Literature Review Report

AE 490 Fall 2021

Hao Liu

wdliu@umich.edu

11/9/2021

1 Introduction

This is a report about what I've done during the undergraduate research, mainly focused on the papers that were used as reference in time order.

Here is a brief introduction to this project. Our general purpose is to have multiple quadrotors track multiple moving target platform. To achieve that, the research is divided into several steps:

- Build the dynamic system of a quadrotor so that it can track certain target speed.
- Find an appropriate tracking policy so that the quadrotor can track a moving traget
- Improve the 1 vs. 1 case to a n vs. n case, leading to the lowest tracking cost.
- Improve the full observation case to partial observation case.

The main reference paper for each part will be introduced in detail in the next part.

2 Literature review and achievements

2.1 Dynamic system and general control

We referred to [1] for the dynamic system of a quadrotor, which system is also used by many other papers. In brief, the state space system is defined as:

$$\mathbf{x}^T = (\dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi}), \tag{1}$$

where x,y, and z are the position of the quadrotor and ϕ,θandψ are the roll, pitch and yaw angle. And based on the dynamic model, we can get

$$\dot{\mathbf{x}} = \begin{bmatrix} -(\cos x_4 \sin x_5 \cos x_6 + \sin x_4 \sin x_6)u_1/m \\ -(\cos x_4 \sin x_5 \cos x_6 - \sin x_4 \sin x_6)u_1/m \\ g - (\cos x_4 \cos x_5)u_1/m \\ x_7 \\ x_8 \\ x_9 \\ x_8 x_9 I_1 - \frac{I_R}{I_x} x_8 g(u) + \frac{L}{I_x} u_2 \\ x_7 x_9 I_2 + \frac{I_R}{I_y} x_7 g(u) + \frac{L}{I_y} u_3 \\ x_7 x_8 I_3 + \frac{1}{I_z} u_4 \end{bmatrix}, \tag{2}$$

where

$$u_1 = b(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \quad (3)$$
$$u_2 = b(\omega_2^2 - \omega_4^2) \quad (4)$$
$$u_3 = b(\omega_1^2 - \omega_3^2) \quad (5)$$
$$u_4 = d(\omega_1^2 + \omega_2^2 - \omega_3^2 - \omega_4^2) \quad (6)$$

Here ω is the rotation speed of each motor b is the thrust factor and d is the drag factor.The purpose of the control is to calculate the desired u_1-u_4, with which we can get the rotation speed of each motor and update the state of the quadrotor.

The overall control system can be separated into two parts and the logic is shown in Fig. 1

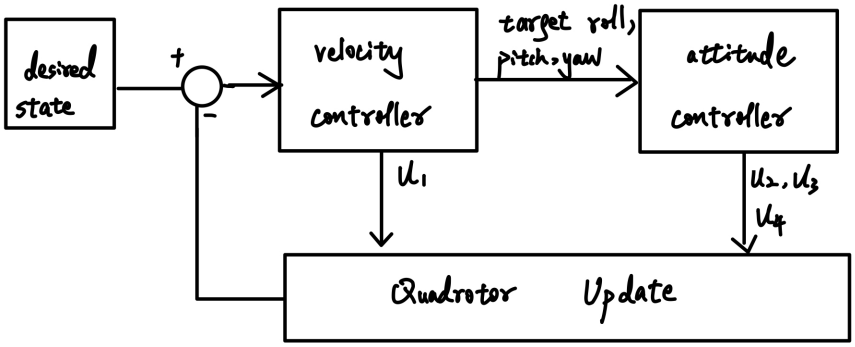


Figure 1: Angles during attitude control

For the velocity control part, we set x_{4d}, x_{5d} and x_{6d} to be the desired roll pitch yaw angle that are transferred to the attitude controller and they are also the desired angle to finish the velocity control. It's obvious that all the desired velocity and attitude can be reached without any yaw angle, thus we can set ψ=x_{6d}=0. In this way, the first three rows of the state can be updated by the following equations:

$$\dot{\mathbf{x}} = \begin{bmatrix} -(\cos x_{4d} \sin x_{5d}) u_1/m \\ -(\cos x_{4d} \sin x_{5d}) u_1/m \\ g - (\cos x_{4d} \cos x_{5d}) u_1/m \end{bmatrix}, \quad (7)$$

We apply simple proportional controller to solve the problem which is shown as follows.

$$\begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \end{bmatrix} = \begin{bmatrix} k_1(x_{1d} - x_1) \\ k_2(x_{2d} - x_2) \\ k_3(x_{3d} - x_3) \end{bmatrix} \quad (8)$$

