DP4: Control of a Quadrotor Drone

Omar Elhayes and Pedro Leite
University of Illinois at Urbana-Champaign, Champaign, IL, 61820

The aim of this report is to explain how we designed, implemented, and tested a controller and observer that enables a quadrotor (aka drone) to race through rings from a starting to a finishing position without crashing. We identified failures (the drone does not reach the end) and explained how the drone was failing. We also set a requirement for the system and verified that requirement to ensure the system is successful. We linearized equations of motion to derive a state-space model and a sensor model based in the rings' center positions, and applied the LQR method to choose a linear state feedback that would, in theory, produce a stable close-loop system that maintains desired values as closely as possible by controlling the torques on the drone's four rotors. Our controller and sensor allowed our drone to race without failure for approximately 98% of the time, with an average time of 11.27 seconds while having to correct for sensor noise.

I. Nomenclature

A, B, C	=	matrices that describe the state-space model
f_z	=	net rotor force (N)
J	=	cost function
K	=	controller gain matrix
L	=	observer gain matrix
P	=	matrix to be solved for in the continuous-time Riccati equation
p_x, p_y, p_z	=	components of position (m)
ψ, θ, ϕ	=	yaw, pitch, and roll angles (rad)
Q, R	=	diagonal matrices to be placed for finding K matrix
τ_x, τ_y, τ_z	=	components of net rotor torque (Nm)
v_x, v_y, v_z	=	components of linear velocity (m/s)
w_x, w_y, w_z	=	components of angular velocity (rad/s)
x, u	=	state and input of linear system
ζ, μ	=	state and input of nonlinear system

II. Introduction

The focus of this project is a quadrotor or drone that has to maneuver from a starting to an ending position while moving through multiple rings. Drones are used everywhere including but not limited to journalism, search and rescue, disaster response, wildlife monitoring, communications relay, and agriculture. Their applications are abundant and, in this project, we set out to design a basic drone. To implement this we had to linearize the dynamic and sensor model of the system (drone), verify that the system is stable, controllable and observable, and add trajectory tracking to enable the drone to move between rings. We then implemented the the controller and observer with the trajectory tracking and tested it in simulation using PyBullet. With the simulation, we had to identify and diagnose sources of failure (drone not reaching the end) in 100 simulations. As an extra step, we can race our drone system against other drones. To do this, we can implement a way for the drone to avoid collision with other drones. Not to mention, we defined a requirement for success for our system and a way to verify this success as seen in the experimental methods section. A snapshot of the system can be seen in Fig. 1.

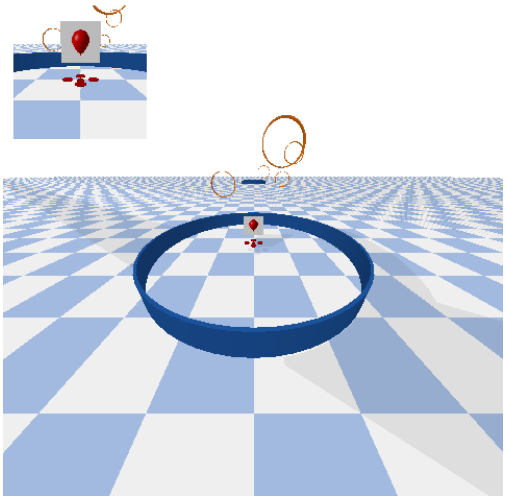


Fig. 1 Snapshot of the system.

III. Theory

A. Derive state-space model

The equations of motion for our system are described as

$$\dot{\zeta} = f(\zeta, \mu) = \begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_y \\ \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \\ \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \\ \dot{w}_x \\ \dot{w}_y \\ \dot{w}_z \end{bmatrix} = f\left(p_x, p_y, p_z, \psi, \theta, \phi, v_x, v_y, v_z, w_x, w_y, w_z, \tau_x, \tau_y, \tau_z, f_z\right). \tag{1}$$

By solving

$$0 = f(\zeta_e, \mu_e), \tag{2}$$

we see that any equilibrium point of (1) has the form

$$\zeta_e = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{9.81}{2} \end{bmatrix}^T \qquad \mu_e = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T. \tag{3}$$

