Bulletin of Electrical Engineering and Informatics
Vol. 15, No. 2, April 2026, pp. 1033∼1048
ISSN: 2302-9285, DOI: 10.11591/eei.v15i2.11309

Model predictive control with safety barrier enforcement for dynamic obstacle avoidance of mobile robots

Michael Andrianto Gunarso, Tua Agustinus Tamba, Jonathan Chandra
Department of Electrical Engineering, Faculty of Engineering Technology, Parahyangan Catholic University, Bandung, Indonesia

Article Info

Article history:

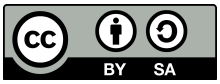
Received Sep 2, 2025
Revised Mar 5, 2026
Accepted Mar 10, 2026

Keywords:

Control barrier function
Mobile robots
Model predictive control
Obstacle avoidance
Safety-critical system

ABSTRACT

This research proposes a model predictive control (MPC) approach with additional barrier function constraint for safe navigation of a differential drive wheeled mobile robot (DDWMR) in the presence of static and dynamic obstacles. The proposed approach uses the kinematic model of DDWMR to initially constructs a stabilizing MPC on the basis of Lyapunov's stability theory. To ensure safe navigation of the DDWMR when obstacles are present, the control barrier function (CBF) concept is subsequently constructed and integrated into the developed MPC framework. The integrated MPC-CBF approach is shown to guarantee both the stability and safety properties of the DDWMR while navigating towards a desired goal position through a workspace filled with obstacles. The good performance of the proposed framework is demonstrated through computer simulations and experimental validation on a Turtlebot3 DDWMR platform. In the real robot experiments, the controller achieved final tracking errors of $[e_x \ e_y] = [0.1286 \ 0.0626]$ m and $e_\theta = 0.021$ rad, while the corresponding simulation errors were $[e_x \ e_y] = [0.0824 \ 0.0698]$ m and $e_\theta = 0.0883$ rad, respectively. These results demonstrate that the developed feedback control method ensures safe, stable, and collision-free motions of the DDWMR.



Corresponding Author:

Tua Agustinus Tamba
Department of Electrical Engineering, Faculty of Engineering Technology, Parahyangan Catholic University
Bandung 40141, West Java, Indonesia
Email: ttamba@unpar.ac.id

1. INTRODUCTION

In recent years, rapid advances in robotics technology have led to significant expansion of robotics applications beyond those commonly used for industrial manufacturing, including the use of mobile robots for applications such as autonomous driving, inspection robots, cleaning robots, and delivery robots [1]. One type of mobile robot in such applications that has received significant attention from researchers and industries is differential drive wheeled mobile robots (DDWMR). The DDWMR can be used to navigate a certain area for different purposes, such as monitoring or actuation purposes, and it moves within the workspace by independently controlling the angular speed of each of its two wheels, which are located on each side of the robot [2]. More specifically, by adjusting the rotational speed and direction of each wheel, the DDWMR can move forward, backward, and/or rotate in its center-of-mass point. To ensure that the DDWMR operates according to the required objectives, extensive research has been conducted in recent years to develop various DDWMR control methods, which range from conventional PID controllers to more advanced controller techniques such as those based on Lyapunov's stability analysis [3] or machine learning and artificial intelligence methods [4].

To achieve safe navigation of DDWMR in dynamic and complex working space/environments, it is particularly crucial to design controllers that not only ensure/guarantee the stability but also preserve the safety properties of the resulting robot motions while performing various desired tasks [5]. Safety can be defined as the condition in which the evolution/trajectory of the system remains within the prescribed safety bounds or within a particularly defined safe set. For mobile robot applications, safety can, for example, be defined in terms of obstacle avoidance task, where the robot must not violate a predetermined safe distance to surrounding objects as it potentially can cause collisions with static/moving obstacles [6]. In recent years, various methods have been proposed to ensure the safety of various robotic applications such as those for the operation of unmanned aerial vehicles [7], quadruped movement [8], multi-robot coordination [9], [10], and robot manipulator task execution [11]. The methods developed in these works include Bayesian optimization algorithm [12], combined A* and dynamic window path planner [13], and artificial potential field (APF) technique [14]. However, most of these developed methods have focused mainly on achieving the stability of the operating robot without specifically taking into consideration the safety aspect of the resulting robot movements. In this regard, more work remains needed to complement the currently developed methods with additional safety-preserving techniques.

One of the methods that has been used to ensure the safety of a system is based on the control barrier function (CBF) concept [15]. The CBF is a relatively recent concept introduced in the control community as a mathematical tool to ensure the safety preservation of a control system. Similarly to Lyapunov's stability theory, the CBF concept determines the conditions to preserve the safety of the trajectories of a system based on the existence of a particular scalar-valued function called CBF which satisfies certain conditions over the considered safe set [16]. Compared to other obstacle avoidance methods, such as APF [14], CBF-based methods aim to determine and guaranty forward invariance of system trajectories by formulating the obstacle avoidance task as a constraint [17]. In this regard, current implementations of CBF-based methods naturally lead to an optimization problem that formulates the requirements of stability and safety properties as constraint functions [18]. From the point of view of real-time implementation, effective optimization-based methods are therefore required to ensure successful implementation of CBF-based methods [19].

This paper proposes the integration of the CBF-based safety preservation approach into the model predictive control (MPC) scheme to construct a dynamic obstacle avoidance method for DDWMR operation. The proposed integration scheme is motivated by the fact that MPC is essentially an optimization-based control design method; therefore, the simultaneous inclusion of both stability and safety properties can be systematically formulated as additional constraints in the optimization formulation [16]. Moreover, the proposed MPC-CBF integration scheme can predict several steps ahead to ensure the safety of the system in a more flexible and less conservative manner [20]. In summary, the main contributions of this research are twofold.

- Firstly, an optimization-based control law is designed to ensure both stable and safe navigation of the DDWMR when required to operate in a workspace filled with random static and/or moving obstacles.
- Secondly, a path planner scheme that generates a predicted motion trajectory to guide the robot's movement towards the goal point is developed to facilitate smooth robot movement between predefined initial and final poses within such a dynamic environment.

In these regards, the proposed controller not only navigates the DDWMR to the desired goal pose, but also prevents collisions between the DDWMR and obstacles using the CBF framework. The satisfactory performance of the proposed safety-preserving control method is shown using computer simulations and then experimentally validated on a real DDWMR platform.

The remainder of this paper is structured as follows. Section 2 describes the research method consisting of the description and mathematical model of Turtlebot3 DDWMR. This section also reviews the basic concept and mathematical formulation of MPC and CBF methods. Section 3 presents the main results of this research on the MPC-CBF integration to ensure robot's ability to avoid both static and dynamic obstacles. This section also presents numerical simulations and experimental results to illustrate the effectiveness of the proposed method. The conclusions and some remarks for potential future work are presented in section 4.

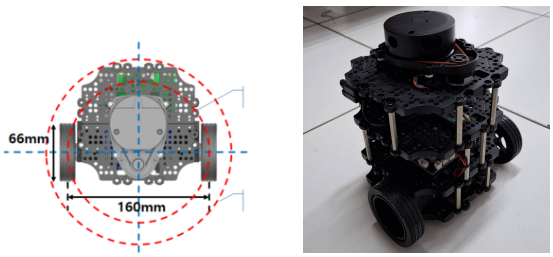
2. METHOD

This section first describes the Turtlebot3 DDMWR platform used for the design and validation of the robot controller developed in this paper. The discrete-time kinematic model of Turtlebot3 DDWMR is also derived and used as the basis for the analysis and design of a suitable control system. The optimization problem

of the MPC design is then formulated to achieve stable navigation of the mobile robot. Finally, integration of the CBF concept into the nominal MPC scheme is proposed to simultaneously achieve both the stability and safety of the DDWMR movement in the presence of static/dynamic obstacles.

