Obstacle Avoidance and Path Tracking for Mobile Robots Using MPC with Linearized Dynamics

Jeremy Chen
Department of Mechanical Engineering
University of California, Berkeley
Berkeley, CA, USA
b_chen@berkeley.edu

Max Freeman
Department of Mechanical Engineering
University of California, Berkeley
Berkeley, CA, USA
max_freeman@berkeley.edu

Felix Hu
Department of Mechanical Engineering
University of California, Berkeley
Berkeley, CA, USA
felix_hyy@berkeley.edu

Junyu Lu
Department of Mechanical Engineering
University of California, Berkeley
Berkeley, CA, USA
junyug27@berkeley.edu

Eric Pham
Department of Mechanical Engineering
University of California, Berkeley
Berkeley, CA, USA
epham1407@berkeley.edu

Fernando Pluma
Department of Mechanical Engineering
University of California, Berkeley
Berkeley, CA, USA
fernando_pluma@berkeley.edu

***Abstract*—This paper presents a Model Predictive Control (MPC)-based framework for path planning of mobile robots, focusing on a two-wheel differential drive robot navigating between two points in a room. The proposed approach incorporates the robot's kinematic and dynamic constraints alongside environmental factors, such as obstacle avoidance, to generate an optimal trajectory. The framework addresses challenges associated with nonlinear dynamics, obstacle navigation, and energy efficiency. Simulation results validate the effectiveness of the method across three scenarios: (1) a perfect model with nonlinear dynamics, (2) a noisy model with nonlinear dynamics, and (3) a linearized approximation of the system. This report details the design, implementation, and performance of the MPC framework in achieving efficient and feasible paths under varying conditions.**

Video link here: https://youtu.be/Zq877qox5KE

I. Introduction

Path planning for mobile robots in dynamic environments poses significant challenges, particularly in the context of energy efficiency and environmental constraints, such as obstacles. The increasing demand for sustainable robotic solutions motivates the development of methods that integrate energy-aware path planning with model-based control techniques. Recent studies have advanced these techniques, including adaptive MPC for real-time trajectory adjustment [1], meta-heuristic algorithms for energy-efficient path planning [2], and MPC with Bayesian optimization for minimizing energy in industrial robotics [3]. There has also been significant interest in novel approaches to optimization-based collision avoidance [4].

This report proposes an MPC-based approach that incorporates the robot's kinematics, dynamics, and energy consumption models to generate optimal trajectories while subject to obstacles and model mismatch. We then demonstrate the effectiveness of this approach by simulating a two-wheeled differential drive robot navigating an environment with obstacles.

II. Controller Design

A. MPC Framework

Our optimization problem is divided into two periods. At $t = 0$, the first time step of the simulation, the Non-Linear MPC (NonLinMPC) is used to generate an optimal N-step reference trajectory, balancing stage costs, terminal costs, and input costs.

After running this, we linearize around the predicted trajectory to compute the A and B matrices for each point in the N-step path, as described above. For all subsequent timesteps, $t > 0$, we use the Linearized MPC model (LinMPC), which functions as a reference tracking controller that seeks to minimize deviation from the reference trajectory generated at $t = 0$. At each simulation timestep, we re-linearize around the newly predicted trajectory. This helps to ensure that the linear model continues to be an accurate approximation.

The general form of the problem we are solving in both models is shown below:

$$\begin{aligned} \min \quad & J(q_k, u_k) \\ \text{s.t.} \quad & q_0 = q(0), \\ & q_N \in \mathcal{Q}_f, \\ & q_{k+1} = f(q_k, u_k), \\ & q \in \mathcal{Q}, \\ & u \in \mathcal{U}, \\ & q_k \notin \mathbb{O}. \end{aligned}$$

In the following sections we will seek to further clarify each of the components of this model.

B. Cost Function

The primary difference between the NonLinMPC and LinMPC problems, aside from the differences in the dynamic models used, lies in the definitions of their cost functions.

