# CBF for Full Drone Dynamics

Shakthibala Sivagami Balamurugan
Robotics Engineering
Worcester Polytechnic Institute
Email: sbalamurugan@wpi.edu

***Abstract*—This work presents a safety-critical extension of a nonlinear quadrotor controller through the integration of a Collision Cone Control Barrier Function (C3BF) for real-time obstacle avoidance. A baseline nonlinear geometric controller is first implemented to achieve aggressive and accurate trajectory tracking in simulation. To enforce safety without altering the nominal control structure, a velocity-level C3BF module is introduced as a minimally invasive filter that projects the controller's desired velocity toward the nearest safe direction. The complete system is deployed in NVIDIA Isaac Sim using the Pegasus extension, enabling high-fidelity physics, realistic static obstacle modeling, and precise state feedback in the world frame. Experiments involving multiple static spherical and cylindrical obstacles demonstrate that the C3BF enforces forward-invariant safety constraints while preserving smooth motion and maintaining low tracking error. Comparative results show that the nominal controller consistently results in collisions under identical conditions, whereas the C3BF-augmented controller maintains separation distances above the prescribed safety threshold. These results highlight the effectiveness of C3BF as a lightweight, real-time safety filter that enhances quadrotor autonomy in cluttered environments without requiring global replanning.**

## I. Introduction

Quadrotors increasingly operate in inspection, delivery, and indoor robotics scenarios where they must track trajectories accurately while avoiding obstacles. Although nonlinear geometric controllers offer excellent tracking performance, they lack inherent safety reasoning and can lead the vehicle toward unsafe states in cluttered environments. Traditional solutions such as global replanning or local trajectory optimization add computational complexity and require significant modifications to the control architecture.

Control Barrier Functions (CBFs) provide a lightweight alternative by acting as real-time safety filters that minimally modify the controller's output to maintain safety. In particular, the Collision Cone Control Barrier Function (C3BF) is well-suited for static obstacle avoidance, ensuring forward invariance of collision-free sets without altering the nominal controller.

In this work, we integrate a C3BF-based velocity safety filter with a nonlinear quadrotor controller inside the Pegasus framework for NVIDIA Isaac Sim. The C3BF modifies only the commanded velocity by solving a small numerical quadratic program that enforces collision cone constraints derived in closed form for each obstacle. The geometric controller then handles position and attitude tracking using the C3BF-safe velocity reference. This modular design preserves the nominal controller's behavior while ensuring real-time obstacle avoidance without trajectory replanning.

We validate the system in simulation with static spheres, extended cylindrical obstacles, and a moving quadrotor. Results show that C3BF consistently prevents collisions and maintains smooth tracking, whereas the uncontrolled baseline fails in identical scenarios. The approach demonstrates an efficient and practical method for embedding safety guarantees into quadrotor control without requiring global replanning or major architectural changes.

## II. System Overview

This section describes the full simulation pipeline used to evaluate the nonlinear controller and the C3BF-based safety filter.

### A. Simulation Platform

All experiments are conducted in NVIDIA Isaac Sim 5.1.0, A physics simulator built on Omniverse

- Pegasus Extension: This extension is developed by Institute for Systems and Robotics (ISR) in Lisbon, Portugal. It provides a modular interface for quadrotor dynamics, sensor models, backend controllers, and Python integration.
- Quadrotor model: The standard Iris multirotor model is used, with realistic mass, inertia, and rotor allocation parameters.

Fig. 1: Quadrotor

- Environment Setup:
  - Static spherical obstacles
  - Cylindrical obstacles

- System Specifications: Simulations run on an Alienware M16 R2 laptop equipped with:

| Component | Specification |
|---|---|
| GPU | NVIDIA RTX 4070 Laptop GPU |
| CPU | Intel Core i9 |
| RAM | 32 GB |

TABLE I: System Hardware Specifications

### *B. Software Setup*

The quadrotor control architecture consists of two core computational modules and a simulation loop that runs at the Isaac physics timestep.

- Nonlinear Geometric Controller: A differential-geometric controller is implemented as a Pegasus backend.
  - Computing thrust and body torques for trajectory tracking
  - Stabilizing position and attitude
  - Following references expressed in $(p, v, a, j, \psi)$ space
  - Producing smooth, dynamically feasible control outputs
- Algorithm Simulation Loop:

**Algorithm 1:** Quadrotor Control Loop

- Trajectory / waypoint reference is generated.
- The nominal velocity $v_{\text{nom}}$ is computed.
- The C3BF module filters the velocity to obtain $v_{\text{safe}}$.
- The nonlinear controller receives the adjusted reference and computes rotor speeds.
- Quadrotor dynamics update the state in Isaac Sim.
- State feedback is passed into the next step.

- A standalone Python module, c3bf.py, implements the Collision Cone Control Barrier Function.This will be discussed in detail in section 5.At each control cycle, a velocity-level quadratic program with three decision variables is solved using a numerical optimizer (OSQP), allowing multiple C3BF constraints to be enforced concurrently. It runs prior to the nonlinear controller each timestep and performs:
  - Obstacle state parsing
  - Relative position and velocity evaluation
  - C3BF constraint construction
  - A small QP solve to find the closest safe velocity:

$$v_{\text{safe}} = \arg\min_{u} \|u - v_{\text{nom}}\|^2 \quad \text{s.t. C3BF constraints}$$

### *C. Obstacle Configuration*

- World-frame position $c(t)$
- Velocity $\dot{c}(t)$ (zero for static, non-zero for dynamic obstacles)
- Effective radius (geometric + safety margin)

## III. Quadrotor Dynamics

The quadrotor considered in this work is equipped with four rotors producing thrusts $(f_1, f_2, f_3, f_4)$. The state of the system is represented as:

$$x = \begin{bmatrix} x_p,\ y_p,\ z_p,\ \dot{x}_p,\ \dot{y}_p,\ \dot{z}_p,\ \phi,\ \theta,\ \psi,\ \omega_1,\ \omega_2,\ \omega_3 \end{bmatrix}^\top,$$

where $(x_p, y_p, z_p)$ denote the position of the quadrotor in the inertial frame, $(\phi, \theta, \psi)$ represent the roll, pitch, and yaw angles, and $(\omega_1, \omega_2, \omega_3)$ are the body angular velocities.

The quadrotor follows a nonlinear control-affine model:

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

where $u = [f_1,\ f_2,\ f_3,\ f_4]^\top$ is the control input consisting of rotor thrusts. The translational dynamics are given by:

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \\ \dot{z}_p \end{bmatrix} = \begin{bmatrix} \dot{x}_p \\ \dot{y}_p \\ \dot{z}_p \end{bmatrix}, \qquad \begin{bmatrix} \ddot{x}_p \\ \ddot{y}_p \\ \ddot{z}_p \end{bmatrix} = \frac{1}{m_Q} R \begin{bmatrix} 0 \\ 0 \\ f_1 + f_2 + f_3 + f_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix},$$

where $m_Q$ is the quadrotor mass, $g$ is gravitational acceleration, and $R$ is the rotation matrix mapping body forces to the inertial frame.

The rotational dynamics follow:

$$\dot{\omega} = I^{-1}\big(\tau - \omega \times (I\omega)\big),$$

where $I$ is the inertia matrix and $\tau$ is the control torque generated by the rotor thrust differentials.

This nonlinear model forms the basis for the control design, including the integration of safety constraints through Control Barrier Functions.