2.1. System description

This paper considers the development of navigation and control algorithms for a TurtleBot3 DDWMR platform developed by ROBOTIS [21]. The specific model used in this research is the TurtleBot3 Burger as depicted in Figure 1 with a top view schematic as shown in Figure 1(a) and the physical view as shown in Figure 1(b). The Turtlebot3 main controller is based on a Raspberry Pi 4B single board computer and is equipped with a second microcontroller board that is developed to support the implementation of control algorithm using an OpenCR module in robot operating system (ROS)-based embedded systems. The robot is also equipped with onboard sensors and actuators such as Dynamixel XL430-W250 actuator and the LDS-02 sensor. The general specification of TurtleBot3 Burger is listed in Table 1.



(a) (b)

Figure 1. The Turtlebot3 Burger DDWMR; (a) top view schematic and (b) physical view

Table 1. Specification of TurtleBot3 Burger

No	Parameters	Description
1	Maximum translational velocity	0.22 m/s
2	Maximum rotational velocity	2.84 rad/s (162.72 deg/s)
3	Dimension ($L \times W \times H$)	138 mm × 178 mm × 192 mm
4	Laser distance sensor (LDS)	360°
5	LDS measurement range	160∼8000 mm

The TurtleBot3 can be operated from a personal computer (PC) using the combination of wired and wireless communication connections, as illustrated in Figure 2. The solid lines in this figure indicate wired connections, while the dashed lines represent wireless connections. The PC communicates with TurtleBot3 through a local IP address assigned to the Raspberry Pi that is embedded in the robot. On the other hand, the connection between TurtleBot3 and its components is done using wired connections. A Raspberry Pi 4B serves as the main controller that is connected to both the LiDAR sensor and the OpenCR 1.0 motor controller. The control input generated by the algorithm designed on the PC is sent to the Raspberry Pi and then implemented on each motor on the wheels of the Turtlebot3 DDWMR via the OpenCR 1.0 module of ROS.

2.2. Kinematic model

The motion analysis and control system design in this paper is performed based on the kinematic model of the Turtlebot3 robot platform, which has the characteristics of conventional DDWMR [22]. This model essentially describes the relationship between the pose (position and orientation) of the robot and its linear and angular velocities. Figure 3 shows the schematic of the DDWMR configuration in a planar workspace between two subsequent discrete times t and $t+1$. Based on this figure, the discrete-time kinematic model of the DDWMR can be derived as in (1):

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + \begin{bmatrix} \cos(\theta_k) & 0 \\ \sin(\theta_k) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_k \\ \omega_k \end{bmatrix} \tag{1}$$

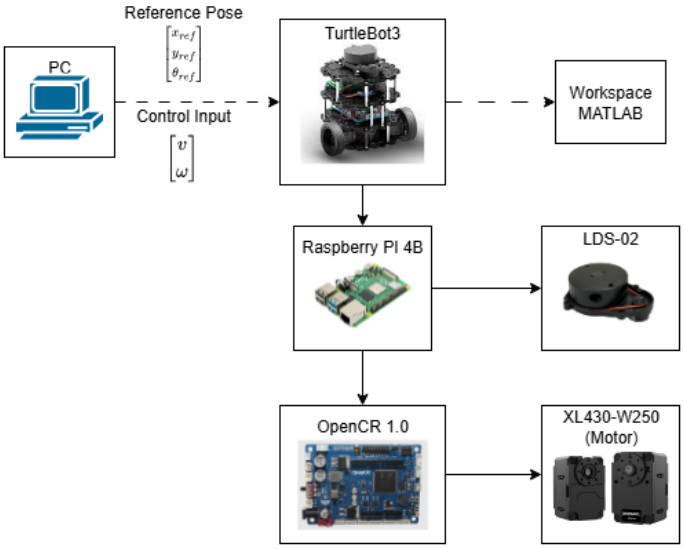


Figure 2. Configuration of the hardware and signal interface of Turtlebot3 Burger

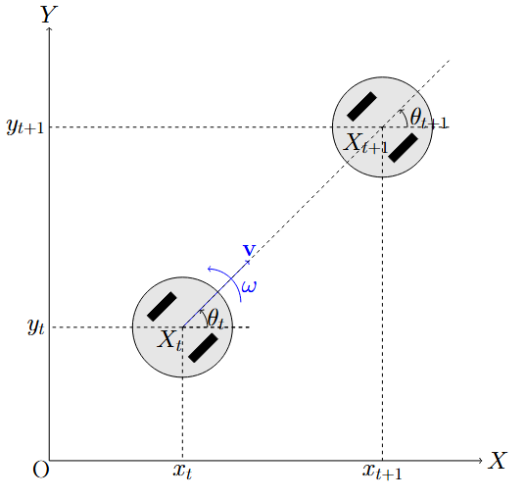


Figure 3. Schematic of kinematics and geometry of Turtlebot3 DDWMR

In (1), $[x_k, y_k, \theta_k]^T$ is the vector of state variables in the kth sample, which consists of robot position $[x_k, y_k]^T$ and orientation θ_k in the planar space, $[v_k, \omega_k]^T$ is the control input vector of the DDWMR consisting of linear velocity v_k and angular velocity ω_k, while $k = 0, 1, 2, \ldots$ is the index of time sampling such that at time t we have $t = k\Delta t$ in which Δt is the sample period. Define $z_k := [z_k^1, z_k^2, z_k^3] \equiv [x_k, y_k, \theta_k]^T$ and $u_k = [v_k, \omega_k]^T$ as vectors of state and control variables, respectively. Then (1) can be rewritten in a compact form (2):

$$z_{k+1} = z_k + f(z_k)u_k \tag{2}$$

where $f(z_k)$ is a matrix function of the state variables defined in (3):

$$f(z_k) = \begin{bmatrix} \cos(z_k^3) & 0 \\ \sin(z_k^3) & 0 \\ 0 & 1 \end{bmatrix} \tag{3}$$

Based on (2), the specific motion problem considered in this paper is that of controlling the DDWMR to perform the point stabilization task from a given initial position and orientation (pose) $z_0 := [z_0^1, z_0^2, z_0^3]^T$ to a desired final pose $z_f := [z_f^1, z_f^2, z_f^3]^T$. To address this problem, this paper uses the MPC-CBF scheme described below.

2.3. Model predictive control formulation

Let $\mathcal{Z} \subseteq \mathbb{R}^n$ be the considered operational space of the DDWMR, and assume that the initial and final poses of the DDWMR belong to certain initial and final subsets $\mathcal{Z}_0 \subset \mathcal{Z}$ and $\mathcal{Z}_f \subset \mathcal{Z}$, respectively, such that $z_0 \in \mathcal{Z}_0$ and $z_f \in \mathcal{Z}_f$. To formulate the stabilization problem of the DDMWR in the MPC framework, we define the following form of stage cost functions as input for the implementation of the MPC design algorithm.

$$p(z_{k+N|k}) = ||z_{k+N|k} - z_{f,t+N|t}||_Q^2 \tag{4}$$

$$q(z_k, u_k) = ||z_k - z_k^f||_Q^2 + ||u_k||_R^2 \tag{5}$$

Note that (4) and (5) essentially define the errors between the current and the desired poses. More specifically, the function $p(\cdot)$ in (4) denotes the difference between the current and the desired poses in each prediction step *N*, while the function $q(\cdot)$ in (5) denotes the deviation between the actual and desired state and the input values. Based on (4) and (5), the MPC design can be formulated as an optimization problem of the following form [23].

$$\min_{u:t+N-1|t} \quad p(z_{t+N|t}) + \textstyle\sum_{k=0}^{N-1} q(z_k, u_k) \tag{6}$$

subject to

$$z_{t+k+1|t} = f(z_{t+k|t}, u_{t+k|t}) \tag{7}$$

$$z_{t|t} = z_t \tag{8}$$

$$z_{t+k+1|t} \in \mathcal{Z} \tag{9}$$

$$u_{t+k|t} \in \mathcal{U} \tag{10}$$

$$z_{t+N|t} \in \mathcal{Z}_f \tag{11}$$

for $k = 0, \ldots, N-1$. The term $p(\cdot)$ in (6) represents the terminal cost, which quantifies the cost of being in a specific state $z_{t+N|t}$ at the end of the prediction horizon N. This terminal cost is utilized to drive the system towards the desired final state. On the other hand, the term $q(\cdot)$ in (6) denotes the stage cost to be evaluated at each time step within the prediction horizon N. The minimization of this stage cost aims to reduce the error and maintains both the control inputs and the state variables to remain within predefined desirable regions. The constraint in (7) captures the kinematic evolution of the system, while (8) sets the values of the state variables at each time t. The constraints (9) and (10) bound both the state and the control variables. Finally, (11) restricts the final predicted value of the state variables to be within the defined terminal set $\mathcal{Z}_f$. In (6)-(11), the time variable t represents the current time step while k represents the prediction horizon index relative to t.