The cost function for the NonLinMPC problem, designed to drive the system to (0, 0, 0) is shown below:

$$J = \sum_{t=0}^{N-1} q_k^\top Q q_k + u_k^\top R u_k + Y\varepsilon_{i,k} + q_N^\top P q_N \tag{1}$$

In contrast, the LinMPC's cost function is defined as a reference tracker, seeking to minimize the error between its current state, q_k, and a reference state, r_k:

$$J = \sum_{t=0}^{N-1} e_k^\top Q e_k + u_k^\top R u_k + Y\varepsilon_{i,k} + e_N^\top P e_N \tag{2}$$

$$e_k = q_k - r_k \tag{3}$$

In both cases, the concept of energy efficiency is captured through the cost functions penalizing excessive control effort.

C. Dynamics

The dynamic model of the robot includes its kinematic constraints and inputs. The robot's states, $[x, y, \theta]$, represent its position and orientation, while the inputs, $\begin{bmatrix}\dot{\phi}_l \\ \dot{\phi}_r\end{bmatrix}$, denote the left and right wheel velocities respectively. The dynamics are defined as:

$$\dot{x} = \cos(\theta)\frac{R}{2}(\dot{\phi}_l + \dot{\phi}_r) \tag{4}$$

$$\dot{y} = \sin(\theta)\frac{R}{2}(\dot{\phi}_l + \dot{\phi}_r) \tag{5}$$

$$\dot{\theta} = \frac{R}{2L}(\dot{\phi}_r - \dot{\phi}_l) \tag{6}$$

where R is the wheel radius and L is half of the distance between the wheels, both of which are in units of meters. These can be seen below in Figure 1.

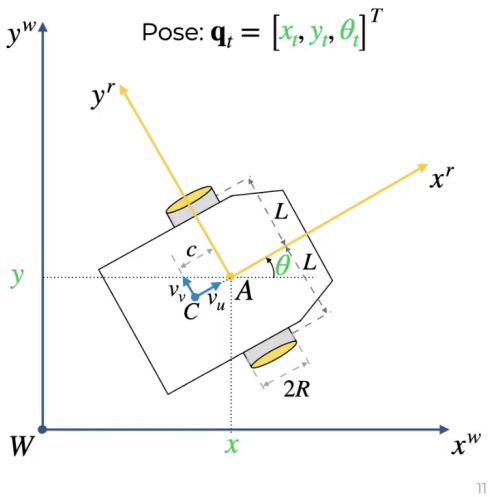

Fig. 1: Labeled figure of the robot

D. Linearization

In the case of the LinMPC model, the dynamics of the predicted trajectory at each timestep were then linearized to reduce computational complexity. This was done by linearizing around every state in the reference path generated at t=0. For an N-step path, this will generate N different sets of A and B matrices, representing the linearised dynamics at each point on the path. This approach is demonstrated in the pseudocode below.

```
DynArr = []
FOR state, input IN
    predictedPath
    Ac = Jacobian_dx(
        state)
    Bc = Jacobian_du(
        input)
    Cc, Dc = [0], [0]
    Ad, Bd = ZOH(Ac, Bc,
         Cc, Dc)
    DynArr = DynArr + [
        Ad, Bd]
ENDFOR

MPC(params, DynArr)
```

E. Discretization

Two different methods of discretization were used in this project. For our linearised system, we used a Zero Order Hold (ZOH), which provides the exact solution for the system when $u(k)$ is held constant across each timestep, as is the case here. This is shown in Eq. (7). In the case of nonlinear dynamics, the dynamic models are discretized using the Euler forward method instead, as shown in Eq. (8). This is because ZOH only works on linear systems.

$$q(k+1) = e^{A_c T_s} \cdot q(k) + u(k) \int_0^{T_s} e^{A_c(T_s - \tau)} B_c \, d\tau \tag{7}$$

$$q(k+1) = q(k) + \Delta t \cdot f(q(k), u(k)) \tag{8}$$

With $q(k+1)$ and $q(k)$ representing the next state and current state, respectively; $u(k)$ being the input at timestep k; and $f(q(k), u(k))$ being the state-space model extracted from Eq. (4), (5), and (6).

