Collision Cone Control Barrier Functions: Experimental Validation on UGVs for Kinematic Obstacle Avoidance

Bhavya Giri Goswami[*1], Manan Tayal[*1], Karthik Rajgopal[1], Pushpak Jagtap[1], Shishir Kolathaya[1]

Abstract— This paper introduces an experimental platform designed for the validation and demonstration of a novel class of Control Barrier Functions (CBFs) tailored for Unmanned Ground Vehicles (UGVs) to proactively prevent collisions with kinematic obstacles by integrating the concept of collision cones. While existing CBF formulations excel with static obstacles, extensions to torque/acceleration-controlled unicycle and bicycle models have seen limited success. Conventional CBF applications in such nonholonomic UGV models have demonstrated control conservatism, particularly in scenarios where steering/thrust control was deemed infeasible. Drawing inspiration from collision cones in path planning, we present a pioneering C3BF formulation ensuring theoretical safety guarantees for such models. The core premise revolves around aligning the obstacle's velocity away from the vehicle, establishing a constraint to perpetually avoid vectors directed towards it. This control methodology is rigorously validated through simulations and experimental verification on the Copernicus mobile robot (Unicycle Model) and FOCAS-Car (Bicycle Model).

Fig. 1: Test setups: Copernicus (left); FOCAS-Car (right).

I. INTRODUCTION

The progress in autonomous technologies has facilitated the deployment of robots in a wide range of environments, often in close proximity to humans. Consequently, the design of controllers with formally assured safety has become a crucial aspect of these safety-critical applications, constituting an active area of research in recent times. Researchers have devised various methodologies to address this challenge, including model predictive control (MPC) [1], reference governor [2], reachability analysis [3] [4], and artificial potential fields [5]. To establish formal safety guarantees, such as collision avoidance with obstacles, it is essential to employ a safety-critical control algorithm that encompasses both trajectory tracking/planning and prioritizes safety over tracking. One such recent approach is based on control barrier functions [6], [7] (CBFs).

A significant advantage of using CBF-based quadratic programs over other state-of-the-art techniques is that they work efficiently on real-time practical applications in complex dynamic environments [4], [5]; that is, optimal control inputs can be computed at a very high frequency. They act as a fast safety filter over existing path planning controllers [8]. CBFs are already being used for UGVs collision avoidance [6], [9]–[13]. Many contributions shown here are for point mass dynamic models with static obstacles, while others are extended
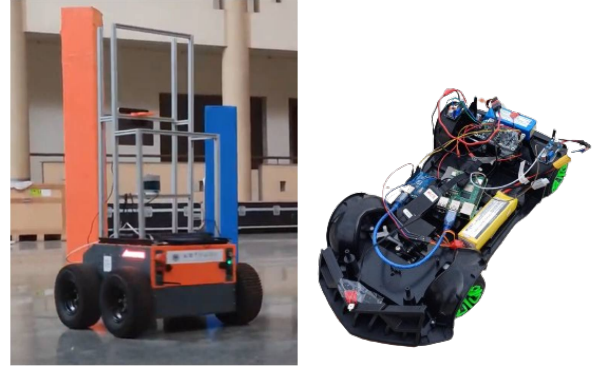
[1]Robert Bosch Center for Cyber-Physical Systems (RBCCPS), Indian Institute of Science (IISc), Bengaluru. bhavya.goswami@uwaterloo.ca {manantayal, karthikrajgo, pushpak, shishirk}@iisc.ac.in

for unicycle models [14]–[16]. However, these extensions are only for velocity-controlled models, not for acceleration-controlled models. On the other hand, the Higher Order CBF (HOCBF) based approaches mentioned in [17] are shown to successfully avoid collisions with static obstacles using acceleration-controlled unicycle models but lack geometrical intuition. Extension of this framework (HOCBF) for the case of moving obstacles is possible, however, safety guarantees are provided for a subset of the original safe set, thereby making it conservative [18]. In addition, computation for a feasible CBF candidate with apt penalty and parameter terms is not that straightforward (Section-II).

Thus, two major challenges remain that prevent the successful deployment of these CBF-QPs for obstacle avoidance in a dynamic environment: a) Existing CBFs are not able to handle the nonholonomic nature of UGVs well. They provide limited control capability in the acceleration-controlled unicycle and bicycle models, i.e., the solutions from the CBF-QPs have either no steering or forward thrust capabilities (Section II), and b) Existing CBFs are not able to handle dynamic obstacles well, i.e., the controllers are not able to avoid collisions with moving obstacles (Section II).

Intending to address the above challenges, we propose a new class of CBFs via the concept of collision cones [19]–[24]. Collision cones have already been incorporated into MPC by defining the cones as constraints [25], but making it a CBF-based constraint has yet to be addressed. In particular, we generate a new class of constraints that ensure that the relative velocity between the obstacle and the vehicle always points away from the direction of the vehicle's approach. Assuming ellipsoidal shape for the obstacles [21], the resulting set of unwanted directions for potential collision forms a conical shape, giving rise to the synthesis of **Collision Cone**

Control Barrier Functions (C3BFs). The C3BF-based QP optimally and rapidly calculates inputs in real-time such that the relative velocity vector direction is kept out of the collision cone for all time. This approach is demonstrated using the acceleration-controlled non-holonomic models (Fig. 1).