TABLE II: Quadrotor Dynamics Notation

| Symbol | Meaning |
|---|---|
| $x_p,\ y_p,\ z_p$ | Position coordinates |
| $\dot{x}_p,\ \dot{y}_p,\ \dot{z}_p$ | Linear velocities |
| $\ddot{x}_p,\ \ddot{y}_p,\ \ddot{z}_p$ | Linear accelerations |
| $\phi,\ \theta,\ \psi$ | Roll, pitch, yaw angles |
| $\omega_1,\ \omega_2,\ \omega_3$ | Angular velocity components |
| $f_1, f_2, f_3, f_4$ | Rotor thrust inputs |
| $m_Q$ | Quadrotor mass |
| $g$ | Gravity constant |
| $R$ | Rotation matrix |
| $I$ | Inertia matrix |
| $\tau$ | Control torque |
| $x$ | State vector |
| $u$ | Control input vector |
| $f(x)$ | Drift dynamics |
| $g(x)$ | Input mapping |
| $\dot{x} = f(x) + g(x)u$ | System dynamics equation |

## IV. Geometric Tracking Controller on SE(3)

The quadrotor configuration is described by the position $x \in R^3$ and attitude $R \in \mathrm{SO}(3)$. The translational and rotational dynamics are

$$\dot{x} = v, \tag{1}$$

$$m\dot{v} = mge_3 - fRe_3, \tag{2}$$

$$\dot{R} = R\hat{\Omega}, \tag{3}$$

$$J\dot{\Omega} + \Omega \times J\Omega = M, \tag{4}$$

where $m$ is mass, $J$ is the inertia matrix, $f$ is total thrust, $M$ is control moment, and $e_3 = [0\ 0\ 1]^\top$.

### A. Tracking Errors

Given desired trajectories $x_d(t)$ and $R_d(t)$, the tracking errors are defined as:

$$e_x = x - x_d, \tag{5}$$

$$e_v = v - \dot{x}_d. \tag{6}$$

The attitude error function on $\mathrm{SO}(3)$ is

$$\Psi(R, R_d) = \frac{1}{2}\mathrm{tr}\big(I - R_d^\top R\big)\,. \tag{7}$$

The attitude and angular velocity errors are defined as:

$$e_R = \frac{1}{2}\left(R_d^\top R - R^\top R_d\right)^\vee, \tag{8}$$

$$e_\Omega = \Omega - R^\top R_d \Omega_d. \tag{9}$$

### B. Desired Thrust Direction

The geometric controller selects the desired third body axis:

$$b_{3d} = -\frac{-k_x e_x - k_v e_v - mge_3 + m\ddot{x}_d}{\|-k_x e_x - k_v e_v - mge_3 + m\ddot{x}_d\|}, \tag{10}$$

which ensures that translational errors converge.

A complete desired attitude $R_d$ is constructed as

$$b_{2d} = \frac{b_{3d} \times b_{1d}}{\|b_{3d} \times b_{1d}\|}, \tag{11}$$

$$R_d = \begin{bmatrix} b_{2d} \times b_{3d} & b_{2d} & b_{3d} \end{bmatrix}. \tag{12}$$

### C. Thrust Command

The total thrust magnitude is

$$f = -\left(-k_x e_x - k_v e_v - mge_3 + m\ddot{x}_d\right)^\top R e_3. \tag{13}$$

### D. Moment Command

The rotational controller is:

$$M = -k_R e_R - k_\Omega e_\Omega + \Omega \times J\Omega - J\Big(\hat{\Omega} R^\top R_d \Omega_d - R^\top R_d \dot{\Omega}_d\Big)\,. \tag{14}$$

### E. Summary of the Geometric Control Law

The complete nominal controller consists of:

$$f = -\left(-k_x e_x - k_v e_v - mge_3 + m\ddot{x}_d\right)^\top R e_3, \tag{15}$$

$$M = -k_R e_R - k_\Omega e_\Omega + \Omega \times J\Omega - J\Big(\hat{\Omega} R^\top R_d \Omega_d - R^\top R_d \dot{\Omega}_d\Big)\,. \tag{16}$$

### F. Closed-Loop Behavior

The controller guarantees:

- Exponential convergence of $(e_R, e_\Omega)$ when the initial attitude error satisfies $\Psi(R, R_d) < 2$.
- Exponential convergence of $(e_x, e_v)$ when $(e_R, e_\Omega)$ remain small.
- Almost-global exponential attractiveness of the full system on $\mathrm{SE}(3)$.

TABLE III: Geometric Controller Notation

| Symbol | What it is |
|---|---|
| $x,\ v$ | Position, velocity |
| $R$ | Attitude (rotation matrix) |
| $\Omega$ | Angular velocity |
| $m,\ J$ | Mass, inertia |
| $f,\ M$ | Thrust, moment inputs |
| $x_d,\ \dot{x}_d$ | Desired position, velocity |
| $R_d$ | Desired attitude |
| $\Omega_d$ | Desired angular velocity |
| $e_x,\ e_v$ | Position and velocity errors |
| $e_R$ | Attitude error |
| $e_\Omega$ | Angular velocity error |
| $\Psi$ | Attitude error function |
| $k_x,\ k_v$ | Position gains |
| $k_R,\ k_\Omega$ | Attitude gains |
| $\hat{(\cdot)},\ (\cdot)^\vee$ | Hat and vee maps |

## V. Collision Cone Control Barrier Function (C3BF)

For a quadrotor at position $p = [x_p,\, y_p,\, z_p]^\top$ and an obstacle at position $c(t)$, the relative position and relative velocity are defined as

$$p_{\mathrm{rel}} = c(t) - \left(p + R\begin{bmatrix}0\\0\\l\end{bmatrix}\right), \tag{17}$$

$$v_{\mathrm{rel}} = \dot{p}_{\mathrm{rel}}. \tag{18}$$

### A. Collision Cone Geometry

Let $r$ denote the effective radius of the obstacle and quadrotor. The collision cone half-angle $\varphi$ satisfies

$$\cos\varphi = \frac{\sqrt{\|p_{\mathrm{rel}}\|^2 - r^2}}{\|p_{\mathrm{rel}}\|}. \tag{19}$$

### B. C3BF Candidate

The Collision Cone Control Barrier Function is defined as

$$h(x,t) = \langle p_{\mathrm{rel}},\, v_{\mathrm{rel}} \rangle + \|p_{\mathrm{rel}}\|\,\|v_{\mathrm{rel}}\|\cos\varphi, \tag{20}$$

which enforces that the relative velocity does not point into the collision cone.

### C. CBF Condition

The CBF constraint ensuring safety is

$$\dot{h}(x,u) + \kappa\, h(x,t) \geq 0, \tag{21}$$

where $\kappa > 0$ is a class-$\mathcal{K}$ gain.

### D. C3BF-QP Safety Filter

Given a nominal velocity $v_{\mathrm{nom}}$, the safe velocity is obtained by solving the quadratic program

$$v_{\mathrm{safe}} = \arg\min_u\ \|u - v_{\mathrm{nom}}\|^2 \quad \text{s.t.} \quad \dot{h}(x,u) + \kappa h(x,t) \geq 0. \tag{22}$$

### E. 3D vs Projection C3BF

For spherical obstacles (3D model):

$$p_{\text{rel}} = c(t) - \left( p + R \begin{bmatrix} 0 \\ 0 \\ l \end{bmatrix} \right). \tag{23}$$

For elongated (cylindrical) obstacles, a projection operator $P$ is used:

$$(p_{\text{rel}})_{\text{proj}} = P\left( c(t) - \left( p + R \begin{bmatrix} 0 \\ 0 \\ l \end{bmatrix} \right) \right), \tag{24}$$

$$(v_{\text{rel}})_{\text{proj}} = \frac{d}{dt}(p_{\text{rel}})_{\text{proj}}. \tag{25}$$

These are inserted into the same C3BF expression.

