International Journal of Robotics and Control Systems
Vol. 4, No. 4, 2024, pp. 2026-2040
ISSN 2775-2658
http://pubs2.ascee.org/index.php/ijrcs

NMPC Based-Trajectory Tracking and Obstacle Avoidance for Mobile Robots

Mohammed Qasim [a,1,*], Abdurahman Basil Ayoub [a,2], Abdulla Ibrahim Abdulla [a,3]

[a] Department of Systems and Control Engineering, Ninevah University, Mosul, 41002, Iraq
[1] mohammed.qasim@uoninevah.edu.iq; [2] abdulrahman.ayoub@uoninevah.edu.iq; [3] abdullah.abdullah@uoninevah.edu.iq
* Corresponding Author

ARTICLE INFO

Article history
Received September 13, 2024
Revised November 06, 2024
Accepted November 16, 2024

Keywords
NMPC;
Mobile Robot;
Trajectory Tracking;
Obstacle Avoidance;
CoppeliaSim

ABSTRACT

This paper presents the design of a Nonlinear Model Predictive Controller (NMPC) for a wheeled Omnidirectional Mobile Robot (OMR) in order to track a desired trajectory in the presence of previously unknown static and dynamic obstacles in the environment around the robot. A laser rangefinder sensor is used to detect the obstacles where each obstacle occupies numerous points of every sensor reading. The points that belong to each obstacle are then clustered together using the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm. This research introduces a novel approach to represent obstacles as multiple rotated ellipses, enabling a more accurate representation of complex obstacle shapes without overestimating their boundaries, thereby allowing the robot to navigate through narrow passages. CoppeliaSim robotic simulator is utilized to create the virtual simulation environment as well as simulate the OMR dynamics. MATLAB with the help of the CasADi toolbox is used for the process of the laser rangefinder readings and the implementation of NMPC, respectively. To validate the effectiveness and robustness of the proposed approach, three simulation scenarios are conducted, each involving distinct trajectories and varying densities of static and/or dynamic obstacles. The proposed control architecture exhibits remarkable performance, enabling the OMR to effectively navigate through narrow passages and avoid multiple static and dynamic obstacles while closely adhering to the desired trajectory.

1. Introduction

Over the past several years, there has been a significant increase in interest in mobile robots, leading to the development of numerous applications such as assistive robots [1]-[3], search and rescue operations [4], [5], and power line inspections [6], [7]. By enhancing robot functionality and performance, researchers tried to replace human effort in dangerous places with mobile robots [8], [9]. Achieving accurate motion control for different kinds of mobile robots on a variety of smooth and rough surfaces in unconventional environments is one of the main problems in this respect, and it has drawn interest from academics all over the world [10]-[13]. In other words, one of the major challenges that designers should take care of is precise trajectory tracking. Offline trajectory planning followed by control system-based trajectory tracking is a common two-layer method for applications in known environments [14]-[16]. However, computationally intensive planning techniques, and the

requirement to repeatedly recalculate trajectories in changing environments, make the two-layer strategy inappropriate in unknown environments [17]-[19].

Model Predictive Control (MPC) is an alternative approach that combines optimum control and optimal trajectory planning into a single optimization problem [20]. By minimizing a performance index and taking into account constraints such as system dynamics and admissible states, control inputs, and obstacles, MPC determines the best control signals and trajectories [21]. MPC has been extensively utilized for trajectory planning and control in a variety of applications in recent years, including industrial automation, aerospace, and automotive engineering [22]. The predictive controller is very adaptable because of its capacity to predict and adjust to changes [23].

A unified MPC-based method, for instance, was presented in [24] for trajectory planning and control, guaranteeing feasibility within traffic limits, both static and dynamic situations constraints. In [25], a realistic and unified trajectory planning and control approach was provided by proposing a formulation for the Dynamic Driving Task (DDT) and an MPC-based solution. An online nonlinear MPC technique for steering multiple autonomous nonholonomic Wheeled Mobile Robots (WMRs) without collisions or deadlocks was reported in [25]. However, terminal constraints and costs were left out of the formulation in order to reduce the computing complexity of the optimization issue. Furthermore, [26] designed a nonlinear MPC (NMPC) technique to stabilize the robot in desired positions and orientations while investigating the motion control of a four-wheeled OMR in dynamic environments. In this work, trajectory planning and control for three-wheeled OMR, are implemented with a single layer NMPC.

A critical aspect of robot control is the ability to maintain trajectory tracking in the presence of obstacles. With the use of path-planning techniques, researchers have managed a mobile robot in the face of obstacles [27], [28]. However, if the path planning approach has a large computing burden, utilizing two control loops (control and path planning) may result in a delayed response to obstacles. Numerous path-planning techniques have been suggested to deal with this problem [29]. Nevertheless, the process of generating a new path can be computationally expensive, and the challenge of avoiding collisions with obstacles remains an active area of research [30]-[33].

The robot needs to be steered quickly in order to avoid collisions with obstacles and maintain its proximity to the reference trajectory. This means that when the robot detects a possible obstacle, it must rapidly deviate from the primary track and then quickly return to the trajectory after overcoming the obstacle. The efficiency and speed of the MPC for directing mobile robots without the requirement for complicated path planning computations or algorithms were examined in [34]-[36]. The controllers demonstrated remarkable precision and efficiency in mimicking overtaking behavior.

Collision-free areas (CFA) can be included in the MPC optimum control problem (OCP) in a variety of ways. The way the CFAs are specified affects the representations' accuracy, complexity, and constraints [37], [38]. More exact descriptions impact computing efficiency by adding complexity and constraints; therefore, there is usually a trade-off between accuracy and efficiency. More straightforward descriptions may be sufficient in sparse surroundings, but they become insufficient in cluttered environments because passageways might be clogged, making optimization impossible as a result of oversimplification [39]. This research proposes a comprehensive NMPC framework that integrates obstacle avoidance as constraints, enabling simultaneous trajectory planning and obstacle avoidance for both static and dynamic obstacles.

In the literature, there are several methods for object detection using the LiDAR sensor [40]-[43]. The best methods make use of point cloud data. A technique is created by [44] for point cloud data based on dynamic environment perception. Reliable dynamic obstacle avoidance is accomplished using this approach by enclosing the dynamic obstacles with a minimal bounding single shape such as an ellipse or circle for each obstacle. However, in crowded environments, single ellipsoids or other forms may overlap with one another when there is a small space between two or more obstacles. This can make it difficult for the robot to navigate between these obstacles, particularly when the path is narrow. This may lead to the loss of recursive feasibility, or the capacity to avoid obstacles that need a substantial departure from the initial path. Although significant progress has been made in trajectory

tracking and obstacle avoidance, existing approaches often face limitations when dealing with complex environments characterized by intricate obstacle geometries and dynamic obstacles. This research aims to address these challenges by proposing a single-layer NMPC algorithm specifically designed for OMRs.

In this paper, a single-layer NMPC algorithm incorporating obstacle avoidance features for trajectory tracking of OMR is proposed. After gathering data from the laser rangefinder, the DBSCAN algorithm is used to cluster the obstacles. Both static and dynamic obstacles in a constrained environment can be managed in this work. In this study, the environmental obstacles are modeled as multiple ellipses, well suited in scenarios including complex-shaped obstacles. The proposed framework's performance is evaluated through simulations conducted using the CoppeliaSim robot simulator together with MATLAB R2023a. This simulation makes use of the Statistics and Machine Learning Toolbox, and the CasADi Toolbox [45] with the Interior Point OPTimizer (IPOPT) solver [46]. The key contributions of this study can be summarized as follows:

- Each obstacle is represented by multiple ellipses rather than a single ellipse or circle. This approach allows for a more accurate estimation of obstacles, even when their surfaces are complex, avoiding the overestimation that can occur with the single ellipse or circle method.
- Obstacles are detected in real-time using a laser rangefinder, as opposed to relying on obstacles with predefined positions and shapes. The proposed obstacle avoidance method is capable of handling both static and dynamic obstacles.
- The NMPC formulation integrates obstacle avoidance as constraints, enabling a unified approach for both trajectory tracking and obstacle avoidance. This integrated approach, validated through real-time experiments in CoppeliaSim and MATLAB, demonstrates the effectiveness of the proposed method.

