Model Predictive Control Approach for Multi-UAVs Planning and Motion Control

X.Li, P.Yang,

***Abstract*—We implemented and simulated a Model predictive control (MPC) approach for multi-UAVs control. A state-based MPC approach for signal UAV position control was designed and was used to explore the effect of parameters on the results. Meanwhile, we designed a path-planning method and an output MPC method with trajectory tracking and collision avoidance for multiple UAVs in complex environments. A stability analysis is also performed to prove the stability of the approach. See Github for project details:**

https://github.com/PatrickYang-5/MPC_drones.

I. Introduction

Cooperative control of multi-UAVs is one of the popular research directions[1][2][3], which has a wide range of applications in scenarios such as environmental protection, factory inspection, and disaster rescue. A traditional solution is to use global planning methods based on A*[4] and RRT*[5] to obtain a control trajectory and apply a positional PID method to control the UAV to realize the trajectory following. This approach has been successful for simple tasks and a small number of UAVs, but as the task complexity and cluster size increase, due to ignoring optimality and dynamic constraints, the above methods often lead to non-optimal control results and erroneous collisions.

Inspired by the constraints and optimality of MPC[6], we chose to use the MPC method instead of the traditional PID method to achieve a more optimal control method. This project will focus on the following four main tasks:

- Establishment of a MIMO linear discrete dynamics model for quadrotors.
- Regular and output MPC designing.
- Asymptotic stability analysis of the system.
- Pybullet-based simulation system construction and simulation results analysis.

II. Model Establishment

A. Dynamics of Quadrotor

The dynamics model of the quadrotor can be solved based on the Lagrangian dynamics approach[7]. It is worth noticing that the state of the quadrotor can be divided into position and attitude variables. Therefore, in this subsection, we will give the dynamics model separated into position and attitude.

X.Li and P. Yang are master's students at TU Delft, The Netherlands. E-mail addresses: {X.Li-89, P.Yang-5}@student.tudelft.nl.

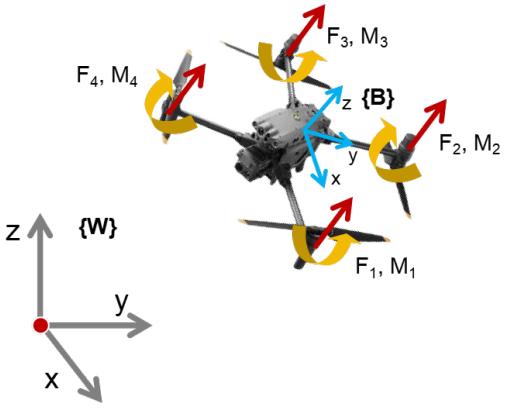

Fig. 1. UAV coordinate system and control variable definitions.

The dynamics model for position variables:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = R_{\phi,\theta,\psi} \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^{4} F_i/m \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}, \tag{1}$$

where, $\ddot{x}, \ddot{y}, \ddot{z}$, and F_i are shown in Fig.1. g is the gravity acceleration along the z-axis. $R_{\phi,\theta,\psi}$ is the rotational matrix transformed from body to world coordinates. This rotational matrix is calculated through an Euler angle in Z-Y-X order:

$$R_{\phi,\theta,\psi} = R_z(\psi)R_y(\theta)R_x(\phi), \tag{2}$$

where,

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}, \tag{3}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \tag{4}$$

$$R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{5}$$

The dynamics model for attitude variables can be calculated using the respective transformations of the rates of the Euler angles:

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R_x^{-1}(\phi) \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R_x^{-1}(\phi)R_y^{-1}(\theta) \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}. \tag{6}$$

Expanding the matrix, combining the Euler angles into a vector, and moving it to the left, we can obtain the following equation:

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \cos\theta\cos\psi & -\sin\psi & 0 \\ \cos\theta\sin\psi & \cos\psi & 0 \\ -\sin\theta & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}. \quad (7)$$

According to the Euler equation, we can get the relationship between Euler acceleration and rotational velocity and individual thrusts F_i:

$$\mathbf{I}\cdot\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \mathbf{I}\cdot\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} l(F_2 - F_4) \\ l(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} \quad (8)$$

where I is the moment of inertia matrix. l is the arm length of the drone and M_i is the i^{th} motor's moment.

B. Dynamics Linearization

The previously established dynamics model contains nonlinear terms, which poses challenges for constructing drones' linear time-invariant (LTI) model. Since our designed UAVs and the corresponding simulation environment are being used in low-speed and small-tilt applications[8], we can make the following assumptions:

- The roll and pith of the UAVs are close to zero.
- The UAVs have small angular velocity and the inertial moment in the Euler equation can be neglected.

