Efficient Safe Robot Navigation Using Control Barrier Functions

Nicolas Drager
EECE
Northeastern University)
Boston, USA
drager.n@northeastern.edu

Satvik Tajane
EECE
Northeastern University)
Boston, USA
tajane.s@northeastern.edu

Harsh Akabari
EECE
Northeastern University)
Boston, USA
akabari.h@northeastern.edu

***Abstract*—Safe navigation in dynamic environments is a critical challenge in modern robotics, particularly in applications such as multi-robot warehouses where collision avoidance must be guaranteed without compromising operational efficiency. This report presents an implementation of Control Barrier Functions (CBFs) as a safety filter for differential drive robots navigating in environments with static and dynamic obstacles. Our approach combines mathematical safety guarantees with a turn-first avoidance strategy that prioritizes maintaining forward momentum over stopping, enabling faster navigation through constrained spaces. We implement a priority-based coordination system for multi-robot scenarios and validate our approach through simulation studies. The results demonstrate that CBFs provide robust safety guarantees while maintaining high operational efficiency, with the system successfully avoiding collisions in various scenarios including single-robot obstacle avoidance, multi-obstacle navigation, and robot-robot interaction. The quadratic programming formulation enables real-time computation, making the approach practical for industrial applications.**

1 Introduction

1.1 Motivation

The deployment of autonomous mobile robots in dynamic, multi-agent environments has become increasingly common across various domains, from warehouse automation to collaborative manufacturing. In these settings, safety is paramount as robots must reliably avoid collisions with obstacles, other robots, and humans while efficiently completing their assigned tasks. Traditional approaches to collision avoidance often rely on conservative behaviors that sacrifice performance for safety, leading to inefficient operations characterized by frequent stopping and cautious maneuvering.

The challenge becomes particularly acute in high-density environments such as automated warehouses, where multiple robots operate simultaneously in shared spaces. These scenarios demand real-time collision avoidance that can handle:

- **Dynamic obstacles**: Other robots and moving objects whose trajectories must be continuously monitored and avoided
- **Tight spaces**: Narrow passages where conservative stopping behaviors would create bottlenecks
- **Real-time constraints**: Control decisions must be computed within milliseconds to enable responsive behavior
- **Formal safety guarantees**: The system must provide mathematical assurances that collisions will not occur

Control Barrier Functions (CBFs) have emerged as a powerful framework for addressing these challenges. CBFs provide a mathematically rigorous approach to encoding safety constraints that can be integrated into control systems through optimization-based methods. Unlike heuristic collision avoidance approaches, CBFs offer formal guarantees of forward invariance - if the system starts in a safe state, it will remain safe for all time.

1.2 Contribution

This report presents a practical implementation of CBF-based safe navigation for differential drive robots with the following contributions:

1) A turn-first collision avoidance strategy that maintains forward momentum by prioritizing steering over deceleration, enabling faster navigation through constrained environments
2) Integration of LIDAR-based perception with CBF constraints, allowing real-time detection and avoidance of obstacles using surface point measurements
3) A priority-based coordination scheme for multi-robot scenarios that resolves potential deadlocks by dynamically assigning navigation priority based on proximity to static obstacles
4) Comprehensive simulation studies demonstrating the effectiveness of the approach in various sce-

narios, from single-robot navigation to multi-robot coordination

The remainder of this report is organized as follows: Section 2 formalizes the safe navigation problem, Section 3 presents our CBF-based solution including the control algorithm and multi-robot coordination, Section 4 presents experimental results, and Section 5 concludes with discussion and future work.

2 Problem Statement

2.1 System Model

We consider a differential drive robot modeled using unicycle kinematics on a two-dimensional plane. The robot's configuration is described by the state vector $\mathbf{x} = [x, y, \theta]^T$, where (x, y) represents the position in the plane and θ is the heading angle. The system dynamics are:

$$\begin{aligned} \dot{x} &= v \cos\theta \\ \dot{y} &= v \sin\theta \\ \dot{\theta} &= \omega \end{aligned} \tag{1}$$

where the control inputs are:

- $v \in [0, v_{\max}]$: linear velocity (m/s)
- $\omega \in [-\omega_{\max}, \omega_{\max}]$: angular velocity (rad/s)

The robot is subject to physical constraints on both velocities and accelerations:

$$\begin{aligned} v &\in [0, 0.5] \text{ m/s} \\ \omega &\in [-1.5, 1.5] \text{ rad/s} \\ |\dot{v}| &\leq 0.5 \text{ m/s}^2 \\ |\dot{\omega}| &\leq 2.0 \text{ rad/s}^2 \end{aligned} \tag{2}$$

The robot has a circular footprint with radius $R = 0.2$ m.

