Efficient Safe Robot Navigation Using Control Barrier Functions

Nicolas Drager, Satvik Tajane, Harsh Akabari

December 9, 2025

Outline

Problem: Safe Navigation in Dynamic Environments

Safety is paramount in robotics

- Primary concern: collision avoidance
- But robots must also reach their goals efficiently

$\Longrightarrow$ **avoid collisions but keep moving**

Application context:

- Multi-robot warehouses: many robots, dynamic environment
- High-speed operations require real-time safety guarantees

Solution: CBF as a Safety Filter

Core Idea

Control Barrier Functions act as a **safety filter**:

- Filters out unsafe commands
- Minimally modifies safe ones for efficient avoidance in the future

Our Approach:

- Mathematical safety guarantees via CBF constraints
- **Turn-first strategy**: prioritize turning over stopping
  - Maintains forward momentum
  - Faster navigation through tight spaces
- Priority-based multi-robot coordination

Control Barrier Functions: Core Concept

Safety Definition

CBF $h(\mathbf{x})$ defines safe region: $\mathcal{C} = \{\mathbf{x} \mid h(\mathbf{x}) \geq 0\}$

Safety Constraint

If control $\mathbf{u}$ satisfies:

$$\dot{h}(\mathbf{x}, \mathbf{u}) + \gamma h(\mathbf{x}) \geq 0, \quad \gamma > 0 \tag{1}$$

then system remains safe: $h(\mathbf{x}) \geq 0$ for all time

Simulated Robot Dynamics & Constraints

Differential drive robot modeled as unicycle:

$$\dot{x} = v\cos\theta, \quad \dot{y} = v\sin\theta, \quad \dot{\theta} = \omega \tag{2}$$

Control Inputs:

- Linear velocity: v
- Angular velocity: ω

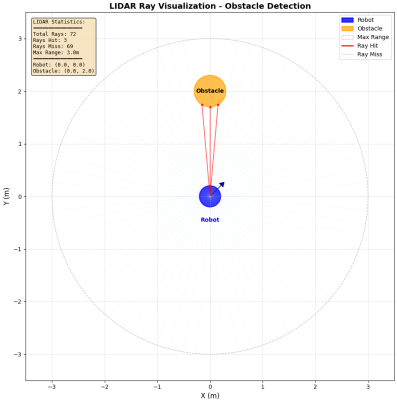
Physical limits:

- Velocity: $v \in [0, 0.5]$ m/s, $\omega \in [-1.5, 1.5]$ rad/s
- Acceleration: $|\dot{v}| \leq 0.5$ m/s^2, $|\dot{\omega}| \leq 2.0$ rad/s^2
- Robot radius: $R = 0.2$ m

LIDAR-Based Surface Detection and Distance Measurement

LIDAR detects *obstacle surfaces*

- 360 rays at 1° resolution to detect surfaces and their distance to robot
- Clustering (optional): group detections to identify objects



Barrier Function

For closest detected surface point $\mathbf{p}_{\text{obs}}$:

Barrier Function

$$h(\mathbf{x}) = \|\mathbf{p}_{\text{robot}} - \mathbf{p}_{\text{obs}}\|^2 - (R_{\text{robot}} + \text{buffer})^2 \tag{3}$$

$\implies$ **Safety buffer** differs dependent on the detected object!
$\implies$ Allows the prioritization of objects, in our case fellow robots

QP-Based Safe Controller

Optimization problem:

$$
\begin{aligned}
\min_{v,\omega} \quad & \|u - u_{\text{des}}\|^2 \\
\text{s.t.} \quad & \dot{h} + \gamma h \geq 0 \\
& v \in [v_{\text{prev}} - a_{\max} dt, v_{\text{prev}} + a_{\max} dt] \\
& \omega \in [\omega_{\text{prev}} - \alpha_{\max} dt, \omega_{\text{prev}} + \alpha_{\max} dt]
\end{aligned}
\tag{4}
$$

Notice:

- Finds control closest to desired while guaranteeing safety
- Acceleration limits embedded in QP bounds (not post-processed)
- $\gamma = 2.0$ balances safety and agility

Prioritizing Turning Over Stopping

Standard case (not heading directly toward obstacle):

$$J(\mathbf{u}) = (v - v_{\text{des}})^2 + 0.5(\omega - \omega_{\text{des}})^2 \tag{5}$$

Obstacle avoidance mode (when $|\text{angle\_diff}| < 60$ and $h < 1.0$):

$$J(\mathbf{u}) = \begin{cases} 8.0v^2 + 0.5(\omega - \omega_{\text{target}})^2 & \text{if } h < 0.3 \\ 4.0v^2 + 0.5(\omega - \omega_{\text{target}})^2 & \text{if } 0.3 \leq h < 0.6 \\ 2.0(v - v_{\text{des}})^2 + 0.5(\omega - \omega_{\text{target}})^2 & \text{if } h \geq 0.6 \end{cases} \tag{6}$$

where ω_{target} encourages turning away from obstacle.

