A model predictive control approach for Quadrotor collision avoidance

Moji Shi (5456320), Benjamin Piet(5614988)

***Abstract*—We study a model predictive control (MPC) approach for obstacle avoidance for a 3D quadrotor. We consider the case with an unknown wind disturbance, and noisy sensors.Our main work can be summarized as following: 1)We linearize the quadrotor system using Jacobian Linearization. 2)We convexify the obstacle constraints. 3) We split the MPC problem into two stages and prove the stability in the second stage. 4)We design an observer to estimate an unknown disturbance.**

I. Introduction

The use of drones in the world is growing, and whether it be for entertainment, deliveries or even military purposes, an automated drone will always need to be able to reach a goal while avoiding obstacles. One could simply have the problem divided in two parts : a planner that compute a trajectory for the drone to follow and a controller (for instance a simple PD controller could work) to follow this trajectory. We have an example of trajectory planning in [1] for a quadrotor for instance. But with MPC, we can do both part at the same time.

We take the case of a simple quadrotor that has to reach a goal, with a number of static obstacles between him and its goal. It is worth noting that we have a 3D workspace, but we focus on the movement in the horizontal plane. Because of this we model the obstacles like cylinder, which could correspond to a real life scenario of a drone that has to avoid trees while crossing a forest for example1.

II. Model definition and linearization

The systems dynamics are nonlinear. The dynamics of quadrotor can be described as:

$$m\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = R\begin{bmatrix}0\\ 0\\ F_1+F_2+F_3+F_4\end{bmatrix} + \begin{bmatrix}0\\ 0\\ -mg\end{bmatrix}$$

$$I\begin{bmatrix}\ddot{\phi}\\ \ddot{\theta}\\ \ddot{\psi}\end{bmatrix} = \begin{bmatrix}L(F_2-F_4)\\ L(F_3-F_1)\\ M_1-M_2+M_3-M_4\end{bmatrix} - \begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} \times I \begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix}$$

where m is the mass of the quadrotor, x, y and z are global position of the quadrotor, ϕ, θ and ψ are roll, pitch and yaw angle of the quadrotor, F_1, F_2, F_3 and F_4 are force provided by each rotor, I is the inertia matrix, L is the lever arm of rotor force, M_1, M_2, M_3 and M_4 are moments provided by each rotor along z axis of the quadrotor, R is the rotation matrix generated from euler angle ϕ, θ and ψ.

Moji Shi and Benjamin Piet are master students at TU Delft, The Netherlands. E-mail addresses: {M.SHI-5, B.C.G.PIET}@student.tudelft.nl.

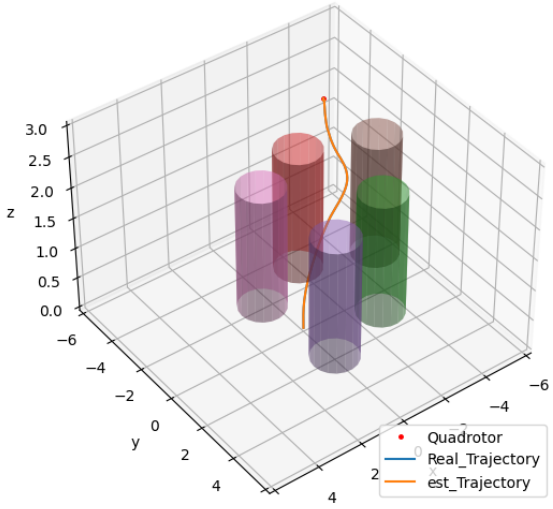

Fig. 1. Example of our method to avoid obstacles

For our obstacle avoidance problem, the yaw angle will be ignored in our case (we consider the orientation of the drone on the horizontal plane to be constant). We consider a 10-dimensional state space and a 4-dimensional input space:

$$\mathbf{x} = \begin{bmatrix}x\\ y\\ z\\ \dot{x}\\ \dot{y}\\ \dot{z}\\ \phi\\ \theta\\ \dot{\phi}\\ \dot{\theta}\end{bmatrix} \qquad \mathbf{u} = \begin{bmatrix}F_1\\ F_2\\ F_3\\ F_4\end{bmatrix}$$

