LiDAR-Enhanced Dynamic Control Barrier Functions for Real-Time Collision Avoidance in an Unknown Environment

Nidhi Agarwal, *Student Member, IEEE*, Shyam Kamal, *Member, IEEE*, Kyle Collins, Kranthi Kumar Deveerasetty, Diwakar Saini, Sandip Ghosh, *Member, IEEE*, Anchal Bhardwaj

***Abstract*—This paper introduces an online LiDAR-based discrete-time barrier function combined with a dynamic obstacle avoidance algorithm to ensure system safety by keeping the state within an invariant set and minimizing collision risks. The online LiDAR-based discrete-time barrier function is a low-level safety controller, particularly in unknown environments, enabling safe navigation under set-based constraints and ensuring safety with both dynamic and static obstacles. The method synthesizes the online LiDAR-based discrete-time barrier function for safe control input corrections. Experimental validation with TurtleBot3 simulations and hardware tests on the Quanser QBot platform demonstrates the effectiveness of the LiDAR-based online LiDAR-based discrete-time barrier function for safe navigation in real-world scenarios.**

***Index Terms*—Wheeled mobile robots (WMRs), dynamic obstacle avoidance algorithm (DOAA), control barrier functions, online LiDAR-based discrete-time barrier function (OLDBF), collision avoidance, safe control design.**

Nomenclature

Nomenclature

CBF	Control barrier function
DBF	Discrete-time barrier function
DOAA	Dynamic obstacle avoidance algorithm
DOF	Degree of freedom
HRI	Human–robot interaction
LiDAR	Light detection and ranging
OLDBF	Online LiDAR-based discrete-time barrier function
ROS	Robot Operating System
WMR	Wheeled mobile robot

I. Introduction

Safe navigation in unknown environments, such as autonomous operations, teleoperation, and human–robot interaction (HRI), is crucial for robotics and unmanned vehicles [1]. Control Barrier Functions (CBFs), based on set invariance theory, transform state constraints into control input conditions, ensuring the system stays within a safe region [2]. The CBF-based safety filter ensures constraints while minimally adjusting the control input [3]- [5]. To synthesize a valid CBF, control-invariant sets for vehicle velocity and input constraints must be defined. These sets ensure that if the system starts within a safe region, it remains within it [7], [8], [9]. CBFs regulate control inputs through the condition: $\exists u$ such that $\dot{h}(x,u) \geq -\alpha(h(x)) \iff C$ is invariant,, where h is the safety function, α controls the change speed, and u is the control input.

Autonomous mobile robots, equipped with sensors like ultrasonic, radar, and LiDAR [11], [12], use these technologies to gather obstacle information [13], [14]. LiDAR is particularly useful for obstacle avoidance in mobile robots [15].

Discrete-Time Barrier Functions (DBFs) [16], [17] differ from continuous-time functions, operating within real-time sampling limits and ensuring safety at slower rates [18]. CBFs have been successfully applied in various fields, such as walking robots, automotive systems, multi-robot systems, and quadrotors. They ensure real-time safety, even in environments requiring quick lateral movements [19]. This work introduces the OLDBF (Online-LiDAR Discrete Barrier Function) concept, expanding OLDBFs by incorporating LiDAR data for real-time navigation in dynamic environments. The discrete-time approach ensures better performance compared to continuous-time methods. The primary contributions are:

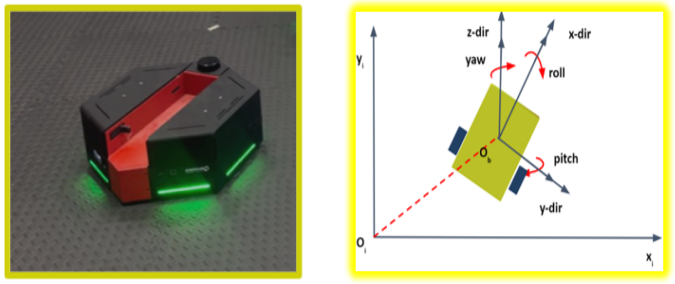

Fig. 1: Typical image of a robot in a laboratory environment with dynamic obstacle avoidance.

Nidhi Agarwal, Shyam Kamal, and Sandip Ghosh are with the Department of Electrical Engineering, Diwakar Saini is with the Department of Ceramic Engineering, Indian Institute of Technology (BHU), Varanasi 221005, India (email: nidhiagarwal.rs.eee20@iitbhu.ac.in; shyamkamal.eee@iitbhu.ac.in; sghosh.eee@iitbhu.ac.in; diwakar.saini.cer23@iitbhu.ac.in).

Kyle Collins and Kranthi Kumar Deveerasetty are with Eagle Flight Research Center, Embry-Riddle Aeronautical University, Daytona Beach, USA (e-mail: kyle.collins@erau.edu, DEVEERAK@erau.edu)

Anchal Bhardwaj is with Banasthali Vidyapith, Rajasthan, 304022, India (email: anchalbhardwaj022@gmail.com).