A. Contribution and Paper Structure

Our specific contributions include the following:

- We formulate a new class of CBFs by using the concept of collision cones. The resulting CBF-QP can be computed in real-time, thereby yielding a safeguarding controller that avoids collision with moving obstacles. This can be stacked over any state-of-the-art planning algorithm with the C3BF-QP acting as a safety filter.
- We formally show how the proposed formulation yields a safeguarding control law for acceleration-controlled wheeled robots with nonholonomic constraints. Note that much of the existing works on CBFs for mobile robots are with point mass models, and our contributions here are for nonholonomic vehicle models, which are more challenging to control. We demonstrate our solutions in both simulations and hardware experiments.

II. BACKGROUND

In this section, we provide the background necessary to formulate our problem of moving obstacle avoidance. Specifically, we first describe the acceleration-controlled unicycle and the bicycle vehicle models considered in our work. Next, we briefly introduce Control Barrier Functions (CBFs) and their importance for real-time safety-critical control for considered vehicle models. Finally, we explain the shortcomings in existing CBF approaches in the context of collision avoidance of moving obstacles.
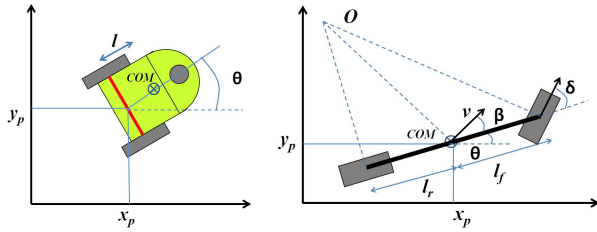

Fig. 2: Schematic of Unicycle (left); Bicycle model (right).

A. Vehicle Models

1) Acceleration controlled unicycle model: The unicycle model encompasses state variables denoted as x_p, y_p, θ, v, and ω, representing pose, linear velocity, and angular velocity, respectively. Linear acceleration (a) and angular acceleration (α) serve as the control inputs. In Figure 2, a differential drive robot is depicted, which is effectively described by the unicycle model presented below:

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \\ \dot{\theta} \\ \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ \alpha \end{bmatrix}. \tag{1}$$

2) Acceleration controlled bicycle model: The bicycle model comprises two wheels, with the front wheel dedicated to steering (Fig.2). The simplified dynamics of the bicycle model with small slip angle assumption [9] are outlined as follows:

$$\underbrace{\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \\ \dot{\theta} \\ \dot{v} \end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix} v\cos\theta \\ v\sin\theta \\ 0 \\ 0 \end{bmatrix}}_{f(x)} + \underbrace{\begin{bmatrix} 0 & -v\sin\theta \\ 0 & v\cos\theta \\ 0 & \frac{v}{l_r} \\ 1 & 0 \end{bmatrix}}_{g(x)} \underbrace{\begin{bmatrix} a \\ \beta \end{bmatrix}}_{u} \tag{2}$$

$$\beta = \tan^{-1}\left(\frac{l_r}{l_f + l_r}\tan(\delta)\right), \tag{3}$$

x_p and y_p denotes vehicle center of mass (CoM) coordinates in an inertial frame. θ is vehicle's orientation with respect to the x axis. a is the linear acceleration of CoM. l_f and l_r are the distances of the front and rear axles from the CoM, respectively. δ is the steering angle of the vehicle and β is the vehicle's slip angle, not to be confused with the tire slip angle (Fig.2).

Since the control inputs a, β are now affine in the dynamics, CBF-QPs can be constructed directly to yield real-time control laws. More model details can be found on [26].

B. Control barrier functions (CBFs)

After providing an overview of the vehicle models, we proceed to formally introduce control barrier functions and their application in ensuring safety within the context of a non-linear control system described by the affine equation:

$$\dot{x} = f(x) + g(x)u. \tag{4}$$

Here, x belongs to the set $\mathcal{D} \subseteq \mathbb{R}^n$ representing the system's state, and u lies in $\mathbb{U} \subseteq \mathbb{R}^m$ representing the system's input. We assume that the functions $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are locally Lipschitz. Given a Lipschitz continuous control law $u = k(x)$, the resulting closed-loop system $\dot{x} = f_{cl}(x) = f(x) + g(x)k(x)$ leads to a solution $x(t)$, with an initial condition of $x(0) = x_0$.

Consider a set $\mathcal{C}$ defined as the *super-level set* of a continuously differentiable function $h : \mathcal{D} \to \mathbb{R}$ yielding,

$$\mathcal{C} = \{x \in \mathcal{D} \subset \mathbb{R}^n : h(x) \geq 0\} \tag{5}$$

$$\partial\mathcal{C} = \{x \in \mathcal{D} \subset \mathbb{R}^n : h(x) = 0\} \tag{6}$$

$$\text{Int}(\mathcal{C}) = \{x \in \mathcal{D} \subset \mathbb{R}^n : h(x) > 0\}. \tag{7}$$

It is assumed that the interior of $\mathcal{C}$ is non-empty, and $\mathcal{C}$ has no isolated points, i.e., $\text{Int}(\mathcal{C}) \neq \emptyset$ and the closure of the interior of $\mathcal{C}$ is equal to $\mathcal{C}$. The system is deemed safe with respect to the control law $u = k(x)$ if for any $x(0) \in \mathcal{C}$, it follows that $x(t) \in \mathcal{C}$ for all $t \geq 0$.