With the state and input we can define the nonlinear system as $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$. For the design of our MPC controller, we are going to linearize the model for higher computational efficiency and stability analysis. Here we are going to use Jacobian linearization for nonlinear systems[2]. For linearization, we first need to define the equilibrium point of the system

where the system remains static around this point:

$$\bar{\mathbf{x}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \bar{\mathbf{u}} = \begin{bmatrix} \frac{mg}{4} \\ \frac{mg}{4} \\ \frac{mg}{4} \\ \frac{mg}{4} \end{bmatrix} \quad s.t. \quad f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = 0$$

We then further assume that the state and input deviate only a bit away from the equilibrium point(i.e. $\phi, \theta \sim 0$ and $F_1 + F_2 + F_3 + F_4 \sim mg$) and define the deviation from equilibrium point as δ_x and δ_u. In this case, the linearization of state space can be formulated as:

$$\begin{aligned} \dot{\delta_x} &= f(\bar{\mathbf{x}} + \delta_x, \bar{\mathbf{u}} + \delta_u) \\ &\approx f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) + \frac{\partial f}{\partial \mathbf{x}}|_{\bar{\mathbf{x}}, \bar{\mathbf{u}}} \delta_x + \frac{\partial f}{\partial \mathbf{u}}|_{\bar{\mathbf{x}}, \bar{\mathbf{u}}} \delta_u \\ &= A_c \delta_x + B_c \delta_u \end{aligned}$$

where the state space matrix can be formulated as:

$$A_c = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B_c = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & \frac{1}{m} & \frac{1}{m} & \frac{1}{m} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{L}{I} & 0 & -\frac{L}{I} \\ -\frac{L}{I} & 0 & \frac{L}{I} & 0 \end{bmatrix}$$

After that we can discretize the state space with time step as dt:

$$\begin{aligned} A &= I_n + A_c \cdot dt \\ B &= B_c \cdot dt \end{aligned}$$

where $n = 10$ is the size of state space and we have discretized state space:

$$\delta_x^+ = A\delta_x + B\delta_u$$

Considering the equilibrium point is all zero, it can also be written as:

$$x(k+1) = Ax(k) + B\delta_u$$

where $x(k)$ denotes state at k time step and for convenience we would directly use u to denote control deviation δ_u in the rest of the paper.

III. Model predictive control design

In this part, we will explain how we designed our MPC model. We will first present a general state-based MPC, and then see how we modified it by adding state disturbances and noisy sensors.

A. State-based MPC

The general model of our MPC can be defined as follows :

$$\begin{aligned} J(x_0, u) &= \sum_{k=0}^{N-1} \ell(x(k), u(k)) + V_{\mathrm{f}}(x(N)) \\ s.t. \qquad & x(k+1) = Ax(k) + Bu(k) \\ & u \in \mathbb{U}, x \in \mathbb{X} \\ & x(N) \in \mathbb{X}_{\mathrm{f}} \end{aligned}$$

We consider the MPC problem in two different stages. The first stage would be time-variant MPC where the obstacle constraints would change with the current state at every time step. After some time the obstacle constraints would not change(which means the convex zone already contains the target point), we compute the convex workspace once and don't modify it. In this case, the second stage would be time-invariant MPC where the constraints remain the same and we are able to add terminal cost and terminal set and have stability analysis.

1) Obstacle Constraint: Because we want to do convex optimisation, we need to design our constraints such that :

- the resulting workspace is convex;
- this workspace excludes the obstacles

To do this, we define a hyperplan for each of the obstacles that is tangent to the obstacle, and "facing" the drone. The convex workspace is then the intersection of the halfplanes defined by the previously mentioned hyperplan . It can thus be translated in the form of a linear constraint $A_{obs}(x(k))x(k) \leq b_{obs}(x(k))$, with the two matrix A_{obs} and b_{obs} that depends on the current position of the drone (see figure 2). This constraints is only applied on the x and y position of the drone.

It is also worth noting that it is in this convex workspace that we search for an intermediate goal, which is the target that the drone tries to reach at each time step. This intermediate goal is simply the closest point in the convex workspace to the final goal .

The major problem with this approach is that we can have the drone getting stuck when the goal, an obstacle and the drone are perfectly aligned. To solve this issue, we simply add a small random value to the "slope" of the hyperplan, so the intermediate goal is not in the alignment between goal, obstacle and drone.