Solving Eqn.(7) and Eqn.(8) and setting

$$\alpha = sinx_{4d} \quad (9)$$
$$\beta = sinx_{5d}, \quad (10)$$

we can get the following results if $\beta \neq 0$:

$$\beta = \pm \left[(\tfrac{g-\hat{x}_3}{\hat{x}_1})^2 + 1 \right]^{-\frac{1}{2}} \tag{11}$$

$$u_1 = m\sqrt{\frac{\hat{x}_1^2}{\beta^2} + \hat{x}_2^2} \tag{12}$$

$$\alpha = \hat{x}_2 \frac{m}{u_1} \tag{13}$$

Here β is negative if $\hat{x}_1$ is positive ad vice versa. If $\beta = 0$, we set

$$u_1 = mg.$$

With α and β calculated, we can calculate the desired x_{4d} and x_{5d} and make them an input into the attitude controller.

In the attitude controller, since the gyroscope term (g(u)) is very small compared with other terms thus we can neglect them first and the system becomes:

$$\begin{bmatrix} \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \end{bmatrix} = \begin{bmatrix} x_8 x_9 I_1 + \frac{L}{I_x} u_2 \\ x_7 x_9 I_2 + \frac{L}{I_y} u_3 \\ x_7 x_8 I_3 + \frac{1}{I_z} u_4 \end{bmatrix}. \tag{14}$$

We apply a feedback linearization in order to obtain a linear system:

$$\begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} f_2(x_7, x_8, x_9) + u_2^* \\ f_3(x_7, x_8, x_9) + u_3^* \\ f_4(x_7, x_8, x_9) + u_4^* \end{bmatrix} \tag{15}$$

with new input variables u_2^*, u_3^* and u_4^*. To make it a linear system, the following requirements must be meet:

$$x_8 x_9 I_1 + \frac{L}{I_x} f_2(x_7, x_8, x_9) = K_2 x_7 \tag{16}$$

$$x_7 x_9 I_2 + \frac{L}{I_y} f_3(x_7, x_8, x_9) = K_3 x_8 \tag{17}$$

$$x_7 x_8 I_3 + \frac{1}{I_z} f_3(x_7, x_8, x_9) = K_4 x_9 \tag{18}$$

so that the system can be written as

$$\begin{bmatrix} \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \end{bmatrix} = \begin{bmatrix} K_2 x_7 + \frac{L}{I_x} u_2^* \\ K_3 x_8 + \frac{L}{I_y} u_3^* \\ K_4 x_9 + \frac{1}{I_z} u_4^* \end{bmatrix}, \tag{19}$$

where $K_{2,3,4}$ are constants and

$$\begin{bmatrix} u_2^* \\ u_3^* \\ u_4^* \end{bmatrix} = \begin{bmatrix} w_2(x_{4d} - x_4) \\ w_3(x_{5d} - x_5) \\ w_4(x_{6d} - x_6) \end{bmatrix}. \tag{20}$$

$w_{1,2,3}$ are constants and $x_{4d,5d,6d}$ are the desired roll, pitch and yaw angle. For detailed proof of this transformation, please refer to [1].

With all the calculations above, we can calculate the input **u** at each time step and have the quadrotor track a target speed as well as keep itself stable. Here we first neglect the g(u) term in Eqn.(2) since it's very small compared with other term. Some simulation result is shown below. First, the initial roll, pitch and yaw angle is ($\phi = 30°$, $\theta = -20°$, $\psi = 10°$) and the target state is steady state with all the velocities to be zero. We got the following simulation result in Fig. 2.

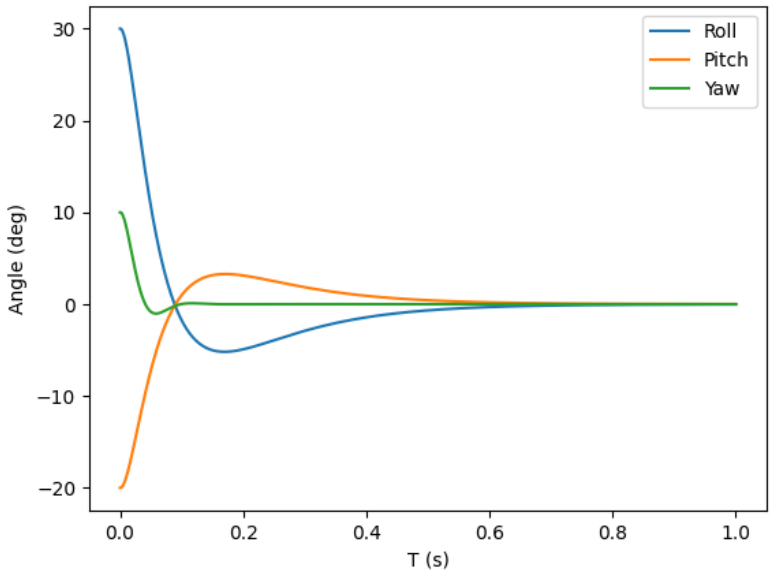


Figure 2: Angles during attitude control

It's easy to see that the system comes to stale after about 0.6 seconds.