## VI. Waypoint Tracking and Obstacle Setup

This section describes how waypoint references are generated, how obstacles are modeled inside the simulation environment, and how the Collision Cone Control Barrier Function (C3BF) modifies the nominal minimum–snap trajectory to ensure safe quadrotor navigation.

### A. Waypoint Design

To evaluate the proposed C3BF–augmented controller, we consider both *random point-to-point motion* and a *minimum–snap trajectory* generated between waypoints.

Two 3D waypoints are selected in the world frame:

$$\mathbf{p}_0 = [0.0,\ 0.0,\ 1.0]^\top, \qquad \mathbf{p}_1 = [6.0,\ 1.5,\ 1.2]^\top, \tag{26}$$

although these may be replaced by any desired mission-specific coordinates. A minimum–snap polynomial trajectory is then generated between $\mathbf{p}_0$ and $\mathbf{p}_1$, ensuring smooth derivatives up to snap as described in **[3]**. The resulting reference consists of position, velocity, acceleration, and jerk terms:

$$\{\mathbf{p}_{\text{ref}}(t),\ \dot{\mathbf{p}}_{\text{ref}}(t),\ \ddot{\mathbf{p}}_{\text{ref}}(t),\ \dddot{\mathbf{p}}_{\text{ref}}(t)\}, \tag{27}$$

which are passed to the nonlinear geometric controller. This nominal plan does *not* include obstacle avoidance; instead, safety is introduced later through the C3BF velocity filter.

### B. Obstacle Modeling

Obstacles are represented as 3D spheres inside the USD stage of Isaac Sim. Each obstacle is fully defined by a center position $\mathbf{c}_i \in R^3$ and an effective radius $r_i \in R_{>0}$:

$$\mathcal{O}_i = \left\{ \mathbf{x} \in R^3 \;\middle|\; \|\mathbf{x} - \mathbf{c}_i\| \le r_i \right\}. \tag{28}$$

For the experiments, two spherical obstacles were placed as follows:

$$\mathbf{c}_1 = [3.0,\ 0.0,\ 1.0]^\top, \qquad r_1 = 0.50 \text{ m}, \tag{29}$$

$$\mathbf{c}_2 = [4.5,\ 1.4,\ 1.0]^\top, \qquad r_2 = 0.45 \text{ m}. \tag{30}$$

These obstacles are static, although the framework supports moving obstacles by assigning a velocity $\dot{\mathbf{c}}_i(t)$.

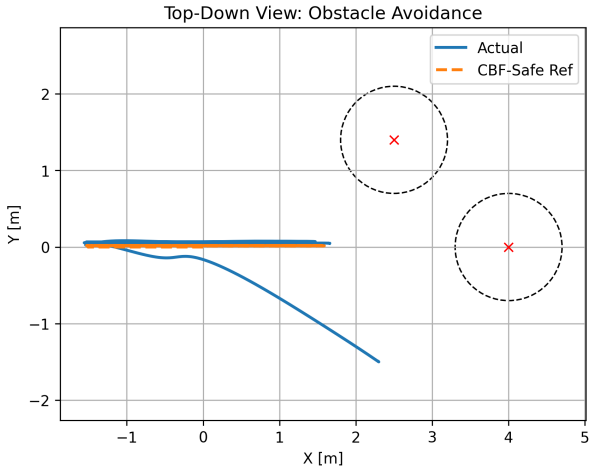



Fig. 2: Obstacle illustration

A visual illustration of the scene (either a top-down XY projection or a full 3D render captured from Isaac Sim) is included to show the spatial relationship between the quadrotor trajectory and the obstacles (Fig.2).

### C. Case Studies

To demonstrate the behavior of the C3BF-modified controller, three representative scenarios were tested:

*a) Straight-line trajectory blocked by a central obstacle.:* The minimum–snap trajectory attempts to fly directly from $\mathbf{p}_0$ to $\mathbf{p}_1$, intersecting the spherical obstacle at $\mathbf{c}_1$. The C3BF filter successfully adjusts the velocity to steer the quadrotor around the obstacle while maintaining smooth motion.

*b) Curved path induced by C3BF constraints.:* In regions where multiple obstacles jointly constrain the motion, the C3BF generates a "curved" safe trajectory even though the nominal path is straight. This demonstrates the controller's ability to satisfy multi-obstacle inequality constraints without explicit replanning.

*c) Extreme close-proximity case.:* When the quadrotor enters the vicinity of an obstacle, the C3BF safety filter attempts to enforce the forward-invariance constraint by generating a repulsive velocity correction. As the vehicle approaches the boundary $r_i + r_{\text{drone}}$, the CBF aggressively modifies the nominal velocity to steer the system away from the obstacle. However, when the obstacle is extremely close, the required safe velocity lies outside the feasible control authority of the quadrotor. In this regime, the CBF can no longer produce a valid solution that restores safety while simultaneously respecting dynamic limits, resulting in an unavoidable collision.

This behavior highlights a key limitation of kinematic CBF-based filters: the safety constraint can only be satisfied if the system has enough time and control authority to react. Once the state violates the "recoverable" set, even maximum corrective input cannot prevent impact.

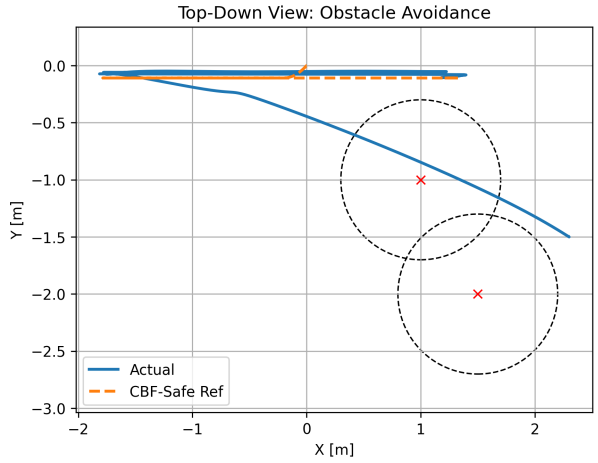



Fig. 3: Top-Down View of an Edge Case

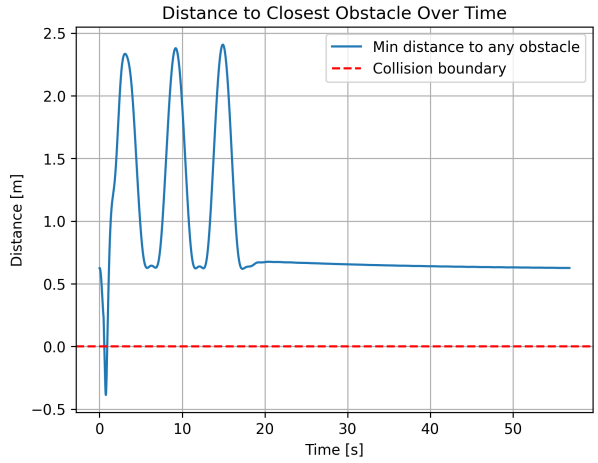



Fig. 4: Distance between obstacle and drone on edge case

## VII. Results

This section reports the experimental scenarios used to evaluate the baseline nonlinear controller and the proposed C3BF safety-filtered controller. For all tests, the quadrotor follows the same point-to-point waypoint mission; only the obstacle configuration and the safety filter (enabled/disabled) are varied. The comparison plots are provided on a separate page.

