AROB Practical Work

Alberto Zafra Navarro & Daniel Sanz Valtueña

***Abstract*— Model Predictive Control (MPC) is well-regarded for real-time control through objective function optimization, particularly in autonomous robots such as quadrotors, representing a significant advancement in dynamic system control strategies. This study emphasizes MPC's potential in drone trajectory tracking by implementing a simplified version of this control algorithm, showcasing its adaptability across various scenarios and its ability to efficiently optimize control inputs for improved performance.**

The implemented algorithm serves as a foundational tool for further exploration and refinement of drone control strategies. Utilizing the CasADi library and IPOPT solver, MPC enhances trajectory tracking in quadrotor drones. Integrated into the Gazebo Simulator with the hector_quadrotor package, MPC accurately predicts drone behavior and optimizes control inputs based on dynamic models. Trajectories are planned using the mav_trajectory_generation package, tailored specifically for rotary-wing micro aerial vehicles.

Experimental evaluation across diverse scenarios assesses MPC's performance using metrics such as global trajectory error, step-wise accuracy, mean velocity deviation, and total flight time. Results demonstrate significant improvements over conventional methods, highlighting the effectiveness of optimized parameters such as position error weights and lookahead horizon.

This study underscores MPC's precision and efficiency in drone control, thereby advancing the capabilities of autonomous aerial systems. Leveraging CasADi and IPOPT, our framework establishes robust MPC-based trajectory tracking, facilitating enhanced drone navigation and control strategies.

I. Introduction

The application of Model Predictive Control (MPC) [1] in autonomous robots, particularly in quadrotor dynamics, represents a significant advancement in control strategies for dynamic systems. MPC is well-known for its capability to optimize control actions by predicting future system behavior over a defined prediction horizon. This predictive ability allows MPC to handle complex systems such as autonomous robots, where real-time decision-making based on future states is crucial for stability, security, and performance.

This project focuses on exploring the implementation and benefits of MPC as a control strategy for quadrotors. Specifically, it examines how MPC leverages predictive models to anticipate system states and optimize control inputs over a finite time horizon. By forecasting the system's future behavior a few time steps ahead (lookahead), MPC effectively balances the trade-offs between different control objectives, such as trajectory tracking, collision avoidance and energy efficiency, ensuring input smoothness and thereby controlling velocity inputs to minimize significant changes over iterations.

In the context of autonomous robots, the MPC controller for quadrotors offers a robust framework to navigate and interact with dynamic environments. This report delves into the theoretical foundations of MPC, its adaptation to quadrotor dynamics, and practical considerations in implementation. Furthermore, it explores case studies and simulations that demonstrate the efficacy of MPC in achieving precise and agile maneuvers, showcasing its potential to advance the field of autonomous robotics.

II. Algorithm description

Model Predictive Control is an optimization-based control algorithm. It continuously solves a finite horizon optimization problem at each time step, using the current state of the system as the initial condition. The key features of MPC include:

- **Prediction Model**: MPC uses a dynamic model of the system to predict future states over a specified prediction horizon.
- **Optimization**: At each time step, MPC solves an optimization problem to determine the control inputs that minimize a cost function subject to system dynamics and constraints.
- **Receding Horizon**: The optimization is performed over a moving (receding) horizon, which means that at each time step, the horizon shifts forward, and the optimization is repeated using the updated state information.

III. Mathematical Formulation

A. Objective Function

The goal of MPC is to minimize a cost function $J_k(x_k, u_k)$ over the prediction horizon. The cost function, presented in Equation 1, in this project, includes terms for tracking the desired trajectory and penalizing control effort to ensure smoothness:

$$J_k(x_k, u_k) = \sum_{i=k}^{k+N} \left[w_{\text{position}} \cdot (x_i - x_i^*)^2 + w_{\text{smooth}} \cdot (u_i)^2 \right] \tag{1}$$

Where:

- x_i: State vector at time step i (position in x, y, z).
- u_i: Control input vector at time step i (velocities v_x, v_y, v_z).
- x_i^*: Desired state (target trajectory).
- $w_{\text{position}}, w_{\text{smooth}}$: Weighting factors for position error and control effort, respectively.
- N: The lookahead time steps horizon.