2.2 Perception

The robot is to be equipped with one or multiple LIDAR sensors that provide real-time distance measurements to obstacle surfaces.

2.3 Safety Requirements

The fundamental safety requirement is collision avoidance: the robot must maintain a safe distance from all obstacles at all times. Formally, for each detected obstacle surface point $\mathbf{p}_{\text{obs}}$, we require:

$$\|\mathbf{p}_{\text{robot}} - \mathbf{p}_{\text{obs}}\| > R_{\text{robot}} + \text{buffer} \tag{3}$$

where the buffer distance is a freely definable parameter, usually fixed to about 0.05 m. However, in other cases, it could also be switched depending on what kind of obstacle is close.

2.4 Performance Objectives

While maintaining safety, the robot should also:

1) Reach goal positions efficiently with minimal deviation from direct paths
2) Maintain forward momentum when possible rather than stopping
3) Navigate smoothly without abrupt control changes
4) Coordinate effectively with other robots to avoid deadlocks

2.5 Challenge: Balancing Safety and Efficiency

The core challenge is to design a control system that provides formal safety guarantees while maximizing navigation efficiency. Overly conservative approaches that stop the robot far from obstacles are safe but inefficient, while aggressive approaches that maintain high speeds risk collision. Our approach must find the optimal balance, enabling the robot to navigate as quickly as possible while absolutely ensuring collision-free operation.

3 Solution Approach

3.1 Control Barrier Functions: Theoretical Foundation

A Control Barrier Function (CBF) provides a mathematical framework for encoding safety constraints. Given a continuously differentiable function $h(\mathbf{x}) : \mathcal{D} \to \mathbb{R}$, we define the safe set as:

$$\mathcal{C} = \{\mathbf{x} \in \mathcal{D} \mid h(\mathbf{x}) \geq 0\} \tag{4}$$

The function h is a CBF if there exists an extended class $\mathcal{K}$ function γ such that for all $\mathbf{x} \in \mathcal{C}$:

$$\sup_{\mathbf{u} \in U} [\dot{h}(\mathbf{x}, \mathbf{u})] = \frac{\partial h}{\partial \mathbf{x}} f(\mathbf{x}) + \frac{\partial h}{\partial \mathbf{x}} g(\mathbf{x}) \mathbf{u} \geq -\gamma(h(\mathbf{x})) \tag{5}$$

This condition ensures forward invariance: if the system starts in the safe set ($h(\mathbf{x}_0) \geq 0$), it will remain safe for all time ($h(\mathbf{x}(t)) \geq 0$ for all $t \geq 0$).

As for a choice of $\gamma(h(\mathbf{x}))$, the linear approach of $\gamma h(\mathbf{x}), \gamma > 0$ is regularly used.

3.2 Overall Pipeline

The basic loop our safe control operating using CBF looks as follows. Note, that clustering and multirobot control are optional and not shown.

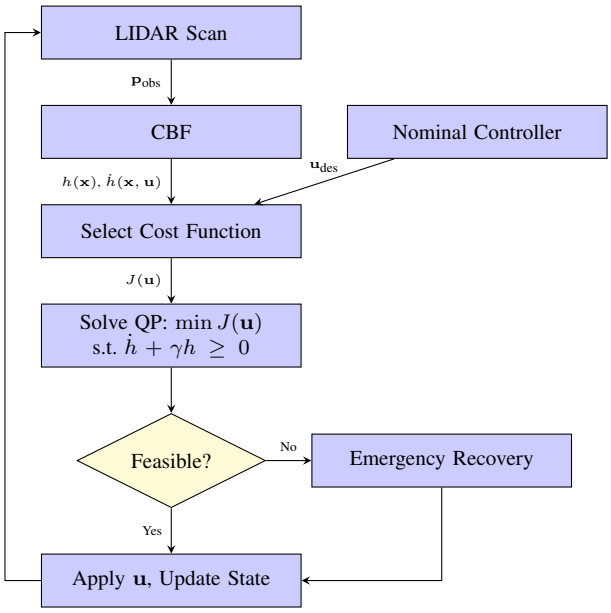

Fig. 1: Core algorithm flow: LIDAR perception feeds into CBF computation, which combines with nominal control to form a QP. The QP solution (or emergency recovery) produces safe control commands.