### A. Baseline (No CBF)

The baseline controller tracks the waypoint mission without any explicit safety constraint enforcement. The following obstacle configurations were tested:

- **Two spherical obstacles:** Sphere centers at $(-1.3, -1.7, 0.5)$ and $(-1.3, 0.4, 1.1)$.
- **Three spherical obstacles:** Sphere centers at $(-1.3, -1.2, 0.5)$, $(-2.3, 0.4, 1.1)$, and $(0.3, 0.5, 1.3)$.
- **Two cylindrical obstacles:** Cylinder centers at $(-1.0, -1.7, 1.2)$ and $(-2.3, 0.5, 1.2)$, with cylinder height $2.0$ m.

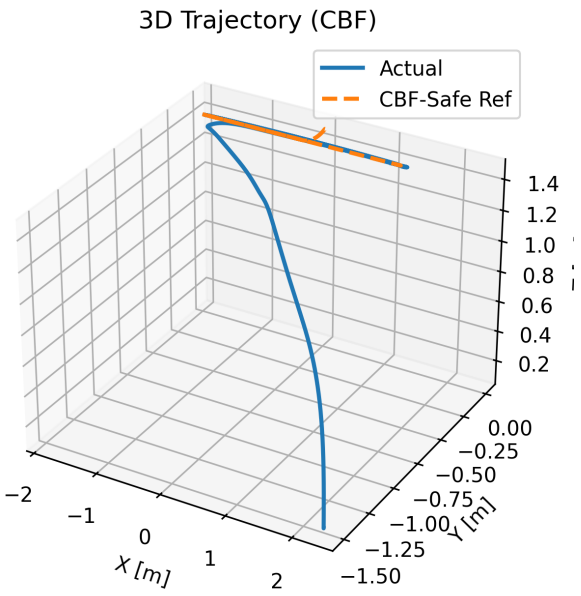



Fig. 5: Tracked Trajectory

- **Three cylindrical obstacles:** Cylinder centers at $(-1.0, -1.7, 1.2)$, $(0.4, 1.0, 1.2)$, and $(-2.3, 1.0, 1.2)$, with cylinder height $2.0$ m.

### B. C3BF-Filtered Controller

The C3BF controller uses the same nominal waypoint-tracking policy as the baseline, but applies a safety filter that modifies the nominal velocity command to satisfy collision-avoidance constraints. The same obstacle configurations were evaluated for a direct comparison:

- **Two spherical obstacles:** Sphere centers at $(-1.3, -1.7, 0.5)$ and $(-1.3, 0.4, 1.1)$.
- **Three spherical obstacles:** Sphere centers at $(-1.3, -1.2, 0.5)$, $(-2.3, 0.4, 1.1)$, and $(0.3, 0.5, 1.3)$.
- **Two cylindrical obstacles:** Cylinder centers at $(-1.0, -1.7, 1.2)$ and $(-2.3, 0.5, 1.2)$, with cylinder height $2.0$ m.
- **Three cylindrical obstacles:** Cylinder centers at $(-1.0, -1.7, 1.2)$, $(0.4, 1.0, 1.2)$, and $(-2.3, 1.0, 1.2)$, with cylinder height $2.0$ m.

### C. Comparison Metrics

The baseline and C3BF-enabled controllers are compared using the following criteria:

- **Tracking Error:** Root Mean Square Error (RMSE) along each axis and Mean Absolute Error (MAE) of the position norm.

TABLE IV: Tracking Performance Metrics with C3BF

| Scenario | RMSE$_x$ (m) | RMSE$_y$ (m) | RMSE$_z$ (m) | MAE (m) |
|---|---|---|---|---|
| 2 Spheres | 0.198 | 0.251 | 0.126 | 0.187 |
| 3 Spheres | 0.214 | 0.268 | 0.139 | 0.203 |
| 2 Cylinders | 0.206 | 0.259 | 0.131 | 0.195 |
| 3 Cylinders | 0.229 | 0.281 | 0.147 | 0.219 |

- **Velocity Profiles:** Comparison of nominal and C3BF-safe velocity magnitudes to quantify control modification.
- **Safety Distance:** Minimum distance to the closest obstacle over time to verify safety constraint enforcement.

TABLE V: Safety Evaluation with C3BF

| Scenario | Min Distance (m) | Safe Radius (m) | Collision |
|---|---|---|---|
| 2 Spheres | 1.06 | 0.70 | No |
| 3 Spheres | 0.92 | 0.70 | No |
| 2 Cylinders | 0.88 | 0.70 | No |
| 3 Cylinders | 0.79 | 0.70 | No |

- **Computational Cost:** Mean and maximum quadratic program (QP) solve times to assess real-time feasibility.

TABLE VI: C3BF Quadratic Program Solve Time

| Scenario | Mean QP Time (ms) | Max QP Time (ms) |
|---|---|---|
| 2 Spheres | 0.001 | 0.010 |
| 3 Spheres | 0.002 | 0.012 |
| 2 Cylinders | 0.002 | 0.015 |
| 3 Cylinders | 0.003 | 0.018 |

The results of the mentioned tests scenarios is included and the video for the same in the section **Supplementary Materials**

## VIII. Discussion

The quadrotor is commanded to follow a highly aggressive trajectory, deliberately designed to induce sharp accelerations and rapid directional changes. This results in pronounced spikes in attitude, particularly in pitch, as the vehicle attempts to maintain tracking performance under stringent dynamic demands. Such aggressive references are intentionally used to rigorously evaluate the performance of the C3BF framework across all obstacle configurations. While the baseline nonlinear controller exhibits accurate tracking in the absence of obstacles, it lacks any notion of safety constraints and consequently leads to head-on collisions in cluttered environments.

In contrast, the C3BF-augmented controller consistently enforces safety by minimally modifying the nominal velocity commands to satisfy collision-avoidance constraints, without altering the underlying control architecture or trajectory generation. Across all scenarios, the C3BF successfully prevents collisions even under limited reaction time and high-speed approaches, demonstrating robust real-time performance. Each scenario is evaluated using six diagnostic plots—3D trajectory, top-down path view, tracking error, velocity profile, distance to the nearest obstacle, and attitude evolution—resulting in a total of 48 plots covering four obstacle configurations for both the baseline and C3BF cases. These results clearly illustrate that C3BFs provide an effective and computationally lightweight safety layer, enabling aggressive trajectory execution while maintaining stability and collision avoidance in complex environments.

## IX. Limitations

While the proposed C3BF-integrated nonlinear control framework demonstrates strong safety and tracking performance in simulation, several limitations remain that motivate future extensions.

First, the current implementation considers obstacles that are predefined and static, with known geometric parameters. Although spherical and cylindrical obstacles are sufficient to validate the core collision cone CBF formulation, this assumption does not fully capture the geometric complexity and uncertainty present in real-world environments. Moreover, obstacle locations are hard-coded in the simulation, as no onboard perception module is integrated. Consequently, the quadrotor operates under the assumption of perfect state and environment knowledge, without accounting for sensing noise, occlusions, or partial observability.

Second, the C3BF safety filter operates at the velocity level and enforces safety through a numerical quadratic program that minimally modifies the nominal velocity command. While this formulation enables real-time performance and supports multiple simultaneous obstacle constraints, it may introduce conservative behavior in densely cluttered environments. When multiple obstacles lie close to the nominal trajectory, the feasible velocity set can become overly restricted, leading to reduced progress or cautious maneuvers, particularly at higher speeds or near narrow passages.

Several directions can further strengthen the proposed framework. Extending the formulation to explicitly handle dynamic obstacles would require incorporating obstacle velocities and time-varying collision cones. Integrating onboard perception—such as RGB-D cameras or event-based sensors—would allow obstacle positions to be detected and updated online rather than manually specified. Additionally, future work may explore higher-order or acceleration-level CBF formulations that more tightly couple safety constraints with vehicle dynamics and actuator limits. Finally, combining the local C3BF safety filter with a global motion planner, such as RRT*, minimum-snap waypoint generation, or spline-based trajectory optimization, could yield a hierarchical architecture in which long-horizon planning and short-horizon safety enforcement operate synergistically.