The second test was proceeded with hovering initial condition and stable desired state with the speed to be 0.5 m/s in all the three directions. The result is shown in Fig. 3.

However, when we wanted to calculate rotation speed of each motor, it turned out that, at some time steps, ω^2 was negative, which indicated that there's no possible motor speed to realize such control policy. By testing, we found that negative ω^2 usually happened at the beginning of the control, which is usually the most intense part. So we guessed that the reason for the negative ω^2 was the too intense control input. Thus, we chose to control the magnitude of Eqn.(8) and Eqn.(20) to get a less intense control input, especially Eqn.(20). The less reasonable the initial state is, the smaller the maximum magnitude can be set and the slower the control is. For example, for the simulation result in Fig. 3, (from hovering state to another steady flight), the maximum magnitude for Eqn.(8) can be set to 100 and the maximum magnitude of Eqn.(20) can be set to 2.5 and get the result shown as Fig. 4 . However, for the simulation result in Fig. 2, (from certain angled state to hovering state),

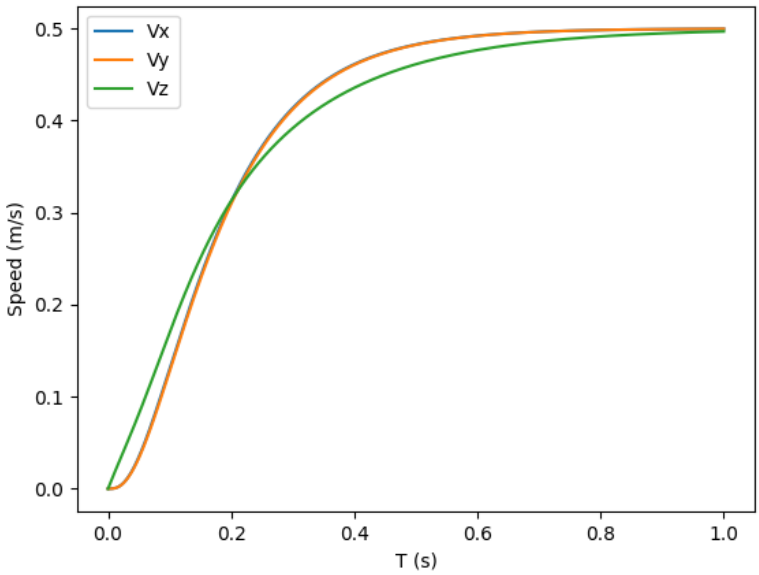


Figure 3: Speed during velocity control

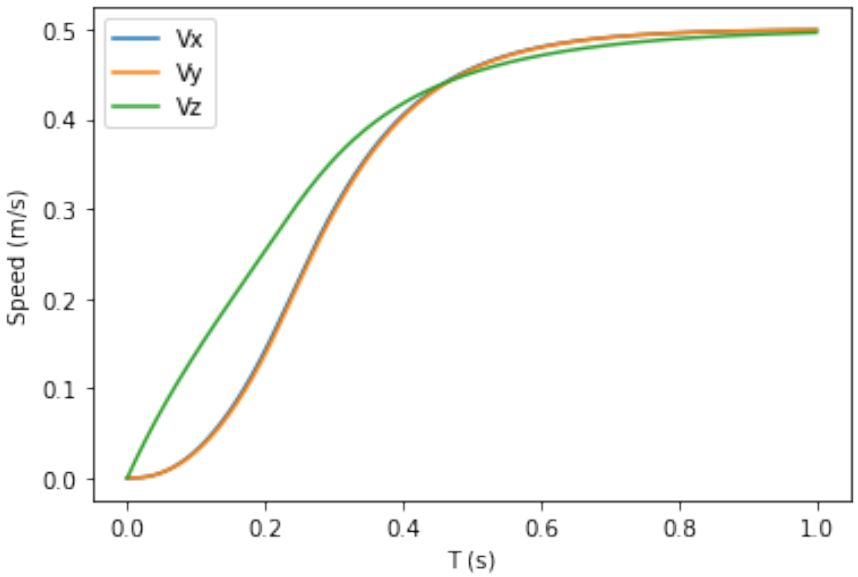


Figure 4: Speed control with control input constrain

since, unless set manually, it's impossible for a hovercraft reach an angled state by itself without any other speed or acceleration, the initial state has little physical meaning which means it's much harder to be controlled. In this case, the maximum magnitude for Eqn.(8) can only be set to 2 and the maximum magnitude of Eqn.(20) can only be set to 0.6 and get the result shown as Fig. 4 .