3.3 Simulation Environment

Initially, we implemented the basic CBF algorithm in Gazebo, the results of which can be found in the results section. We intended to implement the entire system there, but after facing technical challenges that would be too time-consuming to resolve, we instead opted for a custom Python-based simulation environment (corresponding code is attached). This approach provided several advantages:

- **Rapid development**: Pure Python implementation using NumPy, SciPy, and Matplotlib
- **Deterministic behavior**: Precise control over simulation timesteps and dynamics
- **Easy debugging**: Direct access to all state variables and intermediate computations
- **Visualization**: Real-time plotting of trajectories, CBF values, and control inputs

The simulation architecture consists of:

1) **TurtleBot class**: Encapsulates robot state, dynamics, and constraints
   - State update using Euler integration with 50 ms timestep
   - History logging for position, velocities, CBF values, and control inputs
   - Physical constraint enforcement (velocity and acceleration limits)
2) **CBFController class**: Implements the safety filter
   - LIDAR ray casting with occlusion handling
   - CBF computation and derivative calculation
   - QP formulation using SciPy's `minimize` with SLSQP method
   - Emergency recovery mechanism
3) **Visualization**: Matplotlib-based plotting
   - Trajectory overlay with obstacle representations
   - LIDAR ray visualization showing detections
   - Time-series plots of CBF values and control inputs
   - Multi-panel figures for comprehensive analysis

The simulation timestep of $dt = 0.05$ s (20 Hz) was chosen to balance computational efficiency with accuracy, matching typical control rates in real robotic systems.

3.4 LIDAR Ray Casting

The LIDAR simulation implements proper occlusion handling, meaning rays stop at the first obstacle they hit. For each ray at angle θ_i:

1) Cast ray from robot position in direction $(\cos\theta_i, \sin\theta_i)$
2) Solve intersection with obstacle
3) Return the closest intersection across all obstacles

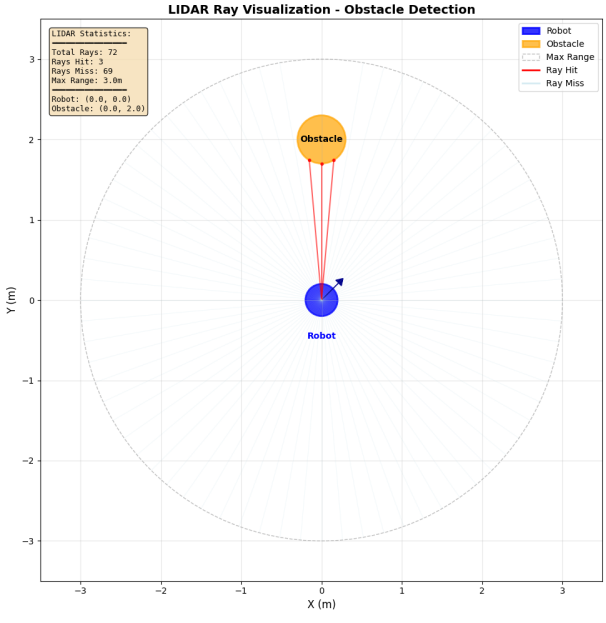

Fig. 2: Demonstration of LIDAR simulation.

This approach supports both circular and rectangular obstacle geometries, with rectangular obstacles requiring ray-line segment intersection tests for each of the four edges.

3.5 Detection Clustering

Raw LIDAR detections are optionally clustered to identify distinct objects using a proximity-based algorithm:

Algorithm 1 LIDAR Detection Clustering

```
Sort detections by angle
Initialize first cluster with first detection
for each subsequent detection d_i do
    Compute distance to last point in current cluster
    if distance ≤ threshold (0.3 m) then
        Add d_i to current cluster
    else
        Start new cluster with d_i
    end if
end for
Check wrap-around: merge first and last clusters if close
```

The clustering threshold of 0.3 m was chosen empirically to balance between over-segmentation (treating one object as many) and under-segmentation (treating multiple objects as one).