2.4. Control barrier function formulation

The CBF concept is concerned with the inclusion of a safety property in the design of control systems through the enforcement of specific state constraints in addition to those already/normally defined for transient performance or stability requirements [24]. More specifically, consider a control system defined in a set $\mathcal{Z} \subseteq \mathbb{R}^n$ whose state evolution satisfies the following differential equation.

$$\dot{z} = f(z) + g(z)u(z), \quad z(0) := z_0 \tag{12}$$

where $z \in \mathcal{Z}$ is the state variable, u is the control input, while $f(z)$ and $g(z)$ are Lipschitz continuous functions. Assume that one is interested in verifying whether a particular subset $\mathcal{S} \subset \mathcal{Z}$ of the state space $\mathcal{Z}$ is *safe*, in the sense that it is *forward invariant* such that every initial state $z_0 \in \mathcal{S}$ then $z(t) \in \mathcal{S}$ for all time $t \in [0, t_{\max}]$ in

which the solution of (12) exists. Similarly to the idea in Lyapunov's stability concept, such a safety property can be decided on the basis of the existence of a function $h(z)$ which is referred to as the CBF. Given the evolution of the system (12), the CBF $h(z)$ is one that satisfies conditions (13) to (15) [25]:

$$h(z) \geq 0, \quad \forall z \in \mathcal{S} \tag{13}$$

$$\dot{h}(z) \geq 0, \quad \forall z \in \partial\mathcal{S} \tag{14}$$

$$\dot{h}(z) := \frac{\partial h(z)}{\partial z} f(z) + \frac{\partial h(z)}{\partial z} g(z)u \geq 0, \quad \forall z \in \mathcal{I}(\mathcal{S}) \tag{15}$$

where $\partial\mathcal{S}$ and $\mathcal{I}(\mathcal{S})$ denote the boundary and interior of the set $\mathcal{S}$, respectively. In this paper, the concept of CBF is adopted in DDWMR movement as the ability to avoid collisions when encountering obstacles, as illustrated in Figure 4. Specifically, the CBF concept is used to ensure that DDWMR does not enter the unsafe area occupied by the obstacle that is denoted by a region within a circle with a radius of $r_{obs} + r_s$.

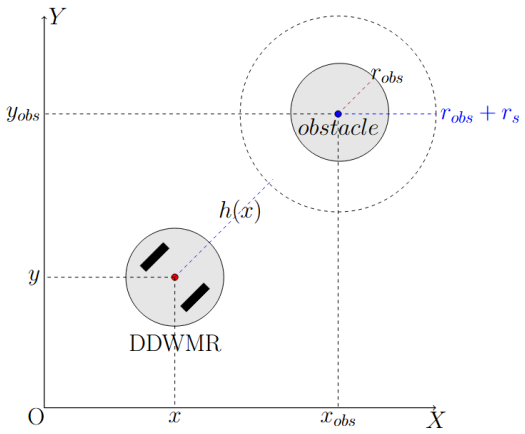


Figure 4. Illustration of obstacle avoidance scheme using CBF

To take into account the CBF concept in the design of a controller with obstacle avoidance feature, the CBF of the form (16) is constructed for the kinematic model of the DDWMR in (2):

$$h(z) = (z_k^1 - z_{obs}^1)^2 + (z_k^2 - z_{obs}^2)^2 - (r_{obs} + r_s)^2 \tag{16}$$

where z is the vector of state variables of DDWMR, z_{obs}^1 and z_{obs}^2 denote the coordinates of the planar position of the obstacle on the X and Y axes, respectively, r_{obs} is the radius of an assumed circle that encloses the obstacle, and r_s is a fixed distance added as a safety margin to the obstacle radius. Taking the time derivative of (16), one may conclude according to (15) that the DDWMR should satisfy (17) over the course of its movement around any obstacle.

$$\dot{h}(z, u) \geq -\gamma h(z) \tag{17}$$

where $0 \leq \gamma \leq 1$ is a class κ function [26]. In order to take the condition (17) into account within the MPC formulation (4) and (5), this paper considers its discrete-time formulation of the form (18):

$$\Delta h(z_k, u_k) \geq -\gamma h(z_k) \tag{18}$$

Since $h(z_k, u_k) := h(z_{k+1}) - h(z_k)$, one may expand (18) to obtain (19):

$$h(z_{k+1}) - h(z_k) + \gamma h(z_k) \geq 0 \tag{19}$$

where $h(z_k)$ is the CBF evaluated at time k and $h(z_{k+1})$ is the CBF evaluated at time $k+1$. Note that inequality (19) essentially enforces the safety (i.e., forward invariance) of the system, as it ensures that if the system state z_k starts in the safe set, it will remain within the safe set in the next time step $(k+1)$ for all $k = 0, \ldots, N$.

3. RESULTS AND DISCUSSION

This section describes the results obtained when implementing the proposed MPC-CBF method for the obstacle avoidance scenario of the DDWMR movement. The main tasks evaluated in the simulation/experiment are to avoid the obstacle while tracking desired waypoints toward the goal pose. The results are presented for three simulation/experiment scenarios, namely: i) integrated MPC-CBF controller, ii) static obstacle avoidance, and iii) dynamic obstacle avoidance scenarios. Furthermore, performance validation of the proposed controller is evaluated on three different platforms, namely using MATLAB simulation, Gazebo simulation, and real experiment using TurtleBot3 Burger platform. In all scenarios, the DDWMR is assumed to have a linear velocity of 0.22 m/s and an angular velocity of 2.84 rad/s. The MPC-CBF controller parameters are set with a safety radius r_{safe} of 0.3, a sampling time $\Delta T = 0.1$, and a prediction horizon of $N = 8$. To evaluate the point stabilization task, the initial pose is set to $z_0 = [0, 0, 0]^T$ and the final pose is set to $z_f = [3, 3, 0]^T$. The video demonstration of the presented results is available at `bit.ly/MPC-CBF-Turtlebot`.

3.1. Model predictive control-control barrier function integration

This evaluation scenario considers the navigation problem of the DDWMR from an initial pose to a desired goal pose while ensuring obstacle avoidance during its movement. This paper addresses this problem by integrating the CBF concept into the MPC design framework. This is done by adding the CBF condition (19) to the set of constraints in the MPC formulation (6)-(11) to obtain the safety-preserving MPC-CBF formulation of the form (20)-(26) [27]:

$$\min_{u:t+N-1|t} \quad p(z_{t+N|t}) + \sum_{k=0}^{N-1} q(z_k, u_k) \tag{20}$$

subject to

$$z_{t+k+1|t} = f(z_{t+k|t}, u_{t+k|t}) \tag{21}$$

$$z_{t|t} = z_t \tag{22}$$

$$z_{t+k|t} \in \mathcal{Z} \tag{23}$$

$$u_{t+k|t} \in \mathcal{U} \tag{24}$$

$$z_{t+N|t} \in \mathcal{Z}_f \tag{25}$$

$$\Delta h(z_{t+k|t}, u_{t+k|t}) + \gamma h(z_{t+k|t}) \geq 0 \tag{26}$$

for $k = 0, \ldots, N-1$. Comparing the MPC formulation in section 2.3. with the one above, it can be seen that (26) is a CBF constraint to guaranty the safety of the robot movement. One algorithmic implementation of the solution method for solving (20)-(26) [27] based on the CasADI solver [28] is summarized in the pseudocode of Algorithm 1.