F. Obstacle Avoidance

Another crucial aspect of our implementation was obstacle avoidance. We chose to model the obstacles as circles, allowing us to describe each obstacle compactly with just three values:

$$\mathcal{O}_i = [xO_i, yO_i, r_i].$$

We can then go from this definition to writing the implementation of the obstacle avoidance constraint:

$$(x_k - xO_i)^2 + (y_k - yO_i)^2 \geq (r_i + d_{\min} - \varepsilon_{i,k})^2 \tag{9}$$

$$\varepsilon_{i,k} \leq 0.75 \cdot d_{\text{min}} \tag{10}$$

where d_{min} is a variable passed into the MPC controller that defines the desired minimum distance from the obstacle. The variable $\varepsilon_{i,k}$ is a slack variable that allows the controller some leeway in how close to the obstacle it can go. This is particularly helpful in maintaining feasibility when dealing with a model mismatch, as discussed in the Results section. This variable is weighted heavily in the cost function to motivate the controller to only move closer to the obstacle if it needs to maintain feasibility.

III. Results and Discussion

A. Simulation Setup

The MPC controller's performance is evaluated in a simulated environment featuring a differential drive robot navigating a set environment with a few obstacles in between. The robot starts at (-10, -10, 0) and aims to reach the origin (0, 0, 0). The simulation parameters are:

Parameter	Value	Description
R_{wheel}	0.02	Wheel Radius [m]
L	0.15	Wheel Base [m]
q_0	$[-10, -10, 0]$	Initial State
q_f	$[0, 0, 0]$	Final State
q_{LB}	$[-500, -500, -2\pi]$	State Lower Bound
q_{UB}	$[500, 500, 2\pi]$	State Upper Bound
d_{min}	$4L$	Obstacle Min. Distance
N	20	Prediction Horizon
dt	0.1	Sampling Time [sec]
Q	$\begin{bmatrix} 50 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 50 \end{bmatrix}$	State Weight
R	$\begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}$	Input Weight
P	$\begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix}$	Terminal Weight
Y	10000	Slack Cost Weight

TABLE I: Summary of simulation parameters

B. Result

The first model was implemented using non-linear dynamics for both finding the optimal path and tracking it. This approach produced smooth convergence of state variables, as demonstrated in the trajectory and state evolution graphs shown in Figures 2 and 3, respectively. For this model, the slack variables remained minimal, with a peak value of 0.2, as the robot did not experience any disturbances that required it to infringe on the obstacle space. The second model introduced