We can ascertain the safety provided by the controller $k(x)$ through the use of control barrier functions (CBFs), as defined below.

Definition 1 (Control barrier function (CBF)): Given the set $\mathcal{C}$ defined by (5)-(7), with $\frac{\partial h}{\partial x}(x) \neq 0 \ \forall x \in \partial\mathcal{C}$, the function h is called the control barrier function (CBF)

defined on the set $\mathcal{D}$, if there exists an extended class $\mathcal{K}$ function[1] κ such that for all $x \in \mathcal{D}$:

$$\sup_{u \in \mathbb{U}} \left[\underbrace{\mathcal{L}_f h(x) + \mathcal{L}_g h(x)u}_{\dot{h}(x,u)} + \kappa\left(h(x)\right) \right] \geq 0 \qquad (8)$$

where $\mathcal{L}_f h(x) := \frac{\partial h}{\partial x} f(x)$ and $\mathcal{L}_g h(x) := \frac{\partial h}{\partial x} g(x)$ are the Lie derivatives.

As per [7] and [27], any Lipschitz continuous control law $k(x)$ that satisfies the inequality $\dot{h} + \kappa(h) \geq 0$ ensures the safety of $\mathcal{C}$ given $x(0) \in \mathcal{C}$, and it leads to asymptotic convergence towards $\mathcal{C}$ if $x(0)$ is outside of $\mathcal{C}$. Notably, CBFs can also be defined solely on $\mathcal{C}$, ensuring only safety. This will find utility in the bicycle model (2), which is elaborated upon later.

C. Controller synthesis for real-time safety

Having described the CBF and its associated formal results, we now discuss its Quadratic Programming (QP) formulation. CBFs are typically regarded as *safety filters* which take the desired input (reference controller input) $u_{ref}(x,t)$ and modify this input in a minimal way:

$$u^*(x,t) = \arg\min_{u \in \mathbb{U} \subseteq \mathbb{R}^m} \|u - u_{ref}(x,t)\|^2$$
$$\text{s.t. } \mathcal{L}_f h(x) + \mathcal{L}_g h(x)u + \kappa\left(h(x)\right) \geq 0 \qquad (9)$$

This is called the Control Barrier Function based Quadratic Program (CBF-QP). The reference control input can be from any SOTA algorithm. If $\mathbb{U} = \mathbb{R}^m$, then the QP is feasible, and the explicit solution is given by

$$u^*(x,t) = u_{ref}(x,t) + u_{safe}(x,t),$$

where $u_{safe}(x,t)$ can be analytically derived usign KKT conditions [18], [26].

D. Classical CBFs and moving obstacle avoidance

We now explore collision avoidance in Unmanned Ground Vehicles (UGVs). In particular, we discuss the problems associated with the classical CBF-QPs, especially with the velocity obstacles. It's proved in our previous work [18], [26], [28] how a simple Ellipse CBF (10) is not a valid with acceleration-controlled unicycle (1) and bicycle model (2).

$$h(x,t) = \left(\frac{c_x(t) - x_p}{c_1} \right)^2 + \left(\frac{c_y(t) - y_p}{c_2} \right)^2 - 1, \quad (10)$$

It is worth mentioning that for the acceleration-controlled unicycle models (1), we can use another class of CBFs introduced specifically for constraints with higher relative degrees: HOCBF [29]–[31] given by:

$$h_2 = \dot{h}_1 + \kappa(h_1), \qquad (11)$$

[1]A continuous function $\kappa : [0,d) \to [0,\infty)$ for some $d > 0$ is said to belong to class-$\mathcal{K}$ if it is strictly increasing and $\kappa(0) = 0$. Here, d is allowed to be ∞. The same function can be extended to the interval $\kappa : (-b, d) \to (-\infty, \infty)$ with $b > 0$ (which is also allowed to be ∞), in which case we call it the extended class $\mathcal{K}$ function.

where h_1 is the equation of ellipse given by (10).

Apart from lacking geometrical intuition, h_2 for acceleration-controlled unicycle model will result in a conservative, safe set as per [17, Theorem 3]. For acceleration controlled bicycle model (2) with same HOCBF h_2 (11), if $\mathcal{L}_g h_2 = 0$ then we can choose $\dot{c}_x, \dot{c}_y$ in such a way that $\mathcal{L}_f h_2 + \frac{\partial h_2}{\partial t} \not\geq 0$ which results in an invalid CBF. For a detailed proof and comparison refer to [18], [28]. The results are summarised in Table-I.

However, our goal in this paper is to develop a CBF formulation with geometrical intuition that provides safety guarantees to avoid moving obstacles with the acceleration-controlled non-holonomic models, which is presented next.

III. COLLISION CONE CBF

We will now introduce our proposed approach, in which a collision cone, defined for a pair of objects, represents a predictive set used to assess the potential for a collision between them based on their relative velocity direction. Specifically, it signifies the directions in which either object, if followed, would lead to a collision between the two. Throughout this paper, we consider obstacles as ellipses, treating the vehicle as a point. Henceforth, when we mention the term "collision cone", we are referring to this scenario, with the center of the UGV as the point of reference and the velocity and positions of the obstacle relative to this point.