Algorithm 1. MPC-CBF method with safety guarantee

Define:
  System Model $z_{k+1} = f(z_k, u_k)$
  Initial State = z_0
  Prediction Horizon = N
  Sampling Time = ΔT
  Input Constraint: $\mathcal{U}$
  CBF Constraint: $h(\mathrm{X})$
  Weight Cost: Q, R
Input: Reference Pose (z_f)
while $\|z - z_f\| \geq 0.15$ **do**
  Obtain pose measurement at time $t = z_t$
  Retrieve obstacle position measurement

13: Solve optimization problem using CasADI [28]:

$$
\begin{aligned}
\min_{u:t+N-1|t} \quad & p(\mathrm{z}_{t+N|t}) + \sum_{k=0}^{N-1} q(\mathrm{z}_k, u_k) \\
\text{s.t.} \quad & \mathrm{z}_{t+k+1|t} = f(\mathrm{z}_{t+k|t}, u_{t+k|t}),\ k = 0, \ldots, N-1 \\
& u_{t+k|t} \in \mathcal{U} \\
& \mathrm{z}_{t|t} = \mathrm{z}_t \\
& z_{t+N|t} \in z_f \\
& \Delta h(X_{t+k|t}, u_{t+k|t}) \geq -\gamma h(z_{t+k|t}),\ k = 0, \ldots, N-1
\end{aligned}
$$

14: Apply the first control input u_t to system
15: Wait for ΔT and shift the prediction horizon
16: **end while**

Figure 5 shows the closed-loop block diagram of the proposed MPC-CBF control design scheme. All computation and control operations are executed on an Ubuntu 20.04 operating system equipped with an Intel Core i7-8700 processor (3.2–4.6 GHz) and 32 GB of memory. The optimization problem is solved using the CasADi solver. The controller receives the reference pose as input and is designed to produce the final pose with minimal error. The controller computes the control inputs from the given reference pose to achieve the point stabilization task, which is then executed by the TurtleBot3 DDWMR platform. The sensor feedback is provided by a LiDAR sensor to provide the obstacle positions and by the odometry for the actual robot position. These feedback information are then used to calculate the CBF and position error of the DDWMR.

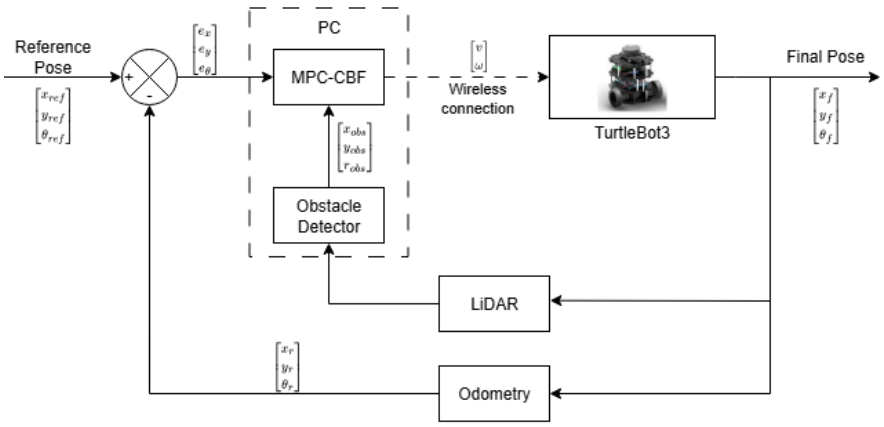


Figure 5. Closed-loop block diagram of the MPC-CBF integration

3.2. Computational cost analysis

To quantify the computational cost, the total execution time and the number of MPC iterations were measured for multiple runs. In a no-obstacle ahead of the robot scenario, the proposed algorithm requires 23.82 seconds over 214 ± 3.61 iterations when the obstacle detector was enabled and 22.4 ± 0.5 seconds over 205 iterations when the detector was disabled. This leads to an average computation time of approximately 0.11 second per iteration and 0.109 second per iteration. In the case of a given obstacle ahead of the robot scenario, the algorithm requires 33.1 ± 14 second over 321 ± 144 iterations, which corresponds to 0.104 seconds/iteration on average. These time measurements indicate that the per-iteration computational cost remains constant (0.1-0.11 second per MPC iteration step), while the increase in total runtime in the obstacle case is primarily due to larger number of required control iterations to avoid an obstacle. It should be noted that these measurements were performed on a PC running ROS Noetic on Ubuntu 20.04 operating system equipped with a stock Intel Core i7 8700 and a 32 GB memory stick.

3.3. Static obstacle scenario

To simulate the obstacle avoidance scenario of the proposed MPC-CBF integration scheme, an assumed static obstacle is placed initially at position $(x = 1.5, y = 1.5)^T$ with a radius of $0.15m$. Figure 6 shows the simulation environments designed in MATLAB [29] software. In this figure, the red and green dots represent the initial and final positions of the DDWMR, respectively, the blue circle represents the static obstacle, and the orange dashed circles represent the safety margin of the designed CBF.

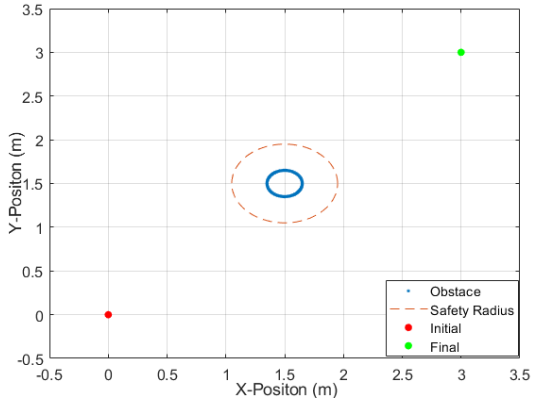
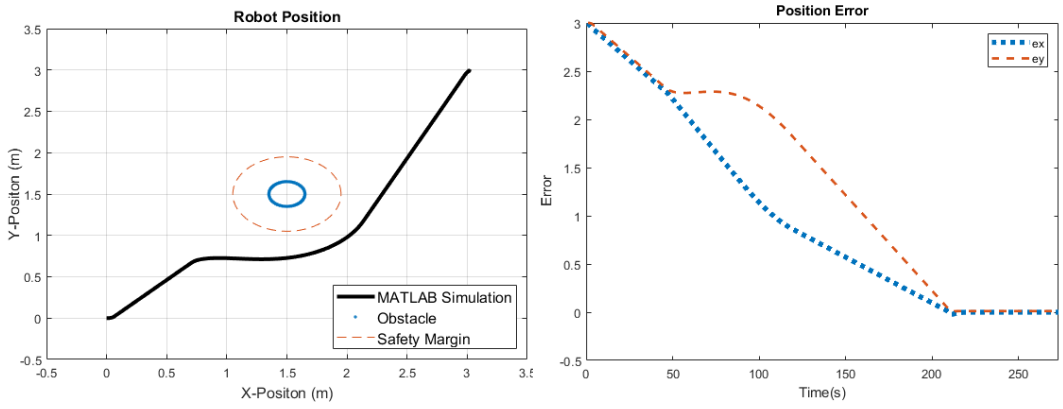


Figure 6. Illustration of the static obstacle scenario

3.3.1. MATLAB simulation of static obstacle scenario

The simulation results obtained using MATLAB are shown in Figure 7. In particular, Figure 7(a) shows the motion of the robot along the X and Y axes, while Figure 7(b) represents the error between the robot pose and the final pose over time. Throughout the trajectory in Figure 7(a), it can be seen that the robot remains inside the admissible set and does not violate the safety margin. In Figure 7(b), the blue dashed line denotes the error along the X axis while the orange line denotes the error along the Y axis. The decreasing error from the robot confirms that the system successfully reaches the point stabilization task with a minimum error.