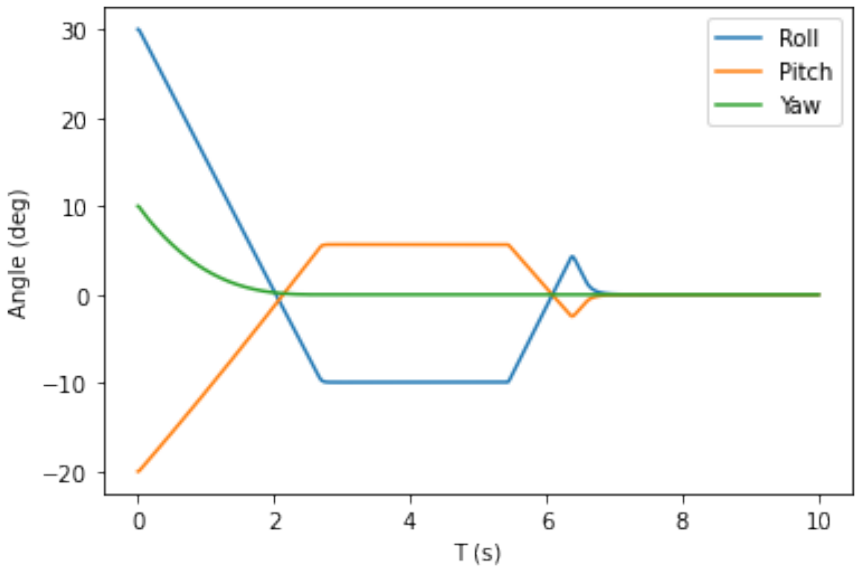


Figure 5: Angle control with control input constrain

Finally, we proved that the constrained control policy, if set proper constrain magnitude, can help track from initial states with physical meanings to different final states which also have physical meanings.

2.2 Tracking policy

Section 2.2 is the tracking simulation based one section 2.1. Instead of tracking some desired speed , the tracking target becomes to a moving platform. Generally speaking, we add a tracking controller of a classical missile guidance problem to provide the desired $\dot{\vec{v}}$, with which we can calculate u_1 and the desired roll, pitch and yaw angle for the attitude controller. The improved block diagram is shown as Fig.6

We referred to [2] for the tracking and landing policy. In [2], it constructed a tracking system based on the dynamic system shown in [1]. Instead of directly combining [1] and [2], we applied our constrained control policy in this part to make sure the control policy is not too intense to reach. The landing policy is just a first order control policy and the tracking policy is derived from classical missile guidance problem. Different from the state system in the previous section, positions of quadrotors were added to the states since the target of the control at this section is to reach certain target position. For now, the quad state is defined as

$$\mathbf{x}^T = (\dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi}, x, y, z). \tag{21}$$

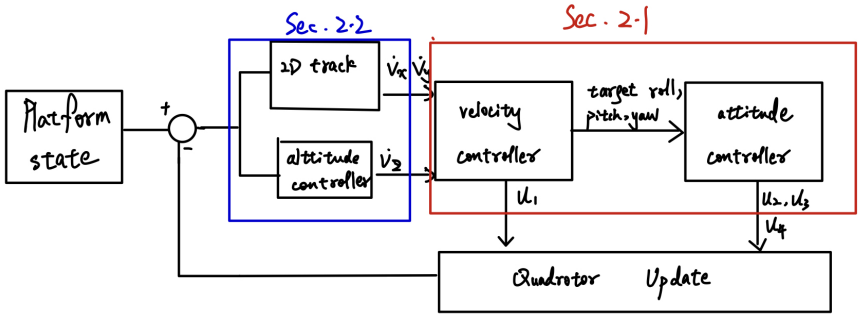


Figure 6: Block diagram of the tracking process

Two more new states, the distance state and the platform state, are also defined here:

$$\mathbf{distance}^T = (R, \sigma) \tag{22}$$

$$\mathbf{x}_p^T = (x_p, y_p, z_p, v_{P_x}, v_{P_y}, v_{P_z}) \tag{23}$$

The definition of distance is shown in Fig. 7. The platform state is composed of the position