B. System Dynamics

The system is modeled with discrete-time dynamics, that implements the basic uniform movement equation, where Δt is the time discretization step, as shown in Equation 2.

$$x_{k+1} = x_k + \Delta t \cdot u_k \tag{2}$$

C. Constraints

The control algorithm incorporates constraints on both the state and control inputs to ensure feasible and safe operation of the quadrotor. These constraints are essential for maintaining the system within operational limits and following the desired trajectory.

- **State constraints** $x_k \in X$: Ensure that the quadrotor's trajectory adheres to the planned path and avoids obstacles or restricted areas. This is critical for mission success and safety.
- **Control constraints** $u_k \in U$: Limit the control inputs, specifically the velocities, to prevent excessive or unsafe maneuvers. These constraints ensure that the quadrotor operates within its physical capabilities and adheres to safety regulations.

Mathematically, the control constraints can be expressed as:

$$u_{\min} \leq u_{k+i} \leq u_{\max} \tag{3}$$

where $u_{\min}$ and $u_{\max}$ represent the minimum and maximum allowable control inputs, respectively, and u_{k+i} is the control input at time step $k+i$.

IV. Implementation Details

To correctly implement the aforementioned MPC algorithm, it was necessary to set up various tools for simulating a quadrotor accurately.

In this case, the *hector quadrotor* [2] package was used for simulating the quadrotor within the *Gazebo Simulator* [3]. Furthermore, the previously mentioned package contains all the necessary functionalities for controlling the drone through velocity commands.

On the other hand, the *mav_trajectory_generation* package [4] offers tools for polynomial trajectory generation and optimization, particularly suited for rotary-wing micro aerial vehicles (MAVs) like quadrotors. This package was used to plan the desired trajectory for the drone. Ideally, the quadrotor would follow the reference path accurately, as the trajectory would have been designed based on the constraints of the device and environment.

Nevertheless, the low-level control limitations of the system may lead to inaccuracies in following the planned trajectory. Hence, the MPC trajectory tracking algorithm was implemented according to the architecture presented in Figure 1. Where a node containing the polynomial trajectory planner would feed the MPC node with the reference poses and velocities of the drone within a lookahead time step horizon.

Subsequently, the closed-loop controller would be responsible for optimizing the previously presented cost function in order to minimize the position error, while ensuring a smooth movement that does not compromise the drone's integrity. It is worth mentioning that both the position and velocity errors are computed using the onboard odometry sensors of the quadrotor. Furthermore, the optimization problems have been solved using the CasADi library [5], a powerful tool for nonlinear optimization and algorithmic differentiation [6].

Finally, after optimization, the resulting velocities are fed to the quadrotor, ensuring its correct movement.

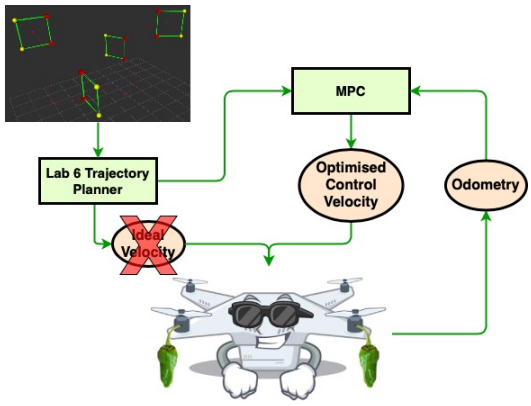


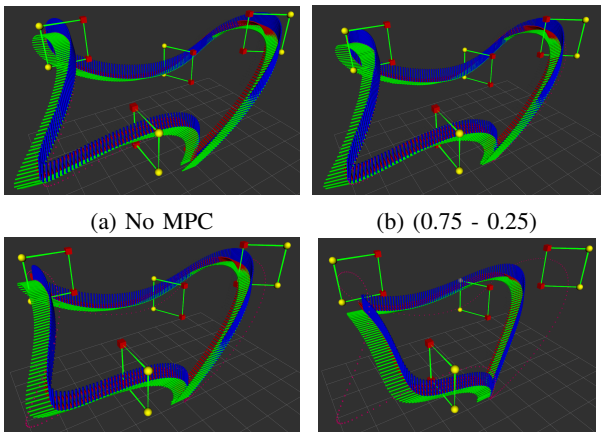
Fig. 1: Illustration graph of the quadrotor control system.