Based on the two assumptions, we can rewrite eq.7 and eq.8 as follows:

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}, \quad (9)$$

$$\mathbf{I}\cdot\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} l(F_2 - F_4) \\ l(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix}. \quad (10)$$

C. LTI Model

The basic LTI model has the following general form:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned}. \quad (11)$$

In the scenario of UAVs, we choose the generalized positions and velocities for translation and rotation as the system's state, and the thrust of each motor as the system's input. In order to strictly follow the general form of the basic LTI model, we introduce gravitational acceleration into the state vector, even though it is a constant and not a state.

$$\begin{aligned} x &= \begin{bmatrix} x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}\ \phi\ \theta\ \psi\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}\ g \end{bmatrix}^T \\ y &= \begin{bmatrix} x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}\ \phi\ \theta\ \psi\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}\ g \end{bmatrix}^T. \\ u &= \begin{bmatrix} F_1\ F_2\ F_3\ F_4 \end{bmatrix}^T \end{aligned} \quad (12)$$

Since using the forces of four motors makes the B matrix complicated, we modified the definition of the inputs to reduce the complexity of the B array, and the modified inputs are as follows:

$$u = \begin{bmatrix} U_1\ U_2\ U_3\ U_4 \end{bmatrix}^T, \quad (13)$$

where,

$$\begin{aligned} U_1 &= F_1 + F_2 + F_3 + F_4 \\ U_2 &= l(F_2 - F_4) \\ U_3 &= l(F_3 - F_1) \\ U_4 &= l(T_1 - T_2 + T_3 - T_4) \\ T_i &= K_M F_i / K_f \ \ i = 1, 2, 3, 4 \end{aligned}. \quad (14)$$

Based on the linearized dynamics model described in the previous sections, we can calculate the A, B, C, and D matrix. Due to the high dimensionality of the matrices, it is hard to show here, please refer to the program for the calculation results.

III. Model predictive control design

According to the dynamic model of UAVs proposed in the previous section, we designed three MPC control strategies for different tasks:

- Regular MPC without global path planning for target position control tasks.
- Output MPC combined with global path planning for single UAV path-tracking tasks.
- Output MPC approach suitable for multi-UAVs path tracking and obstacle avoidance.

A. General Form of MPC

The MPC controller is designed to minimize a stage cost function $\ell(x, u)$ over a finite horizon N subject to the system dynamics and constraints. The MPC problem is solved at each time step k by solving the following optimization problem:

$$\begin{aligned} \min_{\mathbf{u}} \quad & \sum_{i=0}^{N-1} \ell(x(k+i), u(k+i)) + V_f(x(k+N)) \quad (15) \\ \text{s.t.} \quad & x(k+i+1) = f(x(k+i), u(k+i)), i = 0, \ldots, N-1 \\ & x(k+i) \in \mathbb{X}, \quad u(k+i) \in \mathbb{U}, i = 0, \ldots, N-1 \\ & x(k+N) \in \mathbb{X}_f \end{aligned}$$

where $x(k)$ is the state vector at time k, $u(k)$ is the control input at time k, f is the system dynamics, $\mathbb{X}$ and $\mathbb{U}$ are the state and control input constraints, respectively, and $\mathbb{X}_f$ is the terminal set.

B. Regular MPC Design

The goal of the regular MPC approach is to make the UAV control hover at the target position while keeping the UAV state and inputs optimal during operation. To realize this goal, we designed the process and terminal costs as follows:

$$\begin{aligned} &\ell(x(k+i), u(k+i)) = \\ &x(k+i)'^T Q x(k+i)' + u(k+i)^T R u(k+i), \\ &V_f(x(k+N)) = x(N)'^T P x(N)' \end{aligned} \quad (16)$$

where $x(\cdot)' = x(\cdot) - x_{targ}$ and x_{targ} is the target position with other states being zero so that it becomes the new origin and the MPC controller will try to regularize the system to it.

The Q and R are repetitively tuned based on the simulation results, and the corresponding values for which good results can be achieved are as follows:

$$\begin{aligned} Q &= diag(\begin{bmatrix}150 & 150 & 150 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8\end{bmatrix}) \\ R &= diag(\begin{bmatrix}10 & 10 & 10 & 10\end{bmatrix}) \end{aligned} \tag{17}$$

In order to make the system optimal and controllable, P is chosen to be the solution of the unconstrained LQR problem with infinite horizon, and this solution is obtained in this project by solving the discrete algebraic Riccati equation.

For the state constraints, based on the definitions in the simulation environment and URDF model, as well as referring to the performance of common UAVs in reality, we set the linear speed limit of the UAV in the XYZ-axis in the interval $[-2, 2]m/s$.

In order to be consistent with the small angle assumptions made in the previous linearization of the dynamics model, we restrict the angle of RPY rotation to the interval $[-0.5, 0.5]rad$. The corresponding angle is 28.5°, which meets the simplification made in the linearization.

The rotational speed of the UAV has an upper limit, so the inputs to the system should also be designed with corresponding constraints. We refer to the acceleration parameter of common UAVs and set the input limit to a force that makes the UAV realize 0.5 times the acceleration of gravity and cannot provide a reverse force. The effect of the input on the acceleration of the RPY rotation is also limited to small values (maximum 0.15).

The control input constraints are represented in compact form as $H_x x(k) \leq h_x$ and $H_u u(k) \leq h_u$ where H_x and H_u are the 12x12 and 8x4 matrix:

$$H_x = \begin{bmatrix} O_{3\times3} & I_3 & O_{3\times3} & O_{3\times3} \\ O_{3\times3} & -I_3 & O_{3\times3} & O_{3\times3} \\ O_{3\times3} & O_{3\times3} & I_3 & O_{3\times3} \\ O_{3\times3} & O_{3\times3} & -I_3 & O_{3\times3} \end{bmatrix} \tag{18}$$

$$H_u = \begin{bmatrix} \mathbf{1/m} & 0 & 0 & 0 \\ \mathbf{-1/m} & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{-1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{-1} & 0 \\ 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{-1} \end{bmatrix} \tag{19}$$

where $O_{N\times M}$ is the zero matrix with N rows and M columns and I_k is the k-dimensional unit array.

h_x and h_u are the vectors:

$$\begin{aligned} h_x &= \begin{bmatrix}2 & 2 & 2 & 2 & 2 & 2 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5\end{bmatrix}^T \\ h_u &= \begin{bmatrix}1.5g & -0.5g & 0.15 & 0.15 & 0.15 & 0.15 & 0.15 & 0.15\end{bmatrix}^T \end{aligned} \tag{20}$$

The terminal set used in this project is obtained by calculations, which will be presented in section[IV].