The remainder of this paper is organized as follows: Section 2 provides a description of the OMR's kinematic model. Section 3 presents the NMPC framework; Section 4 presents simulation results to strengthen the recommended control strategy; and Section 5 concludes the study.

2. OMR Kinematic Model

To analyze the kinematics of an OMR shown in Fig. 1, it is advantageous to employ two reference frames: a stationary inertial (global) frame (I) and a moving frame (R) attached to the robot's center of rotation. By combining the robot's orientation, θ_I, relative to the inertial frame with its position coordinates, x_I and y_I, a set of generalized coordinates can be established to describe the robot's configuration within the workspace. The robot's motion is controlled through the application of translational velocities (u, v), and angular velocity, ω. Consequently, the state vector and the input vector can be defined as $\mathbf{x} = [x_I \quad y_I \quad \theta_I]^T \in \mathbb{R}^3$ and $\mathbf{u} = [u \quad v \quad \omega]^T \in \mathbb{R}^3$, respectively. The kinematic model of the OMR is presented as follows [47], [48]:

$$\dot{\mathrm{x}}(t) = \mathrm{f}(\mathrm{x}(t), \mathrm{u}(t)), \mathrm{x}(0) = \mathrm{x}_0,$$
$$\begin{bmatrix} \dot{x}_I \\ \dot{y}_I \\ \dot{\theta}_I \end{bmatrix} = \begin{bmatrix} \cos\theta_I & -\sin\theta_I & 0 \\ \sin\theta_I & \cos\theta_I & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ \omega \end{bmatrix} \quad (1)$$

The distance from the OMR's center of mass to each wheel is denoted by L. The Omni-wheels are not steerable and positioned at angular offsets of π/3, π, and -π/3 rad relative to the robot's coordinate system. Incorporating the wheel velocities, the lower-level kinematic model relative to the OMR's coordinate frame is formulated as follows:

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \frac{1}{r} \begin{bmatrix} \sqrt{3}/2 & -1/2 & L \\ 0 & 1 & L \\ -\sqrt{3}/2 & -1/2 & L \end{bmatrix} \begin{bmatrix} u \\ v \\ \omega \end{bmatrix} \quad (2)$$

where the radius of each robot wheel is indicated by r , and the vector of wheel angular velocities is defined as $\dot{\mathbf{q}} = [\dot{q}_1 \quad \dot{q}_2 \quad \dot{q}_3]^T \in \mathbb{R}^3$. The OMR features three identical wheels, each equipped with an electric motor and positioned equidistantly at a distance L from the OMR's center of mass. The maximum achievable wheel velocity is determined by the electric motor's specifications. The wheel velocities are subject to the constraint $\dot{q}_{max}$, such that $\forall i : \dot{q}_i \leq \dot{q}_{max}$, where the subscript ($i$ = 1, 2, 3) denotes the wheel index. This constraint arises from the limitations imposed by the coil resistance and counter electromotive force on the motor's current and voltage.

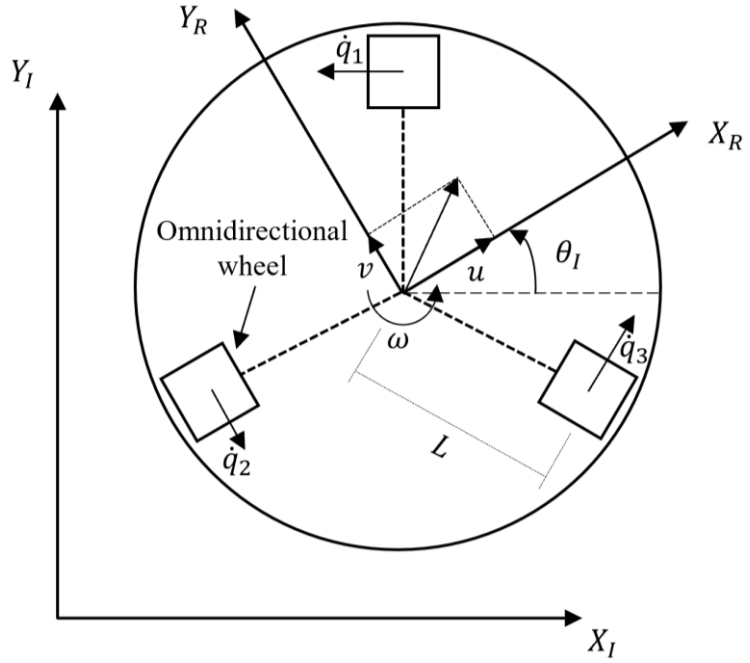

Fig. 1. OMR kinematics diagram

The desired translational velocity of the OMR in the trajectory tracking problem is not a freely selectable parameter but is instead dictated by the time-parameterized reference trajectory. Consequently, the trajectory tracking task necessitates the OMR to accurately follow the desired position and velocity profiles distinct for individually given time instant. Analogous to the OMR model (1), a reference trajectory for OMR can be formulated as follows:

$$\dot{\mathbf{x}}_r = \begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} \cos\theta_r & -\sin\theta_r & 0 \\ \sin\theta_r & \cos\theta_r & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_r \\ v_r \\ \omega_r \end{bmatrix} \tag{3}$$

In the reference trajectory problem, the reference state $\mathbf{x}_r = [x_r \quad y_r \quad \theta_r]^T \in \mathbb{R}^3$, and the associated reference input velocities $\mathbf{u}_r = [u_r \quad v_r \quad \omega_r]^T \in \mathbb{R}^3$ are functions of time t. In order to regulate (1) and monitor (3), an error state $\mathbf{x}_e$ can be characterized by the following:

$$\mathbf{x}_e = \begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} x_r - x_I \\ y_r - y_I \\ \theta_r - \theta_I \end{bmatrix} \tag{4}$$

A control problem can be formulated based on the tracking task by considering the error state and its associated dynamics. By appropriately selecting the error state in a rotated reference frame (4), the model can be simplified. The derivation of the error state model is as follows:

$$\dot{\mathbf{x}}_e = \begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\theta}_e \end{bmatrix} = \begin{bmatrix} \cos\theta_r \cdot u_r - \sin\theta_r \cdot v_r - \cos\theta_I \cdot u + \sin\theta_I \cdot v \\ \sin\theta_r \cdot u_r + \cos\theta_r \cdot v_r - \sin\theta_I \cdot u - \cos\theta_I \cdot v \\ \omega_r - \omega \end{bmatrix} \tag{5}$$

Considering the error model (5), the trajectory tracking objective is to select appropriate robot inputs such that the tracking error $\mathbf{x}_e$ converges to zero. In addition, the control algorithm must ensure safe navigation with the environment. In other words, the controller must consider the map boundaries, previously unknown obstacles, and the OMR's geometry. Furthermore, the saturation limits of the OMR's inputs must be carefully addressed during the controller design process. The NMPC architecture presented in the subsequent section effectively tackles these challenges.