2) State Constraint: Based on the assumption we made for linearizing the model, roll and pitch angle should be small. We assume that they are small than 0.5 rad. To be sure to stay linearisable, we thus impose the state constraints that can be represented in compact form as $A_{state}x(k) \leq b_{state}$, where A_{state} and b_{state} are defined as:

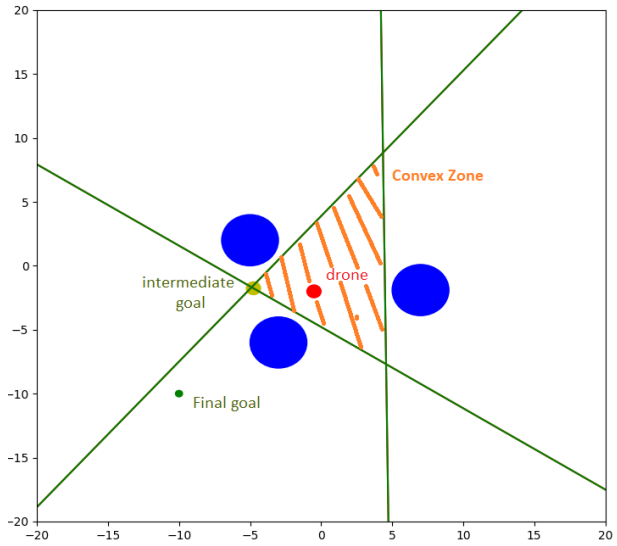

Fig. 2. Illustration of convexification for obstacle constraints

$$A_{state} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix} \quad b_{state} = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}$$

3) Input Constraint: The input constraints are set based on maximum force that can be provided by rotors and can be formulated as $A_u u(k) \leq c_u$ where:

$$A_u = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \quad b_u = \begin{bmatrix} \frac{mg}{4} \\ \frac{mg}{4} \\ \frac{mg}{4} \\ \frac{mg}{4} \\ \frac{mg}{4} \\ \frac{mg}{4} \\ \frac{mg}{4} \\ \frac{mg}{4} \end{bmatrix}$$

4) Stage Cost: We design the stage cost function:

$$\ell(x, u) = (x - x_{ref})^T Q (x - x_{ref}) + u^T R u$$

with $Q = I_n$ and $R = 0.02 \cdot I_4$. We chose these weight matrix because we want to prioritize state regulation over input minimization. The values comes from iterative tuning to find a reasonable MPC cost.

5) Terminal Cost and Terminal Set: We define the terminal cost function by

$$V_{\mathrm{f}}(x) = x^T P x$$

with P the solution of the discrete algebraic Riccati equation (DARE) used to solve the unconstrained infinite-horizon LQR problem. The terminal set can thus be defined by:

$$\mathbb{X}_{\mathrm{f}} = \{x \mid x^T P x <= c\}$$

with c a constant that we choose. We took $c = 0.01$, a choice that would be further explained in part III. As mentioned previously, we only use the terminal cost and set after that the intermediate goal is the same as the final goal, because until then we have time varying constraints.

6) Time Horizon: For the first part, we took a small N = 10 for better computing speed. This value is enough because the intermediate state is usually quite close to the drone position. For the second part we took a larger N = 30 because the final goal can be far away, and thus requires more time step to reach. Like for the value of the weight matrix Q and R, the value for dt and N were found through iterative tuning and testing.

B. Disturbances and noisy sensor

1) Disturbance and noise modeling: For this part and onward, we keep the same MPC design, but we change the dynamical model. Here we are going to add an unknown disturbance d on the state, and use an output y with sensory noise. The resulting model is as follows

$$x^+ = Ax + Bu + B_d d$$
$$y = Cx + C_d d + v$$

with $d = [d_x, d_y, d_z]^T$ a 3-dimensional disturbance that is going to represent a wind disturbance in three directions. Because of this, it will affect only the speed of the drone which means the B_d and C_d can be formulated as:

$$B_d = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad C_d = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

For the output, we will consider that we can measure the entire state (GPS for position, IMU for the rest or the Optitrack system which can measure the full state easily). In this case, we could formulate the output matrix: $C = I_n$.

