Design a Nonlinear MPC Controller for Autonomous Mobile Robot Navigation System Based on ROS

Hiep Do Quang
University of Economics-Technology for Industries, Hanoi, Vietnam
Email: dqhiep@uneti.edu.vn

Thang Le Tran
Controls, Automation in Production and Improvement of Technology Institute Hanoi, Vietnam
Email: ltrantahang@gmail.com

Tien Ngo Manh
Institute of Physics, Vietnam Academy of Science and Technology, Hanoi, Vietnam
Email: nmtien@iop.vast.vn

Cuong Nguyen Manh* and Toan Nguyen Nhu
Hanoi University of Science and Technology, Hanoi, Vietnam
Email: {manhcuong313.ng, toan.nn110899}@gmail.com

Nam Bui Duy
VNU University of Engineering and Technology, Hanoi, Vietnam
Email: duynam.robotics@gmail.com

***Abstract*—Navigation is an essential problem for self-propelled robots, in which the orbital control system plays a decisive role in the robot's behavior and navigating the robot to move in space. This paper presents the tutorial approach for the design and construction of the Nonlinear Model Predictive Control controller for the four-wheeled Omni robot in the orbital tracking problem. The designed controller is implemented in the navigation stack system for the robot to follow the specified trajectory smoothly and avoid collision with surrounding obstacles simultaneously. The results are presented in both theory and simulation cases owing to the ROS platform to illustrate the validity and effectiveness of the method for the trajectory tracking problem combined with the navigation stack system. Besides the advantage of the designed navigation stack is also demonstrated through comparison with the TEB planner.**

***Index Terms*—Nonlinear Model Predictive Control (NMPC), Omni robot, trajectory tracking, navigation, Robot Operating System (ROS)**

I. Introduction

Currently, mobile robotics is one of the most extensive research fields and becoming more popular in the industry. Mobile robots can move automatically in many different environments. In an indoor environment with many obstacles, where the robot's moving space is limited, omnidirectional robots are commonly used in indoor environments such as factories, warehouses, laboratories. Therefore, the increasingly advanced wheel structure has been researched and manufactured to help the robot move flexibly in all directions. The Omni wheel is a popular omnidirectional wheel used for mobile robotic applications in the home. The Omni wheel has a series of free-moving rollers centrally mounted and perpendicular to the wheel. Therefore, in addition to the ability to rotate around the axis, the Omni wheel can slide to help the robot move more flexibly [1], [2].

Nonlinear Model Predictive Control (NMPC), referred to as the receding horizon control, is an optimization-based time-domain method for the closed-loop control of the nonlinear system. The primary advantage of NMPC is that it can explicitly handle system constraints in the control process, which are generally neglected in traditional methods [3]-[7]. However, most mechanism systems always exit many different constraints due to the limited capacity of actuators, the conditions of surrounding environments, or some economic considerations. As in traditional linear MPC, NMPC computes control actions at each control interval using a combination of model-based

Manuscript received December 6, 2021; revised February 12, 2022.
*Corresponding author

prediction and constrained optimization. However, for a nonlinear system like our robot, along with the requirement for fast response of the controller, the linear controller will have many challenges and may lead to divergence in some cases. Besides, by using the NMPC controller, the controller can consider the nonlinear properties of the system. Therefore, NPMC is a preferable method when solving engineering issues in the presence of system constraints. The mobile robot system's constraints can come from each wheel's maximum velocity, acceleration, and distance to the surrounding obstacles to avoid the collision. In [8], an MPC controller is proposed for the trajectory tracking problem of omnidirectional wheeled mobile robot based on linearizing the kinematic model to predict the future system's states, and the constraint on control signal is taken into account. Caireta et al [9] present a model predictive control for the mecanum-wheeled robot using its dynamic model to minimize energy consumption and simultaneously avoid obstacles when moving towards the determined goal. In [10], a concept of virtual vehicle and model predictive controller are combined to ensure that the mobile robot can follow the desired trajectory while satisfying motion constraints.

Navigation is a primary but vital task for any automated mobile robot system. Much research [11]-[14] has concentrated on the smooth trajectory tracking of mobile robots as they navigate in the environment. Theoretically, when the robot is assigned to maneuver to a specified destination in the environment, it first computes the continuous path which connects the current position of the robot with its goal, then computes a set of appropriate commands, i.e., velocity, acceleration, to follow the trajectory while avoiding collision with surrounding obstacles. Therefore, this paper is presented as a tutorial approach towards the problem of designing an effective controller for the mobile robot to follow the previously designed trajectory of the navigation system. Based on the ability to predict the robot's future state, the NMPC controller is exceptionally suitable for environmental obstacle avoidance problems. Hence, the robot can identify obstacles and compute the optimal control signals to move more smoothly than previous algorithms [15]. The proposed navigation stack-based NMPC is implemented based on the Robot Operating System (ROS) platform and the assistance of the ACADO toolkit. The Omni-robot is fully designed to work analogously to the existing system and put in the environment simulated by Gazebo software with many obstacles.

II. Robot Modeling

A. Kinematic Modeling

The 4-wheeled omnidirectional robot model is presented in Fig. 1 with its four wheels 90 degrees apart.

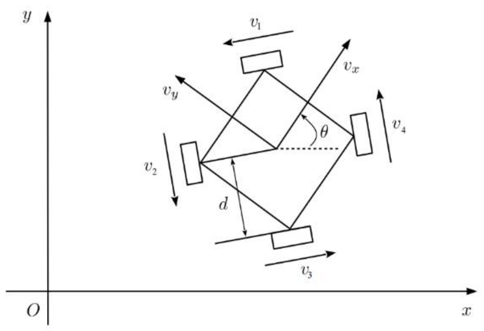

Figure 1. The robot model in the global coordinate

Particularly, the distribution of four wheels on the robot is $\alpha_1 = \frac{\pi}{4}$, $\alpha_2 = \frac{3\pi}{4}$, $\alpha_3 = \frac{5\pi}{4}$, $\alpha_4 = \frac{7\pi}{4}$ respectively. Besides, the wheel radius is denoted as R, the distance from the center of the robot to each wheel is d. The velocity of the robot includes the linear velocity v_x, v_y in the x, y axis, respectively, the angular velocity ω. The forward kinematic equation has the following form:

$$\boldsymbol{u} = \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} = \frac{R}{3}\begin{bmatrix} \frac{-\sqrt{2}}{8} & \frac{-\sqrt{2}}{8} & \frac{\sqrt{2}}{8} & \frac{\sqrt{2}}{8} \\ \frac{\sqrt{2}}{8} & \frac{-\sqrt{2}}{8} & \frac{-\sqrt{2}}{8} & \frac{\sqrt{2}}{8} \\ \frac{1}{4d} & \frac{1}{4d} & \frac{1}{4d} & \frac{1}{4d} \end{bmatrix}\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} \quad (1)$$