Emergency Recovery

When QP fails (constraint infeasible):

1. Test $\dot{h}$ for left/right turns
2. Choose direction maximizing $\dot{h}$
3. Try $v = 0.05$ m/s + turn (if safe)
4. Else: pure rotation $v = 0$

Key: Intelligent recovery maintains progress vs. static stopping

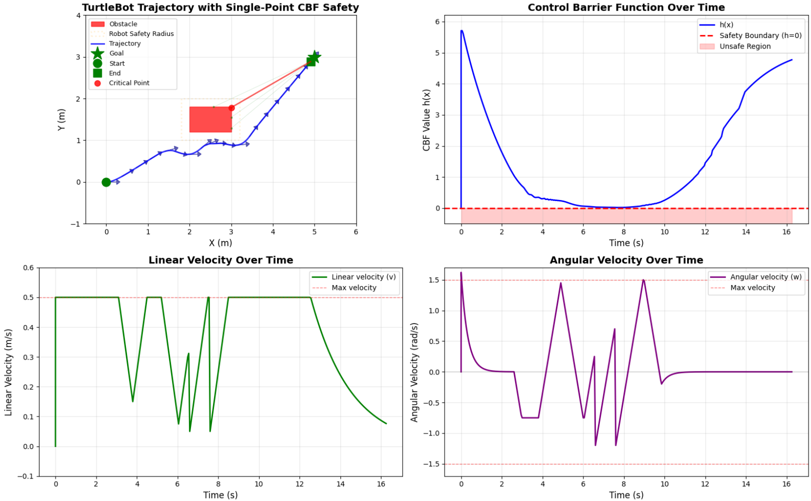
Multi-Robot Priority System

Problem: Equal priorities $\rightarrow$ both robots stop/deadlock
Solution: Dynamic priority based on context

- Robot closer to static obstacle gets priority
- Priority robot: buffer $= -0.05$ m for other robots
- Non-priority robot: buffer $= 0$ m for other robots

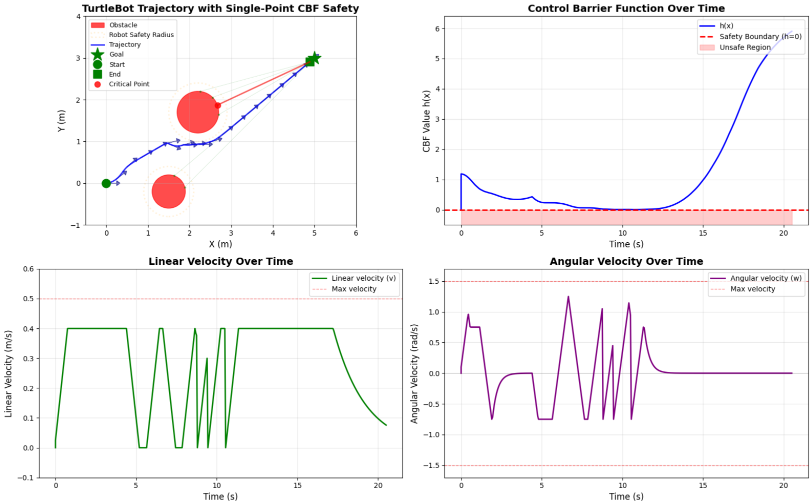
Result: Priority robot can "push" through, non-priority yields

Scenario: One Robot with Obstacle



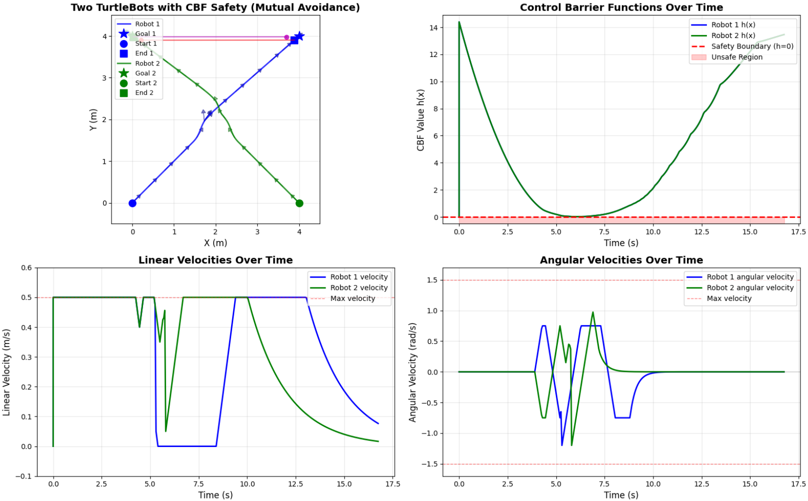
Observation: Obstacle is detected and avoided by turning!

Scenario: One Robot with Two Obstacles



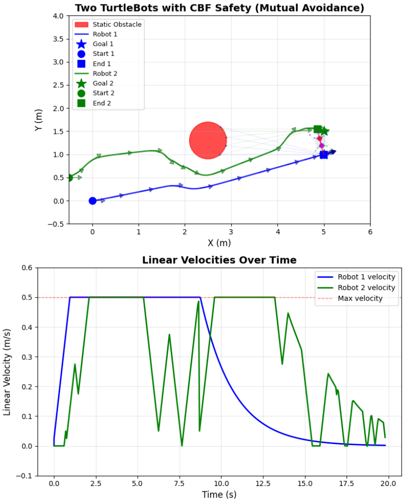
Observation: Robot slows down and passes

Scenario: Two Robots Crossover



$\Longrightarrow$ One went in front of the other causing a emergency stop!

Scenario: Two Robots with Obstacle (Priority System)



$\Longrightarrow$ The robot closest to the obstacle is the dominant one!

Contributions

- **Turn-first strategy**: Maintains momentum while ensuring safety
- **Priority system**: Resolves multi-robot deadlocks
- **Intelligent recovery**: Handles QP failures gracefully

Potential Future Work

Algorithmic improvements:

- Velocity-dependent γ: more conservative at high speeds
- Multi-point CBF: constrain multiple surface points (prevents corner-cutting)
- Predictive CBF: account for changing closest point during motion

System extensions:

- Scale to $N > 2$ robots with decentralized coordination
- Integration with global path planning
- Hardware validation on physical platforms