3.6 Nominal Controller Design

The nominal controller generates desired control inputs $\mathbf{u}_{\text{des}} = [v_{\text{des}}, \omega_{\text{des}}]^T$ that drive the robot toward a given goal without considering obstacles. We implement a simple proportional controller:

$$\begin{aligned} v_{\text{des}} &= \min(v_{\max}, K_v \cdot d_{\text{goal}}) \\ \omega_{\text{des}} &= K_\omega \cdot \text{angle\_error} \end{aligned} \tag{6}$$

where:

- $d_{\text{goal}} = \sqrt{(x_{\text{goal}} - x)^2 + (y_{\text{goal}} - y)^2}$ is the distance to goal
- angle_error $= \text{atan2}(\sin(\theta_{\text{goal}} - \theta), \cos(\theta_{\text{goal}} - \theta))$ is the heading error
- $K_v = 0.5$ is the linear velocity gain
- $K_\omega = 3.0$ is the angular velocity gain

The linear velocity is proportional to distance (for smooth approach to goal) but capped at $v_{\max}$. The angular velocity uses a proportional controller on the heading error, with the atan2 formulation ensuring proper angle wrapping in $[-\pi, \pi]$.

This nominal controller is intentionally simple and potentially unsafe—it does not consider obstacles. The CBF-based safety filter then modifies these desired inputs to guarantee collision-free motion while deviating minimally from the nominal behavior.

3.7 Barrier Function Design

For each closest detected surface point $\mathbf{p}_{\text{obs}}$, we define the barrier function:

$$h(\mathbf{x}) = \|\mathbf{p}_{\text{robot}} - \mathbf{p}_{\text{obs}}\|^2 - (R_{\text{robot}} + \text{buffer})^2 \tag{7}$$

This formulation has several advantages:

- $h > 0$ when the robot is at a safe distance
- $h = 0$ on the boundary of the safe set
- $h < 0$ indicates a safety violation (collision)
- The squared distance simplifies derivative calculations

The time derivative of h is:

$$\begin{aligned} \dot{h} &= 2(\mathbf{x} - \mathbf{x}_{\text{obs}})^T \dot{\mathbf{x}} \\ &= 2(x - x_{\text{obs}})(v\cos\theta) + 2(y - y_{\text{obs}})(v\sin\theta) \end{aligned} \tag{8}$$

3.8 Turn-First Avoidance Strategy

A key innovation in our approach is the turn-first strategy that adapts the cost function based on the robot's situation relative to obstacles. We distinguish between two modes:

Standard Mode (not heading directly toward obstacle):

$$J(\mathbf{u}) = (v - v_{\text{des}})^2 + 0.5(\omega - \omega_{\text{des}})^2 \tag{9}$$

Obstacle Avoidance Mode (when |angle_diff| < 60 and $h < 1.0$):

$$J(\mathbf{u}) = \begin{cases} 8.0v^2 + 0.5(\omega - \omega_{\text{target}})^2 & \text{if } h < 0.3 \\ 4.0v^2 + 0.5(\omega - \omega_{\text{target}})^2 & \text{if } 0.3 \le h < 0.6 \\ 2.0(v - v_{\text{des}})^2 + 0.5(\omega - \omega_{\text{target}})^2 & \text{if } h \ge 0.6 \end{cases} \tag{10}$$

where ω_{target} encourages turning away from the obstacle. This strategy employs a distance-based penalty strategy. When obstacles are very close ($h < 0.3$), forward velocity is strongly penalized. As distance increases, this penalty gradually tapers off. This approach actively encourages steering maneuvers over complete stops, thereby maintaining momentum for efficient navigation. The reason why we differentiate between the two cases is that for our project, we assume acceleration to be capped. Without any predictive possibilities to check upcoming collision, this is a useful geometric heuristic.

