EVALUATING MODEL PREDICTIVE CONTROL IN DIFFERENTIAL DRIVE ROBOTICS: COMPREHENSIVE LITERATURE REVIEW

Student: Eppacher, Kevin, BSc. **PK:** 2310331013
Peer ReviewerIn: Novotny, Georg, MSc.

***Abstract*—This thesis explores the application of Model Predictive Control (MPC) algorithms for differential drive robots, focusing on dynamic obstacle avoidance and safe navigation. Given the rising importance of autonomous mobile robots in various sectors, from industrial to domestic applications, there is a critical need for advanced control systems that can ensure efficient and safe operation in dynamic environments. This study conducts a comprehensive literature review to assess current MPC methodologies, emphasizing control algorithms capable of adjusting to sudden obstacles and changing conditions. The research identifies the essential parameters and constraints that influence the effectiveness of MPC, including model dynamics, control inputs, and environmental interactions. Through a detailed examination of state-of-the-art MPC applications, the study highlights the strengths and limitations of existing approaches and suggests directions for future research. Specifically, the analysis reveals a gap in the implementation of MPC algorithms that can adeptly handle dynamic collision avoidance.**

***Keywords*—Model Predictive Control, Differential Drive Robots, Obstacle Avoidance, Autonomous Mobile Robots, Control Algorithms**

I. Introduction

Autonomous mobile robotic systems that move using wheels, tracks or legs are playing an increasingly important role in various areas of technology and daily life. Their ability to perform tasks independently has led to an impressive rise in their popularity and applications [1].

The areas of application of these robots are diverse: In industry and the service sector, they make a significant contribution to transporting goods safely and without collisions, for example as automated guided vehicles. In the military sector, they are essential for life-saving measures and for defusing bombs in crisis areas. Tracked robots and quadrupeds demonstrate special abilities in difficult terrain [2].

Mobile robots have also found a permanent place in everyday life. For example, they support households as intelligent vacuum cleaners or lawn mower robots, thereby helping to automate everyday tasks. In the area of autonomous passenger transport, self-driving cars are revolutionizing the way we travel and get around.

In the scientific study of mobile robots, the fulfillment of specific criteria and requirements is of central importance. These autonomous systems must be able to effectively compensate for internal and external disturbances [2]. A key aspect here is the use of control algorithms that can compensate for deviations in model parameters and dynamics [2]. This is particularly important in scenarios with a high risk of collision and under rapidly changing conditions. For example, mobile robots in industrial production lines must be able to react flexibly to sudden obstacles such as employees or objects. In such environments, any delay can have a significant impact, which is why rapid path recalculation is essential [3]. Similar challenges exist for service robots such as vacuum cleaners or lawn mowing robots operating in domestic environments. Here, unforeseen obstacles such as children playing or pets can block the robot's predetermined path [3]. The ability to react quickly and efficiently to such obstacles and adapt the route accordingly is of great importance for the safety and efficiency of these robots [3, 4].

The Model Predictive Control (MPC) method is often used to control these systems effectively. MPC is an advanced controller that makes it possible to predict with a model and control the future movement and behavior of the robot. To realize this, the MPC requires various parameters such as the dynamic or kinematic model of the robot, its current states, target functions and restrictions regarding speed, acceleration and obstacle avoidance [4].

This study provides a detailed literature review and guidance to determine the most suitable MPC algorithm for differential drive robots. The focus is on the development of a control algorithm that can dynamically avoid obstacles and ensure safe navigation in its environment. In addition, the study identifies existing research gaps in the current technological landscape and provides valuable insights into the areas that require further research and development.

This paper is divided as follows: Chapter 2 introduces the theoretical foundations of MPC and describes the kinematic model of the mobile robot under investigation. Chapter 3 explains the methodology, including the analysis of selected publications, to understand the application of MPC in differential drive mobile robots. Chapter 4 presents the current state of the art and discusses recent developments and challenges in MPC applications for mobile robots. The results, presented in Chapter 5, show different approaches to MPC trajectory tracking and obstacle avoidance. Chapter 6 summarizes the discussion and highlights the effectiveness of MPC and the need for further research. Finally, Chapter 7 provides a summary of the key findings and an outlook on future research directions. Chapter 5 discusses the State-of-the-Art and provides insights, into which approaches fit into a future implementation. Chapter 6

describes how a future implementation would be carried out.

II. Preliminary Information

This section provides a concise description of the theoretical concepts relevant to this paper, in particular concerning the operation of MPC and the selection of the kinematic model for the mobile robot under investigation.

A. Model Predictive Controller

An MPC is an advanced control approach that is particularly suitable for complex systems with significant constraints and safety requirements. At the core of an MPC system is a mathematical model of the system to be controlled, which describes the dynamics of the process or system. This model is typically given in the form of a state space representation in Equation 1.

$$x_{k+1} = Ax_k + Bu_k \tag{1}$$

Here is x_k the state vector of the system at the time k, u_k the control vector,A the system matrix, which describes the influence of the current state on the next state, and B the input matrix, which represents the influence of the control on the next state. The model is used to predict the future behavior of the system over a specific time horizon. These predictions are then fed into the cost function to determine the optimal control strategy, as can be seen in Fig. 1.

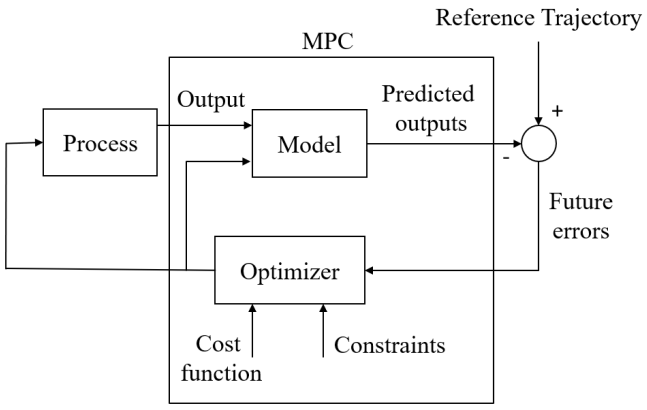

Fig. 1: Block diagram of an MPC [5]