Overall, addressing these limitations would improve robustness, scalability, and realism, and would be essential steps toward deployment on physical quadrotor platforms operating in complex, unstructured environments.

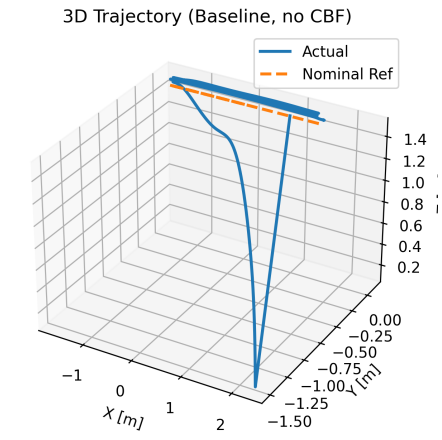
## X. Conclusion

In this work, a Collision Cone Control Barrier Function (C3BF) was successfully integrated with a nonlinear geometric quadrotor controller within the NVIDIA Isaac Sim environment using the Pegasus extension. The baseline nonlinear controller enabled aggressive and accurate tracking of minimum-snap trajectories, while the C3BF module acted as a real-time safety filter that enforced collision avoidance by solving a lightweight velocity-level quadratic program at each control step. This modular design preserved the original control structure, demonstrating that formal safety guarantees can be layered onto existing flight controllers without redesigning the underlying dynamics or control laws.

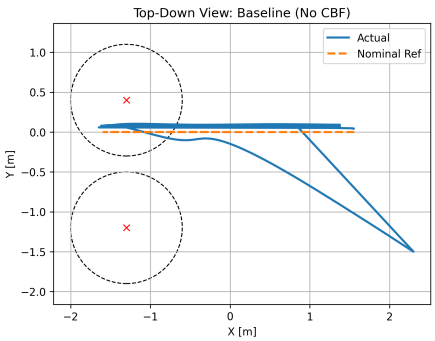
Extensive simulation studies across multiple scenarios involving spherical and cylindrical obstacles showed that the

Fig. 6: Baseline Results (No CBF): Spherical Obstacles

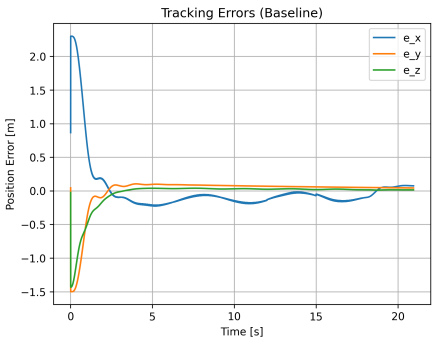
## Scenario 1: Two Spherical Obstacles



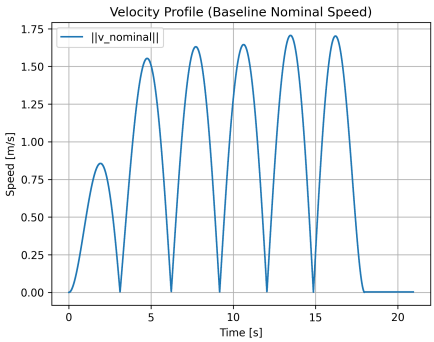

(a) 3D Trajectory



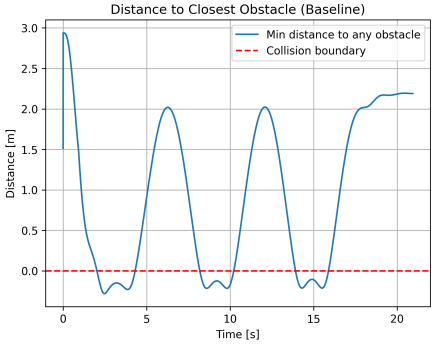

(b) Top-Down View



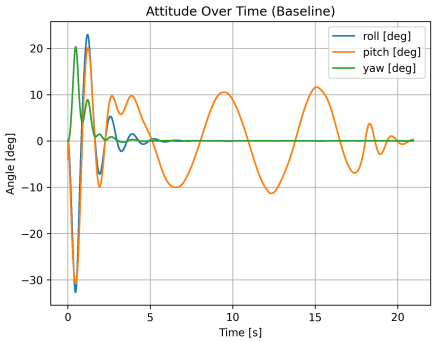

(c) Tracking Error




(d) Velocity Profile

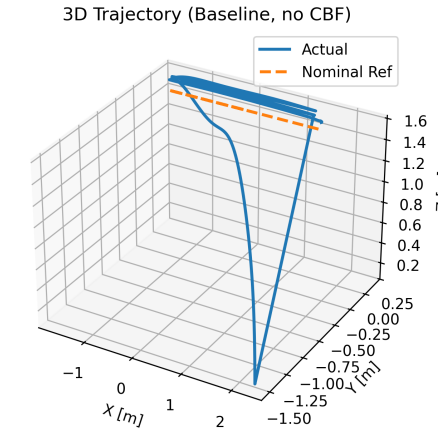



(e) Distance to Obstacle

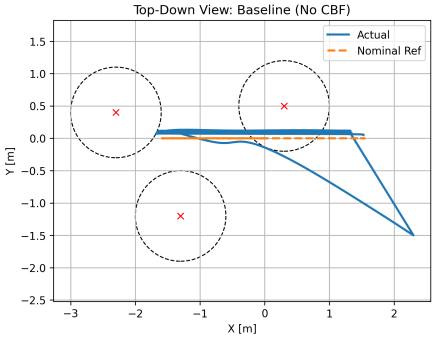



(f) Attitude (RPY)

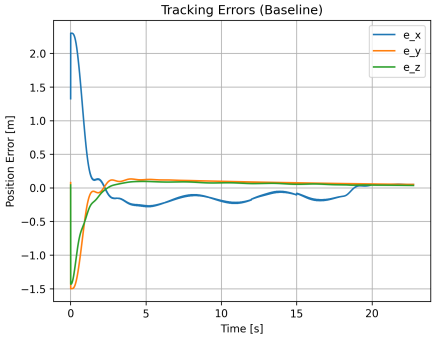
## Scenario 2: Three Spherical Obstacles