(a) (b)

Figure 7. Simulated trajectories; (a) position error and (b) with static obstacle

Figure 8 shows the control input signals in the form of linear and angular velocities obtained from the solution of the MPC-CBF optimization problem. The blue line in this figure represents the linear velocity and the orange line represents the angular velocity of the robot. In these plots, it can be seen that the linear velocity remains constant and started to decrease when the robot is near the goal point. For the angular velocity, it can be seen that the higher change of the value occurs when the robot avoids the obstacle. In these regards, it can be concluded that the control input obtained does not exceed the constraint set in the MPC-CBF controller.

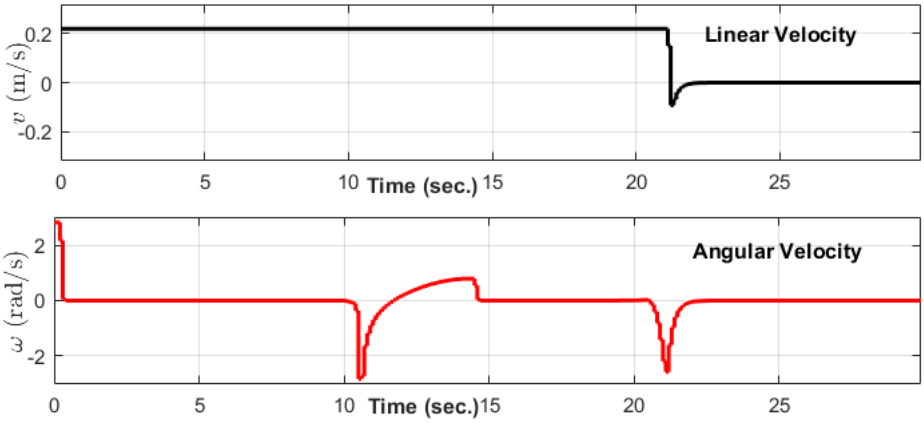
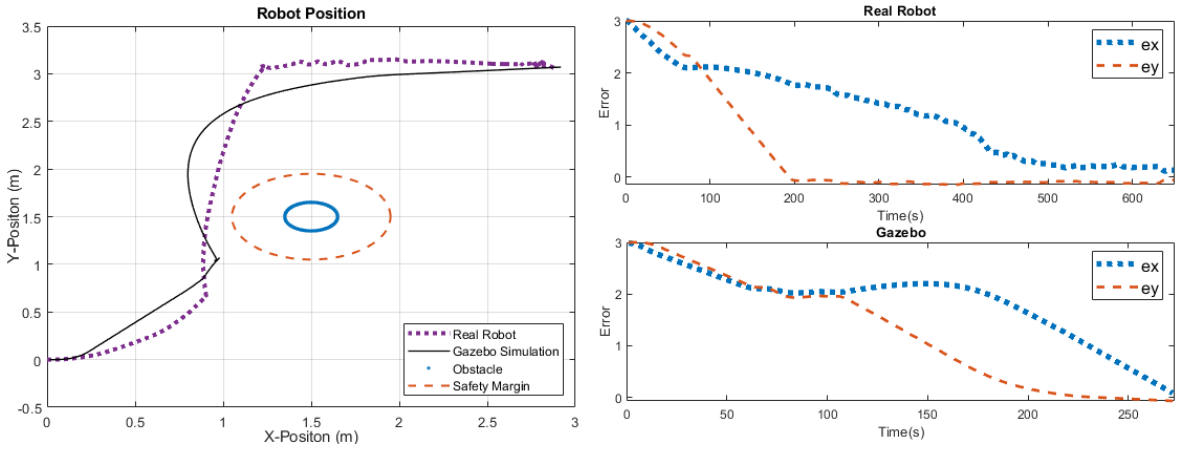


Figure 8. Control inputs consisting linear (top) and angular (bottom) velocities with static obstacle

3.3.2. Static obstacle scenario in Gazebo simulation and real robot implementation

Figure 9 shows the trajectories and position error obtained using the Gazebo simulation and the real robot. In Figure 9(a), the purple dashed lines denote the position of the real robot and the black lines denote the trajectories obtained from Gazebo simulation. It can be seen that the trajectories in Gazebo simulation avoid the obstacle by turning aggressively, while those in the real robot smoothly avoid the obstacle. However, both the Gazebo simulation and real robot experiment results successfully navigate into the final pose while maintaining the safety distance between the robot and the obstacle indicated by the decreasing error. Figure 9(b) also shows the robot tracking errors in Gazebo simulation and real robot experiment. The blue- and orange-dotted lines denote the position error along the X-axis and the Y-axis over time, respectively. It can be seen that the error value from the Gazebo simulation decreases and converges to zero faster than the real robot. This is because in the Gazebo the robot is assumed to have a perfect measurement and without noises for both the odometry and the LiDAR sensors, whereas in the real robot there are many noises affecting the measurement.

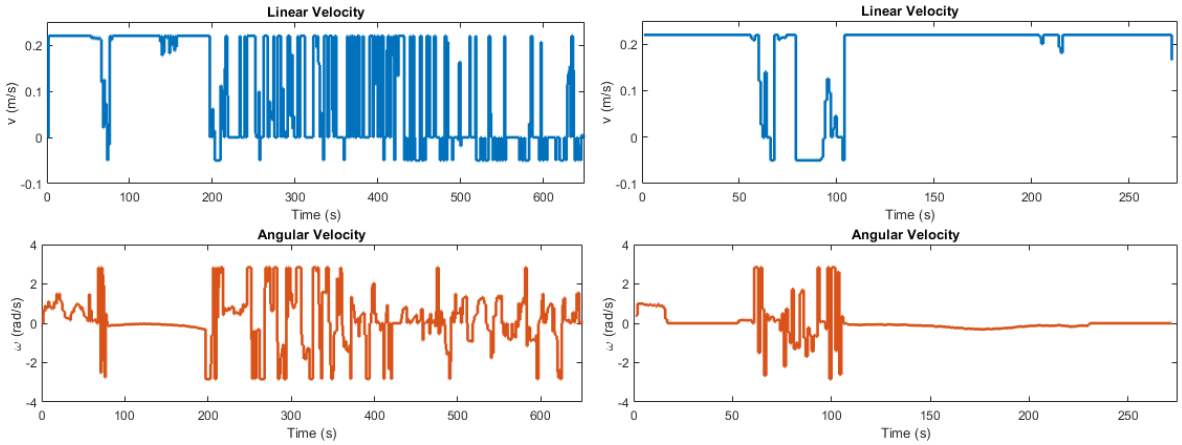


(a) (b)

Figure 9. Simulated and experimental trajectories; (a) position error and (b) with static obstacle

Figure 10 illustrates the robot control inputs that consist of linear and angular velocities. Figure 10(a) shows the robot control input performed using the real robot, while Figure 10(b) shows the control input conducted by the Gazebo simulation. The blue line denotes the linear velocity, and the orange line indicates the angular velocity for both systems. It can be seen that the control input obtained when using the real robot has more oscillation than the value obtained using the Gazebo simulation. The observed oscillations are mainly caused by LiDAR measurement noise that occurs when the robot detects obstacles. Despite the presence of

these oscillations, both methods generated control inputs that remained within the specified limits.