The core functionality of the MPC lies in minimizing a cost function$J$ in Equation 2

$$\begin{aligned} J = \sum_{k=0}^{N-1} \big(&(x_{\text{ref},k} - x_k)^T Q (x_{\text{ref},k} - x_k) \\ &+ (u_{\text{ref},k} - u_k)^T R (u_{\text{ref},k} - u_k) \big) \end{aligned} \tag{2}$$

where x_k is the state vector of the system at the time k , u_k is the control vector, N is the prediction horizon and $x_(ref, k)$ or $u_(ref, k)$ are the reference states and controls. Q and R are positive definite weight matrices that quantify the relative importance of the states and controls in the cost function and have typically only diagonal entries. The optimization takes place via the control states u_k within the control horizon to minimize the cost function and achieve the desired system responses. The optimizer is the tool that solves the optimization problem to find the optimal control sequence. For linear MPC systems, quadratic programming is often used, while for NMPC, due to the nonlinear (NL) system dynamics, nonlinear programming methods such as Sequential Quadratic Programming (SQP) or interior point methods are used.

The optimizer considers both hard and soft constraints. Hard constraints, such as maximum velocities or forces, are integrated directly into the optimization problem and must be strictly adhered to. Soft constraints, such as preferred states or controls, are integrated into the cost function and weighted by penalty terms so that minor violations are tolerated but penalized.

The advantages of MPC include the ability to control multivariable systems (MIMO), the incorporation of state and control constraints, the consideration of obstacle avoidance, and the flexibility to deal with nonlinear systems and changing model parameters, leading to the development of adaptive MPC systems.

Potential disadvantages are the computing capacity requirements, particularly for nonlinear MPC (nMPC) or adaptive MPC systems. However, modern computers and algorithms have evolved significantly so that these challenges can increasingly be overcome.

MPC can deal with nonlinear systems by using the nonlinear model within the prediction horizon. This extends the applicability of MPC to a wide range of systems that cannot be adequately described by linear models. Furthermore, the model within the MPC can be adaptively adjusted to dynamically account for changes in system behavior or the environment, resulting in an adaptive MPC. This flexibility makes MPC a powerful tool for controlling complex dynamic systems.

B. Robot State-Space Model

The state space model of a differential drive robot is described by a set of Equations that define the relationship between the control inputs, the motion parameters, and the robot's position in space. The model is based on the kinematic properties of the robot, as shown in Fig. 2.

Equation 3 connects the linear velocity v and the angular velocity ω of the robot with the rotational speeds of its wheels ω_l and ω_r. Here, r_R and r_L represent the radii of the right and left wheels, respectively, and L represents the distance between the wheels. This relationship allows the robot's movement to be controlled based on wheel speeds.

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \frac{1}{2} \begin{bmatrix} r_R & r_L \\ \frac{r_R}{L} & -\frac{r_L}{L} \end{bmatrix} \begin{bmatrix} \omega_l \\ \omega_r \end{bmatrix} \tag{3}$$

In comparison to Equation 3, describes Equation 4 the individual wheel velocities, which is effective for control algorithms, to move the robot to a target location.

$$\begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix} = \begin{bmatrix} \frac{1}{R_r} & \frac{L}{2R_r} \\ \frac{1}{R_l} & -\frac{L}{2R_l} \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{4}$$

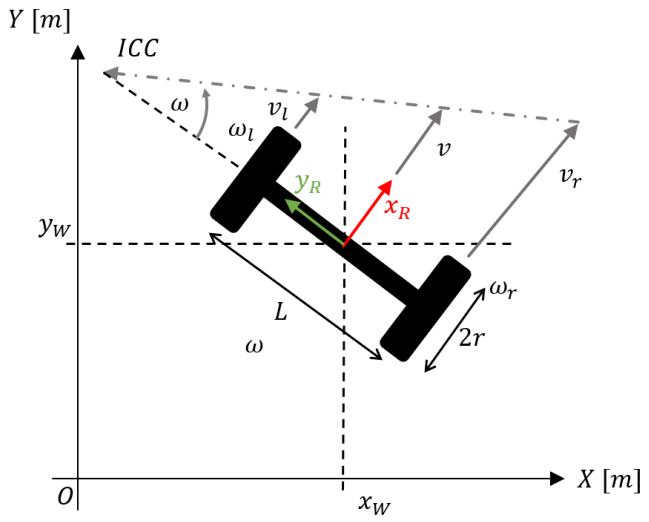

Fig. 2: Kinematization of a differential drive Robot [2]

R_r and R_l are the individual wheel radii and $2L$ is the defined as the wheelbase. The state variables of the robot is defined by Equation 5 and contains the position x and y as well as the orientation θ of the robot. The control variables, given in Equation 6, consists of the linear velocity equation and the angular velocity ω, which control the movement of the robot [2].

$$\mathbf{x} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T \tag{5}$$

$$u = \begin{bmatrix} v & \omega \end{bmatrix}^T \tag{6}$$

Equation 7 describes the rates of change of the robot's position and orientation with respect to its current orientation and speed. These dynamics are crucial for predicting the robot's future position and orientation based on its current motion parameters. The linear velocity v is defined in Equation 8 and the angular velocity ω is defined in Equation 9 [2].

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{7}$$

$$v = \frac{v_l + v_r}{2} = \frac{\omega_l r_l + \omega_r r_r}{2} \tag{8}$$

$$\omega = \frac{v_l - v_r}{L} \tag{9}$$

Finally, the discrete-time model in Equation 10 describes the evolution of the state vector over time. This model allows the calculation of the robot's future position and orientation at discrete time intervals Δt, which is essential for planning and controlling robot movement in dynamic environments [1].

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + \Delta t \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} u_k \tag{10}$$

III. Methods

In this literature study, selected publications are evaluated about specific criteria to gain a deeper understanding of the implementation of MPC in mobile differential-driven robots. The central focus is the analysis of the inputs and outputs of the MPC systems, the investigation of whether the modeling is based on first principles approaches or data-driven methods, as well as the evaluation of the selected weights for the matrices Q and R [2, 4]. Furthermore, both the defined soft - and hard constraints, especially with regard to collision avoidance, as well as the permanent errors and settling times of the systems are examined. Particular attention is paid to the robustness of the systems against internal and external disturbances as well as to the implementation of special features and functions for use in mobile differential-driven robots [6]. Additionally, how the systems were implemented in practice, including use in simulations or on physical hardware, is discussed. This comprehensive analysis determines the optimal control strategies and improves application efficiency in real-world scenarios.