Finally, v is a sensor noise with:

$$\sigma[v] = \begin{bmatrix} 0.05 \\ 0.05 \\ 0.05 \\ 0.001 \\ 0.001 \\ 0.001 \\ 0.001 \\ 0.001 \\ 0.001 \\ 0.001 \end{bmatrix} \quad \mathbb{E}[v] = 0$$

For the MPC loop, we are doing the same thing that was presented in lecture 5, that is summed up in figure 3

2) Observer Design: Lets first explain how we designed the observer. We are going to use a Luenberger observer[3] on the augmented dynamics which would be efficient given

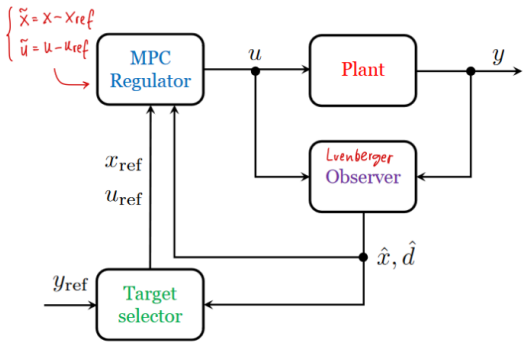

Fig. 3. MPC loop for output feedback from lecture 5

that the model is linearized. The augmented state space can be formulated as:

$$\begin{bmatrix} x^+ \\ d^+ \end{bmatrix} = \begin{bmatrix} A & B_d \\ I & 0 \end{bmatrix} \begin{bmatrix} x \\ d \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u$$
$$y = \begin{bmatrix} C & C_d \end{bmatrix} \begin{bmatrix} x \\ d \end{bmatrix} + v$$

With this Luenberger observer, we use linear feedback from the output to adjust our estimation at each time step :

$$\begin{aligned} \begin{bmatrix} \hat{x}^+ \\ \hat{d}^+ \end{bmatrix} &= \begin{bmatrix} A & Bd \\ I & 0 \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{d} \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u + \begin{bmatrix} L_1 \\ L_2 \end{bmatrix} (y - \begin{bmatrix} C & C_d \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{d} \end{bmatrix}) \\ &= \widetilde{A} \begin{bmatrix} \hat{x} \\ \hat{d} \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u + \widetilde{L}(y - \widetilde{C} \begin{bmatrix} \hat{x} \\ \hat{d} \end{bmatrix}) \end{aligned}$$

where $\hat{x}$ and $\hat{d}$ are estimated states in observer and $\widetilde{A}$, $\widetilde{C}$ and $\widetilde{L}$ are compact expression of the state space matrix. In order to make the system observable and to get the observer to converge, two requirements need to be satisfied:

- The augmented system is observable.
- Matrix $\widetilde{A} - \widetilde{L}\widetilde{C}$ is stable.(i.e. all of the eigenvalues should have norm less than 1.)

For the first condition we can easily check that (A,C) is observable and that

$$rank \begin{bmatrix} I - A & -Bd \\ C & C_d \end{bmatrix} = n + n_d = 10 + 3$$

So according to Hautus Lemma for observability in lecture 5, the augmented system is observable.

For the second condition, we defined the 13 eigenvalues that $\widetilde{A} - \widetilde{L}\widetilde{C}$ should have, and we used the function place from the "control" python library. The eigenvalue of $\widetilde{A} - \widetilde{L}\widetilde{C}$ should be the same as that of its transpose:

$$\begin{aligned} det|\lambda I - (\widetilde{A} - \widetilde{L}\widetilde{C})| &= det|\lambda I - (\widetilde{A} - \widetilde{L}\widetilde{C})^T| \\ &= det|\lambda I - (\widetilde{A}^T - \widetilde{C}^T\widetilde{L}^T)| \end{aligned}$$

With this transfer we can directly apply "place" function in control package to find proper $\widetilde{L}$. Like for the rest we found the final eigenvalues through iterative testing to get at the end the following values [-0.1, -0.1, -0.1, -0.03, -0.03, -0.03, 0.3, 0.3, 0.6, 0.6, -0.05, -0.05, -0.05].