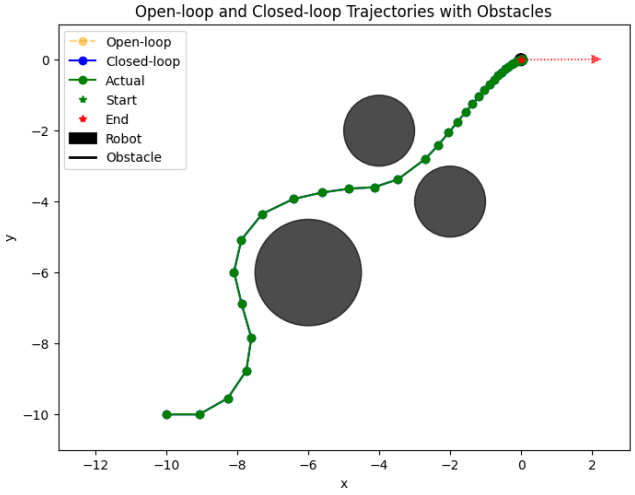

Fig. 2: Model 1-Open and Closed Loop Trajectories with Obstacles

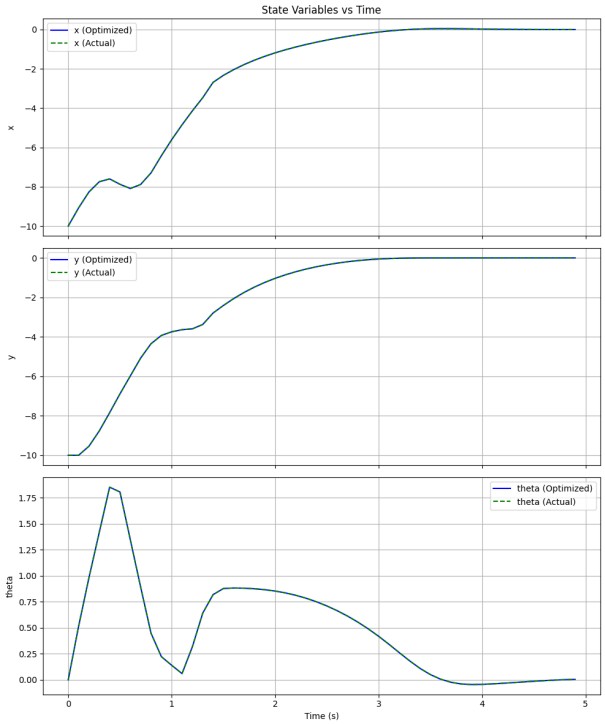

Fig. 3: Model 1-State Variables vs. Time

random noise to simulate uncertainty in the robot's state estimation. This noise was modeled as a Normal distribution with a standard deviation of (0.05, 0.05, 0.05), corresponding to uncertainties of approximately 5 cm in position and 3 degrees in orientation. The results here, shown in Figures 5 and 6 below, still showed a successful convergence to the setpoint, although with minor oscillations. This demonstrates the robustness of the model under uncertainty.

The results here predictably showed higher values for slack variables, with a peak of 0.35. This increase indicates that the model needed to rely more heavily on slack to account for mismatches and maintain feasibility.

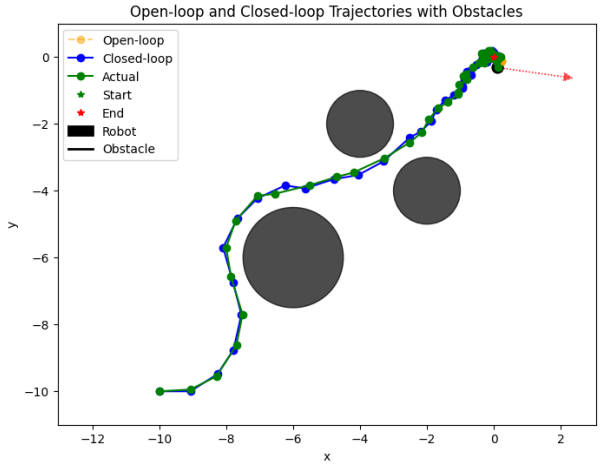

Fig. 4: Model 2-Open and Closed Loop Trajectories with Obstacles

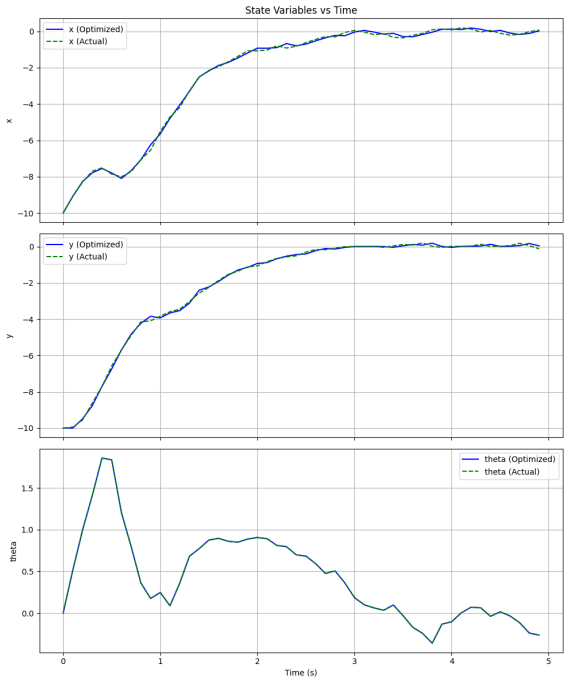

Fig. 5: Model 2-State Variables vs. Time

The third model introduced our linearized dynamics and reference tracking model (LinMPC). This model performed reasonably well during most of the trajectory but exhibited instability upon approaching the setpoint.

The cause of this instability is unclear. It could result from an implementation bug in the control algorithm or represent a fundamental limitation of the model. For example, at the setpoint, the robot was required to execute very small and precise control inputs. Any deviation in these inputs therefore appeared magnified, leading to unstable behavior. This suggests the need for further investigation into the model's stability at the terminal condition.