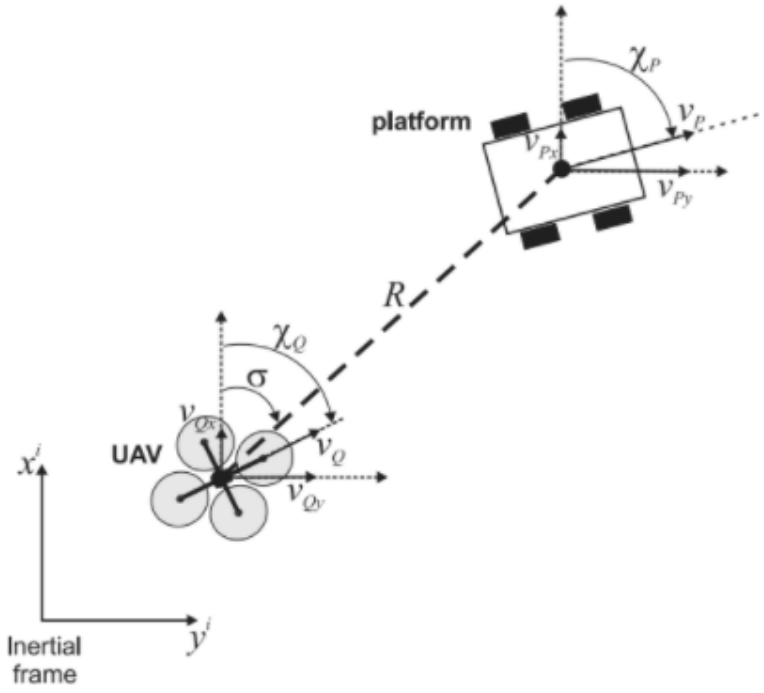


Figure 7: Definition of distance state[2]

of the platform and velocity of the platform. The distance and the quad speed is now updated

by:

$$\dot{R} = v_{P_x}\cos\sigma + v_{P_y}\sin\sigma - v_{Q_x}\cos\sigma - v_{Q_y}\sin\sigma \quad (24)$$

$$\dot{\sigma} = \frac{1}{R}(v_{P_y}\cos\sigma - v_{P_x}\sin\sigma - v_{Q_y}\cos\sigma + v_{Q_x}\sin\sigma) \quad (25)$$

$$\dot{V_{Q_x}} = -(1/T_1)V_{Q_x} + (1/T_1)u_1 \quad (26)$$

$$\dot{V_{Q_y}} = -(1/T_2)V_{Q_y} + (1/T_2)u_2 \quad (27)$$

Here, if R, the distance between the quadrotor and the target is smaller than a designed threshold, $u_1 = v_{P_x}$ and $u_2 = v_{P_y}$. Else,

$$u_1 = v_{P_x} + T_1 R\cos\sigma - T_1\frac{\sigma}{R}\sin\sigma \quad (28)$$

$$u_2 = v_{P_y} + T_2 R\sin\sigma - T_2\frac{\sigma}{R}\cos\sigma \quad (29)$$

The key issue is to apply this track controller with the quadrotor dynamic control system shown in the previous section. We use Eqn. (26) and Eqn. (27) to take the place of the first two rows in Eqn. (8). For the third row in Eqn. (8), the control logic is shown as Fig. 8, where the transfer functions can be written as:

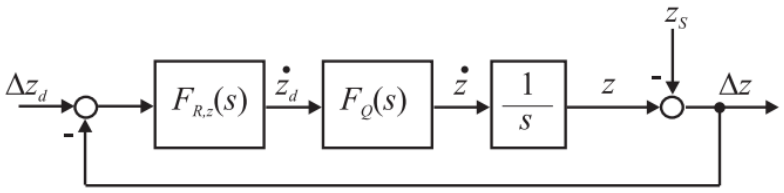


Figure 8: Altitude control loop. [2]

$$F_Q(s) = \frac{V_z(s)}{V_{zd}(s)} \approx \frac{1}{T_3 \times s + 1} \quad (30)$$