Since there is a distance between the starting position and the target position of our design, we choose the prediction horizon N to be 100 in order to ensure the terminal set is reachable within the horizon. Such a challenge became one of the motivations for us to design and implement the output MPC. That is, by using global path planning to generate trajectory positions such that the UAV only needs to reach the nearby target point's terminal set within a short prediction horizon, which can greatly reduce the required prediction horizon.

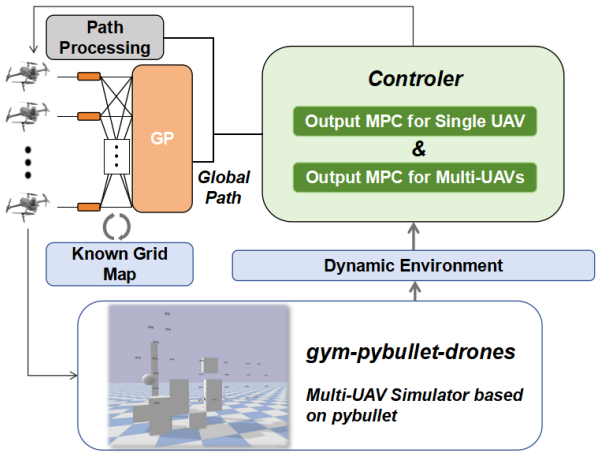

Fig. 2. System schematic with global planner and MPC in the planning and control section.

C. Output MPC for Single UAV

1) Global Planner: We designed UAV path planning algorithms based on A* and RRT*, modified A* for multi-UAVs applications, and used a zero-order keeper to obtain a smoothed discrete path sequence. The path planning approach is not the core of this project and therefore will not be described in detail here.

2) Output MPC Designing: During the state feedback process, not all states are observable due to the limitations of the sensors carried by the UAV. To represent this phenomenon in our simulation, we modify the observation model of the dynamics model to make the following states observable:

- The XYZ-axis absolute position of the UAV.

i.e., only these three terms of the C array in the dynamics equation are 1.

We take the position of the path point from the trajectory as y_{ref}, in order to realize the output MPC, we need to solve for the x_{ref} and u_{ref} corresponding to the y_{ref}.

These values can be obtained by solving the optimal target selection (OTS) problem, which we use can be represented by the following equation:

$$\begin{aligned} \min_{\mathbf{x}_{ref},\mathbf{u}_{ref}} & \sum_{i=0}^{N-1} \ell(x(k+i), u(k+i)) \\ \text{s.t.} & \begin{bmatrix} I-A & -B \\ C & 0 \end{bmatrix} \begin{bmatrix} x_{ref} \\ u_{ref} \end{bmatrix} = \begin{bmatrix} 0 \\ y_{ref} \end{bmatrix} \\ & (x_{ref}, u_{ref}) \in \mathbb{Z} \\ & Cx_{ref} \in \mathbb{Y} \end{aligned} \tag{21}$$

After solving for x_{ref} and u_{ref} by solving for x_{ref} and u_{ref} during each iteration, we modify the cost function in mpc to the following function:

$$\begin{aligned}\min_{\mathbf{u}} \quad & \sum_{i=0}^{N-1} \ell(x(k+i), u(k+i)) + V_f(x(k+N)) \qquad (22)\\ & \ell(x(k+i), u(k+i)) = \\ & x(k+i)'^T Q x(k+i)' + u(k+i)'^T R u(k+i)' \\ & V_f(x(k+N)) = x(N)'^T P x(N)'\end{aligned}$$

where $x(\cdot)' = x(\cdot) - x_{ref}$ and $u(\cdot)' = u(\cdot) - u_{ref}$.

Since the robot model is the same as previously established, in the output MPC for a single UAV, we use the same matrix parameters as designed in the Regular MPC.

D. Output MPC for Multi-UAVs

Obstacle avoidance among multiple UAVs is an important goal in the development of UAV cluster technology. The avoidance between multiple UAVs can be realized by excellent global planning, for example, using the modified A* algorithm UAVs designed in this project can realize the path separation in the global planning process. However, the modified global planning has the problem of high time complexity and non-optimal solutions.

Therefore, we will design special constraints for the MPC to accomplish the obstacle avoidance task among multiple UAVs, and thus obtain better control results with less time complexity.

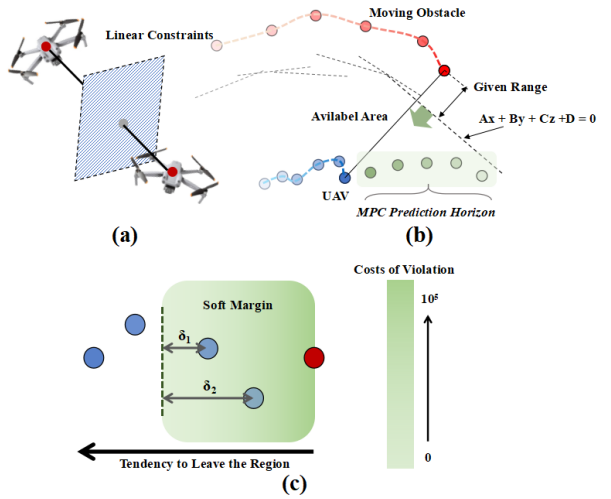

Fig. 3. Constraints design approach, where (a) expresses that the constraints between drones are planar linear constraints, (b) expresses the change in constraints over the runtime, and (c) expresses the features of the soft constraint design.