- Online synthesis of OLDBFs, ensuring safety beyond convex dangerous sets for nonlinear discrete-time systems and establishing forward invariance of safety sets.
- The DLF-based method maintains a larger minimum safety distance between WMRs and static/dynamic obstacles, outperforming traditional velocity obstacle techniques.
- We test the OLDBF with logical control on a QBot platform with a 2D 360° LiDAR sensor for navigating environments with both static and dynamic obstacles (Fig. 1).

The remainder of this paper is organized as follows: Section II presents notation and an overview of CBFs; Section III introduces the proposed control design with OLDBF and stability analysis; Section IV demonstrates OLDBFs' application with simulation and experimental results; Section V includes conclusion.

II. Notation and Preliminary Concepts

A. Notation

Let $\mathbb{R}^n$ be the n-dimensional real vector space, and $\mathbb{Z}$ the set of positive integers. The absolute value of a scalar is $|\cdot|$, and the Euclidean norm in $\mathbb{R}^n$ is $\|\cdot\|$. The open ball centered at x with radius ϵ is $\mathcal{B}_\epsilon(x)$. The transpose of a matrix $\mathbf{A}$ is $\mathbf{A}^\top$, and the interior and boundary of a set S are $\text{Int}(S)$ and ∂S, respectively. The distance from $\mathbf{x}$ to S is: $\|\mathbf{x}\|_S = \inf_{s\in S} \|\mathbf{x}-s\|$. For an essentially bounded function $g : \mathbb{R} \to \mathbb{R}^n$, the infinity norm is: $\|g\|_\infty = \sup_{t\in\mathbb{R}} \|g(t)\|$. The operators $\max$ and $\min$ denote the maximum and minimum values of a set.

B. Problem Formulation

We consider a general nonlinear, time-invariant control-affine system described by

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

where $x \in \mathbb{R}^n$ denotes the state vector and $u \in \mathbb{R}^m$ is the control input. The functions $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are assumed to be Lipschitz continuous in x, and the origin $x^* \equiv \mathbf{0}$ is an equilibrium point of the system. At each time step, the sensor output is represented by the state vector $s_t \in \mathcal{S} \subset \mathbb{R}^n$, where $|Xi$ is the number of LiDAR beams. The robot's motion is modeled as a discrete-time nonlinear system,

$$z(k+1) = \mathcal{F}(z(k), u(k)), \quad z(0) = z_0, \tag{2}$$

where $k \in \mathbb{Z}$ is the discrete time index, $z \in \mathcal{D} \subset \mathbb{R}^n$ is the system state, $u(k) \in \mathcal{U} \subset \mathbb{R}^m$ is the control input, and $\mathcal{F} : \mathcal{D}\times\mathcal{U} \to \mathbb{R}^n$ is a Lipschitz continuous function describing the system dynamics.

C. Control Barrier Function

Consider the system (2). The goal is to design algorithms that regulate the states on the boundary of a specified invariant set using a CBFs, ensuring the system remains within the invariant set while optimizing performance. To ensure safety, verify that trajectories remain within the forward-invariant safe set $\mathcal{D} \subseteq \mathbb{R}^n$, as discussed in [2].

Definition 1: (Zero-Super Level Set) For the system (2) with locally Lipschitz $\mathcal{F}$, the zero-super level set $\mathcal{S}$ of $h : \mathbb{R}^n \to \mathbb{R}$ is: $\mathcal{S} := \{z(k) \mid h(z(k)) \geq 0\}$.

Definition 2: [9] (Class $\mathcal{K}$ Function) A function β_1 belongs to class $\mathcal{K}$ if: 1) strictly increasing, 2) $\beta_1(0) = 0$.

Definition 3: [9] (Class $\mathcal{KL}$ Function) A function β_2 belongs to class $\mathcal{KL}$ if: 1) for each s, $\beta_2(r, s) \in \mathcal{K}$, 2) for each r, $\beta_2(r, s)$ is decreasing with $\beta_2(r, s) \to 0$ as $s \to \infty$.

Definition 4: [9] (Controlled Invariance Set) The set $\mathcal{S}$ is controlled invariant if, for each $z(k) \in \mathcal{S}$, there exists a control $u(k) \in \mathcal{U}$ such that $\mathcal{F}(z(k), u(k)) \in \mathcal{S}$.

D. Forward Invariance of Safety Set via Barrier Functions

The safety set is defined as: $\mathcal{S} := \{z(k) \in \mathcal{D} \mid h(z(k)) \geq 0\}, \quad \partial\mathcal{S} := \{z(k) \mid h(z(k)) = 0\}$. The barrier function h ensures the system stays within the safety set. Forward invariance is guaranteed if: $\dot{h}(z(k)) = \nabla h(z(k))\cdot\dot{z}(k) \geq 0, \quad \forall z(k) \in \mathcal{S}$.

1) Assumption for Forward Invariance of Safety Set:

Assumption 1: Consider the set $\mathcal{S}$ to be forward invariant when h is a barrier function. A sufficient condition for forward invariance is $\mathcal{S}$ is a non-empty compact set and $\frac{\partial h}{\partial z(k)} = 0$ for all $z(k) \in \partial\mathcal{S}$, i.e., on the boundary of the safety set.