Consider a UGV defined by the system (4) and a moving obstacle. This is visually depicted in Figure 3. To account for the UGV's dimensions, we over-approximate [21] the obstacle as a conservative circle that encompasses the ellipse (with radius $r = \max(c_1, c_2) + \frac{w}{2}$), where c_1, c_2 are the dimensions of the ellipse and w is the maximum dimension of the UGV. This over-approximation enhances the safety threshold and removes complexity related to the orientation of the obstacle.

For a collision to occur, the relative velocity of the obstacle must be directed towards the UGV. Consequently, the relative velocity vector should not point into the shaded pink region labeled EHI in Figure 3, which takes the form of a cone.

Let $\mathcal{C}$ denote this set of safe directions for the relative velocity vector. If there exists a function $h : \mathcal{D} \subseteq \mathbb{R}^n \to \mathbb{R}$ that satisfies the conditions outlined in Definition 1 on $\mathcal{C}$, then we can assert that a Lipschitz continuous control law derived from the resulting Quadratic Program (QP) (9) for the system ensures that the vehicle will avoid colliding with the obstacle, even if the reference control signal u_{ref} attempts to steer them towards a collision course. This innovative approach, which avoids the pink cone region, gives rise to what we term as *Collision Cone Control Barrier Functions (C3BFs)* [18], [26], [28], formulated as:

$$h(x,t) = <p_{\text{rel}}, v_{\text{rel}}> + \|p_{\text{rel}}\| \|v_{\text{rel}}\| \cos\phi \qquad (12)$$

where ϕ is the half angle of the cone (Fig. 3). $< \cdot , \cdot >$ denotes the inner product of two vectors. p_{rel} p_{rel} is the relative position vector between the vehicle COM and the center of the obstacle and v_{rel} is the relative velocity vector defined in the next section.

TABLE I: Comparison between the Ellipse CBF (10), HOCBF (11) and the proposed C3BF (12) for different UGV models.

CBFs	Vehicle Models	Static Obstacle (c_x, c_y)	Moving Obstacle $(c_x(t), c_y(t))^\dagger$
Ellipse CBF	Unicycle (1)	Not a valid CBF	Not a valid CBF
Ellipse CBF	Bicycle (2)	Valid CBF, No acceleration	Not a valid CBF
HOCBF	Unicycle (1)	Valid CBF, No steering	Valid CBF, but conservative
HOCBF	Bicycle (2)	Valid CBF	Not a valid CBF
C3BF	Unicycle (1)	Valid CBF in $\mathcal{D}$	Valid CBF in $\mathcal{D}$
C3BF	Bicycle (2)	Valid CBF in $\mathcal{C}$	Valid CBF in $\mathcal{C}$

$\dagger$ $(c_x(t), c_y(t))$ are continuous (or at least piece-wise continuous) functions of time
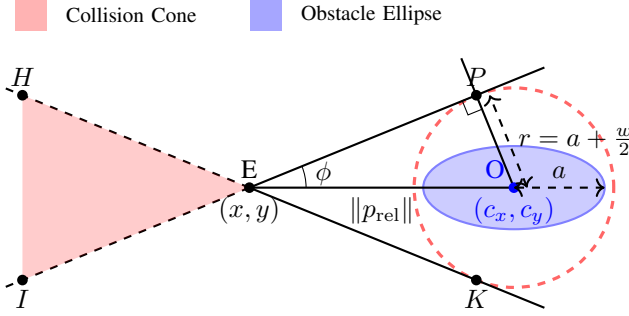

Fig. 3: Construction of collision cone for an elliptical obstacle considering the UGV's dimensions (width: w).

A. Application to systems

1) Acceleration-controlled unicycle model: p_{rel} is given by:

$$p_{\text{rel}} := \begin{bmatrix} c_x - (x_p + l\cos\theta) \\ c_y - (y_p + l\sin\theta) \end{bmatrix}. \qquad (13)$$

where l is the distance of the body center from the differential drive axis (Fig.2) and v_{rel} is calculated by taking derivative of p_{rel} as:

$$v_{\text{rel}} := \begin{bmatrix} \dot{c}_x - (v\cos\theta - l\sin\theta * \omega) \\ \dot{c}_y - (v\sin\theta + l\cos\theta * \omega) \end{bmatrix}. \qquad (14)$$

Theorem 1: Given the acceleration-controlled unicycle model (1), the proposed CBF candidate (12) with $p_{\text{rel}}, v_{\text{rel}}$ defined by (13), (14) is a valid CBF defined for the set $\mathcal{D}$.

2) Acceleration-controlled Bicycle model: p_{rel} and v_{rel} is given by:

$$p_{\text{rel}} := \begin{bmatrix} c_x - x_p & c_y - y_p \end{bmatrix}^T \qquad (15)$$

$$v_{\text{rel}} := \begin{bmatrix} \dot{c}_x - v\cos\theta & \dot{c}_y - v\sin\theta \end{bmatrix}^T. \qquad (16)$$

Here, v_{rel} is NOT equal to the relative velocity $\dot{p}_{\text{rel}}$. However, for small β, we can assume that v_{rel} is the difference between obstacle velocity and the velocity component along the length of the vehicle $v\cos\beta \approx v$ (Fig.3).