Assuming that

$$\boldsymbol{H}_1 = \frac{R}{3}\begin{bmatrix} \frac{-\sqrt{2}}{8} & \frac{-\sqrt{2}}{8} & \frac{\sqrt{2}}{8} & \frac{\sqrt{2}}{8} \\ \frac{\sqrt{2}}{8} & \frac{-\sqrt{2}}{8} & \frac{-\sqrt{2}}{8} & \frac{\sqrt{2}}{8} \\ \frac{1}{4d} & \frac{1}{4d} & \frac{1}{4d} & \frac{1}{4d} \end{bmatrix} \quad (2)$$

The inverse kinematic equation can be transformed into:

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \boldsymbol{H}_1^T\left(\boldsymbol{H}_1\boldsymbol{H}_1^T\right)^{-1}\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} \quad (3)$$

The state of the Omni robot in the global coordinate is denoted as $\boldsymbol{x} = (x, y, \theta)^T$. According to Fig. 1, it is derived that:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} = \boldsymbol{H}_2\boldsymbol{u} \quad (4)$$

where

$$\boldsymbol{H}_2 = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}; \boldsymbol{u} = \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} \quad (5)$$

From (4) the robot, the state will be achieved through integrals

$$\boldsymbol{x} = \begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \end{bmatrix} = \begin{bmatrix} x(t_0) + \int \left(v_x(t)\cos\theta(t) - v_y(t)\sin\theta(t)\right)\mathrm{dt} \\ y(t_0) + \int \left(v_x(t)\sin\theta(t) + v_y(t)\cos\theta(t)\right)\mathrm{dt} \\ \omega(t_0) + \int \omega(t)\mathrm{dt} \end{bmatrix} \quad (6)$$

The angle of the robot is always limited in range $[-\pi, \pi]$. So, we always normalize the angle by:

$$\theta = \theta - 2\pi \left\lfloor \frac{\theta + \pi}{2\pi} \right\rfloor \quad (7)$$

Each wheel of a mobile robot is always limited to a limited speed. Therefore, the linear velocity and angular velocity of the robot are also limited. Define the speed constraints of the robot:

$$\begin{gathered} |v_x|, |v_y| \le v_{\max} \\ |\omega| \le \omega_{\max} \end{gathered} \quad (8)$$

B. Discretization

The continuous nonlinear control system in the equation (4) can be expressed as:

$$\dot{\boldsymbol{x}} = f\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right) \quad (9)$$

where $\boldsymbol{x}(t) \in \mathbb{R}^3$ and $\boldsymbol{u}(t) \in \mathbb{R}^3$ are the 3-dimensional states and 3-dimensional control vectors.

In this paper, a Runge-Kutta approach is used to obtain the state at the next time step $\boldsymbol{x}_{k+1}$. Runge-Kutta methods are a class of methods to integrate ODE. In this case, a 4th order Runge-Kutta method is employed. It approximates the solution to a differential equation in (9), with initial condition $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$. The order of the Runge-Kutta method refers to the number of approximations of the slope with a one-time step. The slope at the start point and endpoint are defined by R_1 and R_4. The slopes at the two midpoints are R_2 and R_3 [16].

To determine an approximation of $\boldsymbol{x}_{k+1}$, a weighted average of the slopes at R_1, R_2, R_3, R_4 is used. The midpoint slopes are weighted twice as much as the start and endpoint. The $\boldsymbol{x}_{k+1}$ is obtained as follow:

$$\begin{gathered} \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \frac{h}{6}\left(R_1 + 2R_2 + 2R_3 + R_4\right) \\ t_{k+1} = t_k + h \end{gathered} \quad (10)$$

Considering a sampling period h, a sampling instant k and applying the 4th order Runge-Kutta approximation to (4), we obtain the discrete-time model for robot motion. The individual slopes at each point are defined as:

$$R_1 = f\left(\boldsymbol{x}_k, \boldsymbol{u}\right) = \begin{bmatrix} v_x\cos\theta - v_y\sin\theta \\ v_x\sin\theta + v_y\cos\theta \\ \omega \end{bmatrix} \quad (11)$$

$$R_2 = f\left(\boldsymbol{x}_k + \frac{h}{2}R_1, \boldsymbol{u}\right) = \begin{bmatrix} v_x\cos\left(\theta + \frac{h}{2}\omega\right) - v_y\sin\left(\theta + \frac{h}{2}\omega\right) \\ v_x\sin\left(\theta + \frac{h}{2}\omega\right) + v_y\cos\left(\theta + \frac{h}{2}\omega\right) \\ \omega \end{bmatrix} \quad (12)$$

$$R_3 = f\left(\boldsymbol{x}_k + \frac{h}{2}R_2, \boldsymbol{u}\right) = \begin{bmatrix} v_x\cos\left(\theta + \frac{h}{2}\omega\right) - v_y\sin\left(\theta + \frac{h}{2}\omega\right) \\ v_x\sin\left(\theta + \frac{h}{2}\omega\right) + v_y\cos\left(\theta + \frac{h}{2}\omega\right) \\ \omega \end{bmatrix} \quad (13)$$

$$R_4 = f\left(\boldsymbol{x}_k + hR_3, \boldsymbol{u}\right) = \begin{bmatrix} v_x\cos(\theta + h\omega) - v_y\sin(\theta + h\omega) \\ v_x\sin(\theta + h\omega) + v_y\cos(\theta + h\omega) \\ \omega \end{bmatrix} \quad (14)$$

Based on (11-14) the equation (10) is as follow:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \frac{h}{6}\left(R_1 + 2R_2 + 2R_3 + R_4\right) = \begin{bmatrix} x_k \\ y_k \\ \omega_k \end{bmatrix} + \begin{bmatrix} \frac{hv_x}{6}\alpha - \frac{hv_y}{6}\beta \\ \frac{hv_x}{6}\beta + \frac{hv_y}{6}\alpha \\ h\omega \end{bmatrix} \quad (15)$$

The matrix form of (15) is as follow:

$$x_{k+1} = x_k + h\begin{bmatrix} \frac{\alpha}{6} & \frac{-\beta}{6} & 0 \\ \frac{\beta}{6} & \frac{\alpha}{6} & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{u} \tag{16}$$

where:

$$\begin{aligned} \alpha &= \cos\theta + 4\cos\left(\theta + \frac{h}{2}\omega\right) + \cos(\theta + h\omega) \\ \beta &= \sin\theta + 4\sin\left(\theta + \frac{h}{2}\omega\right) + \sin(\theta + h\omega) \end{aligned} \tag{17}$$

And a compact representation of (16) is as follow:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + g(\boldsymbol{x}_k, \boldsymbol{u}_k)\boldsymbol{u}_k = f(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{18}$$

III. Nonlinear Model Predictive Control (NMPC)

A. Controller Design

In this section, a nonlinear MPC problem is formulated as a quadratic programming problem. The discrete-time system is shown in (18). And the constraints are:

$$\boldsymbol{x}_k \in \mathcal{X}, k = 0,1,...,N; \boldsymbol{x}_{min} \le \boldsymbol{x}_k \le \boldsymbol{x}_{max} \tag{19}$$

$$\boldsymbol{u}_k \in \mathcal{U}, k = 0,1,...,N_u; \boldsymbol{u}_{min} \le \boldsymbol{u}_k \le \boldsymbol{u}_{max} \tag{20}$$

where N denotes prediction horizon, N_u denotes control horizon, $N \ge N_u > 0$.