If we define

$$x = \zeta - \zeta_e \qquad u = \mu - \mu_e, \tag{4}$$

then (1) can be approximated close to the equilibrium point by the linear system

$$\dot{x} = Ax + Bu \tag{5}$$

where

$$A = \left.\frac{\partial f}{\partial \zeta}\right|_{\zeta_e, \mu_e} \qquad B = \left.\frac{\partial f}{\partial \mu}\right|_{\zeta_e, \mu_e} \tag{6}$$

Suppose a continuous-time linear system described by the closed loop system

$$\dot{x} = (A - BK)x, \tag{7}$$

with a cost function defined as

$$J = \int_{t_o}^{\infty} \left(x(t)^T Q x(t) + u(t)^T R u(t) \right) dt, \tag{8}$$

where Q and R are diagonal matrices. The input for the linear state feedback that minimizes the cost J is

$$u = -Kx, \tag{9}$$

where the gain matrix K is given by

$$K = R^{-1} B^T P \tag{10}$$

where P is the matrix found by solving the continuous-time algebraic Riccati equation

$$PA + A^T P - PBR^{-1} B^T P + Q = 0. \tag{11}$$

We chose diagonal matrices

$$Qc = diag \begin{bmatrix} 1.1x10^4 & 1.1x10^4 & 1.1x10^4 & 10^3 & 10^3 & 10^3 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{12}$$

$$Rc = diag \begin{bmatrix} 10^5 & 10^5 & 10^5 & 10^2 \end{bmatrix} \tag{13}$$

that output an optimized controller gain matrix K which should in theory result in a stable system. Taking this into account, we proceeded as shown below.

$$\dot{x}_{err} = (A - LC)x_{err} \tag{14}$$

The next step is to find a linear approximation for our sensor model. The sensor model is represented as

$$o = g(\zeta, \mu) = \begin{bmatrix} p_x \\ p_y \\ p_z \\ \psi \end{bmatrix}, \tag{15}$$

with an output defined as

$$y = o - g(\zeta_e, \mu_e), \tag{16}$$

that can be further simplified and written as

$$y = Cx, \tag{17}$$

this gives us the desired linear approximation of the sensor model.

From there, the next step is to compute the C matrix which can then be written as

$$C = \left. \frac{\partial g}{\partial \zeta} \right|_{\zeta_e, \mu_e} \tag{18}$$

We can use this computed C matrix to find the next necessary component which would be the observer gain matrix. We can compute the L matrix in a similar fashion to K: applying LQR. The weights we chose for our observer LQR are

$$Qo = diag \begin{bmatrix} 10^3 & 10^3 & 10^3 & 6x10^{-3} \end{bmatrix} \tag{19}$$

$$Ro = diag \begin{bmatrix} 2x10^{14} & 2x10^{14} & 2x10^{14} & 10^{16} & 10^{16} & 10^{16} & 10^{-3} & 10^{-3} & 10^{-3} & 10^{-3} & 10^{-3} & 10^{-3} \end{bmatrix}. \tag{20}$$

B. Determine Conrollability and Observability

To verify that the system is controllable, we used the A and B matrices to define the controllability matrix Wc as in Equation (21), where n is the number of states.

$$Wc = [B, AB, A^2B, ..., A^{n-1}B] \tag{21}$$

The system will be controllable if the rank of matrix W is the same as the number of states of A. After calculating W, the rank obtained was 12, which is the same as the number of states. So we can confirm that the system is controllable.

On the other hand, verifying that the system is observable is very similar to checking that the system is controllable. The matrix built is called the observability matrix, W_o, and is built as in Equation (22). The system is observable if the matrix is full rank, that is if the rank is the same as the number of states of A.

$$Wo = [C^T \quad A^TC^T \quad (A^T)^2C^T \quad ... \quad (A^T)^{(n-1)}C^T] \tag{22}$$

The rank obtain when implementing Equation (22) is 12, so it is verified that the system is observable.

Having found the necessary matrices and equilibrium conditions for our system, we can implement our controller by setting target positions at each of the rings, that target position being updated each time our drone reaches a ring. More precisely, the destination is constantly updated as the drone moves, accounting for the desired direction of the next ring, but taking small steps towards to avoid the system commanding an excessive amount of torque to cover the entire distance at once.