(a) (b)

Figure 10. Control inputs in real robot; (a) Gazebo simulation and (b) with static obstacle

3.4. Dynamic obstacle

To simulate the obstacle avoidance scenario of the proposed MPC-CBF integration scheme, an assumed moving obstacle is placed initially at position $(x = 3, y = 3)^T$ with a velocity of 0.1 m/s and a heading angle toward $-3\pi/4$. To evaluate the effectiveness of the proposed MPC-CBF integration scheme, controller performance is also evaluated using three different methods (Turtlebot3, Gazebo, and MATLAB). Figure 11 shows the simulation environments defined in MATLAB in which the red and green dots denote the initial and final positions, respectively, of the DDWMR. In addition, the blue circle represents the obstacle, the orange dashed circles represent the safety margin of the CBF, and the magenta arrow represents the direction of movement from the obstacle.

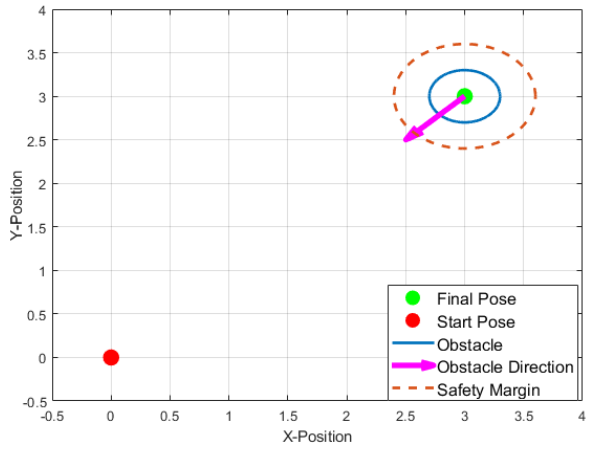
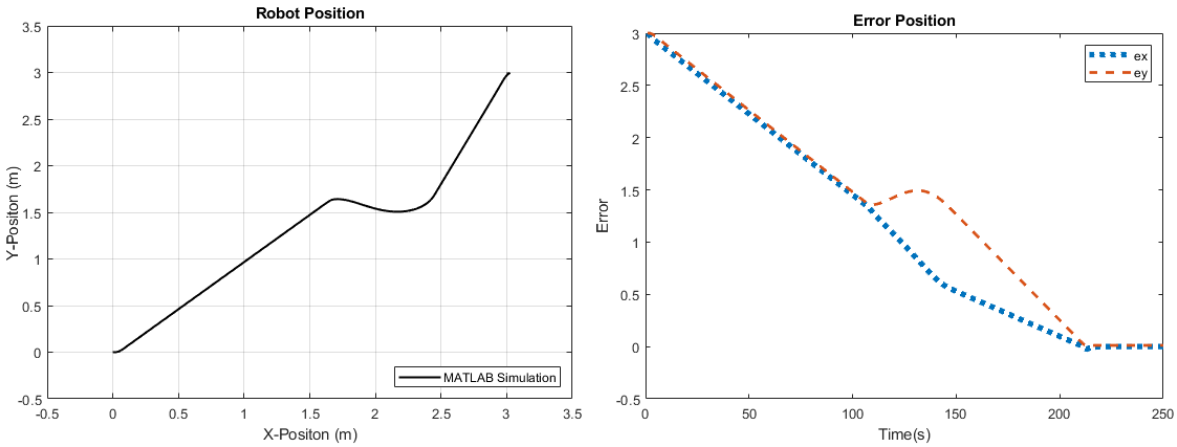


Figure 11. Illustration of dynamic obstacle scenario

3.4.1. MATLAB simulation

Figure 12 shows the MATLAB simulation result of robot trajectories during avoidance of a moving obstacle. The calculated error from the robot over time can be seen in Figure 12(a), where the blue dashed lines represent the position error along the X-axis and the orange dashed lines represent the robot position error along the Y-axis. It can be seen in Figure 12(b) that the deviation of the error in the Y-axis occurs when the robot avoids the moving obstacle. This figure thus demonstrates that the robot successfully performs the point stabilization task while avoiding the obstacle by heading towards $-\theta$ with respect to the global frame.



(a) (b)

Figure 12. Simulated trajectories; (a) position error and (b) with dynamic obstacle

Figure 13 shows the generated control input for both linear and angular velocities, respectively, based on the MPC-CBF controller in the moving obstacle scheme. The blue line represents the linear velocity and the orange line represents the angular velocity of the robot. This figure shows that the linear velocity generated by the solver remains in the maximum limit and started to decrease when the robot was near the goal point. For the angular velocity, it can be seen that the spiking values occurred when the robot changed its orientation. However, the generated control input remains bounded and does not exceed the specified constraint.

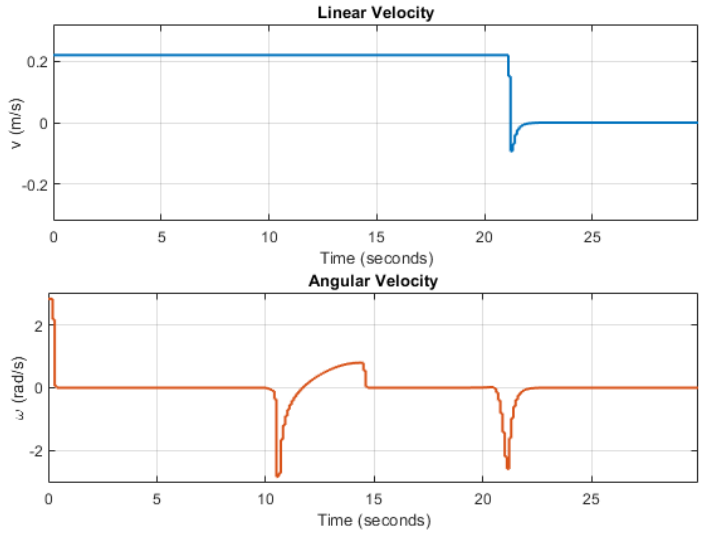
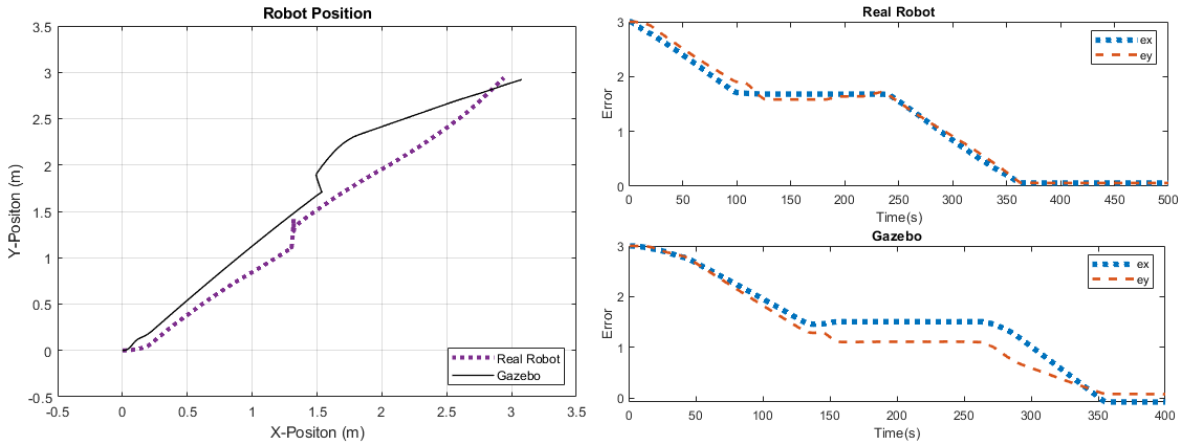


Figure 13. Control inputs consisting linear (top) and angular (bottom) velocities with dynamic obstacle