The safety set $\mathcal{S}$ is safe if $h(z(k)) < 0$ for all $z(k) \in \mathcal{D}\setminus\mathcal{S}$, ensuring the state stays strictly inside the safety set.

E. Discrete-Time Barrier Function (DTCBF) and Controlled Invariance

Let $\mathcal{S} \subseteq \mathcal{D} \subset \mathbb{R}^n$ be a safety set, with $h : \mathbb{R}^n \to \mathbb{R}$ as a barrier function. The system is controlled invariant if for each $z(k) \in \mathcal{S}$, there exists a control input $u(k) \in \mathcal{U}$ such that:

$$h(\mathcal{F}(z(k), u(k))) \geq 0, \quad \forall z(k) \in \mathcal{S}. \tag{3}$$

The variation of the barrier function is constrained by: $\Delta h(z(k), u(k)) := h(\mathcal{F}(z(k), u(k))) - h(z(k)) \geq -\alpha(h(z(k)))$, where $\alpha(h(z(k))) = \gamma h(z(k))$ with $\gamma \in (0, 1]$. Therefore, the DTCBF condition becomes: $h(\mathcal{F}(z(k), u(k))) \geq (1-\gamma)h(z(k)), \quad \forall z(k) \in \mathcal{S}$. Consider the system (2). This ensures that the barrier function decreases by at most a factor of $(1-\gamma)$ at each step, guaranteeing the system stays within the safety set. Over multiple time steps:

$$h(\mathcal{F}(z(k), u(k))) \geq (1-\gamma)^k h(z(k)), \quad \forall k \in \mathbb{Z}. \tag{4}$$

This shows that the barrier function decreases over time at a rate controlled by γ, ensuring the state stays within the safety set $\mathcal{S}$.

Definition 5 (Discrete Barrier Function (DBF):): Let $h : \mathbb{R}^n \to \mathbb{R}$ represent the 0-super level set. A domain $S \subseteq \mathbb{R}^n$ is safe for the system in (2) if there exists an extended class $\mathcal{K}_\infty$ function α such that $L_f h(z(k)) + \alpha(h(z(k))) \geq 0, \quad \forall z(k) \in S$, where $L_f h(z(k)) = \nabla h(z(k))\cdot f(z(k))$ is the Lie derivative, and α is a strictly increasing function satisfying $\alpha(0) = 0$.

This condition ensures that $h(z(k))$ stays non-negative, preventing the state from exiting the safety set S. The function α controls the rate at which $h(z(k))$ can decrease, maintaining safety.

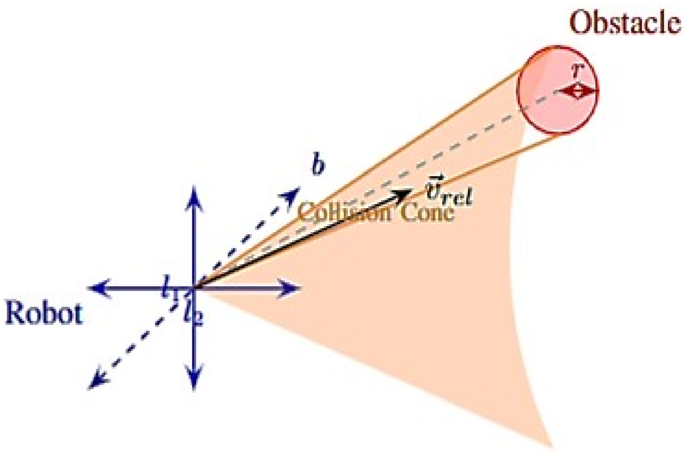

Fig. 2: Construction of collision cone for a collision cone with elliptical obstacle

III. Online LiDAR-Based Discrete Barrier Function (OLDBF)

The OLDBF algorithm uses real-time 2D LiDAR data to adjust the control inputs v and ω on the QBot platform, ensuring safe trajectory planning. If an obstacle is detected, OLDBF dynamically modifies the velocity to avoid collisions, ensuring the system operates within safety boundaries.

A. Sensor Data Representation with Collision Cone

At each time step, the 2D LiDAR sensor emits laser beams to measure distances to obstacles. The output is a distance vector: $\mathbf{p} = [p_1, p_2, \ldots, p_\Xi]^\top$, where p_i ($1 \leq i \leq \Xi$) is the distance from the i-th beam, and Ξ is the total number of beams. Distances beyond the sensor's range are recorded as `inf`. To improve sensor data utility, we take the reciprocals of the distances: $\mathbf{q}_t = \left[\frac{1}{p_1}, \frac{1}{p_2}, \ldots, \frac{1}{p_\Xi}\right]^\top$, where $\mathbf{q}_t \in \mathcal{Q} \subset \mathbb{R}^\Xi$. This amplifies proximity information and maps out-of-range values (`inf`) to zero ($\frac{1}{\infty} = 0$) as described in Fig. 2. The **collision cone** predicts potential collisions based on relative velocities. Obstacles are modeled as circles with radius $r = \max(l_1, l_2) + \frac{b}{2}$, where l_1 and l_2 are the semi-axes of the robot (e.g., width and length for a wheeled mobile robot, WMR), b is the robot's maximum overall dimension.