V. Experimental Results

Three different sets of experiments have been conducted using the aforementioned simulation environment, simulating the quadrotor in Gazebo and visualizing the generated trajectories in RVIZ [7]. Each set of experiments was conducted five times in two different scenarios: an easy scenario (where the drone navigated through four gates) and a hard scenario (where five gates were arranged in a more challenging configuration). Each experiment involved setting specific parameters and evaluating the quadrotor's performance based on the following defined metrics:

- **Global Trajectory Error**: The total error between the desired path and the actual position at each time instant.
- **Step Trajectory Error**: The normalized error computed at each time step, reflecting deviations from the desired trajectory.
- **Mean Velocity Error**: The discrepancy between the path desired velocity and the quadrotor's speed during the whole trajectory.
- **Total Flight Time**: The duration taken to complete the entire trajectory.

The main analysis is focused on the results from the easy scenario, as the results gathered from the hard scenario exhibited a similar behavior from the aforementioned ones, with only slight differences in error magnitudes. Additionally, the standard deviations of the measurements are not explicitly disclosed in Tables I-III, as they all fall within a 10% variation margin.



(a) No MPC (b) (0.75 - 0.25)

(c) (0.5 - 0.5) (d) (0.25 - 0,75)

Fig. 2: Followed trajectories applying different position and smooth weights (w_pos - w_v)

The first set of experiment was focused on evaluating the impact of the algorithm's preferences on the system performance. Therefore, four different system's configurations were tested : No MPC, High Position Error Weight - Low Smoothness Weight, Low Position Error Weight - High Smoothness Weight, and Medium Position Error Weight - Medium Smoothness Weight. Each configuration aimed to evaluate how prioritizing position error versus smoothness of velocity commands influenced trajectory tracking, velocity error, flight time, and control stability.

(w_pos - w_v)	Error[m]	Step error[m]	Vel error[m/s]	Time[s]
No MPC	8.0889	0.0326	7.0613	24.9368
(0.75 - 0.25)	2.4091	0.0194	5.3150	24.9397
(0.5 - 0.5)	5.8632	0.0469	7.6018	25.1042
(0.25 - 0.75)	10.6128	0.0964	9.7271	25.4176

TABLE I: Weights comparison results

The second set of experiments was devoted to assessing the effect of lookahead steps on predictive accuracy and control performance by testing different numbers of steps.

Lookahead [steps]	Error[m]	Step error[m]	Vel error[m/s]	Time[s]
20	2.3572	0.0155	9.1086	24.7955
40	2.3564	0.0166	8.3000	24.9747
60	2.5812	0.0222	4.7674	24.9378
80	2.9517	0.0298	4.7724	24.9691
100	3.5227	0.0424	6.8287	24.4751

TABLE II: Lookahead steps comparison results

During the last set of experiments, various maximum velocities of the drone were tested to determine their impact on trajectory execution and overall performance.

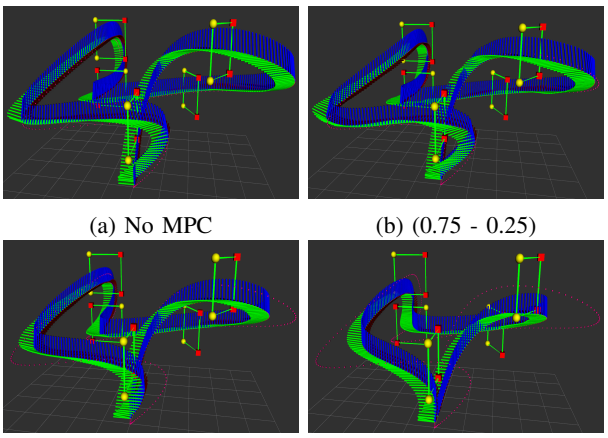
Max velocity	Error[m]	Step error[m]	Vel error[m/s]	Time[s]
1	31.4173	0.1415	15.5301	45.7745
1.5	6.7631	0.0430	8.1169	31.1102
2	2.4091	0.0194	5.3150	24.9397
2.5	2.5891	0.0223	8.1924	22.9216