NMPC can be formulated as an iterative optimization procedure. A cost function is optimized to get an optimal control input vector; then, repeated online calculation of an optimization function is conducted [17]. The advantage of NMPC is that it considers the system constraints in the optimization process, leading to the improvement of the robustness of the overall system.

The standard quadratic form of objective cost function $J(\boldsymbol{x}, \boldsymbol{u})$ of NMPC problem can be defined as follow:

$$J(\boldsymbol{x}, \boldsymbol{u}) = \|\boldsymbol{x}_N\|^2_{\boldsymbol{Q}_N} + \sum_{i=0}^{N-1} \|\boldsymbol{x}_i\|^2_{\boldsymbol{Q}} + \sum_{i=0}^{N_u-1} \|\boldsymbol{u}_i\|^2_{\boldsymbol{R}} \tag{21}$$

subject to:

$$\begin{aligned} \boldsymbol{x}_{i+1} &= f(\boldsymbol{x}_i, \boldsymbol{u}_i) \quad \forall i \in [0, N] \\ \boldsymbol{u}_{min} &\le \boldsymbol{u}_i \le \boldsymbol{u}_{max} \quad \forall i \in [0, N_u - 1] \end{aligned} \tag{22}$$

where $\|\cdot\|$ denotes the Euclidean norm. The term $\|\boldsymbol{x}_N\|^2_{\boldsymbol{Q}_N}$ is the terminal penalty term, $\boldsymbol{Q}$ and $\boldsymbol{R}$ are positive definite weight matrices. The predicted states in the future horizon are $\boldsymbol{x}_i, \forall i = \overline{1, N}$, the predicted control signals in the future horizon $\boldsymbol{u}_j, \forall j = \overline{1, N_u}$. The term $\boldsymbol{u}_{min}$ stands for the lower bound and $\boldsymbol{u}_{max}$ stands for the upper bound.

Similar to the robot state, a reference state vector is defined as ${}^r\boldsymbol{x} = (x_r, y_r, \theta_r)^T$. Define the tracking error as follow:

$${}^e\boldsymbol{x}_j = \boldsymbol{x}_j - {}^r\boldsymbol{x}_j, \quad \forall j \in [0, N] \tag{23}$$

Now in our case, the NMPC method is implemented based on the kinematic model of the Omni robot to find a control sequence $\boldsymbol{u}$ so that the current state $\boldsymbol{x}$ will converge to the desired value. By choosing $N = N_u$, the cost function in (21) is rewritten as follow:

$$J(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}) = {}^e\boldsymbol{x}_N^T \boldsymbol{Q}_N {}^e\boldsymbol{x}_N + \sum_{i=0}^{N-1}\left[{}^e\boldsymbol{x}_i^T \mathbf{Q} {}^e\boldsymbol{x}_i + \boldsymbol{u}_i^T \boldsymbol{R} \boldsymbol{u}_i\right] \tag{24}$$

Subject to

$$\boldsymbol{x}_{i+1} = f(\boldsymbol{x}_i, \boldsymbol{u}_i), \forall i \in [0, N-1] \tag{25}$$

$$\boldsymbol{u}_{min} \le \boldsymbol{u}_i \le \boldsymbol{u}_{max}, \forall i \in [0, N-1] \tag{26}$$

In which

$$\bar{\boldsymbol{x}} = \begin{bmatrix} {}^e\boldsymbol{x}_1 & {}^e\boldsymbol{x}_2 & \cdots & {}^e\boldsymbol{x}_N \end{bmatrix}^T \in \mathbb{R}^{3N} \tag{27}$$

$$\bar{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots & \boldsymbol{u}_{N-1} \end{bmatrix}^T \in \mathbb{R}^{3N} \tag{28}$$

The matrix form of equation **(24)** is as follows:

$$J(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}) = {}^e\boldsymbol{x}_N^T \boldsymbol{Q}_N {}^e\boldsymbol{x}_N + \bar{\boldsymbol{x}}^T \bar{\boldsymbol{Q}} \bar{\boldsymbol{x}} + \bar{\boldsymbol{u}}^T \bar{\boldsymbol{R}} \bar{\boldsymbol{u}} \tag{29}$$

where:

$$\bar{\boldsymbol{Q}} = \begin{bmatrix} Q & 0 & \cdots & 0 \\ 0 & Q & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & Q \end{bmatrix}; \bar{\boldsymbol{R}} = \begin{bmatrix} R & 0 & \cdots & 0 \\ 0 & R & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & R \end{bmatrix}$$

Similarly, we can rewrite (22) in more general form as:

$$\boldsymbol{D}\bar{\boldsymbol{u}} \le \boldsymbol{d}$$

with

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{I}_{3N\times 3N} \\ -\boldsymbol{I}_{3N\times 3N} \end{bmatrix}; \boldsymbol{d} = \begin{bmatrix} \bar{\boldsymbol{u}}_{max} \\ -\bar{\boldsymbol{u}}_{min} \end{bmatrix} \tag{30}$$

B. Sequential Quadratic Programming

At each sampling time instant, the NMPC controller obtains the robot's current input and output measurements, the current robot's states, and the robot model to calculate an optimal future control sequence over a defined predict horizon that optimizes the objective function simultaneously satisfying the constraints of the system. Then, only the first control signal in the sequence is used as the input to control the robot. Among many approaches to solving the NMPC problem in literature such as Interior Point (IP) [18], the Sequential Quadratic Programming (SQP) [19], [20], i.e., the SQP method is often used because of its advantages. The SQP provides highly accurate solutions with fast final convergence. At the same time, the SQP is also suitable for highly non-linear problems such as our robot model. In addition, available tools and libraries such as ACADO [19] are designed to solve SQP problems conveniently. The SQP is powerful enough for actual problems to handle any degree of non-linearity, including non-linearity in the constraints. The main idea of SQP is that it solves in each iteration an inequality constrained QP obtained by linearizing the objective and constraints functions. Hence, to apply the SQP method, at time instant, the objective cost function in (24) is linearized to have the following form:

$$\begin{aligned} J(\bar{x},\bar{u}) &\approx J(\bar{x}(k),\bar{u}(k)) + \nabla_{\bar{u}} J(\bar{x}(k),\bar{u}(k))(\bar{u}-\bar{u}(k)) \\ &+\frac{1}{2}(\bar{u}-\bar{u}(k))^T \nabla_{\bar{u}}^2 J(\bar{x}(k),\bar{u}(k))(\bar{u}-\bar{u}(k)) \end{aligned} \quad (31)$$