IV. State of the Art

The research presented in [7] delves into the integration of robust MPC with adaptive control to enhance trajectory tracking in mobile robotics. This approach employs a kinematic model specifically designed for differentially driven robots, addressing inherent challenges such as model uncertainties and external disturbances. Key to this strategy is the inclusion of a disturbance observer alongside an adaptive controller, which collectively ensure precise and robust trajectory adherence. The control inputs, defined as $u = [u_1, u_2]^T$ in the study, where $u_1 = v_r \cos(\theta_e) - v$ and $u_2 = \omega_r - \omega$, represent adjustments to the robot's linear and angular velocities to maintain its desired path. Furthermore, the study outlines the robot's state vectors and emphasizes the differential drive kinematics through specific equations, with the MPC's effectiveness underpinned by carefully chosen weighting matrices $Q = 300I, 100I$.

Implementing soft constraints, the research ensures that control inputs, input increments, and state variables remain within predefined limits, significantly contributing to the system's reliability and performance. This meticulous control framework enabled the robot to accurately follow a complex figure-8 trajectory with minimal error, showcasing the strategy's effectiveness. Notably, the study's robustness analysis, particularly the incorporation of a disturbance observer, highlights the system's capability to adapt to and compensate for unexpected deviations, ensuring consistent performance. Unlike many studies that rely on simulation environments for validation, this research demonstrates the practical applicability of its findings through direct implementation on a two-wheeled service robot, reinforcing the relevance and potential of the proposed control strategy in real-world applications.

In [8] a MPC is developed and implemented for a differential driven mobile robot. The focus is on tracking a moving target and avoiding obstacles, and the system integrates aspects

of dynamic collision avoidance to enable safe and efficient navigation. As in the previous paper [7], the control variables in Equation 6 and state variables in Equation 5 are defined the same as in [7] and the same model as was used for the kinematic model used in Equation 7. The weighting matrices Q and R are not defined in [8] . In [8] all hard constraints were implemented for obstacle avoidance and physical limitations. The linear velocities v were limited between -0.1 and 0.1 m/s and the angular velocities were limited between -0.2 and 0.2 rad/s. The vector increment Δu in [8] is constrained to be greater 0.

The obstacle avoidance system in [8] is based on a switching algorithm that adjusts the robot's orientation when it encounters obstacles. The algorithm defines two important parameters: the safe distance and the switching distance. The safe distance is the minimum distance the robot must maintain from an obstacle, while the switching distance is the point at which the robot must change its orientation to maintain the safe distance. This orientation adjustment allows the robot to avoid obstacles while continuing to track a moving target. It is important to mention that the positions and sizes of the obstacles are known and are not determined automatically. The MPC has a settling time of 5 seconds with an error of 1 m and the error difference converges to 0 over time. The MPC from [8] is very robust because it automatically avoids known obstacles and calculates a new path this is also the most important technical feature.For the optimization problem, the author in [8] utilized a Quadratic Programming (QP) Algorithm to optimize the control inputs u_k. This approach is facilitated by linearizing the state-space equations at their operating points at each time-step, which results in a convex optimization problem. The convexity of the problem is crucial as it guarantees that any local minimum found by the QP algorithm is also a global minimum, ensuring the optimality of the solution. This method allows for the efficient and effective computation of control actions that navigate the mobile robot towards its target while dynamically avoiding obstacles and respecting the system's constraints. The simulation of MPC was carried out in MATLAB. As outlook in [8] it is suggested that the Obstacle Avoidance algorithm can be extended to not only avoid static objects, but also identify dynamic objects and avoid them in path planning.

The aim of the paper [9] is to develop and demonstrate an approach that integrates trajectory tracking and obstacle avoidance for non-holonomic mobile robots using MPC. It aims to ensure efficient navigation and maneuverability in dynamic environments. In comparison to the other papers, the state variables are again defined in the same way as in Equation 5, but in [9] the control variables are described differently, which is represented in Equation 11.

$$u = u_f + u_b = \begin{bmatrix} v_f \cos\theta + v_b \\ \omega_r + \omega_b \end{bmatrix} \tag{11}$$

In Equation 11 u_f is the feedforward input vector and u_b is the feedback input vector. The control variable is the same as in Equation 6 as well as the robot model in Equation 7. The control variables are limited with an upper and lower bound constraint and the minimum and maximum input increments are also limited.

The obstacle avoidance mechanism is predicated on the establishment of a potential field around each obstacle, characterized by two critical radii: the Obstacle Avoidance Area Radius (L) and the Obstacle Area Radius (l), where $L > l > 0$. This potential field is mathematically represented by a collision detection function, V_a, defined as:

$$V_a = \left(\min \left\{ 0, \frac{d_a^2 - L^2}{d_a^2 - l^2} \right\} \right)^2 \tag{12}$$

In (12), d_a denotes the distance between the robot and the closest point on the obstacle perimeter, calculated by:

$$d_a = \sqrt{\left(\frac{x - x_a}{\alpha} \right)^2 + \left(\frac{y - y_a}{\beta} \right)^2} \tag{13}$$

The parameters (x_a, y_a) represent the coordinates of the obstacle's center, whereas (x, y) denote the robot's current position. The terms α and β are shaping parameters that influence the potential field's geometry, facilitating a smoother transition around obstacles.

The control inputs are then modulated based on the gradient of the potential field with respect to the robot's position. This modulation is captured by the following equations:

$$X_e = x_e + \frac{\partial V_a}{\partial x} \tag{14}$$

$$Y_e = y_e + \frac{\partial V_a}{\partial y} \tag{15}$$

Here, X_e and Y_e are the error components influenced by the obstacle's presence, guiding the robot to adjust its trajectory to avoid collisions. The terms $\frac{\partial V_a}{\partial x}$ and $\frac{\partial V_a}{\partial y}$ represent the partial derivatives of the collision detection function with respect to the robot's x and y coordinates, respectively.