3.4.2. Gazebo vs real robot

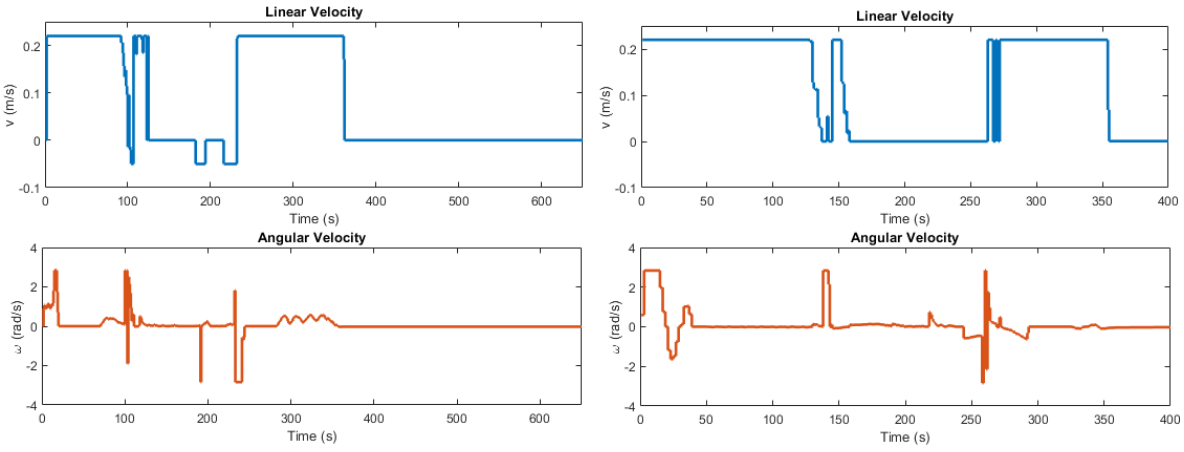
Figure 14 shows the robot trajectories and the error value over time evaluated using the Gazebo simulation and the real robot. The blue dashed lines represent the error position along the X-axis and the orange dashed lines represent the error position along the Y-axis. In Figure 14(a), it can be seen that the trajectories of the Gazebo simulation and the real robot had the same pattern. It can be seen that when the robot approaches the obstacle, the robot will avoid the obstacle by turning into $+\theta$ to avoid the collision. In Figure 14(b), it can be seen that at some interval time, the error obtained did not change. This happened when the robot successfully avoided the obstacle and the robot needed to calculate the controller to continue the point stabilization task.



(a) (b)

Figure 14. Simulated and experimental trajectories; (a) position error and (b) with dynamic obstacle

Figure 15 shows the robot control inputs that consist of linear and angular velocities. The control input generated in the real robot is shown in Figure 15(a), while the control input generated in the Gazebo simulation is shown in Figure 15(b). In these figures, the blue lines represent the robot linear velocity, while the orange lines represent the robot angular velocity. It can be seen that the generated control input for the real robot and in Gazebo simulation have several oscillations compared to those obtained in MATLAB simulation. These oscillations are mainly caused by the LiDAR measurement noise as the robot encountered the obstacle and performed intensive computation to execute the avoidance control task. It also can be seen that the linear velocity in both Gazebo simulation and the real robot at some time intervals goes to zero. This essentially happened after the robot successfully avoided the obstacle and needed to start over the control computation due to the change of its constraint set that is defined by the obstacle.



(a) (b)

Figure 15. Control inputs in real robot; (a) Gazebo simulation and (b) with dynamic obstacle

4. CONCLUSION

This paper has presented a safety-preserving controller design scheme for a DDWMR that incorporates both safety and stability criteria in its formulation. The proposed scheme begins with the development of a nominal point stabilization controller using the MPC scheme. To ensure safety, the CBF scheme is then integrated into the nominal MPC scheme to prevent collisions with static or dynamic obstacles during the robot

movement towards the goal pose. Based on results from numerical simulations and real robot experimental validations, the proposed integrated MPC-CBF control scheme is shown to provide an effective solution for safe and autonomous DDWMR navigation in a workspace filled with static/moving obstacles.

One main limitation of the proposed CBF-based method is the sensitivity of the constructed safe region and control law to noisy sensor measurement in the feedback loop. As a consequence, the implementation of the proposed controller requires a high precision sensor measurement signal. Under a noisy feedback signal, the generated control law may potentially become more conservative and aggressively push the system towards the interior of the safe set. This is because the controller assumes that the system trajectory is already close to the boundary of the safe set, but in reality remains relatively far from the safe boundary set. In this regard, future work will be directed toward exploring robust implementation of the proposed control scheme in the presence of sensor and environmental disturbance/noise signals.

We point out that the MPC-CBF-based method developed in this paper only considers the avoidance of an obstacle moving with a constant speed that is lower than that of the robot and without considering other static/moving obstacles within the workspace. When the robot workspace becomes more complex and the obstacle moves faster than the robot speed, the real time implementation of the MPC-based method generally will be more computationally and mathematically challenging because the robot needs to solve a constrained optimization problem repeatedly over a finite time horizon while accounting for dynamic obstacle motion and system constraints. This is because each additional static/moving obstacle introduces new time-varying collision-avoidance and safety constraints over the prediction horizon, making the number of constraints in the optimization formulation quickly scales up with both the number of obstacles and the length of prediction horizon. In addition, at each control step, the MPC-based method also needs to predict the future trajectory of the robot by simultaneously taking into account both the safe evolution of the robot trajectory and the predicted trajectories of moving obstacles. Finally, additional uncertainty in realistic motion of moving obstacles often requires a robust or stochastic formulation of the safe obstacle avoidance problem, which eventually further increases the computational burden of searching for an appropriate control action. All of these aspects usually must be solved in real time, thereby creating a tight trade-off between the length of prediction horizon, model robustness, safety guaranties, and computational feasibility of the proposed optimization-based method, which needs to be further explored and researched in the future.

FUNDING INFORMATION

The authors gratefully acknowledge the research funding provided for this research by the Institute for Research and Community Service (LPPM) at Parahyangan Catholic University, Indonesia, through the internal research grant scheme with contract number III/LPPM/2025-02/70-P.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Michael Andrianto Gunarso			✓	✓		✓		✓	✓	✓	✓			
Tua Agustinus Tamba	✓	✓			✓	✓	✓		✓	✓		✓	✓	✓
Jonathan Chandra			✓	✓	✓	✓				✓	✓	✓		

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal Analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject Administration
Fu : **Fu**nding Acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this research are available on request from the corresponding author upon reasonable and justifiable reason.

REFERENCES

[1] C. F. Riman and P. E. Abi-Char, "Fuzzy logic control for mobile robot navigation in automated storage," *International Journal of Mechanical Engineering and Robotics Research*, vol. 12, no. 5, pp. 313-323, 2023, doi: 10.18178/ijmerr.12.5.313-323.

[2] A. N. A. Rafai, N. Adzhar, and N. I. Jaini, "A review on path planning and obstacle avoidance algorithms for autonomous mobile robots," *Journal of Robotics*, vol. 2022, no. 1, pp. 1–14, 2022, doi: 10.1155/2022/2538220.

[3] S. I. C. Gulo, T. A. Tamba, and F. Wahab, "Nonlinear path generation and tracking control design for a mobile robot," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 9777-9780, 2023, doi: 10.1016/j.ifacol.2023.10.294.

[4] F. Rashidieranjbar, A. Farhadi, A. Zamanifar, and M. Sorouri, "A systematic literature review: Generative artificial intelligence applications for ground mobile robot navigation," *Computers and Electrical Engineering*, vol. 130, no. 2, 2026, doi: 10.1016/j.compeleceng.2025.110906.

[5] A. M. Ali, C. Shen, and H. A. Hashim, "A linear MPC with control barrier functions for differential drive robots," *IET Control Theory & Applications*, vol. 18, no. 18, pp. 2693–2704, 2024, doi: 10.1049/cth2.12709.