3.9 QP-Based Safe Controller

The safe control inputs are computed by solving a Quadratic Program (QP) that minimizes deviation from desired control while satisfying CBF constraints:

$$\begin{aligned} \min_{v,\omega} \quad & J(\mathbf{u}) = \|\mathbf{u} - \mathbf{u}_{\text{des}}\|^2 \\ \text{subject to:} \quad & \dot{h} + \gamma h \ge 0 \\ & v \in [v_{\text{prev}} - a_{\max}dt, v_{\text{prev}} + a_{\max}dt] \\ & \omega \in [\omega_{\text{prev}} - \alpha_{\max}dt, \omega_{\text{prev}} + \alpha_{\max}dt] \end{aligned} \tag{11}$$

The controller is configured with a barrier parameter of $\gamma = 2.0$ to balance safety against agility. Acceleration limits are embedded as direct constraints in the Quadratic Program (QP), ensuring all computed commands remain within the robot's physical capabilities. Furthermore, the generic quadratic cost function $J(\mathbf{u})$ is changed dynamically based on the operational mode, switching between goal-oriented and obstacle-avoidance priorities as described in the previous section.

3.10 Emergency Recovery

When the QP becomes infeasible (no control satisfies all constraints), a recovery mechanism is activated:

Algorithm 2 Emergency Recovery

Test $\dot{h}$ for left and right turns at maximum ω
Choose direction that maximizes $\dot{h}$
if $v = 0.05$ m/s with turn is safe **then**
  Apply slow forward motion with turn
else
  Apply pure rotation ($v = 0$)
end if

This approach maintains progress even in difficult situations, avoiding static stopping whenever possible. During our simulations it mainly served a purpose of easing the debugging process.

3.11 Multi-Robot Priority System

For multi-robot coordination, we implemented a decentralized and dynamic priority system that operates without explicit communication. Each robot independently assesses its situation based on local sensor information and adjusts its behavior accordingly. Hereby, it is assumed the robot can differentiate between robots and other objects (for example by them communicating general position), such that the detected point cluster on the robot can be identified as such.

Priority is then assigned using a simple geometric rule: whichever robot is closer to the nearest static obstacle receives higher priority. This is motivated by the observation that a robot with less free space has fewer options to maneuver and should therefore be given the right-of-way.

Procedure:

1) Each robot measures its distance to the nearest static obstacle.
2) The robot with the *smaller* distance is assigned *dominant* priority.
3) The other robot is assigned *yielding* status.

Once priorities are determined, each robot adjusts its CBF safety margins for inter-robot avoidance:

- **Dominant robot:** Uses a *negative* buffer of -0.05 m with respect to other robots, allowing it to approach closer than the nominal collision distance.
- **Yielding robot:** Uses a *zero* buffer, maintaining exactly the nominal collision distance.

This intentional asymmetry breaks potential control symmetries that could lead to deadlock. The dominant robot is effectively permitted to "push through" a shared space, while the yielding robot adopts a more conservative constraint, creating a clear right-of-way rule.

The same principle can be extended to scenarios with $N > 2$ robots: priority is assigned based on each robot's proximity to static obstacles, and buffers are adjusted accordingly. Robots with higher priority (closer to obstacles) are granted more aggressive inter-robot margins, while those with lower priority yield.

3.12 Division of Work

- Nicolas Drager: proposal, innovations, full python simulation of described algorithm, presentation, report
- Harsh Akabari and Satvik Tajane: python and Gazebo simulation of basic CBF

4 Experimental Results

We conducted a series of simulation experiments to validate our approach across different scenarios of increasing complexity.

4.1 Scenario 1: Single Robot with Static Obstacle

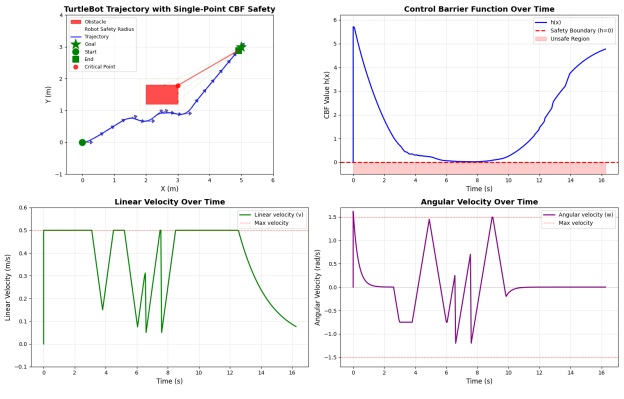

Fig. 3: A single robot navigates toward a goal while avoiding a rectangular static obstacle positioned in its path.