In this project, we first establish a strict planar linear constraint for the UAVs. This constraint is shown in Fig.3(a) and the planar constraints are updated in real-time according to the position of obstacle UAVs according to the method shown in Fig.3(b).

However, strict plane constraints can only deal with collisions with bias, and in direct collision scenarios solution errors will occur (this will be described in detail in section V). To solve this problem, we propose a soft-margin constraint MPC method as shown in Fig.3(c), which allows UAVs to approach each other to a degree but show a tendency to move away from each other. It can be represented by the following optimization problem:

$$\begin{aligned}\min_{\mathbf{u}} \quad & \sum_{i=0}^{N-1} \ell(x(k+i), u(k+i)) + V_f(x(k+N)) \qquad (23)\\ & \ell(x(k+i), u(k+i)) = \\ & x(k+i)'^T Q x(k+i)' + u(k+i)'^T R u(k+i)' \\ & + \delta(k+i)^T K \delta(k+i) \\ & V_f(x(k+N)) = x(N)'^T P x(N)' \\ \text{s.t.} \quad & \mathbf{n} \cdot Pos(k+i) \le b - \delta(k+i)\end{aligned}$$

where $\mathbf{n}$ is the normal vector to the plane constraint, $Pos(k+i)$ is the first three terms of the state $x(k+i)$, i.e., the positional quantities, $\delta(k+i)$ is the violated distance, which participates in the optimization process as an optimization variable.

Except for the new constraint design, all other parameters follow the matrix parameters from the previous sections.

IV. Asymptotic Stability

A. General Assumptions

In this section, we show that the designed MPC asymptotically stabilized the closed-loop system. With this aim, we verify the assumptions of Theorem 2.2, 2.3, 2.14 in the book [9].

Assumption 2.2: (Continuity of system and cost). The functions

$$f : \mathbb{Z} \to \mathbb{X}, \quad \ell : \mathbb{Z} \to \mathbb{R}_{\geqslant 0}, \quad V_f : \mathbb{X}_f \to \mathbb{R}_{\geqslant 0}$$

are continuous, $f(0,0) = 0, \ell(0,0) = 0$ and $V_f(0) = 0$. (Note that $\mathbb{Z} = \mathbb{X} \times \mathbb{U}$).

Assumption 2.3: (Properties of constraint sets). The set $\mathbb{Z}$ is closed and the sets $\mathbb{X}_f \subseteq \mathbb{X}$ is compact. Each set contains the origin.

Assumption 2.14: (Basic stability assumption).

(1) For all $x \in \mathbb{X}_f$, there exists a control law $u = \kappa(x)$ such that

$$\begin{aligned} & f(x, \kappa(x)) \in \mathbb{X}_f \\ & V_f(f(x, \kappa(x))) \le V_f(x) - \ell(x, \kappa(x)) \qquad (24)\end{aligned}$$

This is the control invariant terminal set and control Lyapunov condition.

(2) There exist $\mathcal{K}_\infty$ functions α_1, α_f satisfying

$$\begin{aligned} & \ell(x,u) \ge \alpha_1(|x|), \quad \forall (x,u) \in \mathbb{Z} \\ & V_f(x) \le \alpha_f(|x|), \quad \forall x \in \mathbb{X}_f \qquad (25)\end{aligned}$$

.

Assumption 2.2 is satisfied since the system dynamics f is continuous, and $f(0,0) = 0, \ell(0,0) = 0$ and $V_f(0) = 0$ (for the new origin in our case).

Assumption 2.3 is satisfied since we only have linear constraints in our problem, so the set $\mathbb{Z}$ is closed and the

sets $\mathbb{X}_f \subseteq \mathbb{X}$ is compact. Each set contains the (new) origin, as you can find in the previous section.

Assumption 2.14 (1) is satisfied since the terminal set $\mathbb{X}_f$ is control invariant and the control Lyapunov condition of terminal cost is satisfied because our design for terminal cost is

$$V_f(x) = x(N)'^T P x(N)'$$

where P is the solution to DARE with the aforementioned Q and R, which is proved to have the property of Lyapunov decrease. (2) is satisfied because the stage cost function

$$\ell(x,u) = x(k)'^T Q x(k)' + u(k)'^T R u(k)'$$

is lower bounded by $\alpha_1(|x|)$:

$$\ell(x,u) \geq x(k)'^T Q x(k)' \geq \lambda_{min}(Q)|x|^2 = \alpha_1(|x|)$$

and the terminal cost function $V_f(x)$ is upper bounded by $\alpha_f(|x|)$:

$$V_f(x) = x(N)'^T P x(N)' \leq \lambda_{max}(P)|x|^2 = \alpha_f(|x|)$$

B. Design of Terminal Set

As discussed in the previous section, terminal set $\mathbb{X}_f$ plays an important role in the stability of MPC control. It ensures the MPC optimal control input will be the same as the unconstrained control input defined by the infinite-horizon LQR problem. Within this set $u = \mathcal{K}_N(x) = Kx$ where K is the optimal LQR gain

When designing the terminal set, the three main principles we stuck to were the properties of control invariant, constraint admissibility, and Lyapunov decrease. The last property has already been satisfied by using the solution P of DARE as the weight matrix of the terminal cost. The former two principles are satisfied by constructing a control invariant set under the state and control input constraints. We use the method proposed in [10] to achieve this. Here is the algorithm we used in our code.