By integrating (14) and (15) into the MPC's optimization problem, the robot dynamically prioritizes between closely following the desired trajectory and deviating from it to circumnavigate obstacles. This integrated approach ensures that the robot can navigate through cluttered environments with a high degree of safety and efficiency, as demonstrated by the simulation results.

In [10] a nonlinear MPC (nMPC) approach for controlling a differential driven mobile robot (DDMR) in dynamic environments is presented. The DDMR Model is described in [10] by polar coordinates in Equation 16 instead of by Cartesian coordinates.

$$\begin{cases} e = \sqrt{x^2 + y^2} \\ \phi = \text{atan2}(y, x) \\ \alpha = \phi - \theta \end{cases} \tag{16}$$

where e represents the distance between the robot and the target in the plane, ϕ denotes the angle of the target relative to

the robot's current heading, and α signifies the heading error, calculated as the difference between the target angle ϕ and the robot's orientation θ. The derivation can be seen in Figure 3.

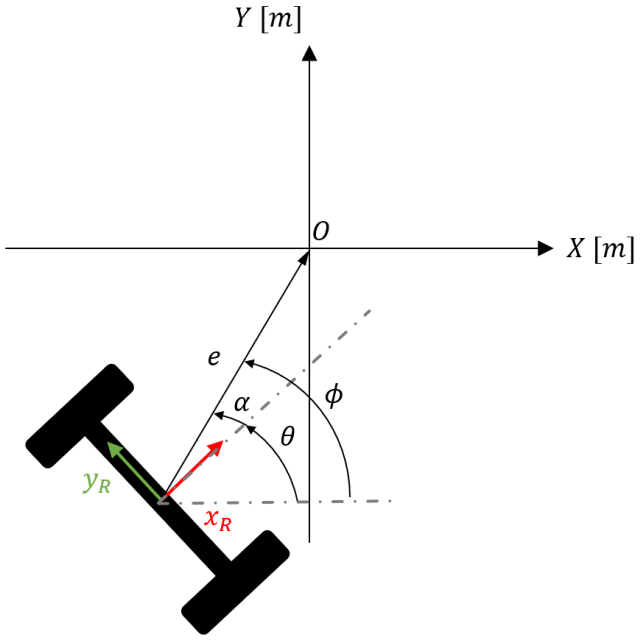

Fig. 3: DDMR described by Polar Coordinates [10]

Polar coordinates provide an intuitive representation of direction and distance. The switch to polar coordinates satisfies the global accessibility rank condition for non-holonomic systems, which facilitates the development of control strategies. This results in the nonlinear state space model in Equation 17

$$\begin{bmatrix} e(k+1) \\ \phi(k+1) \\ \alpha(k+1) \end{bmatrix} = \begin{bmatrix} e(k) \\ \phi(k) \\ \alpha(k) \end{bmatrix} + \Delta t \begin{bmatrix} v(k)\cos(\alpha(k)) \\ -\frac{v(k)\sin(\alpha(k))}{e} \\ \frac{v(k)\sin(\alpha(k))}{e(k)-\Delta T\omega(k)} \end{bmatrix} \tag{17}$$

A main drawback is the singular state, in which the distance e from the DDMR to the goal is 0. The user must make sure, to never get in this state [11]. Polar coordinates are also used in [11] and [12] to describe the kinematics of the DDMR. The cost function of nMPC is represented in Equation 18.

$$J(x_k, y_k) = \phi(x(N)) + \sum_{k=0}^{N-1} l(x_e(k), u(k)) \tag{18}$$

where $l = l_{\text{reg}} + l_{\text{obst}}$ is the cost function. The cost function is divided into the regular cost, l_{reg}, and the cost for obstacle avoidance, l_{obst}, as a soft constraint. The regular cost function is represented in Equation 19

$$\begin{aligned} l_{\text{reg}}(x(k), u(k)) =& x_e(t+k)^\top Q x_e(t+k) \\ &+ u(t+k)^\top R u(t+k) \end{aligned} \tag{19}$$

where x_e is the error between the current DDMR pose and the reference pose, u is the control input, and Q and R are the respective weighting matrices for the state error and control input. The obstacle avoidance cost function, l_{obst}, is defined in Equation 20

$$l_{\text{obst}}(x(k), u(k)) = \frac{s}{\left([d^i(t+k)]^\top d^i(t+k) + \varepsilon\right)} \tag{20}$$

where $d^i(t+k)$ is the distance from the DDMR to the i-th detected obstacle at time k, with a weighting factor s and ε is a small number introduced to prevent a singular state where the cost function converges to infinity. Hard constraints are set on the control states.

In [13] a nMPC was presented for a DDMR with obstacle avoidance. The author selected the nMPC because the Model can be better described than for the conventional MPC, due to the fact that it does not linearize at an operating point. The prediction is more accurate, but also the nonlinear Model has a drawback, for being computationally expensive. For the conventional MPC, a convex optimizer can be selected, which is advantageous due to its well-established theoretical foundations, reliable convergence properties, and availability of efficient algorithms, ensuring robust and computationally efficient solutions for linear systems. The cost function is the same as in Equation 18. The mayor difference from this paper from the other authors, is the hard constraints for obstacle avoidance, which is defined in Equation 28.

$$-\sqrt{(x_R - x_{\text{obst}})^2 + (y_R - y_{\text{obst}})^2} + (r_R + r_{\text{obst}}) \leq 0 \tag{21}$$

where x_R and y_R represent the robot's current position, x_{obst} and y_{obst} denote the position of the obstacle, and r_R and r_{obst} are the safety radii of the objects. In the author's case, the objects are static and predefined in the environment. In [13], the author used the Interior-Point-Method as the optimizer, implemented with the IPOPT Library from CasADi.

In [14], a dynamic model is used instead of a kinematic model for a DDMR. The authors choose a dynamic model, represented by the Equation 22

$$M(q)\ddot{q} + V(q, \dot{q})\dot{q} + B(q)\tau = 0 \tag{22}$$

where $M(q)$ is the moment of inertia matrix, $V(q, \dot{q})\dot{q}$ represents the Coriolis and centrifugal matrix, $B(q)$ is the input matrix, and τ is the input vector. This approach enables more accurate modeling of the robot's physical forces and movements, resulting in more precise control and better performance.