B. OLDBF and Obstacle Avoidance

The minimal distance $d_{\text{min}}(k)$ is the shortest distance from the robot to any obstacle at time step k. The robot's velocity $v_r(k)$ is bounded by: $v_{\text{min}} \leq v_r(k) \leq v_{\text{max}}$, ensuring safety and feasibility. Define the safe set $\mathcal{S}$ as: $\mathcal{S} = \{z \mid d_{\text{min}}(k) \geq d_{\text{safe}}\}$. To ensure safety, the robot state $z(k)$ must remain within the set $\mathcal{S}$. OLDBF integrates set-theoretic principles and linear algebra to impose state-dependent constraints, generating corrective inputs that ensure $z(k) \in \mathcal{S}$.

Remark 1: A continuous function $h \in \mathcal{S}^0 : \mathbb{R}^n \to \mathbb{R}$ is associated with a closed set $\mathcal{S} \subseteq \mathbb{R}^n$ if there exists $\lambda \in (0, 1]$ such that $\sup_{u \in \mathcal{U}} \Delta h(z(k), u) \geq -\lambda h(z(k)), \quad \forall z(k) \in \mathcal{S}$, where $\Delta h(z(k), u(k)) = h(\mathcal{F}(z(k), u(k))) - h(z(k))$, and $\mathcal{F}(z(k), u(k))$ denotes the system dynamics.

A set $\mathcal{S}$ is *control-invariant* if for any $z(k) \in \mathcal{S}$, there exists $u(k) \in \mathcal{U}$ keeping the system within $\mathcal{S}$.

Definition 6: A function $h(z(k)) \in \mathcal{S}^0$ is an OLDBF for a closed set $\mathcal{S} = \{z(k) \mid h(z(k)) \geq 0\}$.

Proposition 1: The set $\mathcal{S}$ is control-invariant if and only if there exists an OLDBF $h(z(k))$.

Proof: We show that the proposed OLDBF keeps the system state within the safe set $\mathcal{S} = \{z(k) \in \mathbb{R}^n \mid d_{\text{min}}(k) \geq d_{\text{safe}}\}$. First, consider, $\mathcal{B}_d(k) = \frac{\Gamma^2}{\Gamma^2 - (\Delta d_{\text{min}}(k))^\sigma}$. As $\Delta d_{\text{min}}(k) \to 0$, $\mathcal{B}_d(k) \to 1$; as $\Delta d_{\text{min}}(k) \to \Gamma^{2/\sigma}$, $\mathcal{B}_d(k) \to \infty$. Thus, $\mathcal{B}_d(k)$ penalizes proximity to unsafe distances. Next, consider, $\mathcal{B}_v(k) = \frac{\Gamma \cdot \|v_r(k)\| \cdot \zeta}{1 + \|v_r(k)\|^\beta}$. As $\|v_r(k)\| \to 0$, $\mathcal{B}_v(k) \to 0$; as $\|v_r(k)\| \to \infty$, $\mathcal{B}_v(k) \to \zeta\Gamma$. Hence, $\mathcal{B}_v(k)$ penalizes high velocities. The combined barrier function is: $\mathcal{B}(z(k), v_r(k)) = \mathcal{B}_d(k) + \mathcal{B}_v(k)$, which grows large near unsafe distances or high speeds, ensuring forward invariance of $\mathcal{S}$. Thus, OLDBF enforces safety and stability, completing the proof. ■

C. Controller Design

We define a control input $u(k)$ to adjust the robot's angular velocity based on the relative distances B_l and B_r to obstacles on the left and right, respectively. The angular velocity ω governs turning (positive for left, negative for right). When $B_l > B_r$, a negative control input triggers a right turn, and vice versa, ensuring smooth maneuvering aligned with the design constraints.

A nominal angular velocity ω_0 is applied when simultaneous obstacles are detected on both sides. The control input $u(k) \in \{u \in \mathcal{U} \mid \mathcal{B}(d_{\text{min}}(k+1), v_r(k+1)) - \mathcal{B}(d_{\text{min}}(k), v_r(k)) \geq -\alpha(\mathcal{B}(z(k)))\}$ ensures forward invariance of $\mathcal{S}$ for system (2).

The control law is given by $u(k) = \omega_0;$ if turning to avoid obstacles on both sides, otherwise:

$$u(k) = \begin{cases} k(B_r - B_l) & \text{if } B_r < B_l \\ k(B_l - B_r) & \text{if } B_l < B_r \end{cases} \tag{5}$$

where k is a proportional gain adjusting turn sensitivity based on obstacle proximity, ω_0 is a fixed angular speed for predefined turns when obstacles are detected on both sides.