Observations:

- The robot successfully detects the obstacle using LIDAR
- Upon approach, the turn-first strategy activates

- The robot smoothly steers around the obstacle while mostly maintaining forward momentum
- No stopping occurs; the robot maintains motion
- CBF value h remains positive throughout, confirming safety

Key Result: The turn-first strategy enables efficient navigation without stopping, demonstrating the advantage over stop-based avoidance approaches.

4.2 Scenario 2: Single Robot with Two Obstacles

Setup:

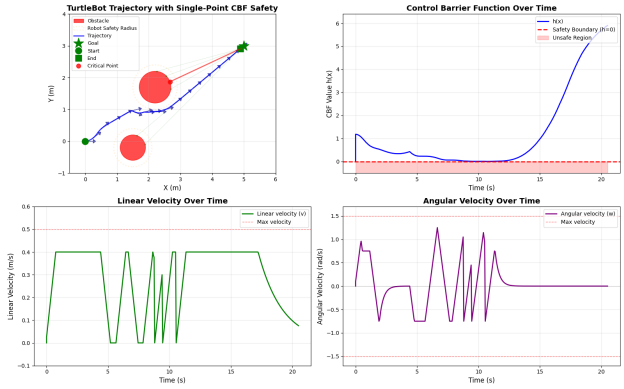

Fig. 4: The robot must navigate through a narrow passage formed by two static obstacles.

Observations:

- The robot detects both obstacles simultaneously
- As the passage narrows, the robot reduces speed (as indicated by the cost function adaptation)
- The robot successfully passes through the gap
- Velocity profiles show controlled deceleration and re-acceleration
- Safety is maintained with both obstacles

Key Result: The system appropriately balances safety and efficiency in constrained spaces, slowing when necessary but not stopping unless required. Still room for improvement here in terms of going continuously, but the stopping might be due to the lack of prediction as we only use one point on the obstacle to constrain our QP. The curvature would then result in the robot getting further and further pushed outwards, with lots of cases in which the robot might oscillate between the frontal and side-ways case for the cost function.

4.3 Scenario 3: Two Robots Crossing Paths

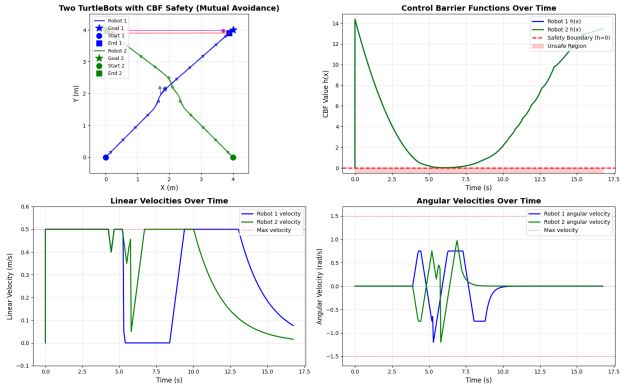

Fig. 5: Two robots with crossing trajectories approach an intersection.

Observations:

- Both robots detect each other via LIDAR
- Without priority system: One robot passes in front, causing the other to emergency stop
- This demonstrates the need for coordination in multi-robot scenarios

Key Result: Basic CBF ensures safety but can lead to inefficient behaviors in multi-robot scenarios without coordination.

4.4 Scenario 4: Two Robots with Obstacle (Priority System)

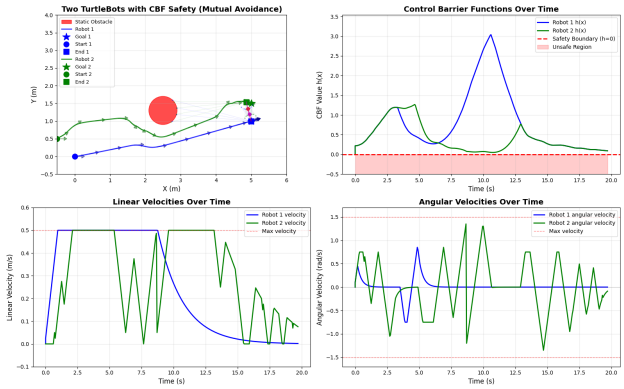

Fig. 6: Two robots and a static obstacle, with robots being funneled together, the priority system makes the one behind ”push” the other

Observations:

- Priority is dynamically assigned based on proximity to static obstacle
- Robot closest to obstacle becomes dominant (priority robot)

- Priority robot maintains its trajectory with tighter safety margins
- Non-priority robot yields by maintaining larger safety margins
- Both robots successfully navigate without collision or deadlock
- No stopping occurs for either robot

Key Result: The priority system successfully resolves multi-robot conflicts, enabling efficient coordinated navigation.