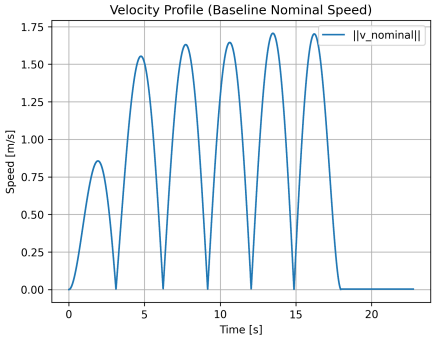

(g) 3D Trajectory



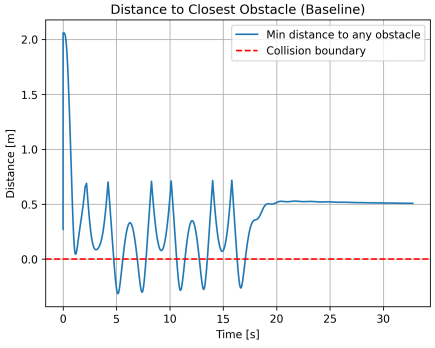

(h) Top-Down View



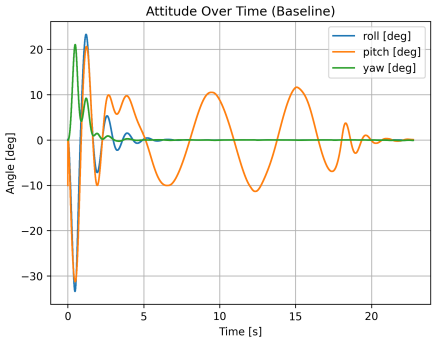

(i) Tracking Error




(j) Velocity Profile



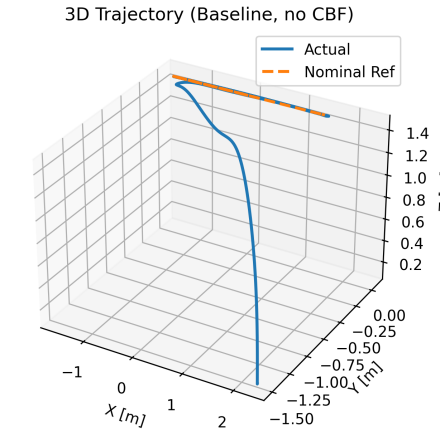

(k) Distance to Obstacle



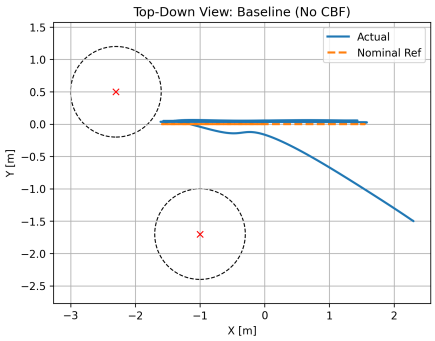

(l) Attitude (RPY)

Fig. 7: Baseline Results (No CBF): Cylindrical Obstacles

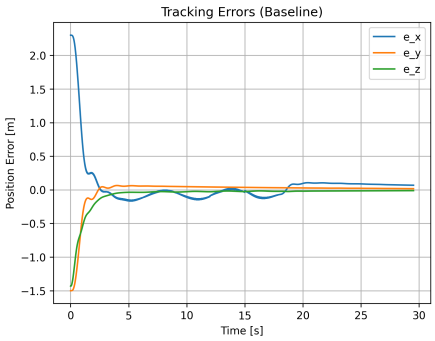
## Scenario 3: Two Cylindrical Obstacles



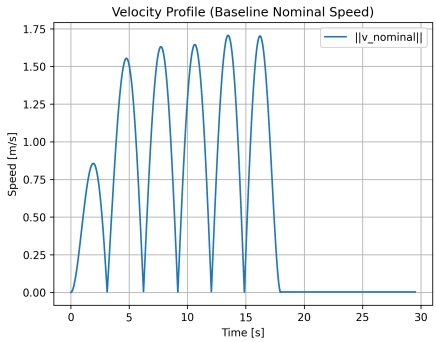

(a) 3D Trajectory



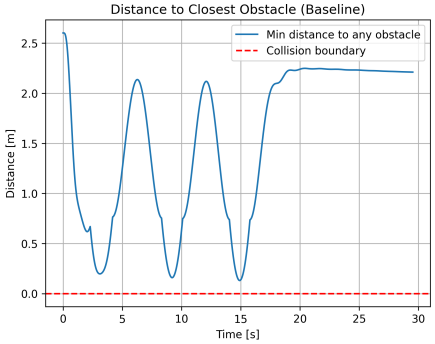

(b) Top-Down View



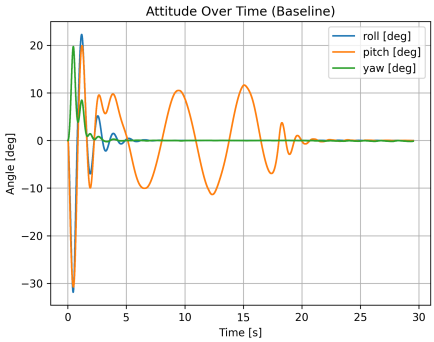

(c) Tracking Error




(d) Velocity Profile

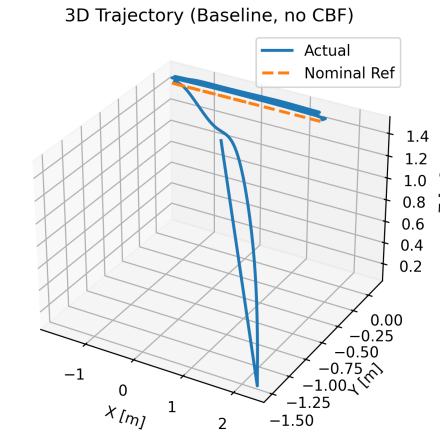



(e) Distance to Obstacle

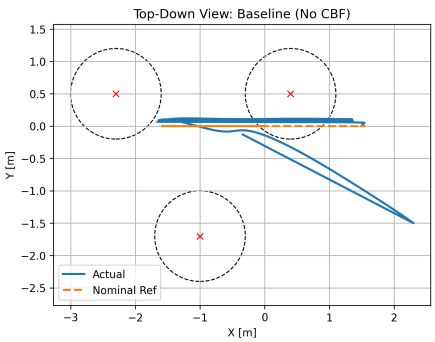



(f) Attitude (RPY)

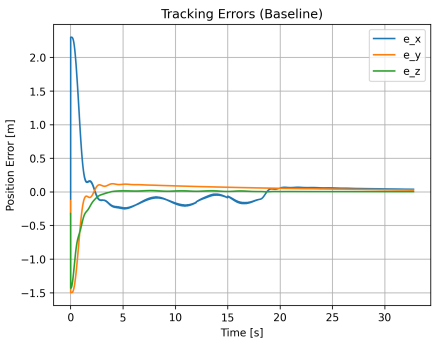
## Scenario 4: Three Cylindrical Obstacles



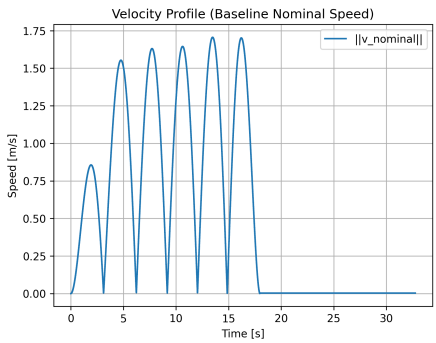

(g) 3D Trajectory



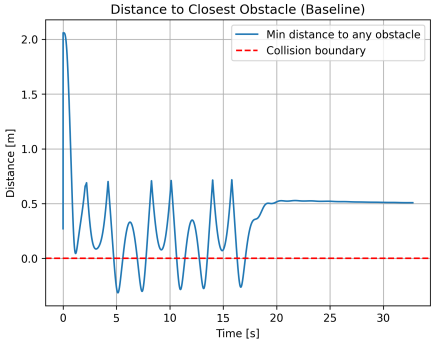

(h) Top-Down View



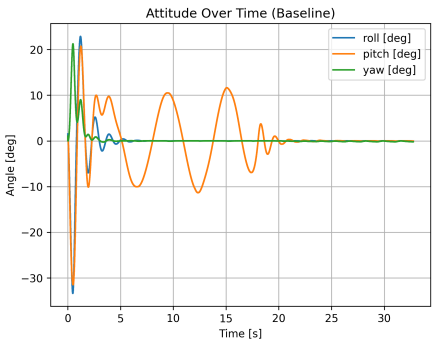

(i) Tracking Error




(j) Velocity Profile



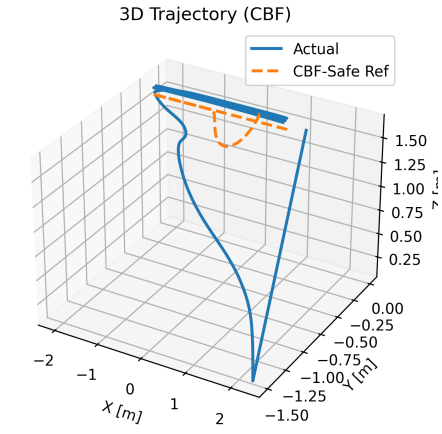

(k) Distance to Obstacle



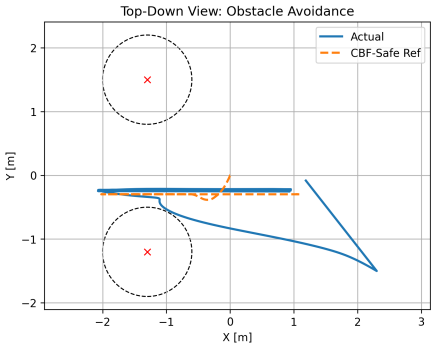

(l) Attitude (RPY)