$$F_{R,z}(s) = K(1 + T_C \times s) \quad (31)$$

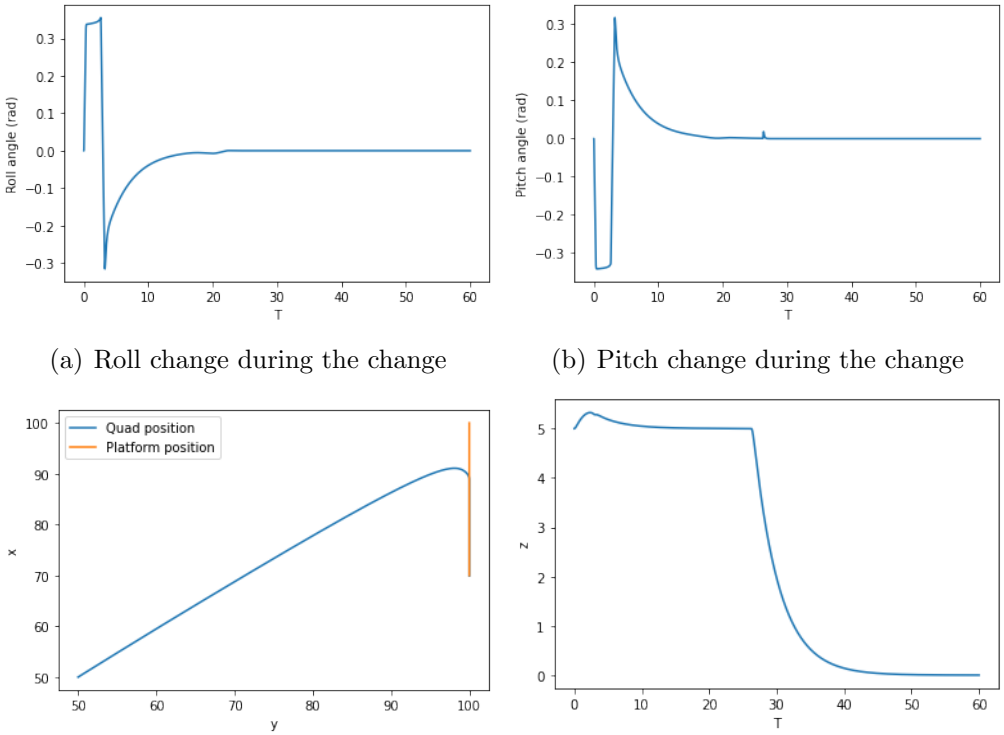
Here z is the altitude of the quadrotor, z_d is the target altitude (initial height during tracking process and the height of the platform during the landing process) and z_s is the surface altitude. $\Delta z = z - z_s$ and $\Delta z_d = z_d - z_s$.TC and k_3 are all constants. Inversely Laplace transform the transfer function, we can get the update equation shown as follows

$$v_{zd} = \frac{z_d - z - TC \times v_z}{1/K - TC} \quad (32)$$

$$a_z = k_3 \times (v_{zd} - v_z), \quad (33)$$

Eqn. (32) and Eqn. (33) took the place of the third row in Eqn. (8). Here, the magnitude constrain we chose for Eqn.(8) is 5 and for Eqn.(20) is 3.

The tracking result as well as its corresponding attitude change and altitude change is shown as Fig. 9.



(a) Roll change during the change

(b) Pitch change during the change

(c) Track

(d) Attitude

Figure 9: Tracking example with $\mathbf{x}_q$=[90,90,5] and $\mathbf{x}_p$=[10,10,0] and $V_{yp} = 0.5$

2.3 Cost assignment

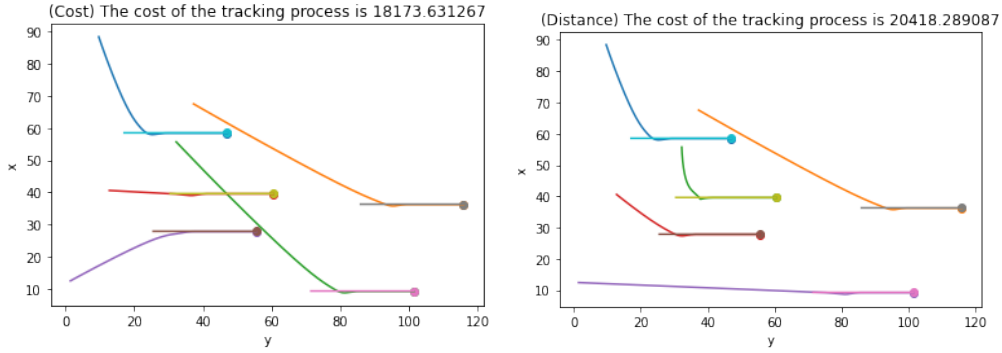
In this section, the problem is upgraded from 1 v.s. 1 problem to n v.s. n problem. The key point here is to find a best quad-platform assignment so that the tracking cost is to the lowest. First, the instant cost need to be defined to better illustrate how hard the tracking process is. Here, the Lyapunov function defined in [2] is taken as the instant cost.

$$\dot{q} = 0.5 \times (R^2 + \sigma^2 + (V_{qx} - V_{px})^2 + (V_{qy} - V_{py})^2) \tag{34}$$

Integrate $\dot{q}$ over time, we can get the tracking cost of one assignment. Then the problem changes to find the assignment with the lowest cost.