In which, the Jacobian $\nabla_{\bar{u}}^2 J$ is calculated as follow:

$$\nabla_{\bar{u}} J = \begin{bmatrix} \frac{\nabla J}{\nabla \boldsymbol{u}_0} & \frac{\nabla J}{\nabla \boldsymbol{u}_1} & \dots & \frac{\nabla J}{\nabla \boldsymbol{u}_{N-1}} \end{bmatrix}^T = \boldsymbol{G} \quad (32)$$

In the case of the objective function expressed as in (24), we have:

$$\frac{\nabla J}{\nabla \boldsymbol{u}_i} = 2\,{}^e\boldsymbol{x}_{i+1}^T \boldsymbol{Q} \frac{\nabla f(\bar{x}_i,\bar{u}_i)}{\nabla \boldsymbol{u}_i} + 2\boldsymbol{R}\boldsymbol{u}_i \quad (33)$$

Considering the Hessian $\nabla_{\bar{u}}^2 J$, because the objective function has the nonlinear least-square form, the Gauss-Newton Hessian approximation with linear convergence property can be employed instead of exact value to avoid complicated computation. We have:

$$\begin{aligned} \left\| \boldsymbol{x}_i - {}^r\boldsymbol{x}_i \right\|_Q^2 &= \left\| f(\boldsymbol{x}_{i-1},\boldsymbol{u}_{i-1}) - {}^r\boldsymbol{x}_i \right\|_{\boldsymbol{Q}}^2 \\ &\approx \left\| \begin{matrix} f^T(\boldsymbol{x}_{i-1}(k),\boldsymbol{u}_{i-1}(k)) \\ +\nabla f(\boldsymbol{x}_{i-1}(k),\boldsymbol{u}_{i-1}(k))(\boldsymbol{u}_{i-1}-\boldsymbol{u}_{i-1}(k)) \\ -{}^r\boldsymbol{x}_i \end{matrix} \right\|_{\boldsymbol{Q}}^2 \\ &= \left\| \boldsymbol{B}_i\boldsymbol{u}_{i-1} + \boldsymbol{C}_i \right\|_{\boldsymbol{Q}}^2 \\ &= \boldsymbol{u}_{i-1}^T\boldsymbol{B}_i^T\boldsymbol{Q}\boldsymbol{B}_i\boldsymbol{u} + 2\boldsymbol{u}^T\boldsymbol{B}_i^T\boldsymbol{Q}\boldsymbol{C}_i + \boldsymbol{C}_i^T\boldsymbol{Q}\boldsymbol{C}_i, \forall i \in [1,N] \end{aligned} \quad (34)$$

with

$$\begin{aligned} \boldsymbol{B}_i &= \nabla f(\boldsymbol{x}_{i-1}(k),\boldsymbol{u}_{i-1}(k)), \forall i \in [1,N] \\ \boldsymbol{C}_i &= f^T(\boldsymbol{x}_{i-1}(k),\boldsymbol{u}_{i-1}(k)) \\ &-\nabla f(\boldsymbol{x}_{i-1}(k),\boldsymbol{u}_{i-1}(k))\boldsymbol{u}_{i-1} - {}^r\boldsymbol{x}_i, \forall i \in [1,N] \end{aligned} \quad (35)$$

Substituting (25) to (24) yields:

$$\begin{aligned} J(\bar{x},\bar{u}) &= f(\boldsymbol{x}_{N-1},\boldsymbol{u}_{N-1}) - {}^r\boldsymbol{x}_{N\boldsymbol{Q}_N}^2 + \boldsymbol{x}_{0\boldsymbol{Q}}^2 \\ &+\sum_{i=1}^{N-1} f(\boldsymbol{x}_{i-1},\boldsymbol{u}_{i-1}) - {}^r\boldsymbol{x}_{i\boldsymbol{Q}}^2 + \sum_{i=0}^{N-1}\boldsymbol{u}_{i\boldsymbol{Q}}^2 \\ &\approx \boldsymbol{B}_N\boldsymbol{u}_{N-1} + \boldsymbol{C}_{N\boldsymbol{Q}_N}^2 + \boldsymbol{x}_{0\boldsymbol{Q}}^2 \\ &+\sum_{i=1}^{N-1}\boldsymbol{B}_i\boldsymbol{u}_{i-1} + \boldsymbol{C}_{i\boldsymbol{Q}}^2 + \sum_{i=0}^{N-1}\boldsymbol{u}_{i\boldsymbol{Q}}^2 \\ &= \bar{\boldsymbol{u}}^T\boldsymbol{H}\bar{\boldsymbol{u}} + 2\bar{\boldsymbol{u}}^T\boldsymbol{M} + \boldsymbol{N} \end{aligned} \quad (36)$$

With

$$\begin{aligned} \boldsymbol{H} &= \begin{bmatrix} \boldsymbol{B}_1^T\boldsymbol{Q}\boldsymbol{B}_1 + \boldsymbol{R} & 0 & \cdots & 0 \\ 0 & \boldsymbol{B}_2^T\boldsymbol{Q}\boldsymbol{B}_2 + \boldsymbol{R} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{B}_N^T\boldsymbol{Q}\boldsymbol{B}_N + \boldsymbol{R} \end{bmatrix} \\ \boldsymbol{M} &= \begin{bmatrix} \boldsymbol{B}_1^T\boldsymbol{Q}\boldsymbol{C}_1 & 0 & \cdots & 0 \\ 0 & \boldsymbol{B}_2^T\boldsymbol{Q}\boldsymbol{C}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{B}_N^T\boldsymbol{Q}\boldsymbol{C}_N \end{bmatrix} \\ \boldsymbol{N} &= x_{0\boldsymbol{Q}}^2 + \sum_{i=1}^{N}\boldsymbol{C}_i^T\mathbf{Q}\boldsymbol{C}_i \end{aligned} \quad (37)$$

According to the Gauss-Newton Hessian approximation, the matrix $\boldsymbol{H}$ is used as the estimated value of the Hessian $\nabla_{\bar{u}}^2 J$. Therefore, using the SQP method, the QP is reformulated as follows:

$$\begin{aligned} \min_{\bar{u}} J_{SQP}(\bar{\boldsymbol{u}}) &= \boldsymbol{G}(\bar{\boldsymbol{u}} - \bar{\boldsymbol{u}}(k)) \\ &+\frac{1}{2}(\bar{\boldsymbol{u}} - \bar{\boldsymbol{u}}(k))^T \boldsymbol{H}(\bar{\boldsymbol{u}} - \bar{\boldsymbol{u}}(k)) \end{aligned} \quad (38)$$

Subject to

$$\boldsymbol{D}\bar{\boldsymbol{u}} \le \boldsymbol{d} \quad (39)$$

Remark 1: Let Ω be the set of indices i for which the i^{th} element of vector $[\boldsymbol{D}\boldsymbol{u} - \boldsymbol{d}]$ equals zero, then Ω includes indices of active constraints. Denote $[\cdot]_\Omega$ be the vector obtained by taking all the i^{th} row of vector $[\cdot]\forall i \in \Omega$.