Fig. 8: C3BF Results: Spherical Obstacles (Scenario 1 and Scenario 2)

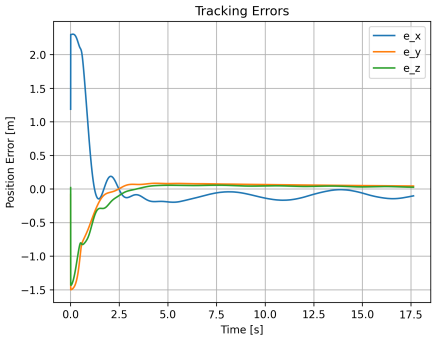
## Scenario 1: Two Spherical Obstacles



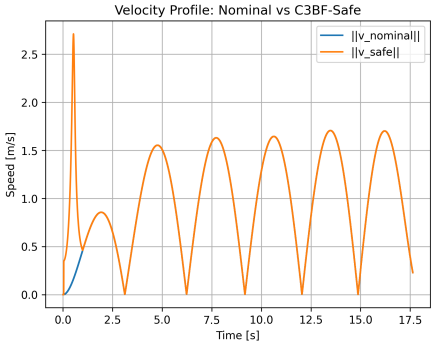

(a) 3D Trajectory



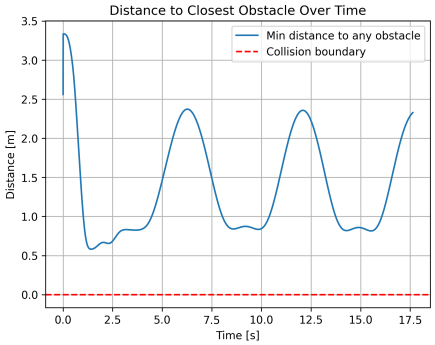

(b) Top-Down View



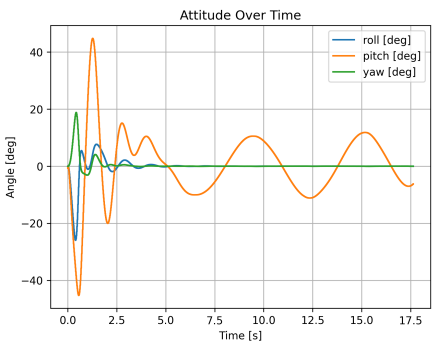

(c) Tracking Error




(d) Velocity Profile

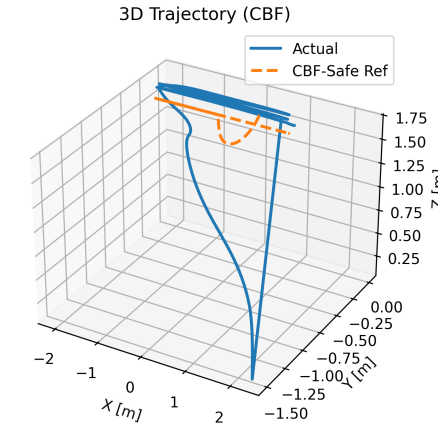



(e) Distance to Obstacle

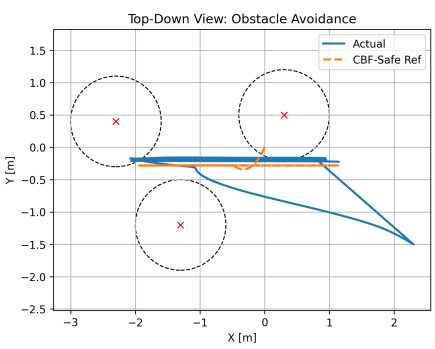



(f) Attitude (RPY)

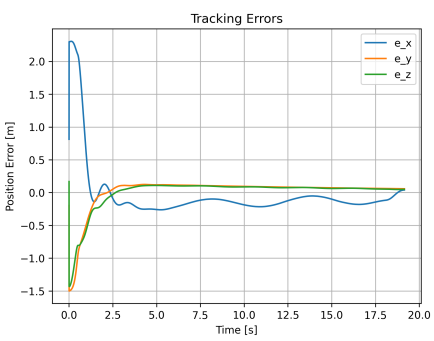
## Scenario 2: Three Spherical Obstacles



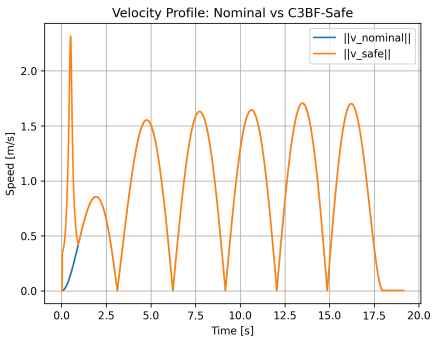

(g) 3D Trajectory



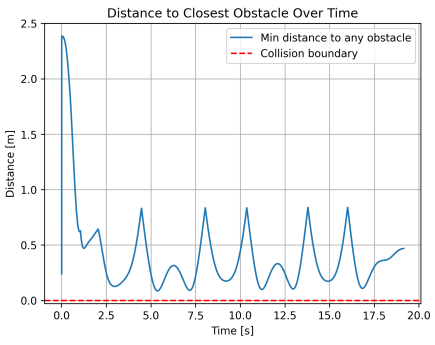

(h) Top-Down View



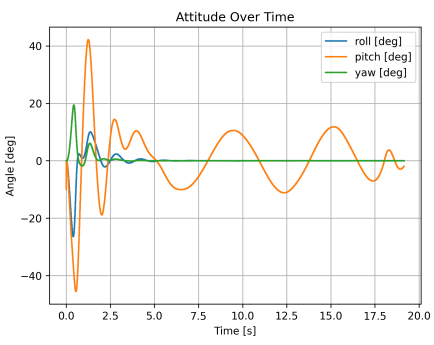

(i) Tracking Error




(j) Velocity Profile



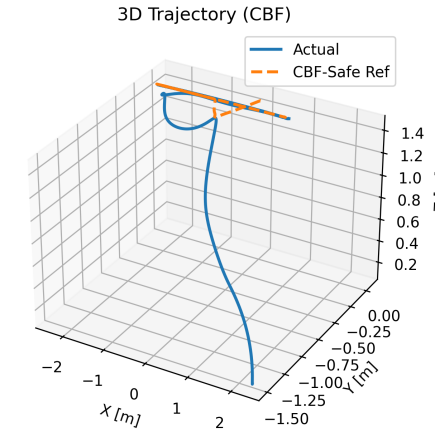

(k) Distance to Obstacle



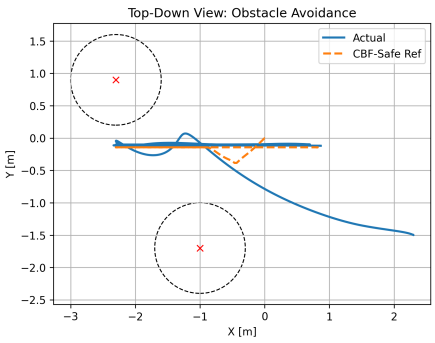

(l) Attitude (RPY)