Firstly, calculate the cost of all the possible assignments and store them in a n×m matrix, where n is the number of quadrotors and m is the number of platforms (n≤m). Here we are talking about n v.s. n cases so n=m. Then apply the Python Optimal Transport package to find the best assignment based on the cost matrix and the number of quadrotors and platforms. The emd (intended for solving Earth Moving Distance problem) function is used to create the assignment matrix. The assignment matrix A is also a $n \times n$ matrix and it is sparse matrix with 1 on the specific assignment element. For example, if the p^{th} quadrotor is assigned to the q^{th} platform, then all the A[p,q]=0 and all other elements on the p^{th} row are 0. To fit the assignment matrix A, the state of quadrotors are stored in a $n \times 12$ matrix and the state of platforms are stored in a $n \times 6$ matrix, where n represents the number of quadrotors and platforms.

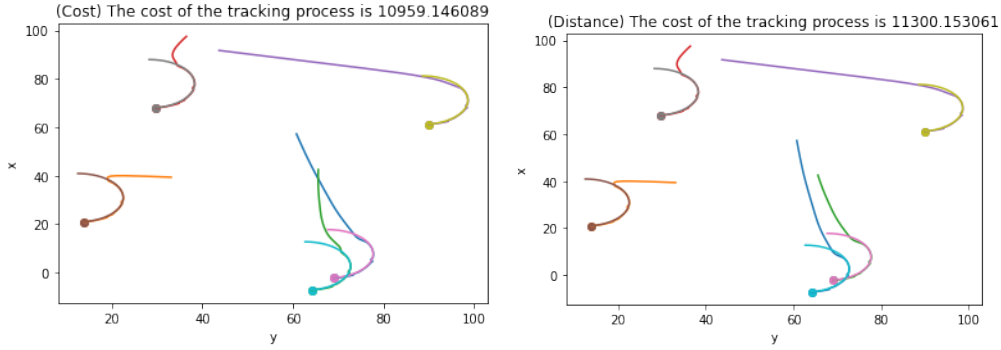
To prove the assignment can give the lowest cost, we calculated the cost of assignment based on distance as is shown in Fig. 10



(a) Based on cost (b) Based on distance

Figure 10: Tracking example with different assignment

The initial positions of all the quadrotors and platforms are generated randomly and we can easily see from the figure that the first assignment based on the cost has a lower cost than the distance-based assignment, proving the correctness of the method. This can also be applied to tracking platforms with non-constant velocity as is shown in Fig.11.



(a) Based on cost (b) Based on distance

Figure 11: Tracking example with different assignment and non-constant velocity platforms

Here, the magnitude constrain we chose for Eqn.(8) is 2 and for Eqn.(20) is 0.6.

2.4 Partial observation

In real life case, usually, it's hard for the quadrotors to monitor the speed of the platforms, leading to a partial observation case. In this part, We use the position of the platform read to estimate the speed of the platform so that the previous control policy still works. The overall control logic is shown as Fig. 12. Here we assume that we can fully observe the quadrotor with different sensors but we can only observe the relative position of the platform.

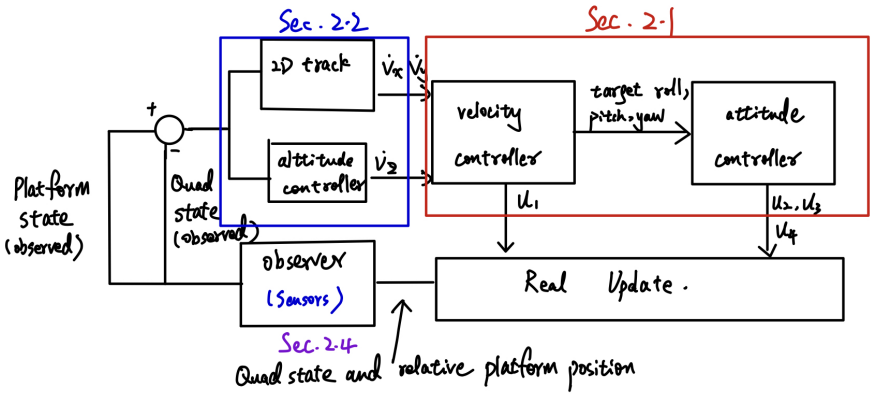


Figure 12: Block diagram of the partial observation tracking process

The main reference here is [3]. The logic of the observation system is shown in Fig. 13. We obtain relative position of the platform (by camera in [3]), based on which we can estimate the real position and speed of the platform. Since in our case, all the coordinates are in the