Theorem 2: Given the bicycle model (2), the proposed candidate CBF (12) with $p_{\text{rel}}, v_{\text{rel}}$ defined by (15), (16) is a valid CBF defined for the set $\mathcal{C}$.

Readers are advice to refer [26, Theorem 1] [28] and [26, Theorem 2] [28] for the detailed proof as the focus of this work is on experimental validation of C3BF.

IV. RESULTS

In this section, we provide the simulation and experimental results to validate the proposed C3BF-QP. We will first demonstrate the results graphically in Python simulations and then on the Copernicus UGV (modeled as a unicycle) and the FOCAS-Car (modeled as a bicycle).

A. Acceleration Controlled Unicycle Model

In both simulation and experiments, we have considered the reference control inputs as a simple proportional controller formulated as follows:

$$u_{ref}(x) = \begin{bmatrix} a_{ref}(x) \\ \alpha_{ref}(x) \end{bmatrix} = \begin{bmatrix} k_1 * (v_{des} - v) \\ -k_2 * \omega \end{bmatrix}, \qquad (17)$$

where k_1, k_2 are constant gains, $v_{des} \leq v_{max}$ is the target velocity of vehicle. We chose constant target velocities for verifying the C3BF-QP. For the class $\mathcal{K}$ function in the CBF inequality, we chose $\kappa(h) = \gamma h$, where $\gamma = 1$.

Note that the reference controller can be replaced by any existing trajectory tracking / path-planning / obstacle-avoiding controller like the Stanley controller [32] or MPC [33]. Thus, by integrating the C3BF safety filter with existing state-of-the-art algorithms, we can provide formal guarantees on their safety concerning obstacle avoidance.

A virtual perception boundary was incorporated, considering the maximum range for the perception sensors. As soon as an obstacle is detected within the perception boundary, the C3BF-QP is activated with u_{ref} given by (17). The QP yields the optimal accelerations, which are then applied to the robot.

1) Python Simulations: We consider different scenarios with different initial poses and velocities of both the vehicle and the obstacle. Different scenarios include static obstacles resulting in a) turning, b) braking and moving obstacles resulting in c) reversing, and d) overtaking, as shown in Fig. 4. The corresponding evolution of CBF value (h) as a function of time is shown in Fig. 5. We can observe that even if $h < 0$ at t = 0 (Fig. 5(b), (c)), the magnitude is exponentially decreasing and becoming positive.

2) Hardware Experiments: Corresponding to the cases tested in Python simulation, experiments were performed on Botsync's Copernicus UGV (Fig. 1). It is a differential drive (4X4) robot with 860 x 760 x 590 mm dimensions. Apart from its internal wheel encoders, external sensors like Ouster OS1-128 Lidar and Xsens IMU MTI-670G sensor were used for localization and minimizing the odometric
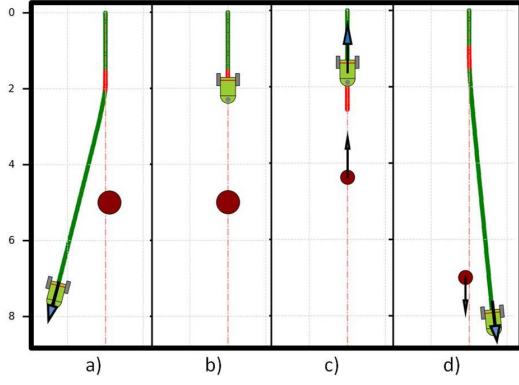
Fig. 4: Different acceleration-controlled unicycle model behaviors with static (a,b) & moving (c,d) obstacle. a) turning; b) braking; c) reversing; d) overtaking. The orange lines represent the points where C3BF is active.
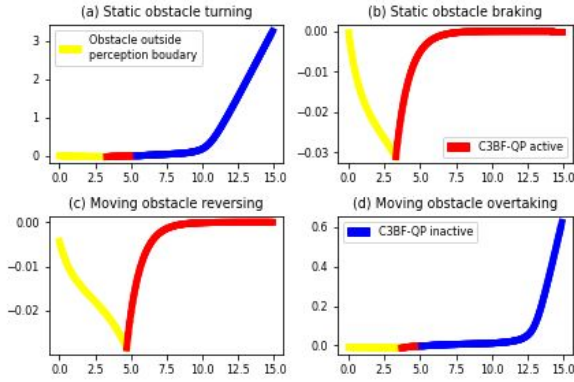

Fig. 5: CBF value (h) (Y-axis) varying with time (X-axis). C3BF-QP active means $u_{safe} \neq 0$ (Red) and C3BF-QP inactive means $u_{safe} = 0$ (Blue).

error through LIO-SAM algorithm [34]. Zed-2 Stereo camera was used to detect the obstacles through masking and image segmentation techniques and to get their positions using depth information as shown in Fig 6.

Our C3BF algorithm combined with the perception stack was first simulated in the Gazebo environment and then tested on the actual robot. We observed braking, turning, reversal, and overtaking behaviors depending on the initial poses of Copernicus, different positions, and velocities of obstacles. These behaviors were similar to the results obtained from Python simulations. We also experimented with various scenarios with multiple stationary obstacles. Even though the behaviors are sensitive to the initial conditions of the robot and obstacle, collision is avoided in all cases. All the results are shown in the attached video link[2].