TABLE III: Maximum velocities comparison results

Hence, as seen in the previously presented Tables I-III and Figures 2-3, an elevated position error weight would result in an accurate trajectory tracking. However, the smoothness weight is essential for controlling the input velocities. Without smooth control, the controller may attempt to perform high velocity changes, potentially damaging the quadrotor (given its maximum acceleration constraint of 2 m/s^2, that might not be enough for satisfying the aforementioned velocity change) or causing oscillatory movements that prevent reaching the desired position.

On the other hand, a lookahead size between 40 and 60 steps is recommended. A short horizon yields an accurate position system that precisely follows the desired trajectory but may fail to maintain the planned velocities due to reduced weight on movement smoothness. Conversely, a longer horizon offers a balanced tradeoff between speed and position accuracy. However, an excessively long horizon might increase errors as the system prioritizes future positions over the immediate desired positions.

Moreover, the velocity constraint must be appropriately set. If the quadrotor moves too slowly, it may not reach the desired positions within the planned time, whereas high velocities might result in oscillations and abrupt control, reducing its stability and accuracy. Therefore, it can be state that the best configuration is the one containing a position error weight of 0.75, a lookahead of 50 steps and a maximum velocity of 2 m/s. Finally, as it has been aforementioned and illustrated in Figure 3, the experiments conducted within the hard scenario exhibit similar behaviour to those performed within the easy scenario.



(a) No MPC (b) (0.75 - 0.25)

(c) (0.5 - 0.5) (d) (0.25 - 0,75)

Fig. 3: Followed trajectories in hard gates circuit applying different position and smooth weights (w_pos - w_v)

VI. Future work

While the current implementation of MPC for the quadrotor has demonstrated promising results by significantly enhancing the basic feed-forward control algorithm, the simplistic approach presented does not fully showcase the trajectory controller's potential. The primary strength of MPC lies in its ability to utilize intricate dynamical models and in-

corporate relevant constraints to derive optimal actions. This paper focuses solely on constraining quadrotor velocity while optimizing a basic kinematic trajectory, thereby limiting the algorithm's visibility of its full capabilities.

To fully realize the potential of the MPC trajectory controller, future enhancements should consider integrating more complex, second-order dynamic models and additional constraints. For instance, incorporating orientation control or obstacle avoidance would greatly enhance the versatility and effectiveness of the implemented system. These improvements would reveal the algorithm's true capabilities and demonstrate its robustness in real-world scenarios.

VII. Conclusions

This study has demonstrated that the MPC algorithm significantly improves performance across various relevant metrics when tracking trajectories. By adjusting the weights that parameterize the cost function, the algorithm can be customized to meet specific requirements, highlighting its flexibility and adaptability. Although the implementation in this paper was basic, it effectively showcased the significant potential of MPC in guiding drones along predefined trajectories, achieving minimal tracking errors.

References

[1] E. F. Camacho and C. Bordons, *Model Predictive Control*. Springer, 2007.

[2] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. von Stryk, "hector_quadrotor," http://wiki.ros.org/hector_quadrotor, 2014, accessed: 2024-06-19.

[3] "Gazebo simulator," https://gazebosim.org/home, 2024, accessed: 2024-06-19.

[4] H. O. R. B. M. P. Markus Achtelik, Michael Burri, "mav_trajectory_generation," https://github.com/ethz-asl/mav_trajectory_generation, 2024, polynomial trajectory generation and optimization, especially for rotary-wing MAVs.

[5] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. (2018) Casadi's documentation. Accessed: 11/06/2024. [Online]. Available: https://web.casadi.org/docs/

[6] ——. (2018) Optimal control problems in a nutshell. Accessed: 11/06/2024. [Online]. Available: https://web.casadi.org/blog/ocp/

[7] "rviz," http://wiki.ros.org/rviz, 2018, accessed: 2024-06-19.