The main idea of this algorithm is to search in the evolution of the system to find two succeeding time steps where the next set is within the previous set. In our code, we built a constraint matrix H

$$H = \begin{bmatrix} H_u & 0 \\ 0 & H_x \end{bmatrix}$$

and via the help of the $K_{aug} = [K, I]^T$, the constraints can be expressed as

$$\begin{bmatrix} H_u & 0 \\ 0 & H_x \end{bmatrix} \begin{bmatrix} u \\ x \end{bmatrix} = H \begin{bmatrix} K \\ I \end{bmatrix} x \leq \begin{bmatrix} h_u \\ h_x \end{bmatrix}$$

The constraints for the next time step can be expressed as

$$H K_{aug} A_k x \leq \begin{bmatrix} h_u \\ h_x \end{bmatrix}$$

Algorithm 1: Compute terminal set

Initialization:
K = Unconstrained LQR gain
$A_K = A - BK$
$K_{aug} = [K; I]$
$k = 0$
For all i = 1, 2, ..., m,
$x_i^* = \underset{x}{\text{argmax}} f_i(K_{aug} A_K^{k+1} x)$
s.t. $f_j(K_{aug} A_K^t x) \leq 0 \quad \forall j \in 0, 1, ..., s,$
$\forall t \in 0, 1, ..., k$

If $f_i(K_{aug} A_K^{k+1} x) \leq 0 \forall j \in 0, 1, ..., s$
return $X_f = \{x \in \mathbb{R}^n | f_j(K_{aug} A_K^t x) \leq 0,$
$\forall j \in 0, 1, ..., s, \forall t \in 0, 1, ..., k\}$

Else k=k+1 and continue

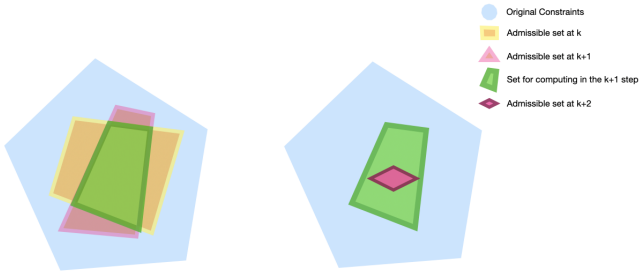

Fig. 4. The computation process of the terminal set. If we get a result like the left one, we use the green union set to compute in the next step. If we get a result like the right one, we return the plum red set as the terminal set

By using this algorithm we can build a control invariant set that won't violate the state and control input constraints. We also visualized the result set of this algorithm in the simulation, it turned out to be a horizon plane at the height of the goal position. It is predictable since we take the differential flatness of the quadrotor into consideration.

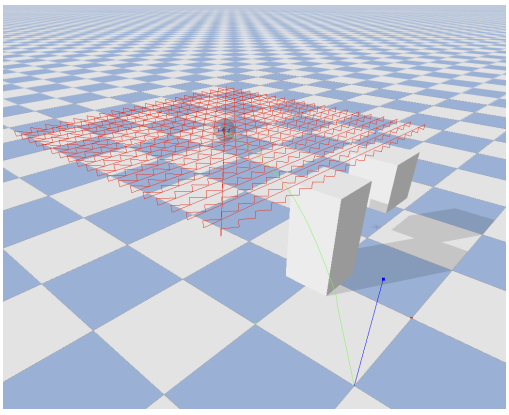

Fig. 5. The visualized terminal set computed by the algorithm. The red region is the representation of the plane of the terminal set and the grey ball is the goal position.

V. Simulations

We established numerical simulation and dynamic simulation environments based on Pybullet. In this section, we present the results of our algorithm implementation and compare three different algorithms in terms of functionality and performance.

A. Impact of Parameters on Regular MPC Performance

1) The Weight Matrices: To determine the effect of the Q and R arrays on the MPC results, we conducted experiments each with controlled variables, for example, maintaining the R constant while adjusting the values of the Q array in the first experiment. The results of the three experiments are shown in Fig.6 and Fig.7.

From Fig.6, it can be seen that as the Q weights increase, the cost of the system to the state error increases, and the system is more oriented to reach the target state quickly while overshooting due to acceleration constraints. The system's optimization for state error saturates when Q reaches near 150, at which point increasing the Q value will have no further effect.

From Fig.7, it can be seen that as the R weights increase, the system's cost to the inputs increases, and the system is more inclined to reach the target state while managing the cost of the inputs. When the system's weights on the inputs are large enough, there is a static difference in the position state along the z-direction, which is caused by the weights of the state errors being obscured by the input weights tend to keep the UAV still.