3. Proposed Controller

3.1. NMPC Theory for Trajectory Tracking

Nonlinear Model Predictive Control (NMPC) has been employed as a promising optimal control strategy for control challenges due to its ability to effectively manage system constraints and incorporate future knowledge into the controller design [49]. For a typical nonlinear system, represented by the following differential equation:

$$\dot{\mathbf{x}}_e(t) = \mathbf{f}\big(\mathbf{x}_e(t), \mathbf{u}_e(t)\big), \text{ subject to } \mathbf{u}_e(t) \in U,\ \mathbf{x}_e(t) \in X,\ \forall t \geq 0, \tag{6}$$

The sets X and U represent the feasible states and inputs, respectively. The error state vector $\mathbf{x}_e(t)$ and the error input vector $\mathbf{u}_e(t)$ are subsets of $\mathbb{R}^3$. The objective of tracking control is to devise an appropriate control strategy that will steer the system (6) toward its steady-state equilibrium point ($\mathbf{x}_e(t) = \mathbf{x}_r(t) - \mathbf{x}(t) = 0$ and $\mathbf{u}_e(t) = \mathbf{u}_r(t) - \mathbf{u}(t) = 0$). The core principle of NMPC entails the following sequential steps:

- Predict the system's evolution over a specified time horizon.
- The optimal input at each time step t, denoted by $\bar{\mathbf{u}}_e(\cdot)$, is determined by the prediction horizon T_p. The objective is to minimize the value of the given objective function within the time interval $[t,\ t + T_p] \rightarrow U$. The bar represents future values that are anticipated but differ from actual values.

$$\begin{gathered} \min_{\mathbf{u}_e}\ J\big(t, \mathbf{x}_e(t), \mathbf{u}_e(t)\big) \\ \text{Subject to: } \dot{\mathbf{x}}_e\,(\tau) = f\big(\mathbf{x}_e(\tau), \mathbf{u}_e(\tau)\big) \\ \mathbf{x}_e(\tau) \in X,\ \ (\tau \in \big[\,t, t + T_p\big]) \\ \mathbf{u}_e(\tau) \in U,\ \ (\tau \in \big[\,t + T_c, t + T_p\big]) \end{gathered} \tag{7}$$

here T_c represents the control horizon, with $T_c \leq T_p$.

- Postulate that the current input is equal to the initial optimal input value, $\bar{\mathbf{u}}_e(t)$.

To ensure that the tracking errors converge to zero, the following cost function is minimized over a finite time interval $[t,\ t + T_p]$:

$$J(t, \mathbf{x}_e(t), \mathbf{u}_e(t)) = \int_t^{t+T_p} (\mathbf{x}_e^T(\tau)\mathbf{Q}\mathbf{x}_e(\tau) + \mathbf{u}_e^T(\tau)\mathbf{R}\mathbf{u}_e(\tau))d\tau \tag{8}$$

where **R** and **Q** denote positive definite symmetric weight matrices. Physical limits imposed as control and system states constraints are often necessary for a realistic system to function. Admissible states and control set as established by the next sets of limitations. The OMR input saturation limits and the box constraint for the map bounds are supplied by equation (9).

The OMR input saturation limit is represented by (u_{min}, u_{max}), (v_{min}, v_{max}) for translational velocities, and (ω_{min}, ω_{max}) for rotational velocities. The sets (x_{min}, x_{max}) in the x-axis and (y_{min}, y_{max}) in the y -axis represents the map borders.

$$\begin{aligned} x_{min} &\leq x_I \leq x_{max} \\ y_{min} &\leq y_I \leq y_{max} \\ u_{min} &\leq u \leq u_{max} \\ v_{min} &\leq v \leq v_{max} \\ \omega_{min} &\leq \omega \leq \omega_{max} \end{aligned} \tag{9}$$

Studies have indicated that the implementation of terminal equality constraints can ensure the stability of NMPC [50], [51]. As detailed in [52], the stability can be established by satisfying the following two fundamental assumptions (a and b):

a. A reference state vector, denoted as $\mathbf{x}_r$, exists within the state space set X of the error state vector $\mathbf{x}_e \in \mathbb{R}^3$. This $\mathbf{x}_r$ represents the desired equilibrium point for the system. To attain this equilibrium, a suitable control input, $\mathbf{u}_r$, must be selected from the admissible control set U.

b. After running the cost function, $J: X \times U \rightarrow \mathbb{R}_0^+$, there is an action $\mathbf{u}_r$ within a feasible control set U that minimizes the cost function, resulting in $\mathbf{f}(\mathbf{x}_r, \mathbf{u}_r)$ equaling zero.

Under Assumption I, the validity of this assumption can be assessed by examining system (5). Given the quadratic nature of the cost function J as defined in equation (8), the second assumption (II) is likewise satisfied. Given that the data for results has been obtained from equation (8) and the desired controller exhibits stability, stability is guaranteed.

3.2. Obstacle Avoidance Algorithm

In order to prevent accidents, path-planning algorithms are frequently utilized without the assistance of controllers. However, this strategy could result in lower reaction times and higher computing expenses, which raises the possibility of accidents. The obstacle avoidance constraints can be used for the reference robot in the model predictive controller to solve this problem. By doing this, predictability can be utilized to make sure the robot stays on the correct trajectory and doesn't run into any obstacles. A reference robot with obstacle avoidance constraints is implemented in order to overcome this issue. In this sense, predictability combined with model predictive control can help robots navigate obstacles safely.

As one of the control performance measures, this work utilizes the obstacle avoidance method as a constraint in the cost function. The laser rangefinder sensor and DBSCAN algorithm are used to characterize the obstacles as elliptical shapes at execution time, which makes it appropriate for narrow passageways. Each obstacle which consists of many points (n_p) is divided into (m) groups where each group contains (k) points. Each group is represented by a rotated ellipse with the following ellipse variables: $(xobs_{ij}, yobs_{ij})$ are the coordinates of each ellipse center, where $xobs_{ij} = mean(x_{ij1}, x_{ij2}, \dots x_{ijk})$, $yobs_{ij} = mean(y_{ij1}, y_{ij2}, \dots y_{ijk})$, $i = 1, 2, \dots n, n,$ is the number of obstacles, $j = 1, 2, \dots m, m,$ denotes the number of ellipses for each obstacle; the semi-major axis $a_{ij} = \sqrt{((x_{ij1} - x_{ijk})^2 + (y_{ij1} - y_{ijk})^2}$; the semi-minor axis $b_{ij} = 0.01$ (a small number because the sensor only detects the obstacle surface); and the rotation of the ellipse is denoted by an angle $\psi_{ij} = atan2(y_{ij1} - y_{ijk}, x_{ij1} - x_{ijk})$. Fig. 2 demonstrates how each obstacle in the robot's environment is enveloped by multiple ellipses.

Given the assumption of prior obstacle ignorance by the controller, a detection mechanism has been devised to identify obstacles within a radius of (M) meters. This detection is based on the system's global position and the available environmental map. As a consequence, obstacle avoidance can only be implemented for objects situated within the sensor's detection range.

The robot radius, the distance between the obstacles and the robot, and other environmental parameters must all be taken into account in order to assist the OMR's safe navigation over the map while avoiding obstacles. In order to avoid both dynamic and static obstacles, the following environmental constraint is taken into consideration:

$$\forall i,j: \frac{\left((x-xobs_{ij})\cos(\psi_{ij})+(y-yobs_{ij})\sin(\psi_{ij})\right)^2}{(a_{ij}+R_b+\beta)^2}+\frac{\left((x-xobs_{ij})\sin(\psi_{ij})-(y-yobs_{ij})\cos(\psi_{ij})\right)^2}{(b_{ij}+R_b+\beta)^2}\geq 1 \quad (10)$$

where R_b is the OMR's radius, and β is the safety margin that can be provided to improve the motion OMR's safe navigation while it is inside a non-obstructed area where a larger β indicates safer navigation since it overcomes the distance reading error. However, excessively large values of β can hinder the robot's ability to navigate through narrow passages due to the creation of unnecessarily large safety margin.