The authors derive the state space vector from this dynamic model $x = [x_c, y_c, \theta, \theta_R, \theta_L, \dot{\theta}_R, \dot{\theta}_L]^T$, which describes the movement of the DDMR. The state space dynamic model of [13] is represented in Equation 23

$$\dot{x} = \begin{bmatrix} S\eta \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & I_{2\times 2} \end{bmatrix} \overline{M}^{-1}(\tau - \overline{V}) \tag{23}$$

where $\eta = \begin{bmatrix} \dot{\phi}_R & \dot{\phi}_L \end{bmatrix}^T$ are the individual wheel velocities and $S(q)^T$ is a transformation matrix. To handle the nonlinear dynamic model within the framework of the NMPC approach, the authors use an input-output linearization technique that transforms the system into a linear form to which the conventional MPC can then be applied. This linearization allows the use of standard optimizers and greatly simplifies the optimization problem.

V. Discussion and Results

In [11], [12] and [10] polar coordinates were used for the state space model of the DDMR, instead of cartesian coordinates. The polar coordinates have the advantage of satisfying the global accessibility rank condition for non-holonomic systems, facilitating effective control despite complex motion constraints. On the other hand, this approach has the main disadvantage, that when the distance between the robot and the goal state is 0, then the robot is in a singular state, and the orientation is undefined. For this reason, cartesian coordinates will be used in the implementation.

The literature often suggests the use of a kinematic model [7, 8, 10] instead of a dynamic model for the control of differential drive mobile robots. A kinematic model that predicts the robot's movement based on its current position and orientation provides a simpler and less computationally intensive alternative to modeling the complex physical forces and movements that would need to be accounted for in a dynamic model. This enables faster implementation of control strategies and more efficient calculation of control commands. In addition, a kinematic model is often sufficiently accurate for many mobile robotics applications, particularly for tasks such as path following and obstacle avoidance. By using a kinematic model, the complexity of the control problem can be reduced, which can lead to improved performance and robustness of the control system

The conventional MPC is a Linear-Time-Invariant (LTI) System, which means, that its model, constraints and cost function are linear and do not change over time. This leads to a convenient optimization problem, in which the cost function is convex [4]. Nonlinear systems can be operated with the linear MPC, when either the nonlinear System is derived at an operating point (adaptive MPC) or is linearized for predefined operating points, in which the model behaves linearly (adaptive MPC). When the model is highly nonlinear, the constraints are nonlinear or the cost function is nonlinear, then the nMPC is suitable for these applications. In the following table are all MPC types compared for specific criteria [15].

In this work, the nMPC is selected, due to the Long-Term Prediction and for the nonlinear hard-constrained obstacle avoidance and for the nonlinear cost function.

There is a wide range of optimizers for Model Predictive Control (MPC), which are listed in Table 2 . In the context of MPC, Quadratic Programming (QP) solves optimization problems where the objective function is quadratic, and constraints are linear. As we progress to more complex and nonlinear control scenarios, other optimizers like Sequential Quadratic Programming (SQP), Interior Point Methods (IPM), Gradient-Based Methods (GBM), Genetic Algorithms (GA), and Particle Swarm Optimization (PSO) come into play, each with its own strengths and limitations. Table 3 offers a concise overview of these optimizers, emphasizing their efficiency, flexibility, scalability, and capacity for handling nonlinearities.

Tab. 1: MPC comparison for DDMR

Criterion [4]	MPC	Adaptive MPC	Gain Scheduled MPC	nMPC
Linear Systems	✓	✓	✓	✓
Nonlinear Systems	×	✓	Partially ✓	✓
Low Computational Demands	✓	×	✓	×
Adaptability to Changes		×	Partially ✓	✓
Long-Term Prediction Accuracy	×	×	×	✓

Tab. 2: Optimizer comparison for MPC

Optimizer / MPC Type	MPC	Adaptive MPC	Gain Scheduled MPC	nMPC
QP [8]	✓	×	✓	×
SQP [16]	✓	✓	✓	✓
IPM [13]	✓	✓	✓	✓
GBM [17]	✓	✓	✓	×
GA	×	×	×	✓
PSO [18]	×	×	×	✓

QP is efficient for linear problems but struggles with nonlinearity. SQP and IPM offer a balance, handling nonlinear problems well with good flexibility and scalability. GBM is a middle-ground option, while GA and PSO are less efficient but excellent at navigating complex nonlinear landscapes.

Tab. 3: Summary of Optimizer Characteristics in MPC

Optimizer	**Efficiency**	**Flexibility**	**Scalability**	**NL handling**
QP [8]	High	Low	Medium	Poor
SQP [16]	Medium	High	Medium	Good
IPM [13]	High	High	High	Excellent
GBM [17]	Medium	Medium	Medium	Good
GA [19]	Low	Medium	Low	Excellent
PSO [18]	Low	Medium	Low	Good

For a DDMR with nMPC and obstacle avoidance, choosing IPM is advantageous due to its superior efficiency in handling the nonlinear dynamics and complex constraints, crucial for real-time performance and precision necessary in obstacle avoidance scenarios.

VI. Summary and Outlook

Given recent developments in MPC´s, particularly when integrated into collision avoidance mechanisms, have demonstrated significant effectiveness in the context of differential-driven robotic platforms. However, the area of dynamic collision avoidance in the context of MPC algorithms remains fertile ground for further scientific investigations. As a future direction for academic exploration, it is suggested that subsequent research efforts could methodologically address this challenge.

To illustrate, consider Figure 4, which presents a comprehensive schema for deploying nMPC.