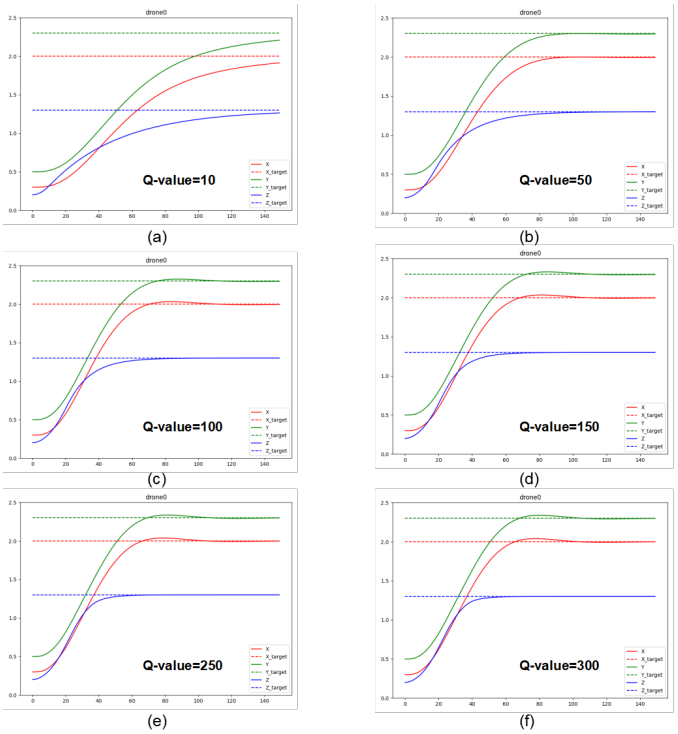

Fig. 6. The effect of Q weights on the performance of the system, where (a)-(f) weights are gradually increased.

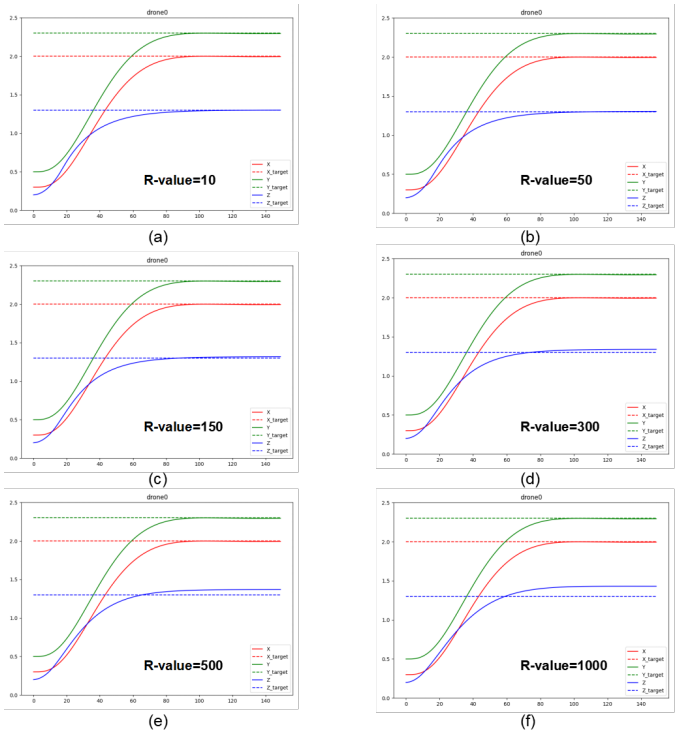

Fig. 7. The effect of R weights on the performance of the system, where (a)-(f) weights are gradually increased.

2) The Prediction Horizon: The prediction horizon also makes a difference in the performance, the longer the prediction horizon leads to a better solution, but it also results in an increase in computing time.

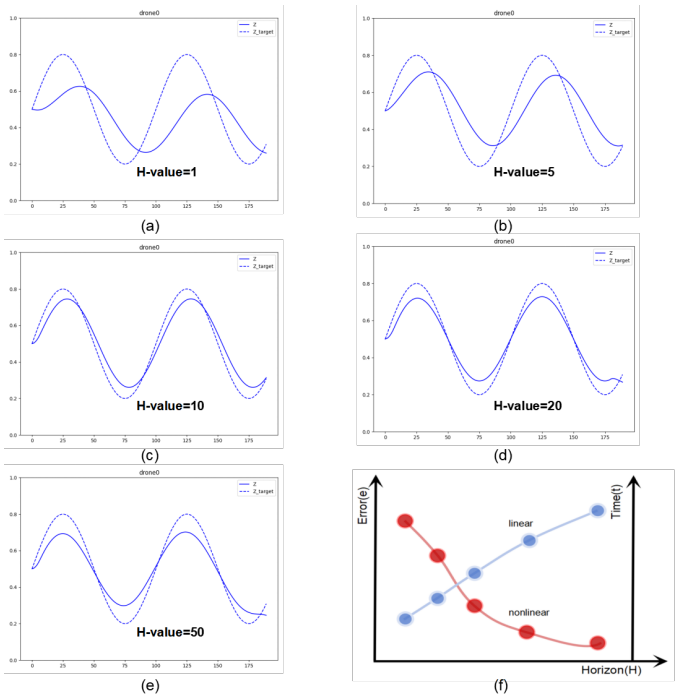

Fig. 8. The errors and computation times corresponding to different prediction horizon H.