D. Stability Analysis

In the OLDBF framework, stability maintains the system within the safe set $\mathcal{S}$, preventing constraint violations and ensuring recursive feasibility, enhances robustness against disturbances, essential for applications like autonomous navigation. The closed-loop system satisfies:

Theorem 1: Consider system (2) with state constraints $z \in \mathbb{R}^n$ and safe set $\mathcal{S} = \{z \in \mathbb{R}^n \mid h(z) \geq 0\}$ where $h(z)$ is an OLDBF. Let $\kappa : \mathbb{R}^n \to \mathbb{R}^m$ be a nominal controller, and $u(z(k))$ the OLDBF-based control. Given $z(0) \in \mathcal{S}$, applying $u(k) = u(z(k))$ ensures: **feasibility** as the control program remains feasible for all $k \geq 0$, **constraint satisfaction:** as the state satisfies $z(k) \in \mathcal{D}$ for all $k \geq 0$ and **recursive feasibility** as the system remains within $\mathcal{S}$ for all $k \geq 0$.

Proof: Given $z(0) \in \mathcal{S}$ shows **feasibility** as the OLDBF condition ensures a control $u(k)$ satisfying $h(z(k+1)) - h(z(k)) \geq -\alpha(h(z(k)))$ where $\alpha(\cdot)$ is a class-$\mathcal{K}$ function, **constraint satisfaction** as from system dynamics $z(k+1) = \mathcal{F}(z(k), u(k))$, and $\mathcal{S} \subseteq \mathcal{D}$, the state remains within $\mathcal{D}$ and **recursive feasibility**, since $h(z(k+1)) \geq 0$, $z(k+1) \in \mathcal{S}$, ensuring forward invariance. Thus, feasibility, constraint satisfaction, and recursive feasibility are guaranteed. ■

Based on the barrier functions B_l and B_r, a logical control strategy generates angular and linear velocities. The pseudocode for the Dynamic Obstacle Avoidance Algorithm (DOAA) is summarized below:

Algorithm 1 An Algorithm for Discrete-time Dynamic Obstacle Avoidance Algorithm (DOAA)

```
Class: Obstacle Tracker
Initialize:
    state ← 'forward'
Function: scan_callback(data)
    Read LiDAR data to obtain left and right distances
    Compute barrier functions $B_l$ and $B_r$ based on the closest obstacles
    Adjust robot control based on $B_l$, $B_r$, and the distances to obstacles
Function: control_bot($B_l$, $B_r$, closest_left, closest_right)
if obstacles are detected on either side then
        Adjust robot motion to avoid obstacles by modifying speed and direction
else
        Continue moving forward with no adjustments
end if
Function: main()
    Initialize ROS node and subscribe to LiDAR data
    Process LiDAR data in real-time via the scan_callback function
    Use the Obstacle Tracker methods to ensure safety and obstacle avoidance =0
```

IV. Application to Simulated and Real-world Environments

A. Model of the Robot

Consider the kinematic model of a wheeled mobile robot (WMR), represented by the following discrete-time state update equation:

$$\begin{bmatrix} z(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} z(k) \\ y(k) \\ \theta(k) \end{bmatrix} + \begin{bmatrix} \cos\theta(k) & 0 \\ \sin\theta(k) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(k) \\ \omega(k) \end{bmatrix} \tag{6}$$

Here, $v(k)$ and $\omega(k)$ represent the robot's linear and angular velocities at the time step k, respectively. The state vector $Z(k) = \begin{bmatrix} z(k), y(k), \theta(k) \end{bmatrix}^T$ describes the robot's position along the X-axis, Y-axis, and its orientation angle $\theta(k)$. The model assumes a non-holonomic constraint, meaning the robot's motion is limited to directions dependent on its current orientation $\theta(k)$.

B. Reference Trajectory Generation

Assume that a reference trajectory is generated using a virtual non-holonomic wheeled mobile robot (WMR) as follows:

$$\begin{bmatrix} z_r(k+1) \\ y_r(k+1) \\ \theta_r(k+1) \end{bmatrix} = \begin{bmatrix} \cos(\theta_r(k)) & 0 \\ \sin(\theta_r(k)) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_r(k) \\ \omega_r(k) \end{bmatrix} \tag{7}$$

Let $Z_r(k) = \begin{bmatrix} z_r(k), y_r(k), \theta_r(k) \end{bmatrix}^T$ represent the reference states of the system, where $z_r(k)$ and $y_r(k)$ denote the positions of the robot along the X-axis and Y-axis, respectively, and $\theta_r(k)$ represents the reference yaw angle of the robot at step k.