IV. Experimental Methods

A. Requirements and Verification

We defined our requirements for success over 100 simulations as the following:

1) The percentage of failure among the simulation would be 10% or less.
2) The average completion time for the drone would be under 30 seconds.

This requirement must be met with initial conditions that entail that the drone is sitting still on the ground at the starting position (initial conditions of zero torque/velocity). We verified this through the use of PyBullet simulations. Through the 100 simulations, we gathered data to plot histograms seen in the results and discussion section. With the data from the simulations, we can verify our requirements.

B. Diagnosis of Failures

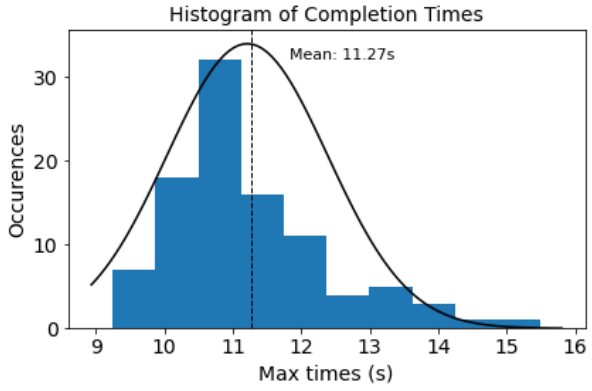
The failures gotten throughout our 100 simulations consisted of runtime failures and the drone not finishing within our conservative time limit, chosen as 20 seconds based on our average completion times. The majority of the failures in our system occur from runtime errors, meaning that the time it took to compute the next time step in our simulation was longer than the set upper limit of 0.01 seconds. This could be improved by optimizing our controller or the drone system itself. Our second largest source of error comes from hitting the walls of the rings. One way this problem was improved was by setting the desired height at the beginning and end of the race to be above that of the ring wall, and at the end to set the torques to zero when it is within a certain distance of the final ring's center coordinates. Another way we dealt with this problem was to set a basic collision avoidance system between the drone and the rings, so that the drone would avoid airspace that touches the outside of the rings as well as areas close to the inside of the rings.

C. Experiments to find best performance based on time ran

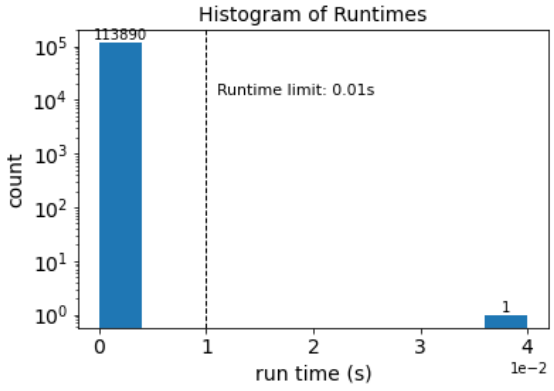
By changing the weights in LQR for both the controller and observer gain matrices we found that, in general, the performance in regards to average completion time was inversely proportional to the performance in regards to successful runs, meaning that in order to improve one, the other would have to be sacrificed. Still, by playing with both the K and L matrices, we managed to get very short completion times while still having high success rates, however that took a lot of manual optimization to find weights that work well together.

V. Results and Discussion

We have shown through simulation that both of our requirements have been met. By simulating 100 runs with no run time errors, we have achieved a 98% success rate with an average completion time of 11.27 seconds. This meets our requirements for success. A histogram of our completion times and run times can be seen in Fig. 2.



(a) A histogram of the completion times for each of our 100 runs.

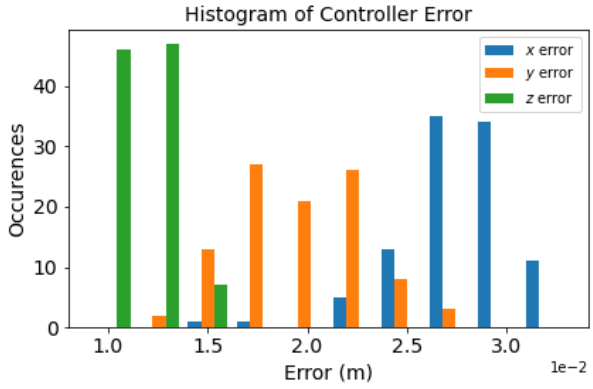


(b) A histogram of the run times of all time steps for all 100 runs.