3) Optimal Target Selector Design: Finally, we should present the target selector. We take the same one that was presented in lecture 5, meaning we search the reference state and input x_r and u_r that minimize

$$J(x_r, u_r) = u^T u + (y_{ref} - Cx_r)^T (y_{ref} - Cx_r)$$
$$s.t. \quad \begin{bmatrix} I - A & -B \\ C & 0 \end{bmatrix} \begin{bmatrix} x_r \\ u_r \end{bmatrix} = \begin{bmatrix} B_d\hat{d} \\ y_{ref} - C_d\hat{d} \end{bmatrix}$$
$$u_r \in \mathbb{U}, x_r \in \mathbb{X}$$

with $y_{ref} = [x_g, y_g, z_g, 0, 0, 0, 0, 0, 0, 0]^T$ the reference output, $[x_g, y_g, z_g]^T$ being the 3D position of the current goal (either final goal, or intermediate goal depending on where the drone is). $\mathbb{U}$ and $\mathbb{X}$ are the feasible input and state space, meaning the space delimited by the input constraint and state constraint that are defined above.

IV. Asymptotic Stability

In this section, we show that the designed MPC asymptotically stabilized the closed-loop system. To do this we will look only at the stability of the regulator MPC (so without the noise and disturbances), because according to the book [4], if the estimation of the disturbance converges, then the disturbed MPC will also be stable if the regulator MPC is stable.

Lets then first prove the stability of our regulator MPC. It is worth noting that, because until the intermediate goal is the same as the final goal our constraints are changing at each time step, we can't prove the stability of our MPC for this first part. We will only look at the case where we have the fixed linear constraints, the intermediate goal equal to the final goal, and where we added a terminal cost and a terminal set.

To prove the asymptotic stability, we will use the theorem 2.19 from [4] and the results from lecture 4. We will simply list the assumptions and justify why they hold in our case.

- linear dynamical model and quadratic stage cost and terminal cost : see previous section for the form of these functions

- $\mathbb{X}, \mathbb{U}, \mathbb{X}_f \subset \mathbb{X}$ compact and $0 \in int(\mathbb{X}_f)$: also quite clear with the form we gave them. The only not trivial part may be that $\mathbb{X}_f$ is included in $\mathbb{X}$, mainly converning the constriants with the obstacle. To deal with this part, we can simply suppose that the final goal is far away enough from the obstacles for the terminal set to be fully included in the constraints set.

- (Weak) Controllability : we can easily check that (A,B) is controllable after looking at the rank of the matrix

$$rank \begin{bmatrix} B & AB & ... & A^9B \end{bmatrix} = n$$

- Terminal cost as a control Lyapunov function in terminal set : because we used the solution of the discrete algebraic Riccati equation (DARE) for our terminal cost, we simply need to check that:

$$\forall x \in \mathbb{X}_f, Kx \in \mathbb{U}$$

where K is calculated from DARE. With this proven, we can use the results from the lecture to confirm that the terminal cost is a control Lyapunov function in the terminal set. With

the chosen form of the terminal set $\mathbb{X}_f = \{x \mid x^T P x \leq c\}$, we simply need to find the largest constant c that allows $Kx \in \mathbb{U}$. To do this, we took the outer approximation of the ellipsoid which is a 10-dimensional "box" P_{out} so that $\mathbb{X}_f \in P_{out}$ and check for every corners4. Some iterative testing got us the value of $c = 0.01$.

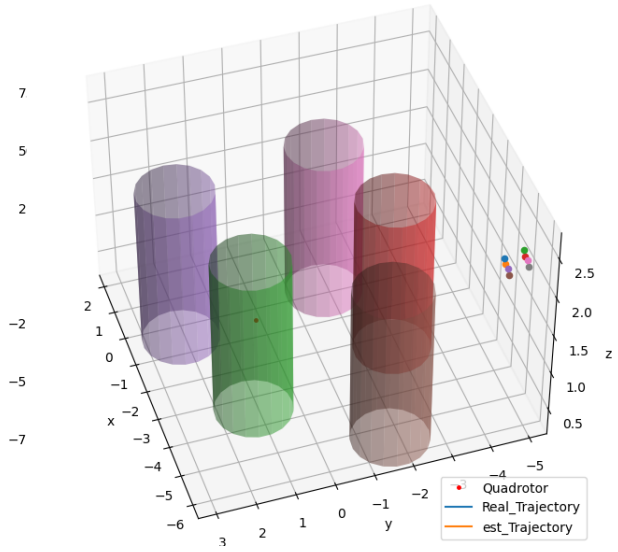

Fig. 4. Projection of P_{out} in 3D world(the cuboid with the points as the vertices)

With these assumptions, we can use theorem 2.19 of [4], and state that our MPC regulator is asymptotically stable. Finally, as mentioned previously, our disturbed system will also be stable if the estimation of the disturbances converges. This however will simply be checked with numerical simulation in the next section.