C. Motion Control

To ensure precise trajectory tracking in practical scenarios, wheeled mobile robots (WMRs) must be controlled by a system capable of efficiently correcting the yaw. A control law that drives the error $e(k) = z(k) - z_r(k)$ to zero, even in the presence of uncertainties, is required for optimal tracking performance. The position error between the reference model and the WMR's coordinates can be expressed as:

$$\begin{bmatrix} z_e(k) \\ y_e(k) \\ \theta_e(k) \end{bmatrix} = \begin{bmatrix} \cos(\theta_e(k)) & \sin(\theta_e(k)) & 0 \\ -\sin(\theta_e(k)) & \cos(\theta_e(k)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

The robot operates at a constant velocity of $v_r(k) = 0.15$ m/s,

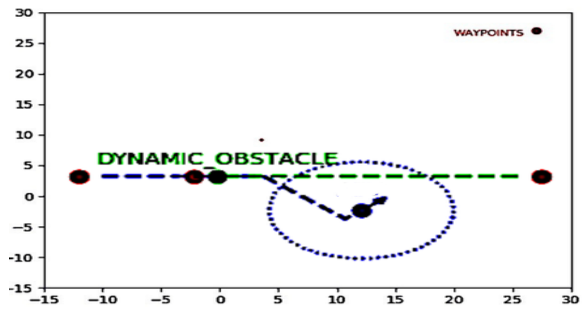

Fig. 3: Simulation with TurtleBot with Dynamic obstacle avoidance, blue dashed line shows barrier function bounds). Full simulation here.

and the angular velocity is constrained by the maximum hardware limit of 0.3 rad/s. Therefore, the error at each step is given by $e(k) = \theta_{\text{ref}}(k) - \theta(k)$.

D. Simulation Setup

In the simulation, a logical control approach was employed to evaluate the OLDBF, utilizing pre-built Robot Operating System (ROS) packages and the TurtleBot3, a differential drive robot as shown in Figs. 3 and 4. The simulation environment is divided into two hemispheres: left and right. The OLDBF establishes a safe perimeter around the robot. Let the minimum distances to obstacles in the left and right hemispheres at

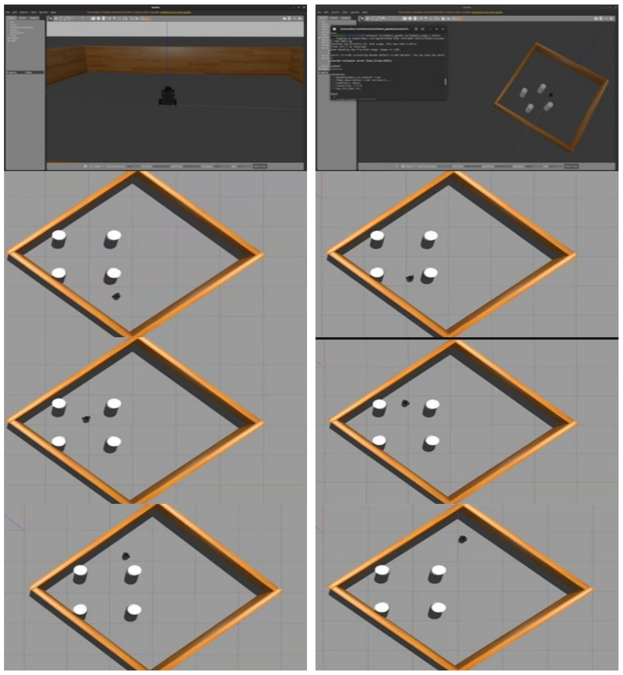

Fig. 4: Snapshots of the dynamic obstacle avoidance of mobile robot in ROS experimental scenario. Link here.

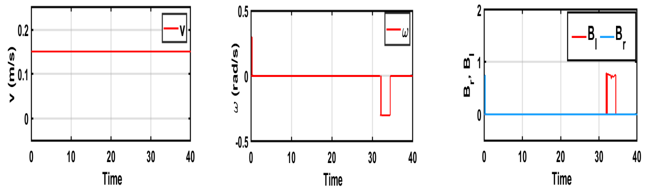

Fig. 5: Right: linear velocity $v(m/s)$; Left: B_l, B_r proposes barrier function where barrier value increases, especially near obstacles, successfully enforcing the safety constraints, Centre: angular velocity $\omega(rad/s)$.

discrete time k be denoted as $d_{l(\min)}(k)$ and $d_{r(\min)}(k)$, respectively. These distances are incorporated into discrete barrier functions $B_l(k)$ and $B_r(k)$.

Define the set of safe time indices as $\mathcal{K}_{\text{safe}} = \{k \in \mathbb{N} \mid B_l(k) > 0 \wedge B_r(k) > 0\}$, where $B_l(k)$ and $B_r(k)$ increase sharply as $d_{l(\min)}(k)$ and $d_{r(\min)}(k)$ approach the safety threshold $d_{\min}$, i.e., $\lim_{d_{l(\min)}(k)\to d_{\min}} B_l(k) \to \infty$ and $\lim_{d_{r(\min)}(k)\to d_{\min}} B_r(k) \to \infty$.

The robot's control input at time k is $u(k) = (v(k), \omega(k))$, where $v(k)$ is linear velocity and $\omega(k)$ is angular velocity. The control law ensures that for all $k \in \mathcal{K}_{\text{safe}}$, $u(k) \in \mathcal{U}_{\text{safe}}(k) = \{(v, \omega) \mid B_l(k) > 0, B_r(k) > 0\}$.

E. Real World Experimental Results

Experimental results are obtained using Quanser's QBot platform as depicted in Figs. 4, 5, and 6. A real-time image of Quanser's QBot platform in the laboratory with dynamic obstacle avoidance as depicted in Figs. 7, 8.