[6] S. Bruggemann, D. Steeves, and M. Krstic, "Simultaneous lane-keeping and obstacle avoidance by combining model predictive control and control barrier functions," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, Cancun, Mexico, 2022, pp. 5285-5290, doi: 10.1109/CDC51059.2022.9992613.

[7] J. Liu *et al.*, "Flexible active safety motion control for robotic obstacle avoidance: a CBF-guided MPC approach," *IEEE Robotics and Automation Letters*, vol. 10, no. 3, pp. 2686–2693, 2025, doi: 10.1109/lra.2025.3534519.

[8] P. J. McConnellogue, M. Van, R. McMullan, R. McElroy, and W. Naeem, "Safety critical control of a quadrupedal robot using MPC-CBF," in *2025 IEEE International Conference on Mechatronics (ICM)*, Wollongong, Australia, 2025, pp. 1-6, doi: 10.1109/ICM62621.2025.10934913.

[9] S. Zhang, O. So, K. Garg, and C. Fan, "GCBF+: a neural graph control barrier function framework for distributed safe multi-agent control," *IEEE Transactions on Robotics*, vol. 41, pp. 1533–1552, 2025, doi: 10.1109/tro.2025.3530348.

[10] C. Jiang and Y. Guo, "Incorporating control barrier functions in distributed model predictive control for multirobot coordinated control," *IEEE Transactions on Control of Network Systems*, vol. 11, no. 1, pp. 547–557, 2024, doi: 10.1109/tcns.2023.3290430.

[11] A. M. Ali, H. A. Hashim, and C. Shen, "MPC based linear equivalence with control barrier functions for VTOL-UAVs," in *2024 American Control Conference (ACC)*, Toronto, ON, Canada, 2024, pp. 1-6, doi: 10.23919/ACC60939.2024.10644310.

[12] F. Berkenkamp, A. Krause, and A. P. Schoellig, "Bayesian optimization with safety constraints: Safe and automatic parameter tuning in robotics," *Machine Learning*, vol. 112, pp. 3713–3747, 2021, doi:10.1007/s10994-021-06019-1.

[13] X. Zhong, J. Tian, H. Hu, and X. Peng, "Hybrid path planning based on SaFe A* algorithm and adaptive window approach for mobile robot in large-scale dynamic environment," *Journal of Intelligent & Robotic Systems*, vol. 99, no. 1, pp. 65–77, 2020, doi: 10.1007/s10846-019-01112-z.

[14] R. Szczepanski, "Safe artificial potential field - novel local path planning algorithm maintaining safe distance from obstacles," *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 4823–4830, 2023, doi: 10.1109/lra.2023.3290819.

[15] 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)*, Naples, Italy, 2019, pp. 3420-3431, doi: 10.23919/ECC.2019.8796030.

[16] J. Schilliger, T. Lew, S. M. Richards, S. Hänggi, M. Pavone, and C. Onder, "Control barrier functions for cyber-physical systems and applications to NMPC," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8623–8630, 2021, doi: 10.1109/lra.2021.3111010.

[17] Q. Nguyen and K. Sreenath, "Robust safety-critical control for dynamic robotics," *IEEE Transactions on Automatic Control*, vol. 67, no. 3, pp. 1073–1088, 2022, doi: 10.1109/tac.2021.3059156.

[18] J. Zeng, Z. Li, and K. Sreenath, "Enhancing feasibility and safety of nonlinear model predictive control with discrete-time control barrier functions," in *2021 60th IEEE Conference on Decision and Control (CDC)*, Austin, TX, USA, 2021, pp. 6137-6144, doi: 10.1109/CDC45484.2021.9683174.

[19] M. Black, G. Fainekos, B. Hoxha, H. Okamoto, and D. Prokhorov, "CBFKit: A control barrier function toolbox for robotics applications," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Abu Dhabi, United Arab Emirates, 2024, pp. 12428-12434, doi: 10.1109/IROS58592.2024.10801431.

[20] H. Abdi, P. Zhao, N. Hovakimyan, and R. Ghabcheloo, "Model predictive control barrier functions: Guaranteed safety with reduced conservatism and shortened horizon," in *2024 American Control Conference (ACC)*, Toronto, ON, Canada, 2024, pp. 1652-1657, doi: 10.23919/ACC60939.2024.10644741.

[21] R. Amsters and P. Slaets, "Turtlebot 3 as a robotics education platform," in *International Conference on Robotics in Education (RiE)*, vol. 10, pp. 170–181, 2019, doi: 10.1007/978-3-030-26945-6_16.

[22] M. D. Akmal, A. R. Al Tahtawi, K. Wijayanto, and F. Wahab, "Trajectory tracking control design for mobile robot using interval type-2 fuzzy logic," *Journal of Fuzzy Systems and Control*, vol. 2, no. 2, pp. 67-73, 2024, doi: 10.59247/jfsc.v2i2.200.

[23] S. Bouzoualegh, E.-H. Guechi, and R. Kelaiaia, "Model predictive control of a differential-drive mobile robot," *Acta Universitatis Sapientiae Electrical and Mechanical Engineering*, vol. 10, no. 1, pp. 20–41, 2018, doi: 10.2478/auseme-2018-0002.

[24] O. Y. Ismael, M. Almaged, and A. I. Abdulla, "Nonlinear model predictive control-based collision avoidance for mobile robot," *Journal of Robotics and Control*, vol. 5, no. 1, pp. 142-151, 2024, doi: 10.18196/jrc.v5i1.20615.

[25] 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, doi: 10.1109/TAC.2016.2638961.

[26] J. Zeng, B. Zhang, and K. Sreenath, "Safety-critical model predictive control with discrete-time control barrier function," in *2021 American Control Conference (ACC)*, New Orleans, LA, USA, 2021, pp. 3882-3889, doi: 10.23919/ACC50511.2021.9483029.

[27] Z. Jian *et al.*, "Dynamic control barrier function-based model predictive control to safety-critical obstacle-avoidance of mobile robot," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, London, United Kingdom, 2023, pp. 3679-

3685, doi: 10.1109/ICRA48891.2023.10160857.
[28] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2018, doi: 10.1007/s12532-018-0139-4.
[29] The MathWorks, Inc., *MATLAB version: 24.1.0.2537033 (R2024a)*. Available: https://www.mathworks.com. (Accessed Aug. 10, 2025).

BIOGRAPHIES OF AUTHORS

Michael Andrianto Gunarso received his B.S. degree in Electrical Engineering from Parahyangan Catholic University in 2025. His research interests include nonlinear control, adaptive control, and safety control system in the field of robotics. He can be contacted at email: michaelandrianto17@gmail.com.

Tua Agustinus Tamba is an Associate Professor at the Department of Electrical Engineering, Parahyangan Catholic University, Indonesia, where he has been a faculty member since 2017. He received both the MSEE and Ph.D. in Electrical Engineering from University of Notre Dame, USA, in 2016, M.Sc. in Mechanical Engineering from Pusan National University, Republic of Korea, in 2009, and B.Eng. in Engineering Physics from Institut Teknologi Bandung, Indonesia, in 2006. His main research interests include dynamical systems, control theory, and optimization with applications in mechatronics, robotics, automation systems, and systems biology. He can be contacted at email: ttamba@unpar.ac.id.

Jonathan Chandra is a Lecturer in the Department of Electrical Engineering at Parahyangan Catholic University (Bandung, Indonesia), where he has been a faculty member since 2024. He received his M.Sc. degree in Mechanical Engineering (Mechatronics & Robotics track) from the University of Groningen, The Netherlands, in 2024 and a B.Eng. in Electrical Engineering with a concentration in Mechatronics from Parahyangan Catholic University, Indonesia, in 2020. His research interests include automation, control systems, and robotics. He can be contacted at email: jonathan.chanda@unpar.ac.id.