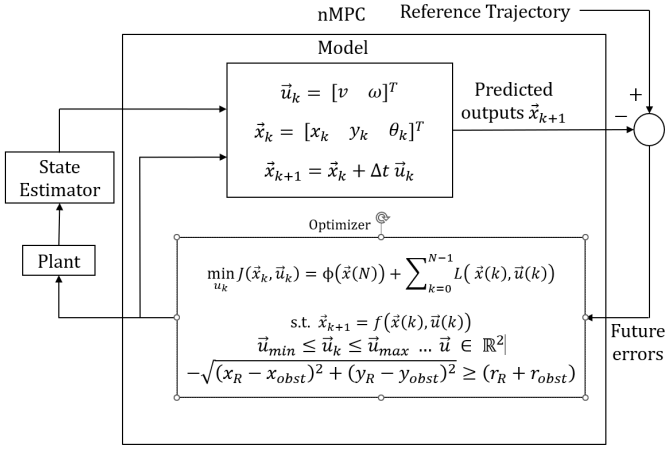

Fig. 4: Blockdiagram of nMPC for future Implementation

The model for the nMPC is described in Equation 10 with u_k described in Equation 6. From the Forwards-kinematic Model, the Robots velocities are calculated in 3. When the optimal control sequence is calculated, the Inverse Kinematic Model in 4 is used to calculate the individual wheel velocities.

The cost function in Equation **??** is divided into the terminal cost $\phi(x(N))$ in Equation 25, which is penalized at the end of the prediction horizon N and the continuous cost $L(x(k), u(k))$ represented in Equation 26.

$$\min_{\vec{u}_k} J(\vec{x}_k, \vec{u}_k) = \phi(\vec{x}(N)) + \sum_{k=0}^{N-1} L(\vec{x}(k), \vec{u}(k)) \tag{24}$$

$$\phi(\vec{x}(N)) = \frac{1}{2} \vec{x}_N^T S \vec{x}_N \tag{25}$$

The cost function in Equation 26 aims to minimize the difference between the current state $x_{R,k}$ and the reference state $x_{\text{ref},k}$ over a future time horizon from k to $k+N-1$. This is achieved by minimizing the squared difference of these states, weighted by a matrix Q. In addition, the use of the control actions $u(k)$ is penalized by a term $u_k^T R u_k$, where R represents the weight of this penalty.

$$L(\vec{x}(k), \vec{u}(k)) = \frac{1}{2} \sum_{k}^{k+N-1} (\vec{x}_{R,k} - \vec{x}_{\text{ref},k})^T Q (\vec{x}_{R,k} - \vec{x}_{\text{ref},k}) + \vec{u}_k^T R \vec{u}_k \tag{26}$$

The formulation of this cost function shows that it aims to keep the state deviations from the reference values $x_{R,k} - x_{\text{ref},k}$ as small as possible, which means that the control approach tries to keep the system as close as possible to the desired state $x_{\text{ref},k}$. At the same time, the dependence on the control states should be kept to a minimum in order to increase the efficiency of the system and avoid excessive energy consumption or wear and tear. The weighting matrices Q and R play a crucial role as they determine the balance between minimizing the state deviation and utilizing the control actions.

For safety reasons, strict limits on the control variables u_k are set in Equation 27 to avoid oversaturation. These limits ensure that stays within a defined range.

$$\vec{u}_{\min} \leq \vec{u}_k \leq \vec{u}_{\max}, \quad \forall k \quad \dots \quad \vec{u} \in \mathbb{R}^2 \tag{27}$$

MPC is distinguished among various control strategies for its capability to forecast future system behaviors and adjust control inputs accordingly. A significant advantage of MPC is its ability to integrate obstacle avoidance mechanisms, a critical aspect in autonomous navigation. This integration is facilitated through the incorporation of specific constraints, such as the collision avoidance condition outlined in Equation 28 from [13].

$$-\sqrt{(x_R - x_{\text{obst}})^2 + (y_R - y_{\text{obst}})^2} + (r_R + r_{\text{obst}}) \leq 0 \tag{28}$$

In this equation, $(x_{\text{obst}}, y_{\text{obst}})$ denotes the position of an obstacle, and $r_r + r_{\text{obst}}$ represents the sum of the radii of the robot and the obstacle, ensuring a safety margin for avoidance.

The implementation of such constraints within MPC frameworks raises concerns regarding computational efficiency, particularly when utilizing occupancy maps to represent environments. Occupancy maps, which mark the presence of obstacles within discrete grid cells, can become computationally demanding in extensive environments. To mitigate this, a localized search radius around the DDMR can be employed to identify nearby obstacles. An even more efficient approach involves detecting features, for example lines, circles, rectangles and ellipses. This method reduces the number of individual elements the MPC must consider, significantly enhancing computational efficiency. Several algorithms for this aggregation process are compared in Table 4, highlighting their effectiveness in various scenarios.

Tab. 4: Comparison of Additional Criteria

Methods	Comp. Demand	Adaptability	Localization
Hough [20]	Med-High	Low	High
Canny [21]	Med	Low	Med
RANSAC [22]	Med-High	Med	High
Template [23]	Low-Med	Low	Med
CNN [24]	High	High	High
Ext. Tracking [25]	High	High	High

Table 5 shows which shapes can be detected by which algorithm, with the primary aim, for circle, rectangle and ellipse shapes.

The comparative analysis of feature detection algorithms reveals distinct methodologies and applications. The Hough Transform, adept at detecting simple geometric shapes, utilizes a parameter space transformation to identify accumulations representing lines and circles. The Canny Edge Detector employs a multi-stage process focusing on edge detection

Tab. 5: Object recognition using various methods

Methods / Objects	Line	Circle	Rectangle	Ellipse
Hough [20]	✓	✓	×	×
Canny [21]	✓	×	×	×
RANSAC [22]	✓	✓	✓	×
Template [23]	✓	✓	✓	✓
CNN [24]	✓	✓	✓	✓
Ext. Tracking [25]	×	×	×	×

through gradient calculation and non-maximum suppression, offering precise edge localization but limited adaptability. Random sample consensus (RANSAC), an iterative approach, excels in modeling geometric shapes within noisy datasets by hypothesizing models from random data subsets and validating these against the entire dataset, showcasing moderate adaptability and high localization accuracy. Template Matching, by sliding a template image over the input image, identifies matching areas, though its effectiveness is constrained by variations in scale and lighting. Convolutional Neural Networks (CNNs), through deep learning, adaptively learn features from images, demanding high computational resources but providing unparalleled adaptability and object recognition capabilities. Extended Object Tracking (Ext. Tracking), essential for tracking objects spanning multiple pixels, combines object extent and kinematics for effective tracking in scenarios like autonomous driving.From Table 4 and Table 5 it can be seen, that RANSAC fullfils the requirements. However, the computational cost increases for large maps. Therefore, the Implementation for the RANSAC Algorithm is only performed in the nearby search area of the DDMR, as illustrated in Figure 5. In future Implementations, the search area could be as large, as the long the prediction horizon of the nMPC is. In Addition, the search area also could be limited from a minimum and maximum range, in which the laser beams of the LIDAR reach. The LIDAR Measurements need to be converted from Polar into Cartesian coordinates, for the use of the RANSAC algorithm. The computational effectiveness must be tested in future Implementations.