V. Numerical simulations

In this section, we run several numerical simulations where we test the function of MPC controller and compare some MPC controllers with different settings.

A. MPC with different time horizon

We first test our MPC controller with different time horizon when there is no disturbance and noise. We test the MPC controller with a time horizon of 5, 10 and 15 respectively5. As illustrated from the figure, the MPC controller with higher time horizon are able to converge quicker. However, when the horizon is more than 10, the improvement can be very small and there should be a trade-off between computational effort and the controller performance. We also record and compare the computational time for each MPC controller.

TABLE I
THE COMPUTATIONAL TIME FOR THE MPC CONTROLLER WITH DIFFERENT TIME HORIZON

Time Horizon	N=5	N=10	N=15
Computation Time(s)	15.3	12.7	13.4

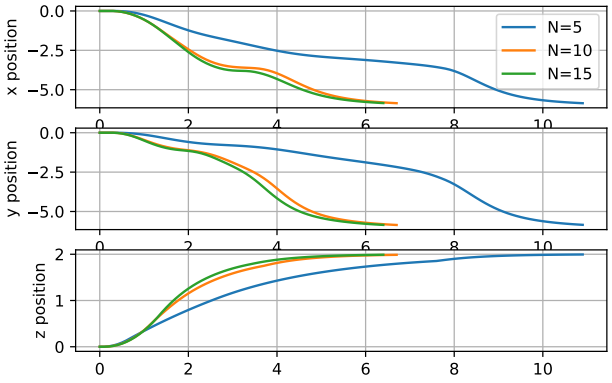

Fig. 5. MPC controller with different time horizon

According to the tableV-A, we have some counterintuitive findings where the controller with least time horizon takes the most time. This can be explained by the fact that the controller with less time horizon takes more time steps to reach the target. So even though for each step it takes less time to compute, the greater number of steps makes it more time-consuming overall.

B. MPC with different cost

In this problem, the major task is to arrive at the target position. In this case, we also want to test how the cost on x and y position would affect the performance of controller. In the following figure6 we have tried with different cost for x and y position.(The cost for rest of the state remains 1 for all the cases.) Through the figure it is easy to conclude that

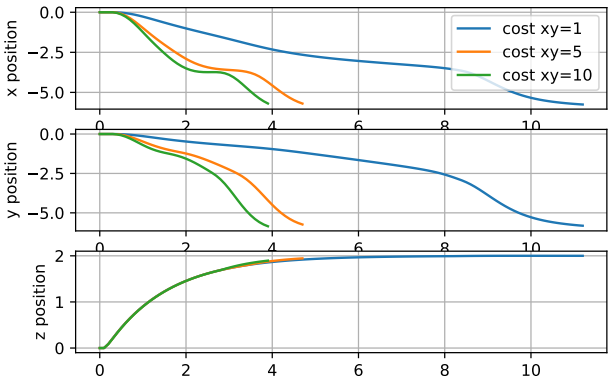

Fig. 6. MPC controller with different cost for xy

more weight on x and y would result in faster convergence for these two state variables. However if we set them as a very large value(e.g. cost of x and y is set to 100) then the solver would fail to find solution at some point. It may be because with such a high cost, going slightly away from the goal to avoid an obstacle is in the end not worth it, and the optimizer simply decides to stay on the same position.

C. MPC for disturbance estimation

In the experiment we consider sensor noise and unknown random disturbances for the speed of the drone in three directions.7 Given the experimental results, although there are

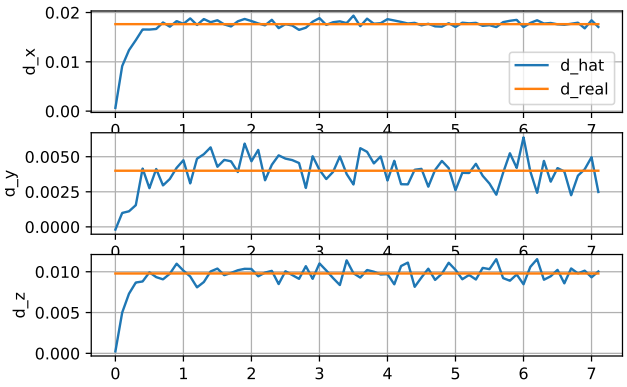

Fig. 7. Disturbance estimator for velocity at xyz directions

still some fluctuations caused by sensor noise(with scale of 0.001 in each direction), we can conclude that the estimation of the disturbance successfully converge to the real disturbances very fast. Even if the noise is relatively very large(e.g. for that in y direction) the observer is still able to identify the disturbance.