The Lagrange function can be established as follow:

$$L(\bar{\boldsymbol{u}},\lambda) = J_{SQP}(\boldsymbol{u}) + \lambda^T(\boldsymbol{D}\bar{\boldsymbol{u}} - \boldsymbol{d}) \quad (40)$$

With λ is Lagrange multiplier.

Then, we have the Karush-Kuhn-Tucker conditions as:

$$\begin{aligned} \frac{L(\boldsymbol{u},\lambda)}{\nabla \bar{\boldsymbol{u}}} &= \nabla J_{SQP}(\bar{\boldsymbol{u}}) + \lambda\boldsymbol{D} \\ &= \boldsymbol{H}(\bar{\boldsymbol{u}} - \bar{\boldsymbol{u}}(k)) + \boldsymbol{G} + \lambda\boldsymbol{D} = 0 \end{aligned} \quad (41)$$

$$\frac{L(\bar{\boldsymbol{u}},\lambda)}{\nabla \lambda} = \boldsymbol{D}\bar{\boldsymbol{u}} - \boldsymbol{d} \le 0 \quad (42)$$

$$\lambda \geq 0 \tag{43}$$

$$\lambda^T \left(\boldsymbol{D}\bar{\boldsymbol{u}} - \boldsymbol{d} \right) = 0 \tag{44}$$

To find the minimizer $\bar{\boldsymbol{u}}^*$ of J_{SQP} satisfying these KKT conditions, the active set strategy is implemented. Assuming that the active set Ω is given at time instant k and $\bar{\boldsymbol{u}}(k)$ is a feasible point. The QP algorithm using active set strategy proceeds as follows:

1. The minimum of the quadratic objective subject (38) to the constraints in the active set treated as equalities is computed by solving the equalities:

$$\boldsymbol{H}\left(\bar{\boldsymbol{u}} - \bar{\boldsymbol{u}}(k)\right) + \boldsymbol{G} + \left[\lambda \boldsymbol{D}\right]_\Omega \tag{45}$$

$$\left[\boldsymbol{D}\bar{\boldsymbol{u}} - \boldsymbol{d}\right]_\Omega = 0 \tag{46}$$

to obtain the update direction $\boldsymbol{p} = \bar{\boldsymbol{u}} - \bar{\boldsymbol{u}}(k)$.

2. Take the largest possible step in direction $\boldsymbol{p}$ that does not break any inactive inequalities:

$$\bar{\boldsymbol{u}} = \bar{\boldsymbol{u}}(k) + \gamma \boldsymbol{p}$$

with $0 \leq \gamma \leq 1$ is opted to guarantee that the inactive inequality constraints are still feasible.

3. If $0 \leq \gamma \leq 1$ adding a limiting number of inequalities to the active set and return to one. Otherwise, take the full step $(\gamma = 1)$ and consider the sign of the Lagrange multipliers whether or not it satisfies the condition (43) and (44):

- If all the inequalities constraints have positive multipliers, terminate the algorithm
- Otherwise, remove the inequality which has the most negative multiplier from the active set and return to step 1.

Remark 2: Theoretically, the number of iterations depends on the initial active set Ω. If the initial active set is chosen properly, $\Omega_0 = \Omega^*$, the algorithm will be terminated in the first iteration. On the other hand, if the initial active set is considerably different from the desired one, many iterations are needed to get the optimal result.

After finishing the iteration step and obtaining the optimal control consequence $\bar{\boldsymbol{u}}^*$ only its first element is used to control the robot system in this sampling time. Then, the output variables are measured and the same process is carried out in the next time instant.

IV. Simulation and Results

Considering the robot model described by (4) and the QP problem is used to solve the trajectory tracking problem with the cost function in (21).

Let us consider our robot which has the limits in the amplitude of the control variables:

$$\boldsymbol{u}_{max} = -\boldsymbol{u}_{min} = \begin{bmatrix} 1 & (\text{m/s}) \\ 1 & (\text{m/s}) \\ \frac{\pi}{4} & (\text{rad/s}) \end{bmatrix} \tag{47}$$

For the tracking problem, we set the sample time is $\text{dt} = 0.2$ (s) and the initial position of the robot is $\boldsymbol{x}_0 = (0,0,0)^T$, and design the described path like the following:

$$\begin{aligned} x_r(t) &= 5sin\left(\frac{\pi}{20}t\right) \\ y_r(t) &= 5cos\left(\frac{\pi}{20}t\right) \\ \theta_r(t) &= -\frac{\pi}{20}t \end{aligned} \tag{48}$$

To implement the NMPC controller, ACADO [19] toolbox is employed which provides several built-in algorithms for an optimization problem. Besides, it also supports both MATLAB and C++ code interface which is convenient to deploy the proposed method in ROS and MATLAB environment.

A. Simulation Using MATLAB

In this section, the MATLAB simulations were conducted to show the effectiveness of the controller in tracking problems with limited control signals. The results show how well the controller performs in the tracking issue. In the simulation, we choose different prediction horizons $(N = 5, N = 10)$ to show their effect on the robot's behavior.

Fig. 2 shows the tracked paths of the controller with different horizons. The larger prediction horizon can provide the robot to track the reference path smoothly and early. With $N = 10$, the tracked path is smoother than with $N = 5$. Furthermore, the controller efforts are shown in Fig. 3. It reveals that the large prediction horizon leads to a smoother controller effort than the small one. Nevertheless, the larger prediction horizon leads the expensive computation. Therefore in practice, we need to consider the power of the hardware to choose the suitable prediction horizon.

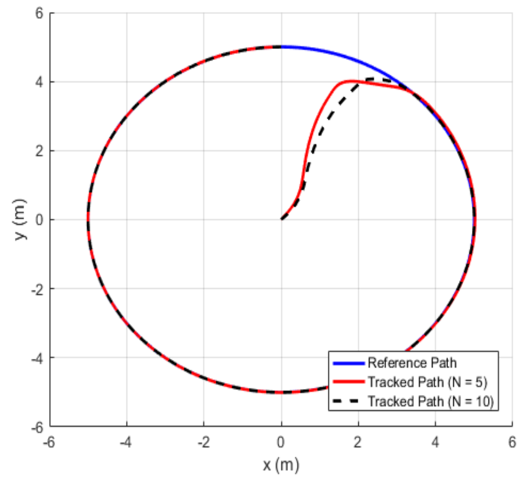

Figure 2. Tracked trajectories

Fig. 3 shows that the robot's linear and angular velocities track the target while the constraints are satisfied. The velocities, in the beginning, are upper bound until they approach the desired path; they converge to the velocities that track the reference path.