Drawing a parallel to the established Move Base Stack within the Robot Operating System (ROS) architecture, one could imagine the integration of a global planner, such as an A* algorithm, tasked with generating intermediate way-points to a specific destination is. These way-points would in turn serve as navigation beacons for the local planner of the DDMR, here represented by the nMPC. The integration of nMPC as a local planner is expected to provide the system with the ability to circumvent dynamically evolving obstacles, thereby improving its operational robustness in complex environments, as illustrated in Figure 6.

The accuracy of state estimation directly influences the control performance. State estimators provide the nMPC with reliable estimates of the system's states, which are crucial for predicting future states and making informed control decisions.

The Extended Kalman Filter (EKF) linearizes around an estimated mean and covariance, providing a balance between

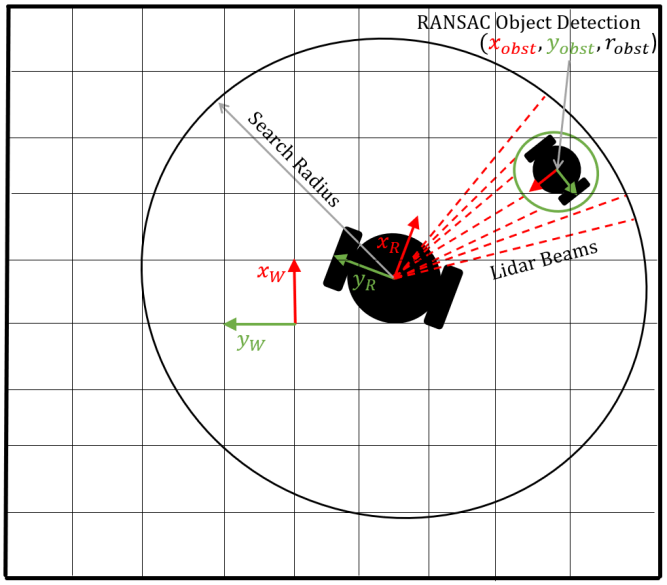

Fig. 5: Sketch for LIDAR RANSAC Object Detection in combination with nMPC

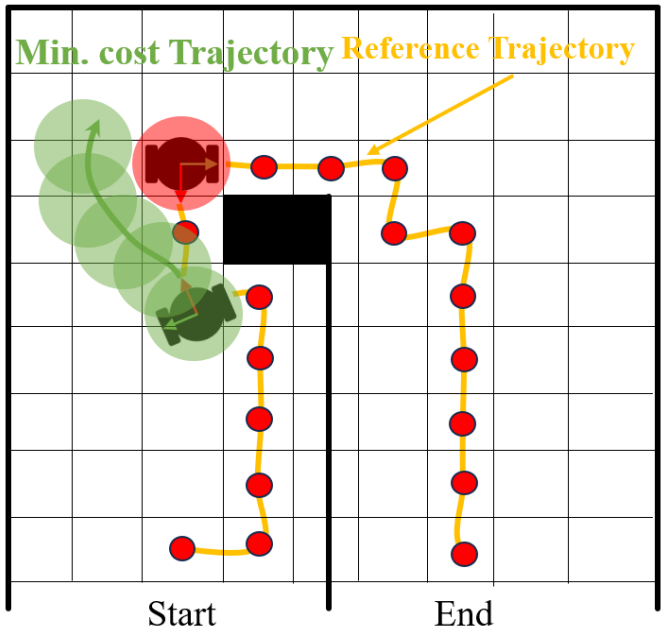

Fig. 6: Example for MPC implementation with dynamic collision avoidance

computational demand and estimation accuracy for mildly nonlinear systems. The Particle Filter (PF) employs a set of random samples or "particles" to represent probability distributions, offering robustness in highly nonlinear scenarios with the trade-off of increased computational load. Graph-Based SLAM (GB SLAM) constructs a graph where nodes represent robot poses or landmarks, and edges represent spatial relationships, excelling in large-scale mapping due to its high scalability and accuracy. FastSLAM combines particle filtering

with a landmark-based map representation, efficiently handling the SLAM problem by decoupling the pose estimation from landmark positions. ORB-SLAM, utilizing Oriented FAST and Rotated BRIEF (ORB) features, stands out for its exceptional performance in large environments and its ability to relocalize and close loops efficiently.

Tab. 6: Comparison of State Estimator Algorithms for Mobile Robot Localization

Algorithm	Accuracy	Computation Time	Error Robustness	Scalability
EKF [26]	High	Medium	Medium	Low
PF [27]	High	High	High	Medium
GB SLAM [28]	Very High	Variable	High	High
FastSLAM [29]	High	Medium	High	High
ORB-SLAM [30]	Very High	Low	High	Very High

Given these considerations, GMapping in ROS is particularly favored for robotic applications due to its effective balance of accuracy and computational efficiency. GMapping, based on Grid-based FastSLAM, provides a practical solution for mobile robots by integrating the strengths of FastSLAM with a grid-based map representation. This combination yields a robust and scalable approach suitable for a wide range of environments, making it an ideal choice for real-world robotics applications where reliable state estimation is crucial for nMPC performance.

REFERENCES

[1] D. Poduval and P. Rajalakshmy, "A review paper on autonomous mobile robots," vol. 2670, 12 2022, p. 030005.

[2] R. P. M. Chan, K. A. Stol, and C. R. Halkyard, "Review of modelling and control of two-wheeled robots," *Annual Reviews in Control*, vol. 37, no. 1, pp. 89–103, 2013.