B. Acceleration Controlled Bicycle Model

We have extended and validated our C3BF algorithm for the bicycle model (2) which is a good approximation of actual car dynamics in low-speed scenarios where the lateral acceleration is small ($\leq 0.5\mu g$, μ is the friction co-efficient) [9] [35]. We first simulated all the same four cases without
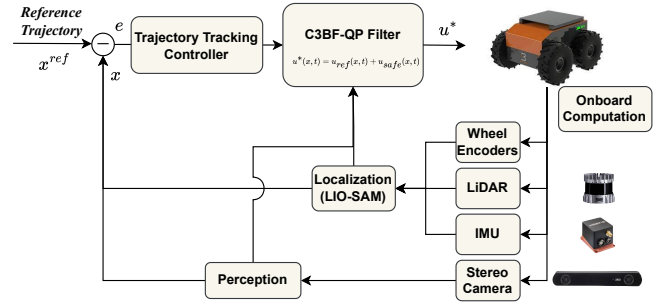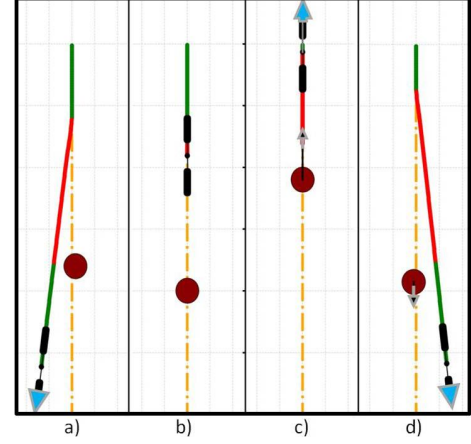

Fig. 6: Experimental Setup for Copernicus UGV

Fig. 7: Different acceleration-controlled bicycle model behaviors with static (a,b) & moving (c,d) obstacle. a) turning; b) braking; c) reversing; d) overtaking.

any trajectory tracking reference controller as we did for the unicycle model (Section IV-A.1) and got similar results as shown in Fig. 7. For the second set of simulations, the reference acceleration a_{ref} was obtained from a P-controller tracking the desired velocity, while the reference steering β_{ref} was obtained from the Stanley controller [32]. The Stanley controller is fed an explicit reference spline trajectory for tracking, with an obstacle on it. The reference controllers were integrated with the C3BF-QP and applied to the robot simulated in Python (Fig. 8) and CARLA simulator.

1) Python Simulations: Fig. 7 shows different behaviors (turning, braking, reversing, overtaking) of the vehicle modeled as a bicycle model without reference trajectory tracking. Fig. 8 shows the trajectory setup along with an obstacle placed towards the end. The vehicle tracks the reference spline trajectory (red), and the reflection of the collision cone is shown by the pink region. The C3BF algorithm preemptively prevents the relative velocity vector from falling into the unsafe set, thus avoiding obstacles by circumnavigating. It can be verified from Fig. (8b) that the slip angle β remains small, which is in line with the assumption used (2).

2) CARLA Simulations: With the small β approximation and using the formulation from Section III-A.2, numerical simulations (using *CVXOPT* library) were performed on virtual car model on CARLA simulator [36] to demonstrate

(a) Ego Vehicle avoiding static obstacle (black arrow: $-v_{\text{rel}}$).

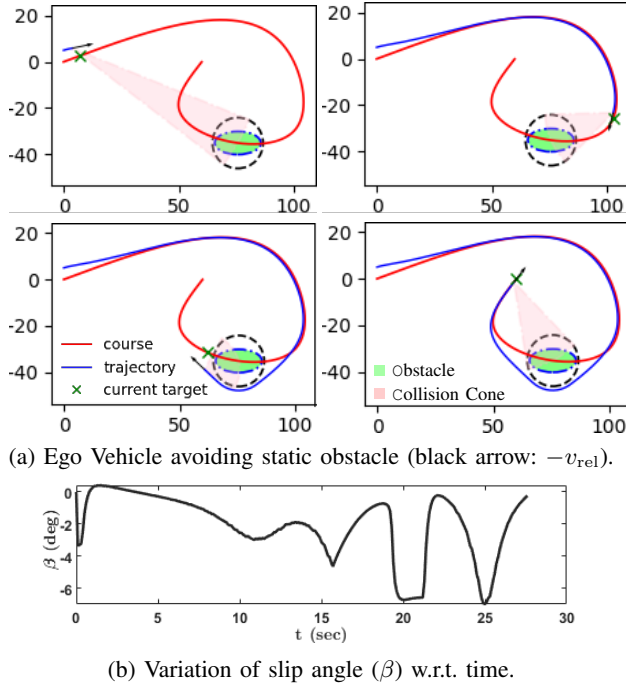(b) Variation of slip angle (β) w.r.t. time.

Fig. 8: Graphical illustration of C3BF on bicycle model (2).