Additionally, since we are using estimated state as the input of our controller, we also want to test if the estimated state is close to real state. The following figure8 shows how the real state, the estimated state and the output state differ with each other. From the plot we can observe that through our estimator, the estimated state is very close to real state. In comparison with output state(with sensor noise), the estimated state is almost the same besides some delays.

VI. Conclusion

In this paper, we designed an MPC controller to solve the problem of obstacle avoidance for a quadrotor. We linearized the quadrotor dynamic model around the equilibrium point for better computational efficiency and stability analysis. We also convexified the obstacle constraint and formulate the state and input constraint to simplify the problem into a convex optimization problem. Additionally, we split the MPC problem into two stages, first with time-varying constraints and the second one is with time-invariant constraint. By doing that, we were able to design the terminal cost and terminal set for the second stage and prove the stability. Besides, we designed an observer so that the MPC controller is able to reject disturbances and noise.

With the designed model, we also did some numerical experiments with different settings of MPC controller. With these results, we discuss the performances of the MPC controller with different time horizon, different cost and we verify that our observer does indeed succeed in estimating the unknown disturbances.

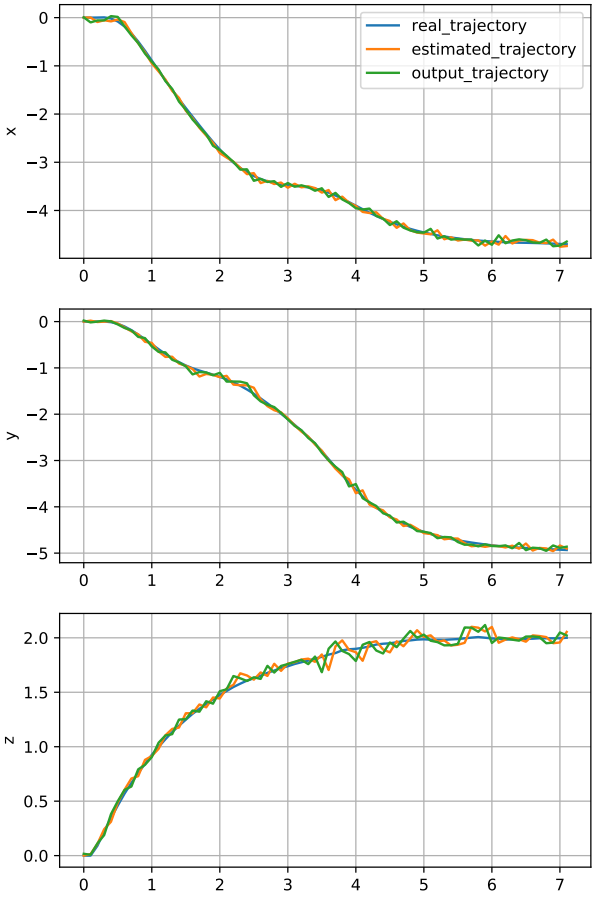

Fig. 8. Comparison between real state, estimated state and output state for xyz position

References

[1] X. Yu, X. Zhou, and Y. Zhang, "Collision-free trajectory generation and tracking for uavs using markov decision process in a cluttered environment," *Journal of Intelligent & Robotic Systems*, vol. 93, no. 1, pp. 17–32, 2019.

[2] M. Adler and P. van Moerbeke, "Linearization of hamiltonian systems, jacobi varieties and representation theory," vol. 38, no. 3, pp. 318–379. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0001870880900080

[3] B. R. Barmish and A. R. Galimidi, "Robustness of luenberger observers: Linear systems stabilized via non-linear control," vol. 22, no. 4, pp. 413–423. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0005109886900464

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