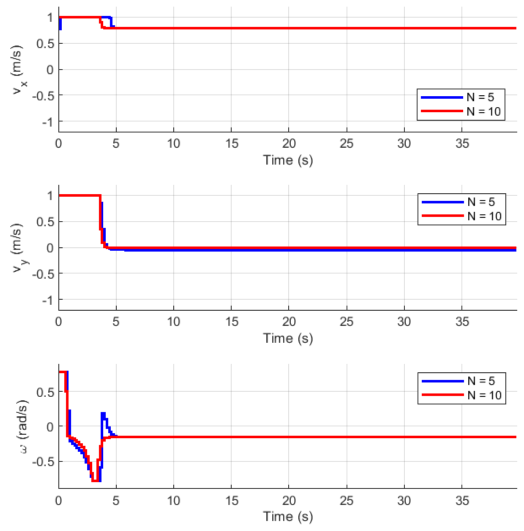

Figure 3. Output control signals of NMPC

The tracking error is converged to zero, shown in Fig. 4. Due to the robot's kinematic model, the convergence of x or y depends on the robot's initial states and orientation. As in Fig. 4, at the beginning of the movement, the error is significant because of initial states and the velocity constraints, and as time goes on, it converges to zero. With the larger prediction horizon, the tracking error converges to zero quickly. Furthermore, the error results can be further verified in Table I, which shows the position's root-mean-square error (RMSE). With $N = 10$, the RMSE of all the axis and the distance error are smaller than $N = 5$.

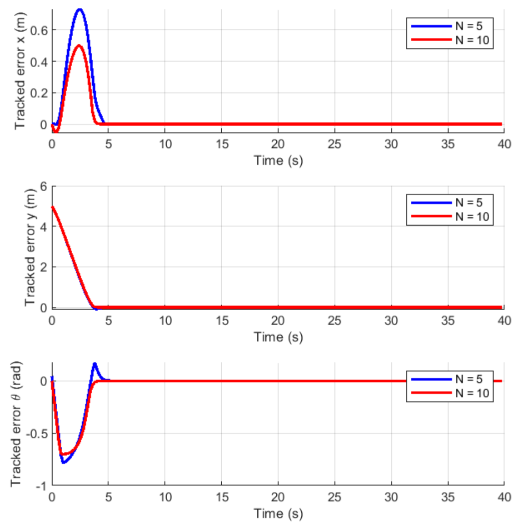

Figure 4. Tracking errors

TABLE I. THE RMSE OF THE CONTROLLER IN DIFFERENTIAL PREDICTS HORIZON

RMSE	$N = 5$	$N = 10$
The x-axis (m)	0.1611	**0.0991**
The y-axis (m)	0.9194	**0.9031**
Distance Error (m)	0.9318	**0.9285**

B. Simulation Using ROS and Gazebo

In this section, the simulations are conducted based on the Gazebo simulator. The visualization is performed by using Rviz to obtain the results. According to [21], [22], the weight matrices chosen must be positive. Thus, in this simulation, we choose the prediction horizon $N = 30$ and weight matrices:

$$\boldsymbol{Q}_N = \boldsymbol{Q} = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}, \boldsymbol{R} = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 5 \end{bmatrix} \quad (49)$$

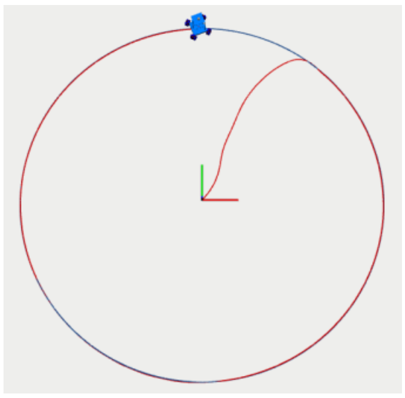

Figure 5. Tracking in the XY plane (RViz)

The obtained path of the robot is given in Fig. 5, where the blue line is the reference path and the red line is the robot path. The red axis denotes the Ox, and the green axis denotes the Oy. The control input signals and the tracking errors of the robot are shown in Figs. 6 and 7. The tracking error in the simulation is given as follow:

$$error = \sqrt{(x_r - x)^2 + (y_r - y)^2} \quad (50)$$

At the beginning of the movement, the vertical axis error increases because of the difference between initial states and the described trajectory and the velocities constraints. After that, the error tracking converges quickly to zero. It means the path was tracked.

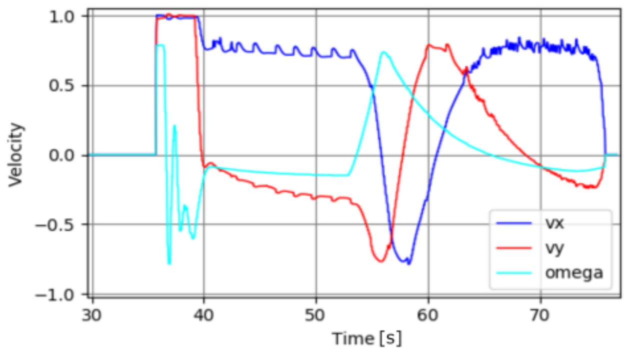

Figure 6. Control input of robot in RViz. vx (m/s), vy (m/s) are the linear velocities along the x, y-axis, respectively; omega (rad/s) is the angular velocity of the robot.

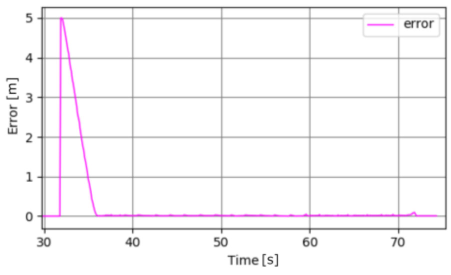

Figure 7. Distance error in Rviz. The error is satisfied following equation (50).

C. *Navigation Stack-Based NMPC*

After ensuring that the NMPC controller has been successfully installed on the ROS, the Navigation stack-based NMPC is built to control the robot instead of the TEB controller [23], [24].

The TEB controller is based on the method called Timed Elastic Band locally optimizes the robot's trajectory for trajectory execution time. Because TEB is based on optimal travel time, its trajectory tracking cannot be fully guaranteed. On the other hand, MPC can optimize the travel distance to follow the given trajectory while still avoiding obstacles in the environment.