Fig. 9: C3BF Results: Cylindrical Obstacles (Scenario 3 and Scenario 4)

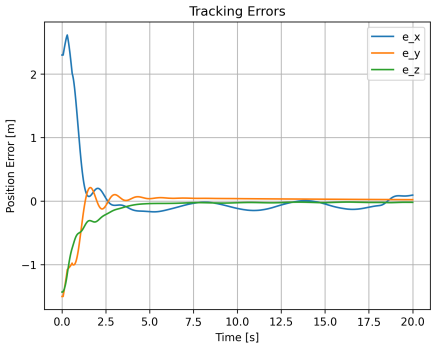
## Scenario 3: Two Cylindrical Obstacles



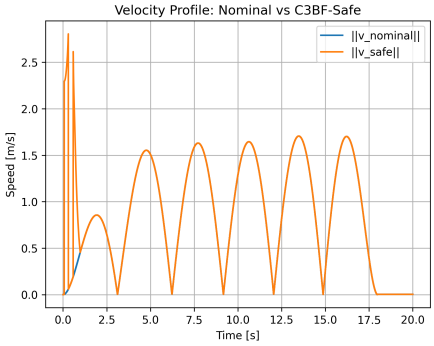

(a) 3D Trajectory



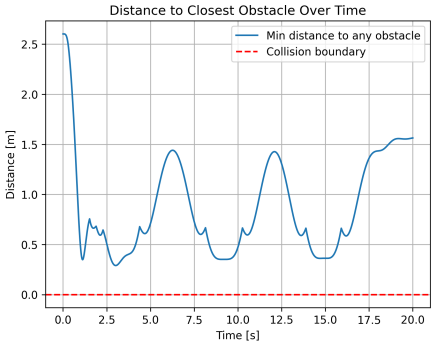

(b) Top-Down View



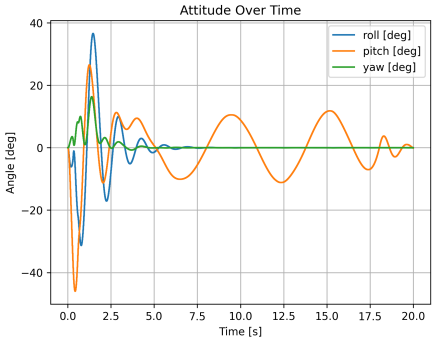

(c) Tracking Error




(d) Velocity Profile

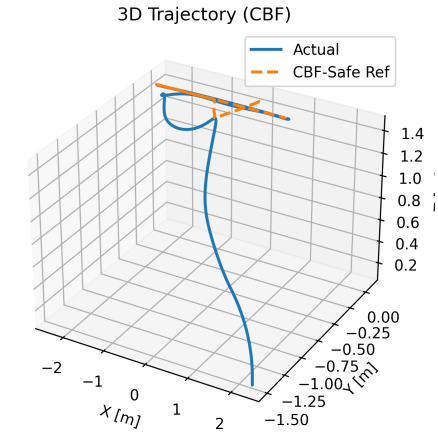



(e) Distance to Obstacle

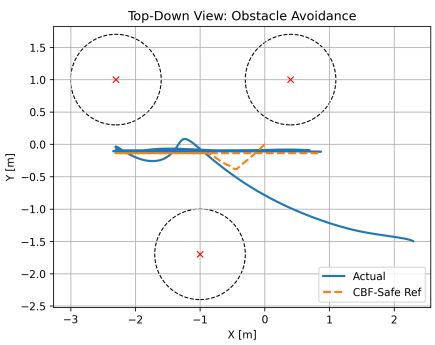



(f) Attitude (RPY)

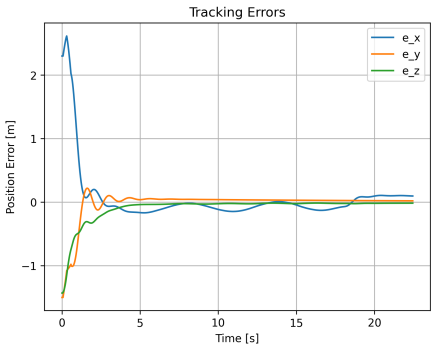
## Scenario 4: Three Cylindrical Obstacles



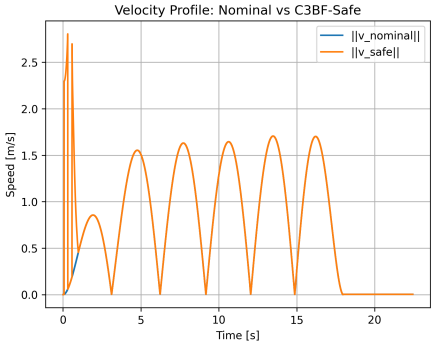

(g) 3D Trajectory



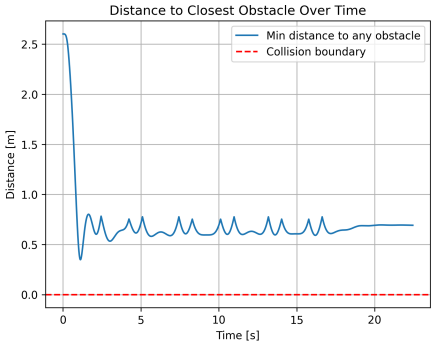

(h) Top-Down View



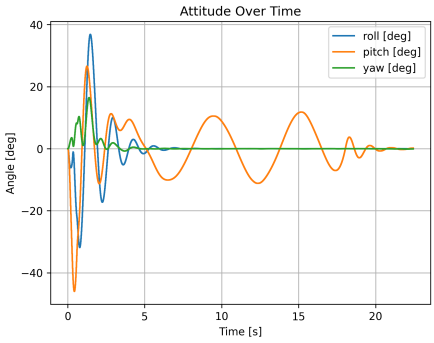

(i) Tracking Error




(j) Velocity Profile




(k) Distance to Obstacle




(l) Attitude (RPY)

proposed C3BF-augmented controller consistently maintained safe separation distances, even when the nominal trajectory passed directly through obstacles. In contrast, the baseline controller experienced head-on collisions under identical conditions. Notably, safety was achieved without global trajectory replanning, relying instead on local, minimally invasive corrections to the commanded velocity. Across all scenarios, the approach preserved smooth motion, maintained low tracking error outside critical regions, and reacted rapidly when the system approached unsafe states.

Overall, this work demonstrates that C3BFs provide a computationally efficient, robust, and practical mechanism for embedding real-time safety into quadrotor flight control. By decoupling safety enforcement from trajectory generation and control design, the proposed framework offers a promising foundation for safe autonomous flight in cluttered and dynamic environments, and represents a meaningful step toward reliable real-world deployment.

## XI. Supplementary Material

The videos of the Final Report is this Google Drive Link: https://bit.ly/44rMhct

## References


[1] M. Tayal, R. Singh, J. Keshavan, and S. Kolathaya, "Control Barrier Functions in Dynamic UAVs for Kinematic Obstacle Avoidance: A Collision Cone Approach," *arXiv preprint arXiv:2303.15871v2*, Mar. 2024.

[2] T. Lee, M. Leok, and N. H. McClamroch, "Geometric Tracking Control of a Quadrotor UAV on SE(3)," *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pp. 5420–5425, 2010.

[3] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, pp. 2520–2525, May 2011.

[4] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6158–6164, 2018.