance $h(q)$ come from enabling and disabling collision objects when required for interaction, like the basket when gripping and the fryer when hanging. At the maximum value of $h(q)$, the robot is only 11 cm away from the frame around it during these behaviors.

wrapped in a sleeve, and we send joint trajectories from an Intel i9-9900KF running ROS.

The cooking environment used in the testing is fully modeled using high-quality meshes used for collision checking. There are 36 collision objects in total, each represented by tens to hundreds of mesh triangles. The primary collision objects of concern are the six baskets, three industrial fryers, the hood vent over the fryers, and the glass pane separating the manipulator from the human workers. Of these objects, the baskets and fryers are the most commonly displaced.

As shown in the figures, the workspace of the manipulator is very densely crowded with obstacles. To complete a behavior, it is common to have less than a few centimeters of clearance between the robot and the surrounding environment. For this reason, planning methods must be minimally conservative, and there is no room for any collision buffer.

For experiments, a minimal cache was utilized to highlight the role of CBFs in re-planning around obstacles. In a commercial setting, with a more populated cache, the CBF would have many more prior trajectories to choose from, meaning that the path modifications would be much smaller. In practice, we find that the cache size saturates at around 200 stored behaviors, and we used roughly 10% of that.

B. Hardware Results

We test our framework's ability to safely re-plan on the two most volatile behaviors: `fryer_to_hanger` and `hanger_to_fryer`, described below. These behaviors see the most change in obstacle position and initial conditions, and are the most commonly re-planned behavior.

Fryer to hanger: The `fryer_to_hanger` behavior moves a basket from the dipped state to the hanging state. The manipulator picks up a basket that has finished cooking and hangs it, allowing the oil to drip off the basket before serving.

Hanger to Fryer: The `hanger_to_fryer` behavior is the reverse of `fryer_to_hanger`, transitioning a basket from the hanging state to the frying state.

Each behavior is tested in two primary configurations: one where the adjacent basket is submerged, and one where it is hanging. For the purpose of this paper, each of the four testing configurations were run 25 times, each with different cached trajectories and planning environments, for 100 total executions. The testing methodology was simple: for each setup, we first run the CBF on the best matching reference trajectory in the limited cache, and then we re-plan using TrajOpt for comparison purposes. Along with the true noise of the localization of the robot and environment, small amounts (several mm) of noise was further injected into the initial conditions and obstacles to ensure that the new trajectory differed significantly from the cache.

The CBF was able to produce a successful, collision-free trajectory in all 100 cases, even with the artificially limited cache size. The average computation time per CBF call was 2 ms, and the average computation time for the entire behavior was 223 ms. This is a significant improvement compared to TrajOpt's average computation time of 5923 ms. The CBF's trajectory computes waypoints every 10 ms compared to TrajOpt's 64 ms, thus no additional local planner needs to be used. Two example trajectories from the CBF are visualized in Fig. 4 with the value of $h(q)$ throughout the motion.

VI. Conclusion

In this work, we showcased control barrier functions (CBFs) for utilization in complex, real-world collision environments in the case of robotic cooking applications. First, we demonstrated how CBFs applied to the kinematics of robotic manipulators guarantee safety for the full-order dynamics. Then, we described the construction of these CBFs for very complex collision obstacle representations. We proposed an algorithm for filtering reference trajectories via CBFs to achieve safety and demonstrated these capabilities in the real-world application of frying foods.

References

[1] J. Schulman et al., "Motion planning with sequential convex optimization and convex collision checking," *Int. J. Robot. Res.*, vol. 33, no. 9, pp. 1251–1270, 2014.

[2] N. Ratliff, M. Zucker, J. A. Bagnell, and S Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009, pp. 489–494.

[3] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robot. Automat. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012.

[4] T. S. Lembono, A. Paolillo, E. Pignat, and S. Calinon, "Memory of motion for warm-starting trajectory optimization," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 2594–2601, Apr. 2020.

[5] S. Banerjee et al., "Learning-based warm-starting for fast sequential convex programming and trajectory optimization," in *Proc. IEEE Aerosp. Conf.*, 2020, pp. 1–8.

[6] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, "Motion planning as probabilistic inference using gaussian processes and factor graphs," *Robot.: Sci. Syst.*, vol. 12, no. 4, 2016.

[7] B. Baginski, "Local motion planning for manipulators based on shrinking and growing geometry models," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1996, pp. 3303–3308.

[8] R. Terasawa et al., "Achievement of dynamic tennis swing motion by offline motion planning and online trajectory modification based on optimization with a humanoid robot," in *Proc. IEEE-RAS 16th Int. Conf. Humanoid Robots*, 2016, pp. 1094–1100.

[9] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 3861–3876, Aug. 2017.

[10] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *Proc. 18th Eur. Control Conf.*, 2019, pp. 3420–3431.

[11] C. T. Landi, F. Ferraguti, S. Costi, M. Bonfè, and C. Secchi, "Safety barrier functions for human-robot interaction with industrial manipulators," in *Proc. 18th Eur. Control Conf.*, 2019, pp. 2565–2570.

[12] A. Singletary, P. Nilsson, T. Gurriet, and A. D. Ames, "Online active safety for robotic manipulators," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 173–178.

[13] A. Singletary, S. Kolathaya, and A. D. Ames, "Safety-critical kinematic control of robotic systems," *IEEE Contr. Syst. Lett.*, vol. 6, pp. 139–144, Jan. 2021.

[14] D. Coleman, I. Sucan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: A MoveIt! case study," *J. Softw. Eng. Robot.*, vol. 5, no. 1, pp. 3–16, 2014.

[15] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL, USA: CRC Press, 2017.

[16] T. G. Molnar, R. K. Cosner, A. W. Singletary, W. Ubellacker, and A. D. Ames, "Model-free safety-critical control for robotic systems," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 944–951, Apr. 2022.

[17] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE J. Robot. Automat.*, vol. 4, no. 2, pp. 193–203, Apr. 1988.

[18] G. Van Den Bergen, "Proximity queries and penetration depth computation on 3D game objects," in *Proc. Game Developers Conf.*, 2001.

[19] T. Sakai, "On riemannian manifolds admitting a function whose gradient is of constant norm," *Kodai Math. J.*, vol. 19, no. 1, pp. 39–51, 1996.

[20] S. Kolathaya and A. D. Ames, "Input-to-state safety with control barrier functions," *IEEE Control Syst. Lett.*, vol. 3, no. 1, pp. 108–113, Jan. 2019.

[21] A. Alan, A. J. Taylor, C. R. He, G. Orosz, and A. D. Ames, "Safe controller synthesis with tunable input-to-state safe control barrier functions," *IEEE Control Syst. Lett.*, vol. 6, pp. 908–913, Jun. 2021.

[22] P. Glotfelter, J. Cortés, and M. Egerstedt, "Nonsmooth barrier functions with applications to multi-robot systems," *IEEE Control Syst. Lett.*, vol. 1, no. 2, pp. 310–315, Oct. 2017.

[23] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," *Math. Program. Comput.*, vol. 12, no. 4, pp. 637–672, 2020.