Due to space constraints, a dynamic obstacle velocity control strategy has been implemented, which ensures that for all time indices k, the velocity of any dynamic obstacle $v_o(k)$ satisfies $v_o(k) < v_r(k)$, where $v_r(k)$ is the robot's linear velocity. Define the set of safe time indices as $\mathcal{K}_{\text{safe}} = \{k \in \mathbb{N} \mid B_l(k) > 0 \wedge B_r(k) > 0\}$, where $B_l(k)$ and $B_r(k)$ are the left and right barrier functions, respectively, as shown in Fig. 5. These barrier functions are designed such that $\lim_{d_{l(\min)}(k)\to d_{\min}} B_l(k) \to \infty$, $\lim_{d_{r(\min)}(k)\to d_{\min}} B_r(k) \to \infty$, where $d_{l(\min)}(k)$ and $d_{r(\min)}(k)$ are the minimum distances to obstacles on the left and right hemispheres, and $d_{\min}$ is the safety threshold. The robot's control input at time k is defined as $u(k) = (v(k), \omega(k))$, where $v(k)$ is the linear velocity

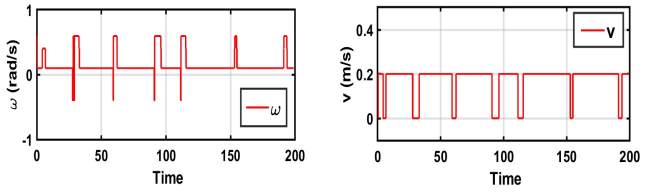

Fig. 6: Left: linear velocity $v(m/s)$; Right: angular velocity $\omega(rad/s)$ in circular trajectory motion. The result incorporates the linear trajectory constraints $v = .15(m/s)$ and $\omega = 0(rad/s)$ with these dynamics.

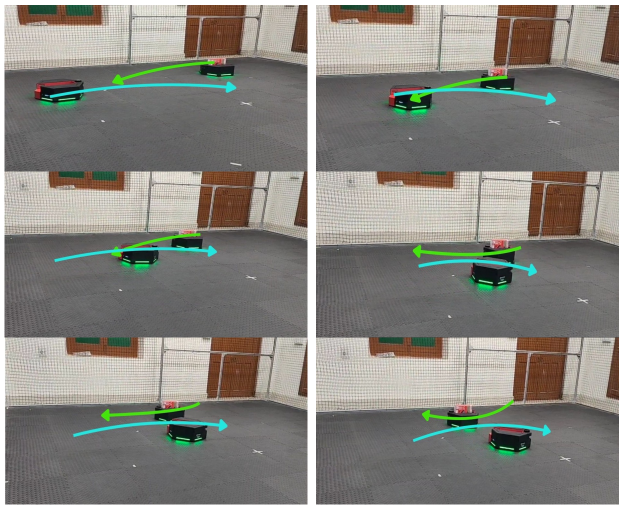

Fig. 7: Snapshots of the dynamic obstacle avoidance of mobile robot in experimental scenario. Link here.

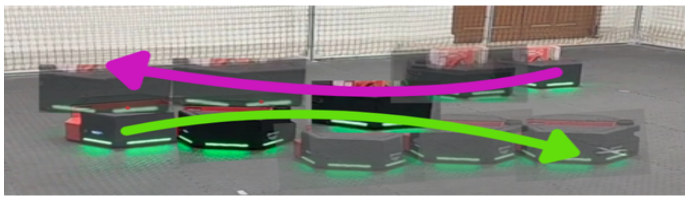

Fig. 8: Typical real-time image of Quanser's QBot platform in the Laboratory with dynamic obstacle avoidance.

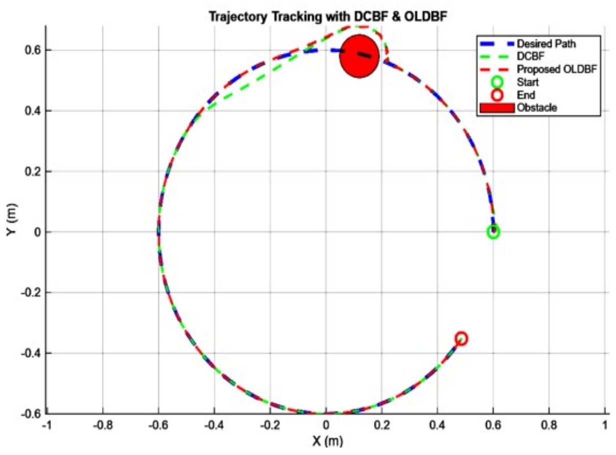

Fig. 9: Circular trajectory tracking with comparison of DCBF and proposed OLDBF

and $\omega(k)$ is the angular velocity. The control law is synthesized such that for all $k \in \mathcal{K}_{\text{safe}}$, $u(k) \in \mathcal{U}_{\text{safe}}(k) = \{(v,\omega) \mid B_l(k) > 0,\ B_r(k) > 0\}$.