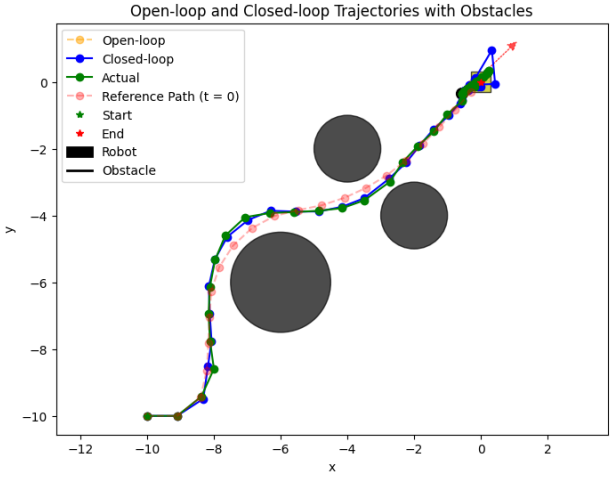

Fig. 6: Model 3-Open and Closed Loop Trajectories with Obstacles

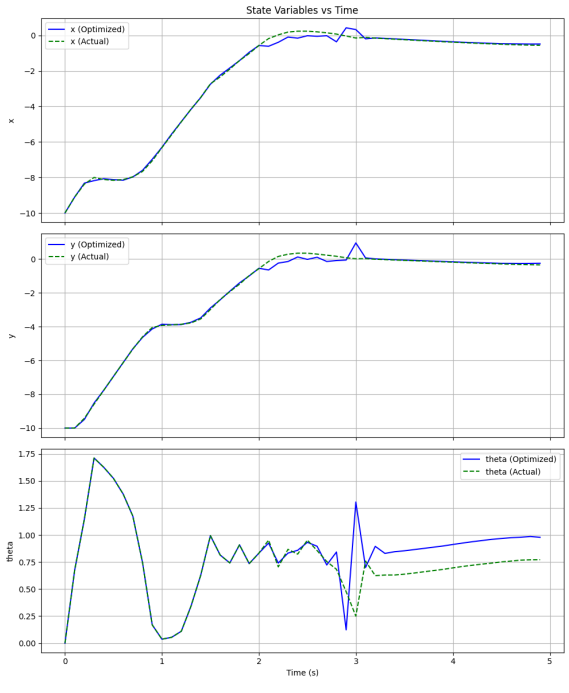

Fig. 7: Model 3-State Variables vs. Time

IV. Conclusion and Future Work

This report presented an MPC-based approach for energy-aware path planning using linearised dynamics in the presence of obstacles. The controller successfully integrated non-linear dynamics, energy constraints, and obstacle avoidance, demonstrating its applicability to real-world scenarios. Future work includes:

- Extending the framework to dynamic obstacles and multi-agent systems.
- Incorporating uncertainty in robot dynamics and environmental factors using stochastic MPC.

In future work, we propose extending this framework to address the challenge of varying dynamics. Specifically, this can be formulated as a Mixed Integer Problem. A binary optimization variable, Z, can be introduced to represent whether the robot is operating in a region with varying dynamics. This variable would be defined as 1 in the presence of varied dynamics and 0 otherwise. The dynamic constraints can then be expressed as:

$$q(k+1) = (1 - Z) \cdot f_1 + Z \cdot f_2 \tag{12}$$

where f_1 and f_2 represent the dynamic equations for the nominal and varied dynamics region, respectively.

The team initially attempted to implement this; however, challenges with implementing this formulation arose. The solver experienced difficulties handling the MIP problem, resulting in infeasibility and excessive computation times. Additionally, attempts to linearize the model yielded trajectories that were sometimes unstable, particularly in the terminal states. These inconsistent results highlighted the limitations of linearization in capturing the robot's true behavior, as observed during simulations.

References

[1] H. Zhang and et al., "Adaptive model predictive control of mobile robot with local path refitting," in *Proceedings of the 2023 IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 1000–1007. [Online]. Available: https://ieeexplore.ieee.org/document/10006003.

[2] L. Chen and et al., "A meta-heuristic path planning algorithm for heterogeneous mobile robots," *Sustainability*, vol. 14, no. 22, 2022. [Online]. Available: https://www.mdpi.com/2071-1050/14/22/15137.

[3] F. Holzmann and et al., "Mpc and bayesian optimization for energy-minimal trajectories in industrial robots," in *European Control Conference (ECC)*, 2024. [Online]. Available: https://www.ias.informatik.tu-darmstadt.de/uploads/Publications/ByType/ECC_2024_final.pdf.

[4] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 972–983, May 2021. DOI: 10.1109/TCST.2019.2949540.