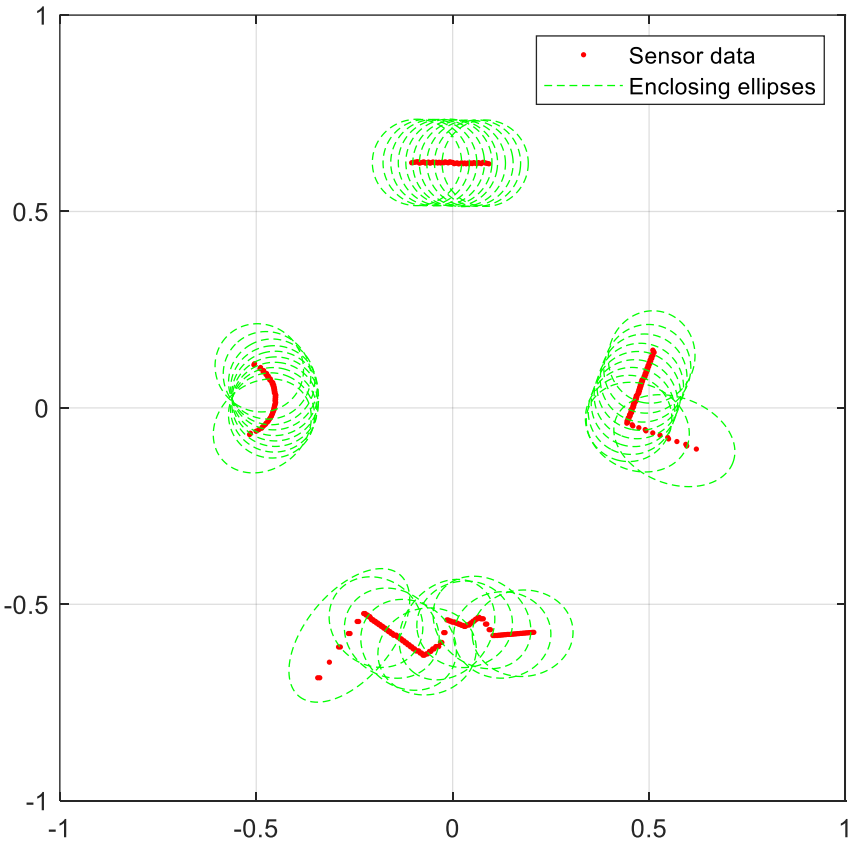

Fig. 2. Obstacle detection and enclosing by multiple ellipses

4. Results and Discussion

The OMR dynamics are simulated using CoppeliaSim simulator, a highly capable robotic simulation tool that facilitates the simulation of both rigid and soft body dynamics. One of the key strengths of CoppeliaSim is its compatibility with various programming languages, including MATLAB, C++, and Python. Moreover, it includes a wide range of built-in tools that greatly enhance and speed up the robot simulation process. It also allows for real-time interaction with the virtual environment.

In each simulation scenario, OMR is presumed to have knowledge of its pose (x_I, y_I, θ_I) in each simulation step. It is equipped with a laser rangefinder capable of detecting obstacles within a 3-meter radius and a 360° field of view, with a measurement accuracy of ±2 centimeters. Data transmission between MATLAB and CoppeliaSim is performed using ZeroMQ remote APIs. MATLAB is chosen since it facilitates the laser range finder data processing and the solution of the optimal control problem. MATLAB uses the DBSCAN algorithm to convert the laser rangefinder raw data into clusters where each cluster belongs to an obstacle in the environment around the robot. DBSCAN algorithm takes two inputs beside the data to cluster which are the minimum number of neighbor points required to form a cluster (minpts) and the minimum distance between two points to

be considered in the same cluster (epsilon). CasADi toolbox in MATLAB is utilized for solving the optimal control problem and system states integration. This is accomplished using multiple shooting method and IPOPT solver. The IPOPT acceptable convergence tolerance is set to 10^{-8} and the maximum number of iterations is selected to be 2000. For system states integration, the Runge-Kutta 4th (RK4) order method is used.

In each simulation scenario, the reference angle θ_r is kept to zero and the DBSCAN parameters are chosen as 5 and 0.1 for minpts and epsilon, respectively. This implies that obstacles represented by fewer than five points are either too distant from the robot or are likely noise artifacts. Additionally, if the distance between two consecutive points exceeds 0.1 meters, they are considered to belong to separate obstacles, which aligns with reasonable assumptions. The controller saturation limits and map margins are chosen as follows

$$\begin{gathered} -0.5 \leq v_x \leq 0.5\ (m/sec) \\ -0.5 \leq v_y \leq 0.5\ (m/sec) \\ -\pi/4 \leq \omega \leq \pi/4\ (rad/sec) \\ -2.5 \leq x \leq 2.5\ (m) \\ -2.5 \leq y \leq 2.5\ (m) \end{gathered}$$

The radii of the robot wheels in (2) and the robot diameter are designed to be $r = 3.5\ mm$ and $R_b = 10$ cm. The safety margin is selected as $\beta = 10$ cm. The NMPC parameters are chosen as follows: the total number of horizon steps is set to $\mathrm{N} = 5$ steps and the time step is selected as $\Delta T = 0.05$ sec, yielding a prediction horizon length of $T_p = 0.25$. The weight matrices in (8) are selected as $Q = diag(10, 10, 10), R = diag(0.1, 0.1, 0.1)$. These parameters and weights are empirically determined to optimize trajectory tracking and obstacle avoidance performance. A potential avenue for future research involves the utilization of optimization algorithms to systematically determine these weights.

The first scenario is presented in Fig. 3 and Fig. 4 in which the OMR follows a figure-8 desired trajectory described by the following.

$$\begin{aligned} x_r &= \cos(wt) \\ y_r &= 0.5\sin(2wt) \end{aligned} \tag{11}$$

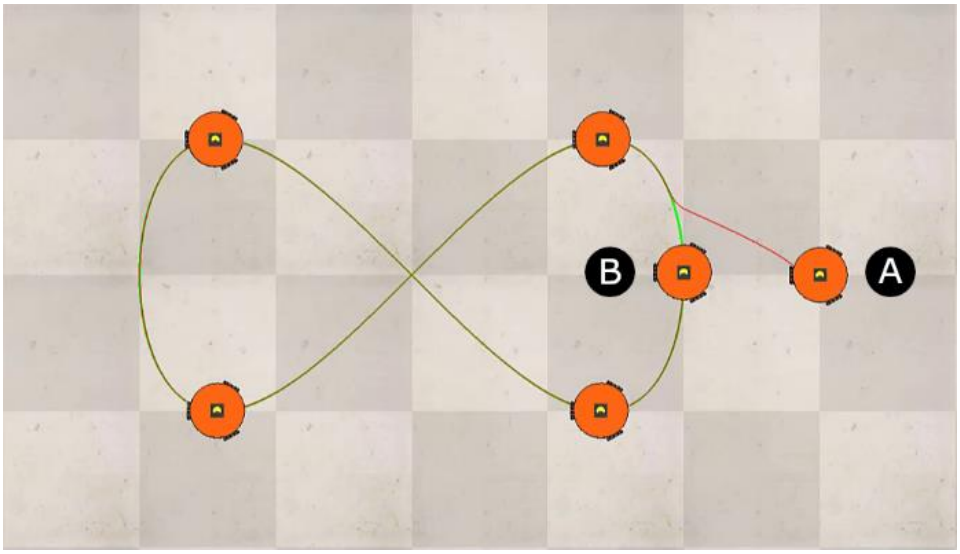

Fig. 3. The OMR follows a figure-8 trajectory

where $w = 0.628$, selected such that the trajectory takes about 10 sec when starting at point B and returning to the same point. The OMR faces no obstacle while following the figure-8 trajectory. It starts at point A and then tracks the desired trajectory with a very small tracking error (see Fig. 4) to