[3] I. Lee, "Service robots: A systematic literature review," *Electronics*, vol. 10, no. 21, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/21/2658

[4]

[5] F. Yakub and Y. Mori, "Effects of roll dynamics for car stability control by laguerre functions," 08 2013, pp. 330–335.

[6] S. Nikravesh, *Nonlinear Systems Stability Analysis: Lyapunov-Based Approach*, 09 2018.

[7] Y. Chen, H. Kong, Z. Li, and F. Ke, "Trajectory-tracking for a mobile robot using robust predictive control and adaptive control," in *2018 3rd International Conference on Advanced Robotics and Mechatronics (ICARM)*, 2018, pp. 30–35.

[8] N. Eslami and R. Amiadifard, "Moving target tracking and obstacle avoidance for a mobile robot using mpc," in *2019 27th Iranian Conference on Electrical Engineering (ICEE)*, 2019, pp. 1163–1169.

[9] J. Zhang, D. Wei, R. Gao, and Z. Xia, "A trajectory tracking and obstacle avoidance approach for nonholonomic mobile robots based on model predictive control," 10 2020, pp. 1038–1043.

[10] C.-H. Hsieh and J.-S. Liu, "Nonlinear model predictive control for wheeled mobile robot in dynamic environment," 07 2012.

[11] D. Gu and H. Hu, "A stabilizing receding horizon regulator for nonholonomic mobile robots," *Robotics, IEEE Transactions on*, vol. 21, pp. 1022 – 1028, 11 2005.

[12] F. Xie and R. Fierro, "First-state contractive model predictive control of nonholonomic mobile robots," in *2008 American Control Conference*, 2008, pp. 3494–3499.

[13] W. Benchouche, R. Mellah, and M. S. Bennouna, "Navigation of a differential drive mobile robot using nonlinear model predictive control," *L2CSP Laboratory, Faculty of Electrical and Computing Engineering, University Mouloud Mammri of Tizi-Ouzou, 15000, Algeria; Mechanical engineering dept, University Kasdi Merbah, Ouargla, 30000, Algeria*, 2021.

[14] S. Bouzoualegh, E. Guechi, and R. Kelaiaia, "Model predictive control of a differential-drive mobile robot," *Acta Universitatis Sapientiae Electrical and Mechanical Engineering*, vol. 10, pp. 20–41, 12 2018.

[15] G. Pannocchia, "Handbook of model predictive control [bookshelf]," *IEEE Control Systems Magazine*, vol. 40, no. 5, pp. 96–99, 2020.

[16] D. Boiroux and J. B. Jørgensen, "Sequential 1 quadratic programming for nonlinear model predictive control," *IFAC-PapersOnLine*, vol. 52, no. 1, pp. 474–479, 2019, 12th IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems DYCOPS 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896319301934

[17] I. Kempf, P. Goulart, and S. Duncan, "Fast gradient method for model predictive control with input rate and amplitude constraints∗∗this research is supported by the engineering and physical sciences research council (epsrc) with a diamond case studentship." *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6542–6547, 2020, 21st IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896320303268

[18] Q. Zou, J. Ji, S. Zhang, M. Shi, and Y. Luo, "Model predictive control based on particle swarm optimization of greenhouse climate for saving energy consumption," in *2010 World Automation Congress*, 2010, pp. 123–128.

[19] W. Naeem, R. Sutton, J. Chudley, F. Dalgleish, and S. Tetlow, "An online genetic algorithm based model predictive control autopilot design with experimental verification," *International Journal of Control - INT J CONTR*, vol. 78, pp. 1076–1090, 09 2005.

[20] H. Bässmann and P. W. Besslich, *Hough-Transformation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 101–121. [Online]. Available: https://doi.org/10.1007/978-3-662-42589-3$_7$

[21] X. Ma, B. Li, Y. Zhang, and M. Yan, "The canny edge detection and its improvement," in *Artificial Intelligence and Computational Intelligence*, J. Lei, F. L. Wang, H. Deng, and D. Miao, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 50–58.

[22] R. Raguram, J.-M. Frahm, and M. Pollefeys, "A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus," in *Computer Vision – ECCV 2008*, D. Forsyth, P. Torr, and A. Zisserman, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 500–513.

[23] M. Annaby and Y. Fouda, "Fast template matching and object detection techniques using ϕ-correlation and binary circuits," *Multimedia Tools and Applications*, vol. 83, pp. 6469–6496, 2024, received: 04 August 2021; Revised: 27 April 2022; Accepted: 21 April 2023; Published: 09 June 2023; Issue Date: January 2024. [Online]. Available: https://doi.org/10.1007/s11042-023-15564-x

[24] S. Patel and A. Patel, *Object Detection with Convolutional Neural Networks*, 10 2020, pp. 529–539.

[25] K. Granström, M. Baum, and S. Reuter, "Extended object tracking: Introduction, overview, and applications," *Journal of Advances in Information Fusion*, vol. 12, 12 2017.

[26] A. Eman and H. Ramdane, "Mobile robot localization using extended kalman filter," in *2020 3rd International Conference on Computer Applications Information Security (ICCAIS)*, 2020, pp. 1–5.

[27] D. Talwar and S. Jung, "Particle filter-based localization of a mobile robot by using a single lidar sensor under slam in ros environment," in *2019 19th International Conference on Control, Automation and Systems (ICCAS)*, 2019, pp. 1112–1115.

[28] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Transactions on Intelligent Transportation Systems Magazine*, vol. 2, pp. 31–43, 12 2010.

[29] J. Luo and S. Qin, "A fast algorithm of simultaneous localization and mapping for mobile robot based on ball particle filter," *IEEE Access*,

vol. PP, pp. 1–1, 03 2018.

[30] J. Lin, J. Peng, Z. Hu, X. Xie, and R. Peng, "ORB-SLAM, IMU and Wheel Odometry Fusion for Indoor Mobile Robot Localization and Navigation," *Academic Journal of Computing Information Science*, vol. 3, no. 1, pp. 131–141, 2020. [Online]. Available: https://doi.org/10.25236/AJCIS.2020.030114