Fig. 9: Ego-vehicle (white) avoiding multiple moving cars (Red) using C3BF controller with graphical collision cones.

the efficacy of C3BF in collision avoidance in real-world like environments (Fig. 9). A virtual perception boundary was used to accommodate the range of perception sensors on real self-driving cars. State information about the obstacle is programmatically retrieved from the CARLA server. The resulting simulation experiments can be viewed in the attached video link[2].

3) Hardware Experiments: Corresponding to the cases tested in Python simulation, experiments were performed on FOCAS-Car (Fig. 1), a rear wheel drive car $1/10th$ scaled model, which is powered by 1800 *mAh* LIPO Battery. The steering actuation is handled by a servo motor TowerPro MG995, and a DC motor actuates the rear wheels. The realization of the proposed controller has been divided into two stages: a high-level controller running ROS (Robotic Operating System) on Ubuntu and a low-level controller realized by an Arduino UNO board. The sampling rate for the ROS master is set to 100 Hz. The Raspberry Pi 4, equipped

with Ubuntu 20.04 LTS and ROS Noetic, is mounted on the car to solve the online optimization problem (C3BF-QP). The controller has been coded as a ROS Node in Python. At the low level, this controller has less computation, and it runs at a frequency of 57600 BaudRate. The global position of the car, as well as the obstacle(s), is measured using PhaseSpace™ motion capture system with a tracking frequency of 960 Hz. The 2 LED Markers are placed in front and rear of the car as shown in Fig. 10 to estimate the state of the car (p, v, θ, ω).

Similar to the Python simulation, we observed braking, turning, and overtaking behaviors and irrespective of the initial conditions, collision is avoided in all cases.
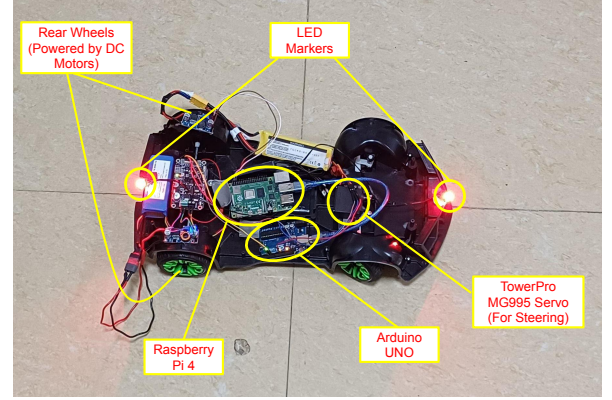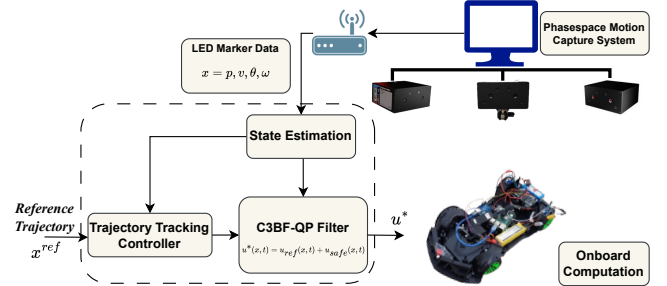

Fig. 10: FOCAS-Car Hardware Setup

Fig. 11: Experimental Setup for FOCAS-Car.

All the simulations and hardware experiments can be viewed in the attached video link[2]

V. CONCLUSIONS

In this paper, we presented a novel real-time control methodology for UGVs for avoiding moving obstacles by using the concept of collision cones. A well-known collision-cone technique used in planning [19]–[22], was extended with control barrier functions (CBFs) with acceleration-controlled nonholonomic UGV models (unicycle and bicycle models). This enabled fast modification of reference control inputs giving guarantees of collision avoidance in real-time. The proposed QP formulation (C3BF-QP) acts as a filter on the reference controller allowing the vehicle to safely maneuver under different scenarios presented in the paper. Our

[2].https://tayalmanan28.github.io/C3BF-UGV/

C3BF safety filter can be integrated with any existing state-of-the-art trajectory tracking/path-planning/obstacle-avoiding algorithms to ensure real-time safety by giving formal guarantees. We validated the proposed C3BF controller in simulations as well as hardware by integrating it with simple trajectory-tracking reference controllers. As part of future work, the focus will be more on self-driving vehicles with complex dynamic models for avoiding moving obstacles on the road.

REFERENCES

[1] S. Yu, M. Hirche, Y. Huang, H. Chen, and F. Allgöwer, "Model predictive control for autonomous ground vehicles: a review," *Autonomous Intelligent Systems*, vol. 1, 2021.

[2] I. Kolmanovsky, E. Garone, and S. Di Cairano, "Reference and command governors: A tutorial on their theory and automotive applications," in *American Control Conference*, 2014, pp. 226–241.

[3] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-jacobi reachability: A brief overview and recent advances," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, pp. 2242–2253.

[4] Z. Li, "Comparison between safety methods control barrier function vs. reachability analysis," *arXiv preprint arXiv:2106.13176*, 2021.

[5] A. W. Singletary, K. Klingebiel, J. R. Bourne, N. A. Browning, P. T. Tokumaru, and A. D. Ames, "Comparative analysis of control barrier functions and artificial potential fields for obstacle avoidance," *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8129–8136, 2021.

[6] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 6271–6278.

[7] 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.