We simulated different values of the prediction horizon and the results are shown in Fig.8 and TABLE.I. From the

TABLE I
RESULT OF PREDICTION HORIZON

H-Value	Computation Time	State Error
1	9.31	123.9
5	13.8	114.2
10	20.8	88.1
20	31.1	64.6
50	67.2	52.3

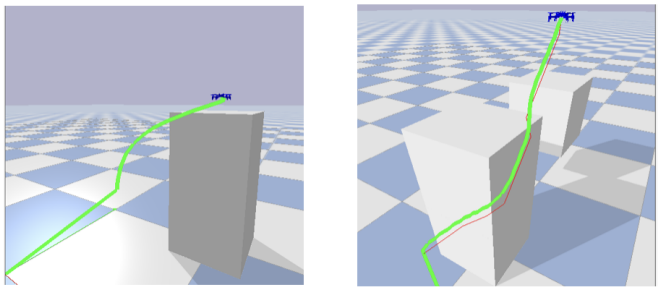

Fig. 9. Tracking results for different MPC controllers.

simulation results, we can see that the computation time grows and the error decreases as H increases. The computation time has a linear relationship with H, while the error shows a nonlinear relationship. H=10 is a balance of time and error.

B. Output MPC for Single UAV

1) Comparison with Regular MPC: Since the Output MPC needs a smaller horizon than Regular MPC (in our case the horizon Regular MPC needs is 10 times larger), the computation time is greatly reduced. Besides, in Regular MPC the controller will try to find a trajectory that costs the least energy (by minimizing the input) while in Output MPC the controller needs to follow the given path. The differences can be observed in the Fig.9.

C. Output MPC for Multi-UAVs

1) Comparison with Output MPC for Single UAV: The single UAV output MPC did not take into account the constraints between UAVs and obviously could not be applied to clusters of multiple UAVs, so we first introduced simple hard constraints for the output MPC. However, as shown in Fig.10(a), without priority and cost adjustment, the planning result produces deadlocks and leads to computational crashes.

As shown in Fig.10(b) and (c), we have made adjustments to the global planner such that the separability of paths is guaranteed during the global planning process. By doing so, the tightly constrained mpc works properly but consumes too much time in global planning and obtains results that are clearly non-optimal solutions.

Faced with the above problems of complex systems, we hope to achieve optimal and fast computation by using a pure mpc method, so we added a slack variable into the inter-UAV constraints which provides soft margins and punished it in the corresponding cost functions for avoiding computational

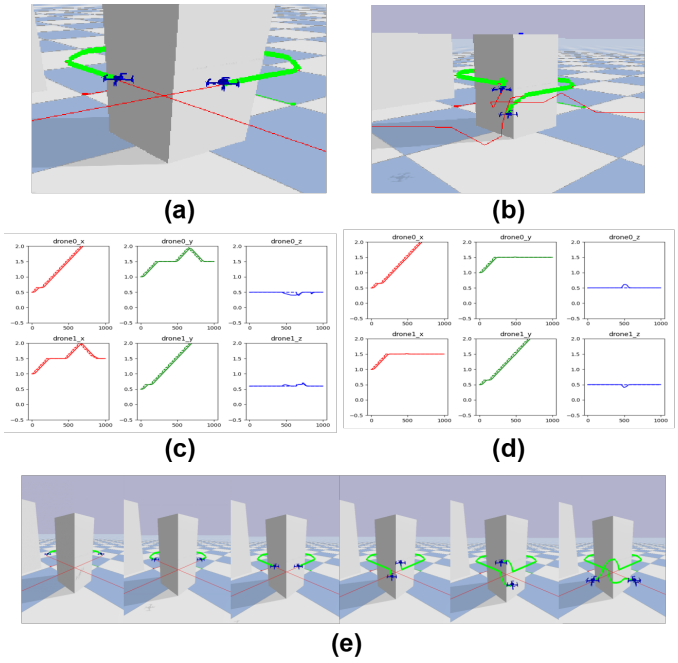

Fig. 10. Comparison of multi-UAVs and single-UAV Algorithms for Output MPCs, where (a) is the point at which the single UAV MPC algorithm crashes in responding to a collision, (b) and (c) is the non-optimal obstacle avoidance result produced by the hard constraint with modified A*, (d) and (e) is the optimal solution of soft margin constraints without a modified planner.

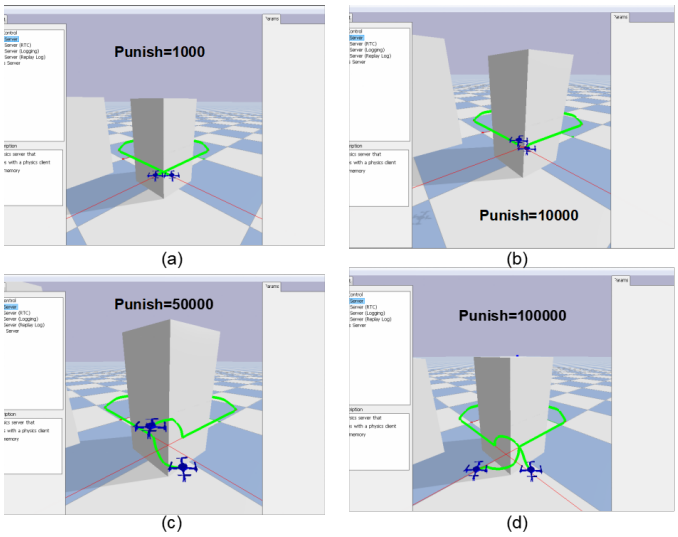

Fig. 11. Different results with different punishment coefficients.

crashes and realizing optimal solution results. Besides the inter-UAV constraints are solved online in the whole process, which ensures a dynamic collision avoidance between UAVs. As shown in Fig.10(d) and (e), this algorithm was applied to the most severe collision scenarios and proved to obtain much better results than the previous solutions.