reach point B. In both Fig. 3 and Fig. 4, the desired trajectory is represented by the green curve while the OMR actual trajectory is represented by the red curve. Fig. 4 (right) shows the control action vector components $\mathbf{u} = [u \quad v \quad \omega]$ which exhibit smooth curves and no aggressive actions honoring the provided input saturation limits. While traditional performance metrics such as Integral Square Error (ISE) and Mean Square Error (MSE) are commonly used, they may not be entirely suitable in this context. Given the significant potential for deviations from the planned trajectory when encountering obstacles, these metrics might not accurately reflect the controller's performance.

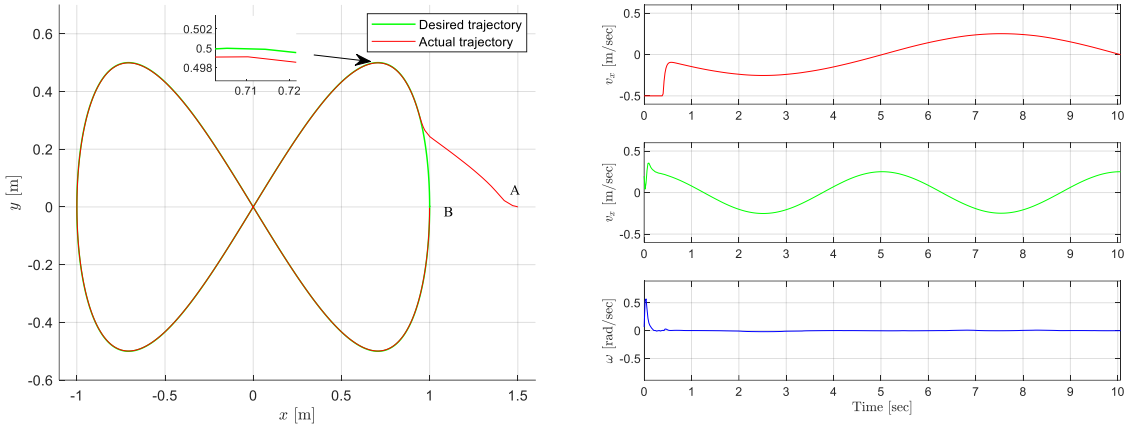

Fig. 4. The OMR follows a figure-8 trajectory (left); control action vector components (right)

In the second scenario, shown in Fig. 5 and Fig. 6, the OMR follows a circular trajectory given by the following

$$x_r = \cos(wt)$$
$$y_r = \sin(wt) \tag{12}$$

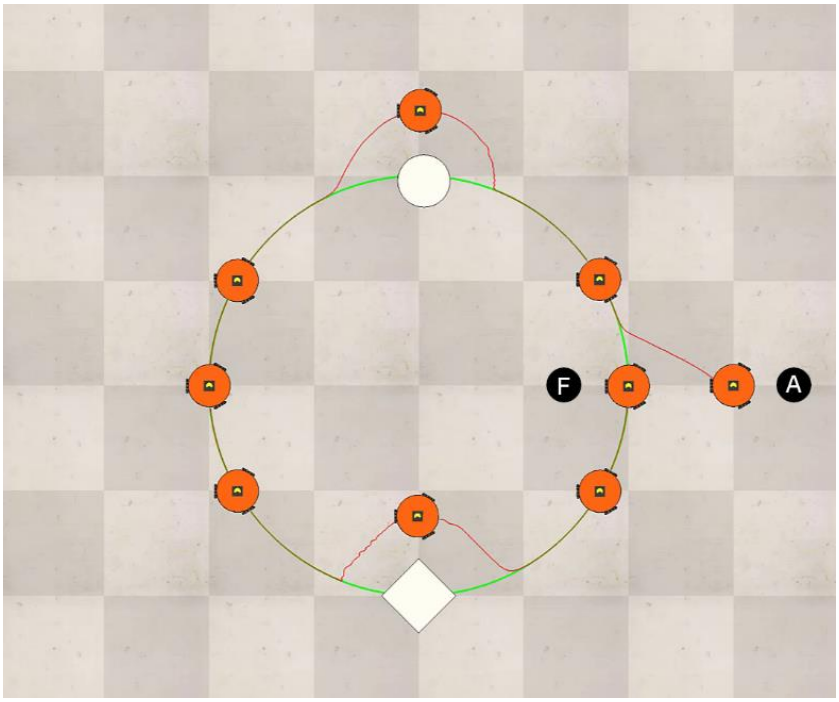

Fig. 5. The OMR tracks a circular trajectory in the presence of static obstacles

where $w = 0.628$, selected such that the trajectory takes about 10 sec when starting at point B and returning to the same point. The OMR starts at point A and moves to follow the desired trajectory

(green curve in Fig. 5 and Fig. 6 while the red curve represents the actual robot trajectory). It faces a static obstacle at point B, moves around the obstacle, and returns to the desired trajectory at point C until it encounters another obstacle at point D, overcomes it, and follows back the desired trajectory at point E to reach point F. The control vector $\mathbf{u} = [u \quad v \quad \omega]$ applied to OMR is shown in Fig. 6 (right) which shows aggressive control action only when facing obstacles.

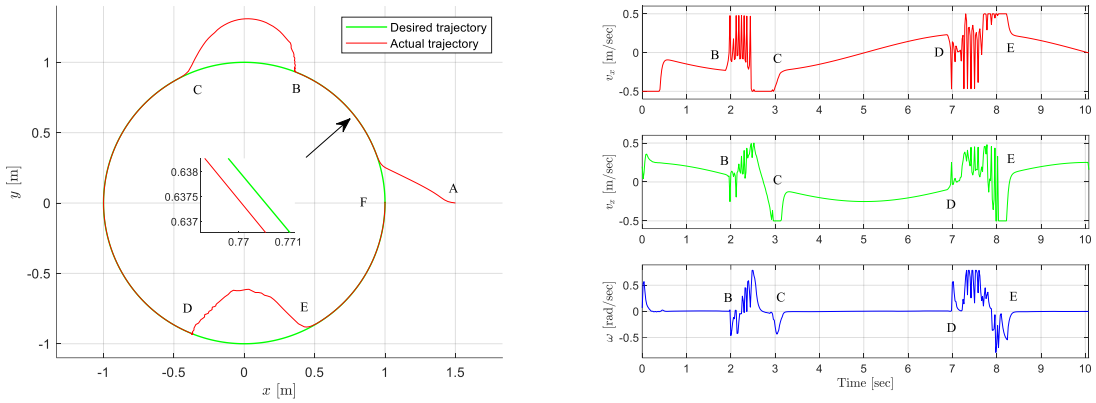

Fig. 6. The OMR follows a circular trajectory (left); control action vector components (right)

The third scenario is more realistic and challenging in which the environment is cluttered with static and dynamic obstacles as shown in Fig. 7 and Fig. 8. This scenario occurs when the OMR is required to move from point A to point K and it has a map of the environment. However, the map may change if the position of static obstacles is altered or by adding new static obstacles to the environment. Additionally, the map may change if there are moving obstacles in the environment such as pedestrians. The environment map the robot possesses does not include the extra two static and two dynamic obstacles (red and blue obstacles in Fig. 7, respectively.) To move from point A to point K, the robot should first plan a path using the environment map it owns and then it should avoid any additional unmapped obstacles in its path. The path is generated using the Open Motion Planning Library (OMPL). There is a built-in plugin in coppeliaSim that wraps the OMPL and it can be used directly inside the coppeliSim. Bidirectional Transition-based Rapidly-exploring Random Tree (BiTRRT) algorithm, available in OMPL, is utilized to plan the path from point A to point K. The generated path does not take into account the red and blue obstacles. The generated path is converted to a trajectory such that it takes about 39 sec to move from point A to point K and provided to the proposed NMPC tracking controller and collision avoidance scheme.