[8] A. Manjunath and Q. Nguyen, "Safe and robust motion planning for dynamic robotics via control barrier functions," in *60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 2122–2128.

[9] S. He, J. Zeng, B. Zhang, and K. Sreenath, "Rule-based safety-critical control design using control barrier functions with application to autonomous lane change," in *American Control Conference (ACC)*. IEEE, 2021, pp. 178–185.

[10] Y. Chen, H. Peng, and J. Grizzle, "Obstacle avoidance for low-speed autonomous vehicles with barrier function," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 1, pp. 194–206, 2018.

[11] T. D. Son and Q. Nguyen, "Safety-critical control for non-affine nonlinear systems with application on autonomous vehicle," in *IEEE 58th Conference on Decision and Control (CDC)*, 2019, pp. 7623–7628.

[12] Y. Chen, A. Singletary, and A. D. Ames, "Guaranteed obstacle avoidance for multi-robot operations with limited actuation: A control barrier function approach," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 127–132, 2021.

[13] M. Abduljabbar, N. Meskin, and C. G. Cassandras, "Control barrier function-based lateral control of autonomous vehicle for roundabout crossing," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 859–864.

[14] W. Xiao, C. G. Cassandras, and C. A. Belta, "Bridging the gap between optimal trajectory planning and safety-critical control with applications to autonomous vehicles," *Automatica*, vol. 129, p. 109592, 2021.

[15] C. Li, Z. Zhang, A. Nesrin, Q. Liu, F. Liu, and M. Buss, "Instantaneous local control barrier function: An online learning approach for collision avoidance," *arXiv preprint arXiv:2106.05341*, 2021.

[16] M. Marley, R. Skjetne, and A. R. Teel, "Synergistic control barrier functions with application to obstacle avoidance for nonholonomic vehicles," in *American Control Conference (ACC)*, 2021, pp. 243–249.

[17] W. Xiao and C. Belta, "High-order control barrier functions," *IEEE Transactions on Automatic Control*, vol. 67, no. 7, pp. 3655–3662, 2022.

[18] M. Tayal and S. Kolathaya, "Control barrier functions in dynamic uavs for kinematic obstacle avoidance: a collision cone approach," *arXiv preprint arXiv:2303.15871*, 2023.

[19] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using the relative velocity paradigm," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, 1993, pp. 560–565 vol.1.

[20] ——, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[21] A. Chakravarthy and D. Ghose, "Obstacle avoidance in a dynamic environment: a collision cone approach," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 28, no. 5, pp. 562–574, 1998.

[22] ——, "Generalization of the collision cone approach for motion safety in 3-d environments," *Autonomous Robots*, vol. 32, no. 3, pp. 243–266, 2012.

[23] D. Bhattacharjee, A. Chakravarthy, and K. Subbarao, "Nonlinear model predictive control and collision-cone-based missile guidance algorithm," *Journal of Guidance, Control, and Dynamics*, vol. 44, no. 8, pp. 1481–1497, 2021.

[24] J. Goss, R. Rajvanshi, and K. Subbarao, *Aircraft Conflict Detection and Resolution Using Mixed Geometric And Collision Cone Approaches*. AIAA Guidance, Navigation, and Control Conference and Exhibit, 2004, ch. 0, p. 0.

[25] M. Babu, Y. Oza, A. K. Singh, K. M. Krishna, and S. Medasani, "Model predictive control for autonomous driving based on time scaled collision cone," in *2018 European Control Conference (ECC)*. IEEE, 2018, pp. 641–648.

[26] P. Thontepu, B. G. Goswami, M. Tayal, N. Singh, S. S. P I, S. S. M G, S. Sundaram, V. Katewa, and S. Kolathaya, "Collision cone control barrier functions for kinematic obstacle avoidance in ugvs," in *2023 Ninth Indian Control Conference (ICC)*, 2023, pp. 293–298.

[27] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *18th European Control Conference (ECC)*, 2019, pp. 3420–3431.

[28] M. Tayal, B. G. Goswami, K. Rajgopal, R. Singh, T. Rao, J. Keshavan, P. Jagtap, and S. Kolathaya, "A Collision Cone Approach for Control Barrier Functions," *arXiv e-prints*, p. arXiv:2403.07043, Mar. 2024.

[29] Q. Nguyen and K. Sreenath, "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints," in *American Control Conference (ACC)*, 2016, pp. 322–328.

[30] W. Xiao and C. Belta, "Control barrier functions for systems with high relative degree," in *IEEE 58th Conference on Decision and Control (CDC)*, 2019, pp. 474–479.

[31] X. Tan, W. S. Cortez, and D. V. Dimarogonas, "High-order barrier functions: Robustness, safety, and performance-critical control," *IEEE Transactions on Automatic Control*, vol. 67, no. 6, pp. 3021–3028, 2022.

[32] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun, "Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing," in *American Control Conference*, 2007, pp. 2296–2301.

[33] J. Zeng, B. Zhang, and K. Sreenath, "Safety-critical model predictive control with discrete-time control barrier function," in *2021 American Control Conference (ACC)*, 2021, pp. 3882–3889.

[34] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and R. Daniela, "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5135–5142.

[35] P. Polack, F. Altché, B. d'Andréa Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 812–818.

[36] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.