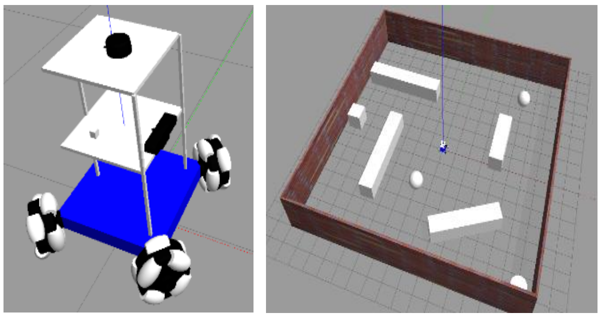

Figure 8. The 3D Omni robot (left) and simulation environment (right)

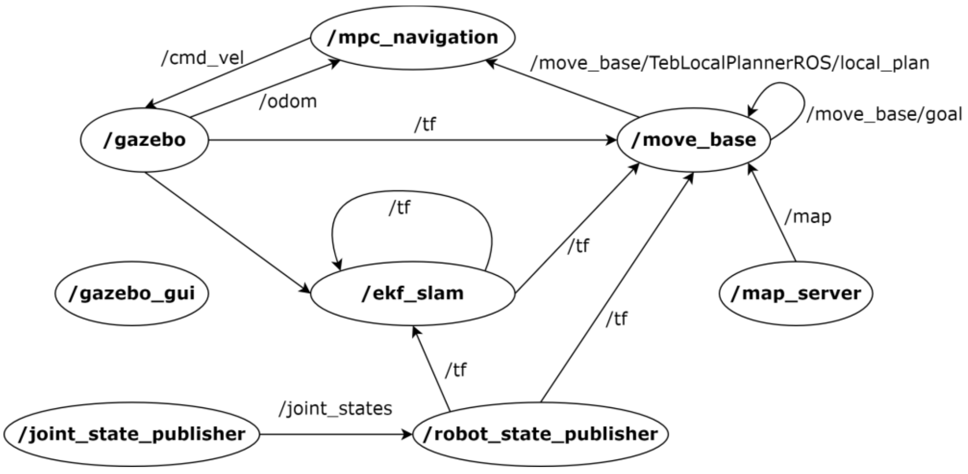

Figure 9. The diagram of Navigation stack-based NMPC

ROS is integrated with Gazebo simulation software. The environment in Gazebo is optimized so that the physical conditions are most similar to the actual environment. To simulate the behavior of the robot, the Omni robot model and the simulated environment were built to evaluate the quality of navigation using the proposed controller. The robot model and environment are shown in Fig. 8.

The diagram of the Navigation stack-based NMPC is shown in Fig. 9. In which, node */gazebo_gui* represents Gazebo simulator, node */gazebo* represents physical environment and robot; node */joint_state_publisher* and */robot_state_publisher* represent robot states in the RViz visualization tool; node */ekf_slam* is used to locate the robot in the environment; node */map_server* represents the map obtained by the robot during the previous mapping; node */move_base* aggregates information from the map, robot, and target location to create a trajectory, and node */mpc_navigation* is the node deployed to perform the trajectory tracking task. The NMPC is substituted for the TEB controller in robot control. As can be seen in Fig. 9, the NMPC node receives the local path signal from TEB and outputs the control signal to the robot.

The comparison between NMPC and TEB was performed to evaluate the performance of the Navigation stack-based NMPC. The global path is generated based on the A* planner and used as the robot's desired trajectory. The robot has an initial position $x_r = 0$, $y_r = 0$ and moves to the target position $x_g = 5$, $y_g = -5$.

The white pixel presented the movable space. The black pixel presented the wall, and the gray pixels presented the unknown space in the map. According to Fig. 10, the red path is the global path navigated to the target position. The yellow path is the local path designed from the NMPC controller. Besides, the local path of the TEB shown in the blue line is added to consider the movement trend of the robot at the same time to compare the local path of the proposed navigator with the TEB. As shown in Fig. 10, the local path of the NMPC navigator is always asymptotic to the global path, while the local path of the TEB tends to be skewed out. Because NMPC considers the Omni robot's kinematic model, NMPC can make optimal predictions to follow the global path. And TEB is designed for many robot models, so its navigability is more limited than the proposed navigator.

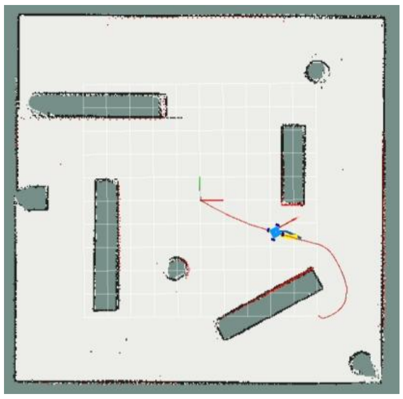

Figure 10. Navigation paths of the robot using NMPC

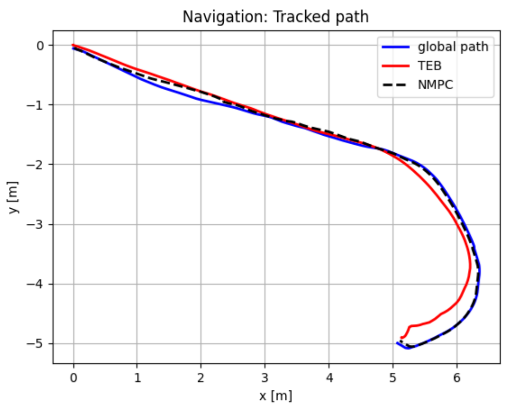

Figure 11. Record navigation paths of the robot

The robot's data under the direction of the two navigators were recorded to compare the performance of the two navigators. Fig. 11 shows the tracked path of the robot based on the NMPC and TEB navigators. According to Fig. 11, the robot's trajectory by using the NMPC navigator is asymptotic to the global path. Because the TEB Navigator is used for various robot models, and the proposed navigator is designed for the Omni robot model, the capabilities of the NMPC navigator are better. In addition, TEB is optimized in terms of robot travel time, and NMPC is optimized for tracking. Thus, the robot's trajectory using the proposed navigator has a better ability to follow the global path, while TEB will make priority moves to the destination, so it can be seen that the robot trajectory with the TEB navigator separates from the global path and moves towards the destination.

V. Conclusion

This paper has designed and built a trajectory tracking control system for the four-wheeled omnidirectional mobile robot based on NMPC. In addition, Navigation stack-based NMPC is also constructed to replace TEB in trajectory tracking control. The controller is built based on the kinematic model of the robot. The obtained results show the effectiveness of the NMPC controller in tracking and the navigation problem.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Hiep Do Quang, Cuong Nguyen Manh and Toan Nguyen Nhu designed the proposed algorithm; Hiep Do Quang and Thang Le Tran implemented and conducted the experiment; Toan Nguyen Nhu and Nam Bui Duy wrote the paper; Cuong Nguyen Manh and Tien Ngo Manh reviewed the paper.

Acknowledgment

This research was funded by Vietnam's national project “Research, develop an intelligent mobile robot using different types of sensing technology and IoT platform, AI, and implemented in radioactive environment monitoring application” of the CT1187 Physics development program in the period 2021-2025.