It can be observed that when the robot moves from point A to point B (see Fig. 8), it adheres precisely to the desired trajectory (the obstacle avoidance algorithm is not triggered). At point B, the robot is about to enter the opening of 0.5 m in the wall and the wall is about to enter the safety margin β which triggers the obstacle avoidance algorithm. This causes a small oscillation in the OMR trajectory. The same happens at point C. At point D, the OMR encounters the unmapped static obstacle (the red one). It successfully bypasses it and returns to the desired trajectory at point E. At point F, the OMR faces the first moving obstacle (with a speed of 0.25 m/s) which moves back and forth in a straight line between positions 1 and 2 (see Fig. 7.) The robot initially halts and oscillates in its current position, unable to find a feasible solution to follow the desired trajectory. Once the obstacle reaches position 2, the OMR moves towards point K. However, the moving obstacle is now moving to position 1 and consequently, the OMR moves away from it causing it to deviate from the desired trajectory. Once the moving obstacle is away from OMR, it returns to the desired trajectory at point G, faces the second static unmapped obstacle, avoids it and reaches point H on the desired trajectory. At point I, the OMR faces the second dynamic obstacle (with a speed of 0.32 m/s) which moves back and forth in a circular motion along positions 3, 4 and 5 (the obstacle at position 5 is not shown to provide a clear view of the scene, see Fig. 7.) It moves away from the obstacle and heads

back to the trajectory as it is leading the desired trajectory this time. The OMR finally reaches the desired point K without collision. Fig 8 (right) presents the control vector $\mathbf{u} = [u \quad v \quad \omega]$. It can be noticed that every time an obstacle is about to enter the safety margin β, the control vector components highly oscillate to ensure collision-free navigation. In this scenario, it is assumed that the moving obstacles speed is less than the robot speed for safe navigation. This represents a limitation of the proposed approach when dealing with dynamic obstacles. Another limitation is the increased computational burden associated with representing each obstacle using multiple ellipses. A potential solution involves the implementation of adaptive mechanisms to determine the optimal number of ellipses for each obstacle.

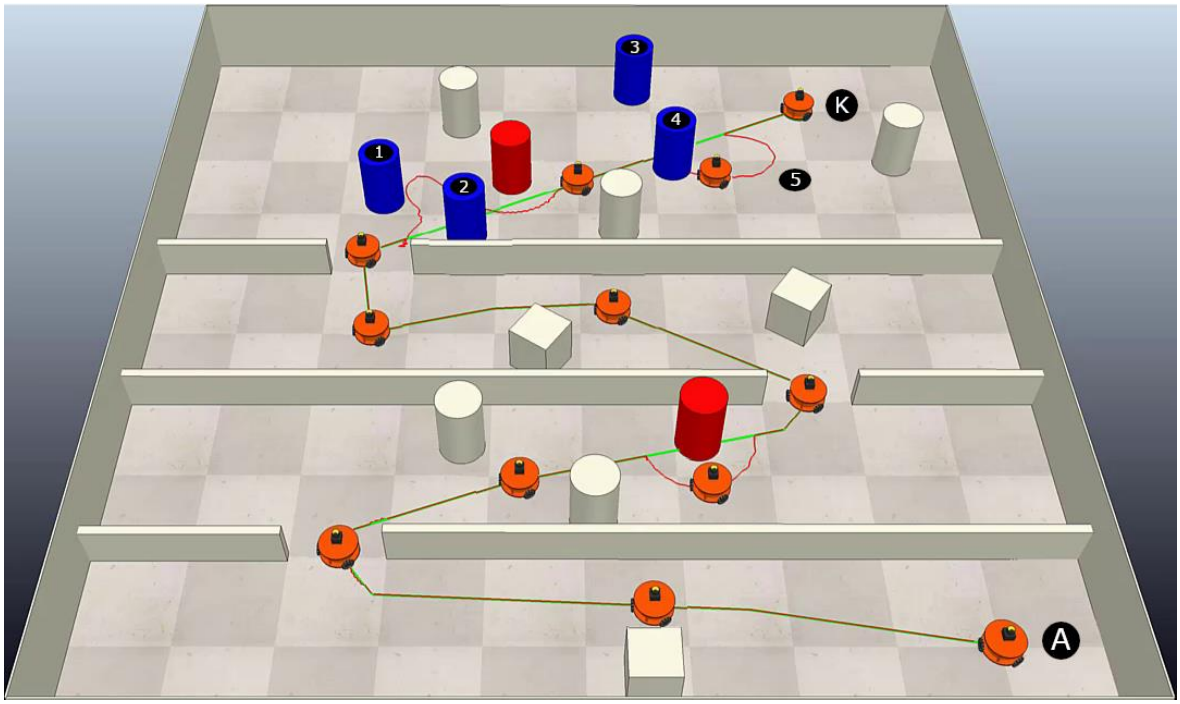

Fig. 7. The OMR follows the desired trajectory in a cluttered environment

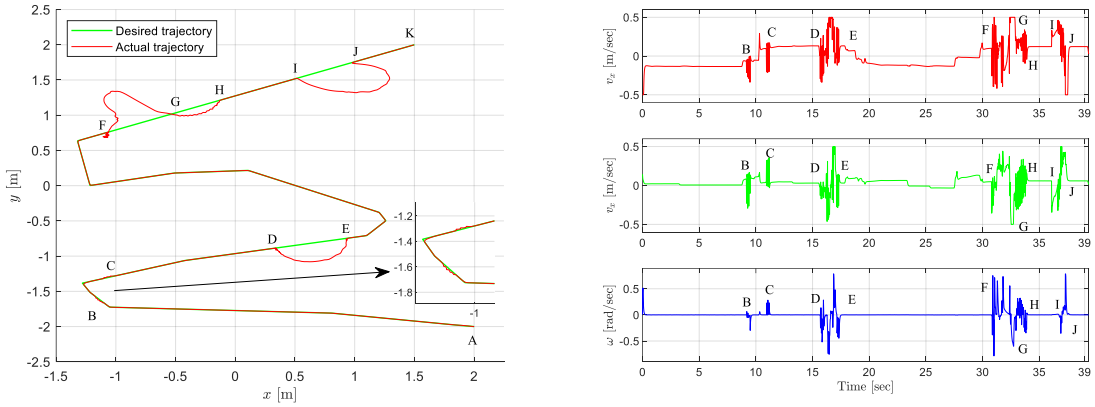

Fig. 8. The OMR follows the desired trajectory (left); control action vector components (right)

5. Conclusion

In this paper, a nonlinear model predictive controller was designed, enabling OMR to track a desired trajectory while at the same time avoiding both static and dynamic obstacles. The detection of obstacles was performed in real-time using a laser rangefinder, without former knowledge of

obstacles' positions and shapes. The DBSCAN algorithm was used to cluster the laser rangefinder data where each cluster represents an obstacle in the environment. Each cluster was then enclosed by multiple rotated ellipses to represent complex-shaped obstacles and seamlessly be integrated with NMPC as inequality constraints. The performance of the proposed framework was evaluated by three scenarios using CoppeliaSim robotic simulator, MATLAB, and CasADi toolbox, showing its effectiveness in cluttered environments. One potential direction for future work is the implementation of the proposed control architecture on a real OMR.