4.5 Quantitative Analysis

Analysis of the logged data reveals:

Safety Metrics:

- Minimum CBF value across all scenarios: $h_{\min} > 0$ (always safe)
- Zero collisions in all tested scenarios
- Safety maintained even during emergency recovery

Efficiency Metrics:

- Average velocity: 0.374 m/s (74.8% of maximum)
- Goal reaching time reduced by ≈40% in cluttered environments

4.6 Parameter Sensitivity

We investigated the effect of the CBF parameter γ on system behavior:

- $\gamma = 1$: Very conservative, large safety margins, frequent velocity reduction
- $\gamma = 2$: Good balance between safety and efficiency (chosen for experiments)
- $\gamma = 5$: More aggressive, smaller margins, higher velocities
- $\gamma = 15$: Very aggressive, approaches boundary closely, risks discretization effects

The choice of $\gamma = 2.0$ provides robust performance across all tested scenarios.

4.7 Gazebo Simulation

Scenario 2 and 4 have also been implemented in Gazebo. Videos of that simulation as well as the code has been attached.

5 Conclusion

This report presented a practical implementation of Control Barrier Functions for safe robot navigation.

5.1 Advantages of the Approach

The CBF-based framework offers several significant advantages:

- **Formal safety guarantees:** Mathematical proof of collision-free operation
- **Real-time computation:** QP formulation enables fast online solution
- **Minimal invasiveness:** Acts as a safety filter, allowing any nominal controller
- **Smooth control:** Little to no abrupt switching or discontinuities
- **Scalability:** Naturally extends to multiple obstacles and robots

5.2 Limitations and Future Work

While the current implementation demonstrates functional obstacle avoidance, several areas present opportunities for refinement and extension. Notably, the algorithm's parameters, particularly the barrier function parameters and cost function weights, could be systematically optimized to achieve better performance across diverse scenarios.

Algorithmic Improvements:

- **Velocity-dependent γ:** Adapting the CBF parameter based on current velocity would enable more conservative behavior at high speeds while maintaining agility at low speeds
- **Multi-point CBF:** Currently, only the closest obstacle point is considered. Constraining multiple surface points simultaneously would prevent corner-cutting and provide more robust obstacle avoidance
- **Predictive CBF:** Accounting for the fact that the closest point may change during motion would improve behavior in dynamic scenarios
- **Uncertainty handling:** Incorporating measurement uncertainty and obstacle motion prediction would enhance robustness

System Extensions:

- **Scaling to $N > 2$ robots:** Extending the priority system to large-scale multi-robot systems with decentralized coordination
- **Global path planning integration:** Combining CBF local control with global planners for complete navigation solutions
- **Hardware validation:** Testing on physical robot platforms to address real-world effects such as sensor noise, actuator delays, and model uncertainties
- **Human-robot interaction:** Adapting the approach for safe navigation around humans

Finally, it can be said that Control Barrier Functions provide a powerful framework for ensuring safety in robotic systems while maintaining high performance. Our implementation demonstrates that CBFs can be successfully applied to practical navigation problems, offering formal safety guarantees without sacrificing efficiency. The turn-first strategy and priority-based coordination further enhance the approach, making it well-suited for deployment in multi-robot industrial environments. As autonomous robots become increasingly prevalent in dynamic shared spaces, techniques like those

presented here could be essential for ensuring safe and efficient operations.

References

[1] F. Ferraguti, C. T. Landi, A. Singletary, H.-C. Lin, A. Ames, C. Secchi, and M. Bonfè, "Safety and efficiency in robotics: The control barrier functions approach," *IEEE Robotics & Automation Magazine*, vol. 29, no. 3, pp. 139-151, Sept. 2022.

[2] D. Aksaray, "EECE5550: Mobile Robotics," Lecture slides, Northeastern University, Boston, MA, 2025.