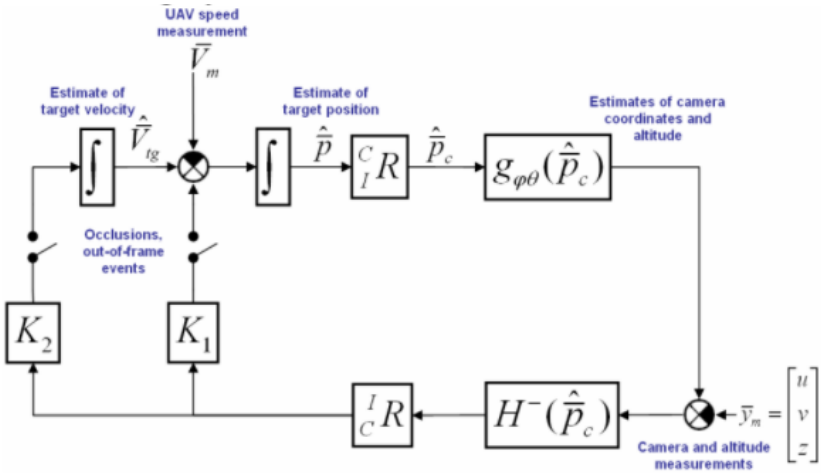


Figure 13: Observation system logic flow[3]

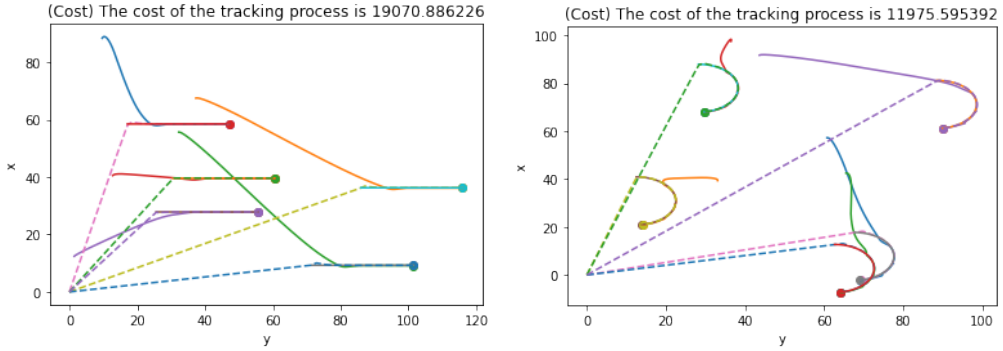
same reference frame, we can neglect the coordinate transformation terms R. The system can be written as:

$$\frac{d}{dt}(\bar{p}_{ob}) = -\bar{V}_m + \hat{\bar{V}}_{tg} + K_1 \times (\bar{p}_{ob} - \bar{y}_m) \tag{35}$$

$$\frac{d}{dt}(\hat{\bar{V}}_{tg}) = K_2 \times (\bar{p}_{ob} - \bar{y}_m) \tag{36}$$

Here $\bar{p}_{ob}$ is the observed position of the platforms, $\bar{y}_m$ is the exact position of the platform (should be the data extracted from the camera as [3], but since it's wasting of time converging the state to the camera reference frame and then transfer back to the global state, here we directly used the exact position), $\bar{V}_m$ is the velocity of the quadrotor and $\hat{\bar{V}}_{tg}$ is the estimated target speed relative to the quadrotor. The exact estimated speed of the platform in the global frame is $\frac{d}{dt}\bar{p}_{ob}$. In our simulation, we set the initial observation position of the platform

to be [0,0,0] and use the observed value in the tracking controller instead of the real position of the platform. We applied partial observation to the simulation the the previous section and achieved the result shown as Fig. 14 , where the dotted lines shows the observed position



(a) Constant speed (b) Non-constant speed

Figure 14: Tracking example with partial observation

of the platforms

Also, in [3], it says this observation method can only be used to those with constant speed but as is shown in Fig. 14, it also works well with non-constant speed target. The limitation seems to be that the target can't change its speed to rapidly or it can be observed.

3 Conclusion and Future

Based on the discussion above, we built a partial-observation tracking system which can be applied to n v.s. n cases. The capability of the model is also justified.

For future study, the model for now is only at theoretical stage and whether the quadrotor can move as the simulation shows is still unknown. The delay of the actuator is not considered in this model as well. We need to build it in the lab for more experimental data so that we can adjust some control parameters. Also, in the model, some coordinate convention is neglected since all the coordinates in the model are in the same global frame. In reality, the data of platform position is actually in the quadrotor coordinate. Some further data adjustment is necessary.

References

[1] H. Voos, "Nonlinear control of a quadrotor micro-uav using feedback-linearization," in *2009 IEEE International Conference on Mechatronics*, 2009, pp. 1–6.

[2] H. Voos and H. Bou-Ammar, "Nonlinear tracking and landing controller for quadrotor aerial robots," in *2010 IEEE International Conference on Control Applications*, 2010, pp. 2136–2141.

[3] V. N. Dobrokhodov, I. I. Kaminer, K. D. Jones, and R. Ghabcheloo, "Vision-based tracking and motion estimation for moving targets using small uavs," in *2006 American Control Conference*. IEEE, 2006, pp. 6–pp.