As illustrated in Fig. 5, the barrier functions B_l and B_r increase near obstacles, effectively enforcing the safety constraints. In the scenario depicted, the robot is tasked to follow a line trajectory with control input $v = 0.15\,\text{m/s}$ and dynamic obstacles placed along its trajectory. The control input ω dynamically adjusts to ensure obstacle avoidance while maintaining the overall trajectory. As shown in Fig. 6, the robot's behavior is demonstrated for a circular path of radius $0.2\,\text{m}$. The reference linear and angular velocities are set as $v = 0.2\,\text{m/s}$ and $\omega = 0.13\,\text{rad/s}$, respectively, enabling the robot to accurately follow the desired circular trajectory. The angular velocity ω and the barrier functions B_l and B_r effectively enforce safety constraints. Specifically, the barrier values increase with higher velocities, particularly in the vicinity of obstacles, thereby ensuring that the robot avoids unsafe conditions. The real-time image of Quanser's QBot platform in Laboratory with dynamic obstacle avoidance can be observed from Figs. 7, and Fig.8. A comparative evaluation of circular trajectory tracking using the standard DCBF and the proposed OLDBF is presented in Fig. 9, highlighting the differences in safety enforcement and trajectory adherence between the two approaches.

V. Conclusion

We have proposed an online LiDAR-based discrete barrier function (OLDBF) combined with a dynamic obstacle avoidance algorithm (DOAA). The OLDBF has been rigorously tested in both ROS simulations using the TurtleBot and real-world environments with the Quanser QBot platform. The OLDBF effectively enabled the robot to navigate while avoiding both static and dynamic obstacles, confirming its capability to ensure safe and efficient path planning.

References

[1] A. Singletary, S. Kolathaya, and A. D. Ames, "Safety-critical kinematic control of robotic systems," *IEEE Control Systems Letters*, vol. 6, pp. 139–144, 2022.

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

[3] 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.

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

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

[6] J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "Combined speed and steering control in high-speed autonomous ground vehicles for obstacle avoidance using model predictive control," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 10, pp. 8746-8763, Oct. 2017.

[7] E. G. Gilbert and K. T. Tan, "Linear systems with state and control constraints: The theory and application of maximal output admissible sets," *IEEE Trans. Autom. Control*, vol. 36, no. 9, pp. 1008–1020, Sep. 1991.

[8] F. Blanchini, "Set invariance in control," *Automatica*, vol. 35, no. 11, pp. 1747–1767, 1999.

[9] H.K. Khalil, *Nonlinear Systems*, 3rd ed., Prentice Hall, 2002.

[10] C. Lee, D. Chung, J. Kim, and J. Kim, "Nonlinear model predictive control with obstacle avoidance constraints for autonomous navigation in a canal environment," *IEEE/ASME Transactions on Mechatronics*, vol. 29, no. 3, pp. 1985-1996, June 2024.

[11] H. Dong, C. -Y. Weng, C. Guo, H. Yu, and I. -M. Chen, "Real-time avoidance strategy of dynamic obstacles via half model-free detection and tracking with 2D LiDAR for mobile robots," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 4, pp. 2215-2225, Aug. 2021.

[12] P. L. Richard, N. Pouliot, and S. Montambault, "Introduction of a LiDAR-based obstacle detection system on the LineScout power line robot," *2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Besacon, France, 2014, pp. 1734-1740.

[13] E. -J. Jung, J. H. Lee, B. -J. Yi, J. Park, S. Yuta, and S. -T. Noh, "Development of a laser-range-finder-based human tracking and control algorithm for a marathoner service robot," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 6, pp. 1963-1976, Dec. 2014.

[14] M. Thuy and F. P. Leon, "Non-linear shape independent object tracking based on 2D LiDAR data," *Proc. IEEE Intell. Veh. Symp.*, pp. 532-537, 2009.

[15] J. Villa, J. Aaltonen, and K. T. Koskinen, "Path-following with LiDAR-based obstacle avoidance of an unmanned surface vehicle in harbor conditions," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 4, pp. 1812-1820, Aug. 2020.

[16] A. Agrawal and K. Sreenath, "Discrete Control Barrier Functions for Safety-Critical Control of Discrete Systems with Application to Bipedal Robot Navigation," *Robotics: Science and Systems*, 2017.

[17] Y. Xiong, D.-H. Zhai, M. Tavakoli, and Y. Xia, "Discrete-Time Control Barrier Function: High-Order Case and Adaptive Case," *IEEE Transactions on Cybernetics*, vol. 53, no. 5, pp. 3231–3239, May 2023.

[18] V. Freire and M. M. Nicotra, "Systematic Design of Discrete-Time Control Barrier Functions Using Maximal Output Admissible Sets," *IEEE Control Systems Letters*, vol. 7, pp. 1891–1896, 2023.

[19] S. Keyumarsi, M. W. S. Atman and A. Gusrialdi, "LiDAR-Based Online Control Barrier Function Synthesis for Safe Navigation in Unknown Environments," in IEEE Robotics and Automation Letters, vol. 9, no. 2, pp. 1043-1050, Feb. 2024,