2) Influence of the Parameters of the Soft constraints: The punishment coefficient is the most important parameter for soft constraints, as shown in Fig.11, we tested the effect of different punishment coefficients on the results. As shown in Fig. 11(a), when the punishment coefficient is small, the

algorithm ignores the constraint to follow the target trajectory. As shown in Fig.11(b), when the punishment coefficient rises but remains small, the UAV avoids the constraint when it is very close. As shown in Fig.11(c), when the punishment coefficient is suitable, the UAVs avoid each other softly. As shown in Fig.11(d), when the punishment coefficient is high, the UAVs produce planning results similar to hard constraints.

VI. Discussion

A. Summary

In this paper, we designed several MPC control strategies for the dynamic system of the quadrotor, which can be applied to navigation and obstacle avoidance with or without the presence of a global planner. We also extended it to a centralized multi-UAV MPC control by adding a slack variable into the inter-UAV constraints, which has a pretty good performance in solving the collision between UAVs with a small predicting horizon. We also visualized the terminal set in the simulation environment, the result also matches our different flatness assumptions.

B. Future Work

This work can be extended in the future by adding a Luenberger observer to get the ability to reject constant disturbance in the output under conditions of both state feedback and output feedback since in reality disturbance is inevitable and sometimes it is difficult to get full-state feedback.

References

[1] A. T. Hafez, A. J. Marasco, S. N. Givigi, M. Iskandarani, S. Yousefi, and C. A. Rabbath, "Solving multi-uav dynamic encirclement via model predictive control," *IEEE Transactions on control systems technology*, vol. 23, no. 6, pp. 2251–2265, 2015.

[2] J. Tang, H. Duan, and S. Lao, "Swarm intelligence algorithms for multiple unmanned aerial vehicles collaboration: A comprehensive review," *Artificial Intelligence Review*, vol. 56, no. 5, pp. 4295–4327, 2023.

[3] S. Ivić, B. Crnković, L. Grbčić, and L. Matleković, "Multi-uav trajectory planning for 3d visual inspection of complex structures," *Automation in Construction*, vol. 147, p. 104709, 2023.

[4] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Jurišica, "Path planning with modified a star algorithm for a mobile robot," *Procedia engineering*, vol. 96, pp. 59–69, 2014.

[5] W. Zhang, L. Shan, L. Chang, and Y. Dai, "Svf-rrt*: A stream-based vf-rrt* for usvs path planning considering ocean currents," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2413–2420, 2023.

[6] S. J. Wright, "Efficient convex optimization for linear mpc," *Handbook of model predictive control*, pp. 287–303, 2019.

[7] K. U. Lee, Y. H. Yun, W. Chang, J. B. Park, and Y. H. Choi, "Modeling and altitude control of quad-rotor uav," in *2011 11th International Conference on Control, Automation and Systems*. IEEE, 2011, pp. 1897–1902.

[8] Y. M. Al-Younes, M. A. Al-Jarrah, and A. A. Jhemi, "Linear vs. non-linear control techniques for a quadrotor vehicle," in *7th International Symposium on Mechatronics and its Applications*. IEEE, 2010, pp. 1–10.

[9] J. Rawlings and D. Mayne, *Model Predictive Control: Theory and Design*. Nob Hill Publishing, 2008.

[10] E. Gilbert and K. Tan, "Linear systems with state and control constraints: the theory and application of maximal output admissible sets," *IEEE Transactions on Automatic Control*, vol. 36, no. 9, pp. 1008–1020, 1991.

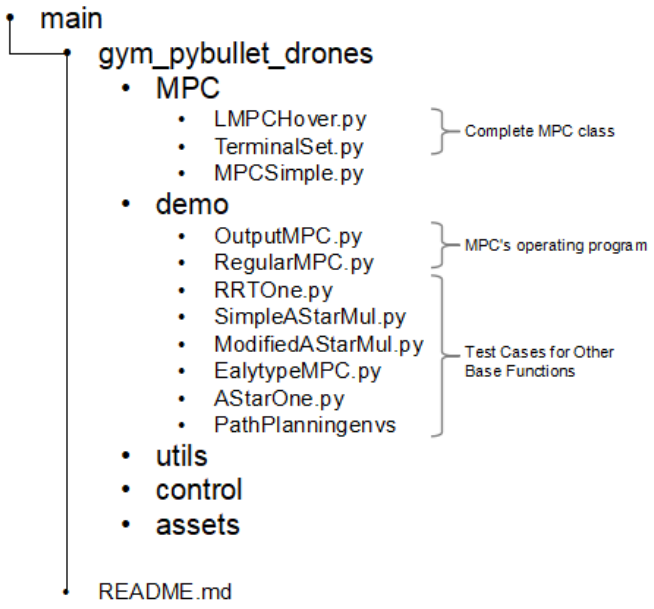

Fig. 12. Documentation and Program Composition of the Project.

Appendix

The file and program composition of this project is shown in Fig.12, where the MPC methods are mainly centralized in the MPC class and the MPC operation program.

Run RegularMPC.py in the drone anaconda environment to get the regular MPC simulation for a single drone, and OutputMPC.py to get the output MPC simulation for multiple drones. OutputMPC uses soft constraints by default, and hard constraints can be applied by changing the interface of the MPC class.