Author Contribution: All authors contributed equally to the main contributor to this paper. All authors read and approved the final paper.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] L. Fiorini *et al*., "Assistive robots to improve the independent living of older persons: results from a needs study," *Disability and Rehabilitation: Assistive Technology*, vol. 16, no. 1, pp. 92–102, 2019, https://doi.org/10.1080/17483107.2019.1642392.

[2] D. P. Losey *et al*., "Learning latent actions to control assistive robots," *Autonomous Robots*, vol. 46, no. 1, pp. 115–147, 2021, https://doi.org/10.1007/s10514-021-10005-w.

[3] M. Qasim and O. Y. Ismael, "Shared Control of a Robot Arm Using BCI and Computer Vision," *Journal Européen des Systèmes Automatisés*, vol. 55, no. 1, pp. 139-146, 2022, https://doi.org/10.18280/jesa.550115.

[4] X. Kan, H. Teng and K. Karydis, "Online Exploration and Coverage Planning in Unknown Obstacle-Cluttered Environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5969-5976, 2020, https://doi.org/10.1109/LRA.2020.3010455.

[5] F. Niroui, K. Zhang, Z. Kashino and G. Nejat, "Deep Reinforcement Learning Robot for Search and Rescue Applications: Exploration in Unknown Cluttered Environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 610-617, 2019, https://doi.org/10.1109/LRA.2019.2891991.

[6] M. Perez-Jimenez, M. A. Montes-Grova, P. Ramon-Soria, B. C. Arrue and A. Ollero, "POSITRON: lightweight active positioning compliant joints robotic arm in power lines inspection," *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 729-736, 2020, https://doi.org/10.1109/ICUAS48674.2020.9214022.

[7] A. B. Alhassan, X. Zhang, H. Shen, and H. Xu, "Power transmission line inspection robots: A review, trends and challenges for future research," *International Journal of Electrical Power & Energy Systems*, vol. 118, p. 105862, 2020, https://doi.org/10.1016/j.ijepes.2020.105862.

[8] Z. Li, J. Deng, R. Lu, Y. Xu, J. Bai and C. -Y. Su, "Trajectory-Tracking Control of Mobile Robot Systems Incorporating Neural-Dynamic Optimized Model Predictive Approach," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 6, pp. 740-749, 2016, https://doi.org/10.1109/TSMC.2015.2465352.

[9] D. Wang, W. Wei, Y. Yeboah, Y. Li, and Y. Gao, "A Robust Model Predictive Control Strategy for Trajectory Tracking of Omni-directional Mobile Robots," *Journal of Intelligent & Robotic Systems*, vol. 98, no. 2, pp. 439-453, 2019, https://doi.org/10.1007/s10846-019-01083-1.

[10] M. N. Alghanim, M. Qasim, K. P. Valavanis, M. J. Rutherford and M. Stefanovic, "Comparison of Controller Performance for UGV-Landing Platform Self-Leveling," *2020 28th Mediterranean Conference on Control and Automation (MED)*, pp. 471-478, 2020, https://doi.org/10.1109/MED48518.2020.9182837.

[11] O. Y. Ismael, M. Almaged, and A. I. Abdulla, "Nonlinear Model Predictive Control-based Collision Avoidance for Mobile Robot," *Journal of Robotics and Control (JRC)*, vol. 5, no. 1, pp. 142-151, 2024, https://doi.org/10.18196/jrc.v5i1.20615.

[12] M. N. Noaman, M. Qasim, and O. Y. Ismael, "Landmarks exploration algorithm for mobile robot indoor localization using VISION sensor," *Journal of Engineering Science & Technology*, vol. 16, no. 4, pp. 3165-3184, 2021, https://jestec.taylors.edu.my/Vol%2016%20Issue%204%20August%202021/16_4_29.pdf.

[13] M. N. Alghanim, M. Qasim, K. P. Valavanis, M. J. Rutherford and M. Stefanovic, "Passivity-Based Adaptive Controller for Dynamic Self-Leveling of a Custom-Built Landing Platform on Top of a UGV," *2020 28th Mediterranean Conference on Control and Automation (MED)*, pp. 458-464, 2020, https://doi.org/10.1109/MED48518.2020.9182807.

[14] B. Wang, Y. Zhang, and W. Zhang, "Integrated path planning and trajectory tracking control for quadrotor UAVs with obstacle avoidance in the presence of environmental and systematic uncertainties: Theory and experiment," *Aerospace Science and Technology*, vol. 120, p. 107277, 2022, https://doi.org/10.1016/j.ast.2021.107277.

[15] O. Y. Ismael, M. Qasim, and M. N. Noaman, "Equilibrium Optimizer-Based Robust Sliding Mode Control of Magnetic Levitation System," *Journal Européen des Systèmes Automatisés*, vol. 54, no. 1, pp. 131–138, 2021, https://doi.org/10.18280/jesa.540115.

[16] O. Y. Ismael, M. Qasim, M. N. Noaman, and A. Kurniawan, "Salp Swarm Algorithm-Based Nonlinear Robust Control of Magnetic Levitation System Using Feedback Linearization Approach," *Proceedings of the 3rd International Conference on Electronics, Communications and Control Engineering*, pp. 58-64, 2020, https://doi.org/10.1145/3396730.3396734.

[17] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011, https://doi.org/10.1177/0278364911406761.

[18] A. Orthey and M. Toussaint, "Sparse Multilevel Roadmaps for High-Dimensional Robotic Motion Planning," *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7851-7857, 2021, https://doi.org/10.1109/ICRA48506.2021.9562053.

[19] A. A. Ravankar, A. Ravankar, T. Emaru and Y. Kobayashi, "HPPRM: Hybrid Potential Based Probabilistic Roadmap Algorithm for Improved Dynamic Path Planning of Mobile Robots," *IEEE Access*, vol. 8, pp. 221743-221766, 2020, https://doi.org/10.1109/ACCESS.2020.3043333.

[20] O. Y. Ismael, M. N. Noaman, I. Kh. Abdullah, "Fick's Law Algorithm Based-Nonlinear Model Predictive Control of Twin Rotor MIMO System," *Journal of Robotics and Control (JRC)*, vol. 5, no. 3, pp. 694-705, 2024, https://doi.org/10.18196/jrc.v5i3.21725.

[21] C. E. Luis, M. Vukosavljev and A. P. Schoellig, "Online Trajectory Generation With Distributed Model Predictive Control for Multi-Robot Motion Planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 604-611, 2020, https://doi.org/10.1109/LRA.2020.2964159.

[22] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: an engineering perspective," *The International Journal of Advanced Manufacturing Technology*, vol. 117, no. 5–6, pp. 1327–1349, 2021, https://doi.org/10.1007/s00170-021-07682-3.