References

[1] K. V. Ignatiev, M. M. Kopichev, and A. V. Putov. “Autonomous omni-wheeled mobile robots,” in *Proc. 2nd International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*, 2016, pp. 1–4.

[2] M. Emam and A. Fakharian. “Path following of an omnidirectional four-wheeled mobile robot,” in *Proc. Artificial Intelligence and Robotics (IRAN OPEN)*, 2016, pp. 36–41.

[3] S. Wang, *et al*., “Trajectory tracking control for mobile robots using reinforcement learning and PID,” *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, pp. 1059–1068, 2020.

[4] Z. Yuan, *et al*. “Trajectory tracking control of a four mecanum wheeled mobile platform: An extended state observer-based sliding mode approach,” in *Proc. IET Control Theory & Applications*, 2019, pp. 415–426.

[5] N. T. Binh, *et al.*, “An adaptive backstepping trajectory tracking control of a tractor-trailer wheeled mobile robot,” *International Journal of Control, Automation, and Systems*, pp. 465–473, 2019.

[6] Kim, Duyen Ha Thi, *et al*. “Adaptive control for uncertain model of omni-directional mobile robot based on radial basis function neural network,” *International Journal of Control, Automation and Systems*, pp. 1-13, 2021.

[7] K. D. H. Thi, *et al*., “Trajectory tracking control for four-wheeled omnidirectional mobile robot using Backstepping technique aggregated with sliding mode control,” in *Proc. First International Symposium on Instrumentation, Control, Artificial Intelligence, and Robotics (ICA-SYMP)*, 2019, pp. 131–134.

[8] C. Wang, *et al*., “Trajectory tracking of an omnidirectional wheeled mobile robot using a model predictive control strategy,” *Applied Sciences*, p. 231, 2018.

[9] Ĩ. Moreno-Caireta, E. Celaya, and L. Ros, “Model predictive control for a mecanum-wheeled robot navigating among obstacles,” *IFAC-Papers On-Line*, pp. 119–125, 2021.

[10] K. Kanjanawanishkul, “MPC-Based path following control of an omnidirectional mobile robot with consideration of robot constraints,” *Advances in Electrical and Electronic Engineering*, pp. 54–63, 2015.

[11] J. Inthiam and C. Deelertpaiboon, “Self-localization and navigation of holonomic mobile robot using Omni-directional wheel odometry,” in *Proc. TENCON 2014 - 2014 IEEE Region 10 Conference*, 2014, pp. 1–5.

[12] H. D. Quang, *et al*., “Mapping and navigation with four-wheeled omnidirectional mobile robot based on robot operating system,” in *Proc. International Conference on Mechatronics Robotics and Systems Engineering (MoRSE)*, 2019, pp. 54–59.

[13] H. D. Quang, T. N. Manh, C. N. Manh, D. P. Tien, M. T. Van, K. N. Tien, and D. N. Duc, “An approach to design navigation system for omnidirectional mobile robot based on ROS,” *International Journal of Mechanical Engineering and Robotics Research*, pp. 1502-1508, 2020.

[14] K. Xu, *et al*., "Design and implementation of a ROS-based autonomous navigation system," in *Proc. IEEE International Conference on Mechatronics and Automation (ICMA)*, 2015, pp. 2220–2225.
[15] B. Cybulski, A. Wegierska, and G. Granosik, "Accuracy comparison of navigation local planners on ROS-based mobile robot," in *Proc. 12th International Workshop on Robot Motion and Control (RoMoCo)*, 2019, pp. 104–111.
[16] G. Campagne, "A nonlinear model predictive control based evasive manoeuvre assist function," Master thesis, Delft University of Technology, 2019.
[17] Z. Li, *et al*., "Trajectory-tracking control of mobile robot systems incorporating neural-dynamic optimized model predictive approach," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 740–749, 2016.
[18] Y. Ding, Z. Xu, J. Zhao, K. Wang, and Z. Shao, "Embedded MPC controller based on interior-point method with convergence depth control," *Asian Journal of Control*, vol. 18, pp. 2064–2077, 2016.
[19] B. Houska, H. J. Ferreau, and M. Diehl, "An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range," *Automatica*, pp. 2279–2285, 2011.
[20] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, 1999.
[21] Z. Yongjie and U. Ozguner, "Constrained model predictive control for nonholonomic vehicle regulation problem," *IFAC Proceedings*, vol. 41, 2008, pp. 9552-9557.
[22] K. Felipe, *et al*., "Mobile robot trajectory tracking using model predictive control," in *Proc. IEEE Latin-American Robotics Symposium*, 2005.
[23] M. P. Pablo, *et al*., "Global and local path planning study in a ROS-Based research platform for autonomous vehicles," *Journal of Advanced Transportation*, pp. 1-10, 2018.
[24] C. Roesmann, W. Feiten, T. Woesch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *Proc. ROBOTIK 2012; 7th German Conference on Robotics*, 2012, pp. 1-6.

Hiep Do Quang received the B.E in Instrumentation and Industrial informatics (2001), M.E (2005) in Instrumentation and Control from Hanoi University of Science and Technology (HUST). He is a lecturer, Faculty of Electrical Engineering. University of Economics-Technology for Industries, Viet Nam. The main researches: AI, robotics, automatic control.

Thang Le Tran graduated with a PhD in Engineering in System Analysis, Control and Information Processing from the Taganrog Insitute of Technology, Southern University - Russian Federation in 2008. Currently a researcher at Controls, Automation in Production and Improvement of Technology Institute, Hanoi, Vietnam. The main research direction is to design embedded control system, stable control and intelligent control.

Tien Ngo Manh, graduated Engineering Degree in automatic control at Hanoi University of Science and Technology (HUST) from 1996-2001. He received his Doctor's Degree in electrical engineering at HUST in 2014. Now, he works at the Institute of Physics, Vietnam Academy of Science and Technology. His main research areas include process control, adaptive control, fuzzy logic and neural network control, Robotics, electro-optical system, image processing.

Cuong Nguyen Manh, graduated Engineering Degree in Control Engineering and Automation at Hanoi University of Science and Technology (HUST), 2020. Now, he is a Master student at HUST, research engineer at Institute of Physics, Vietnam Academy of Science and Technology, and robotics engineer at VinAI Research. His main research area includes adaptive control, optimal control, fuzzy logic and neural network control, and robot operating system programming for robotics.

Toan Nguyen Nhu graduated from Hanoi University of Science and Technology, Vietnam. He is now a master's student at Hanoi University of Science and Technology, Vietnam. His interests includes intelligent control, optimal control, machine learning for robotic systems

Nam Bui Duy is an undergraduate student majoring in Robotics Engineering at the University of Engineering and Technology (UET). Now, he is working at the Institute of Physics, Vietnam Academy of Science and Technology. His main research area includes adaptive control, optimal control, and robot vision.