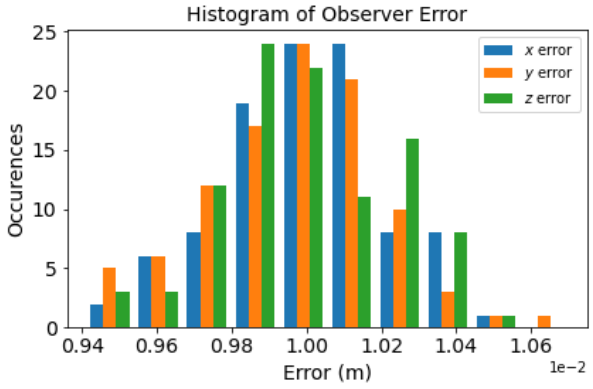
Fig. 2 Plots of the completion times and run times for 100 runs.

As can be seen by Fig. 2a, our completion times seem to fall in a normal distribution, with a mean falling well under the 30 second mark, as it expected from our average value of 11.27 seconds. Also, as observed in Fig. 2b, the vast majority of our run times is within the required 0.01 second range, but there are a few outliers that would normally cause our simulation to fail. These are likely a result of both a not-ideal implementation of our controller, as well as systematic errors deep within the code we were given, and the way the computer prioritizes all the tasks running in it.

We also plotted histograms of the controller and observer error to see how well our system is implemented. Those can be seen in fig 2 below.



(a) A histogram of the RMSE between the drone's actual position and desired position.



(b) A histogram of the RMSE between the drone's actual position and estimated position.

Fig. 3 Histograms of the Root-Mean Squared Error between the actual position, estimated position, and desired position of our drone.

The error plots seen above were taken by subtracting the desired or estimated positions from the actual position, and taken the RMSE of those values. This allows us to see on average how far our drone is from where is should be or is observed to be, which gives us meaningful data about the quality of our implementation. As can be seen from the plots, our controller and observer have been implemented to work fairly well with our definition of success. Both plots seem

to have the majority of its data in the hundredths place, which can be seen more easily in Fig. 3b than in Fig. 3a, but this further confirms the quality of our system.

VI. Conclusion

This report described our approach to design and test a controller and observer system that make a quadrotor drone fly through the center of many vertical rings until arriving at the last horizontal ring. Our first requirement was to have our drone finish its races at least 90% of the time, and our second requirement was for our drone to complete its races in an average time lower than 30 seconds. Both of these requirements were verified by looping through 100 simulations and recording how many of those races were finished, and what the average time across all those runs was. To improve upon this experiment in the future, we might attempt to implement an avoidance system in the controller so that our drone may be able to avoid collision with other drones, thus making it effective in group settings as well. Another way this project could be improved would be by optimizing the implementation of the controller itself so that we can decrease errors due to long run times.

Acknowledgments

We would like to thank Professor Bretl for giving us the base code for this very interesting simulation, as well as the example project which was immensely helpful as a reference on how to better organize our report, our code, and plots, and for all the help during office hours and through CampusWire.

References

Bretl, T., "AE353 (Spring, 2022) Code,"https://github.com/tbretl/ae353-sp22, 2022.

Bretl, T. *DP0: Control of a platform with a reaction wheel in gravity*, University of Illinois at Urbana-Champaign, 2022.

Appendix

Table 1 This is a log of work for Design Project 3.

Day	Task	Person or People
04/12	Linearized the dynamical and sensor model of the system as well as attempted to implement controller class	Omar
04/13	Wrote the Theory section	Pedro
04/14	Successfully implemented the controller class	Pedro
04/20	Wrote the abstract and introduction sections and wrote the requirements and verification section in experimental methods	Omar
04/21	Plotted the histograms for the report, improved Q and R matrices for a better performance	Pedro
04/21	Wrote the rest of the experimental methods as well as the results and discussion, and conclusion sections	Pedro
04/22	Wrote the controllablity and observability section of the Theory. Also added more to the diagnosis of failures section in experimental methods	Omar
04/26	Implemented drone collision avoidance in code	Pedro
04/28	Optimized K and L matrices' LQR	Pedro
05/03	Finished collecting data for plots for report, organized code	Pedro
05/03	Wrote script, recorded, and edited video	Omar and Pedro