[23] N. Guo, B. Lenzo, X. Zhang, Y. Zou, R. Zhai and T. Zhang, "A Real-Time Nonlinear Model Predictive Controller for Yaw Motion Optimization of Distributed Drive Electric Vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 4935-4946, 2020, https://doi.org/10.1109/TVT.2020.2980169.

[24] R. Dempster, M. Al-Sharman, D. Rayside and W. Melek, "Real-Time Unified Trajectory Planning and Optimal Control for Urban Autonomous Driving Under Static and Dynamic Obstacle Constraints," *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10139-10145, 2023, https://doi.org/10.1109/ICRA48891.2023.10160577.

[25] A. Salimi Lafmejani and S. Berman, "Nonlinear MPC for collision-free and deadlock-free navigation of multiple nonholonomic mobile robots," *Robotics and Autonomous Systems*, vol. 141, p. 103774, 2021, https://doi.org/10.1016/j.robot.2021.103774.

[26] M. R. Azizi, A. Rastegarpanah, and R. Stolkin, "Motion Planning and Control of an Omnidirectional Mobile Robot in Dynamic Environments," *Robotics*, vol. 10, no. 1, p. 48, 2021, https://doi.org/10.3390/robotics10010048.

[27] M. A. Daoud, M. W. Mehrez, D. Rayside and W. W. Melek, "Simultaneous Feasible Local Planning and Path-Following Control for Autonomous Driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 16358-16370, 2022, https://doi.org/10.1109/TITS.2022.3149986.

[28] 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, https://doi.org/10.1155/2022/2538220.

[29] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, and P. Wang, "Path planning techniques for mobile robots: Review and prospect," *Expert Systems with Applications*, vol. 227, p. 120254, 2023, https://doi.org/10.1016/j.eswa.2023.120254.

[30] A. Britzelmeier and M. Gerdts, "A Nonsmooth Newton Method for Linear Model-Predictive Control in Tracking Tasks for a Mobile Robot With Obstacle Avoidance," *IEEE Control Systems Letters*, vol. 4, no. 4, pp. 886-891, 2020, https://doi.org/10.1109/LCSYS.2020.2996959.

[31] I. Sánchez, A. D'Jorge, G. V. Raffo, A. H. González, and A. Ferramosca, "Nonlinear Model Predictive Path Following Controller with Obstacle Avoidance," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 16, 2021, https://doi.org/10.1007/s10846-021-01373-7.

[32] M. A. Santos, A. Ferramosca, and G. V. Raffo, "Nonlinear Model Predictive Control Schemes for Obstacle Avoidance," *Journal of Control, Automation and Electrical Systems*, vol. 34, no. 5, pp. 891-906, 2023, https://doi.org/10.1007/s40313-023-01024-2.

[33] Z. Jian *et al*., "Dynamic Control Barrier Function-based Model Predictive Control to Safety-Critical Obstacle-Avoidance of Mobile Robot," *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3679-3685, 2023, https://doi.org/10.1109/ICRA48891.2023.10160857.

[34] R. Zheng and S. Li, "MCCA: A Decentralized Method for Collision and Deadlock Avoidance With Nonholonomic Robots," *IEEE Robotics and Automation Letters*, vol. 9, no. 3, pp. 2710-2717, 2024, https://doi.org/10.1109/LRA.2024.3358623.

[35] L.-L. Wang and L.-X. Pan, "Research on SBMPC Algorithm for Path Planning of Rescue and Detection Robot," *Discrete Dynamics in Nature and Society*, vol. 2020, no. 1, pp. 1–11, 2020, https://doi.org/10.1155/2020/7821942.

[36] Z. Zuo *et al*., "MPC-Based Cooperative Control Strategy of Path Planning and Trajectory Tracking for Intelligent Vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 3, pp. 513-522, 2021, https://doi.org/10.1109/TIV.2020.3045837.

[37] M. Ammour, R. Orjuela and M. Basset, "A MPC Combined Decision Making and Trajectory Planning for Autonomous Vehicle Collision Avoidance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 24805-24817, 2022, https://doi.org/10.1109/TITS.2022.3210276.

[38] D. Huo, L. Dai, R. Chai, R. Xue and Y. Xia, "Collision-Free Model Predictive Trajectory Tracking Control for UAVs in Obstacle Environment," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 59, no. 3, pp. 2920-2932, 2023, https://doi.org/10.1109/TAES.2022.3221702.

[39] Q. Shi, J. Zhao, A. E. Kamel and I. Lopez-Juarez, "MPC Based Vehicular Trajectory Planning in Structured Environment," *IEEE Access*, vol. 9, pp. 21998-22013, 2021, https://doi.org/10.1109/ACCESS.2021.3052720.

[40] M. B. Alatise and G. P. Hancke, "A Review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods," *IEEE Access*, vol. 8, pp. 39830-39846, 2020, https://doi.org/10.1109/ACCESS.2020.2975643.

[41] J. Li, R. Li, J. Li, J. Wang, Q. Wu, and X. Liu, "Dual-view 3D object recognition and detection via Lidar point cloud and camera image," *Robotics and Autonomous Systems*, vol. 150, p. 103999, 2022, https://doi.org/10.1016/j.robot.2021.103999.

[42] M. Ilyas, H. Y. Khaw, N. M. Selvaraj, Y. Jin, X. Zhao and C. C. Cheah, "Robot-Assisted Object Detection for Construction Automation: Data and Information-Driven Approach," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 6, pp. 2845-2856, 2021, https://doi.org/10.1109/TMECH.2021.3100306.

[43] Y. Wu, Y. Wang, S. Zhang and H. Ogai, "Deep 3D Object Detection Networks Using LiDAR Data: A Review," *IEEE Sensors Journal*, vol. 21, no. 2, pp. 1152-1171, 2021, https://doi.org/10.1109/JSEN.2020.3020626.

[44] B. Brito, B. Floor, L. Ferranti and J. Alonso-Mora, "Model Predictive Contouring Control for Collision Avoidance in Unstructured Dynamic Environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4459-4466, 2019, https://doi.org/10.1109/LRA.2019.2929976.

[45] 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, https://doi.org/10.1007/s12532-018-0139-4.

[46] A. Wächter and L. T. Biegler, "Line Search Filter Methods for Nonlinear Programming: Motivation and Global Convergence," SIAM Journal on Optimization, vol. 16, no. 1, pp. 1-31, 2005, https://doi.org/10.1137/S1052623403426556.

[47] J. C. L. Barreto S., A. G. S. Conceição, C. E. T. Dórea, L. Martinez and E. R. de Pieri, "Design and Implementation of Model-Predictive Control With Friction Compensation on an Omnidirectional Mobile Robot," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 2, pp. 467-476, 2014, https://doi.org/10.1109/TMECH.2013.2243161.

[48] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, "Introduction to autonomous mobile robots," *MIT press*, 2011, https://mitpress.mit.edu/9780262015356/introduction-to-autonomous-mobile-robots/.

[49] L. Grüne and J. Pannek, "Nonlinear Model Predictive Control," *Communications and Control Engineering*, 2017, https://doi.org/10.1007/978-3-319-46024-6.

[50] E. F. Camacho and C. Bordons, "Model Predictive control," *Springer London*, 2007, https://doi.org/10.1007/978-0-85729-398-5.

[51] J. B. Rawlings and K. R. Muske, "The stability of constrained receding horizon control," *IEEE Transactions on Automatic Control*, vol. 38, no. 10, pp. 1512-1516, 1993, https://doi.org/10.1109/9.241565.

[52] L. Grüne and J. Pannek, "Nonlinear Model Predictive Control," *Springer London*, 2011, https://doi.org/10.1007/978-0-85729-501-9.