DP4: Control of a Drone

Grace Heaton and Varsha Krishnakumar
University of Illinois at Urbana-Champaign, Champaign, IL, 61820

The objective of this exploration is to design, implement, test, and experiment with a controller that will enable a drone to complete a course containing hoops through which the drone will have to maneuver. This report will discuss the method of linearizing the equations of motion (EOMs) of the drone and deriving a stable controller. Next, simulations and statistical analyses will be conducted to test the validity of the designed controller. The report will set the requirement and verification to compare and analyze the final results. Finally, the results of the experiments will be noted and analyzed.

I. Nomenclature

p_x	=	x position (m)
p_y	=	y position (m)
p_z	=	z position (m)
ψ	=	yaw angle (rad)
θ	=	pitch angle(rad)
ϕ	=	roll angle (rad)
v_x	=	linear velocity along body-fixed x-axis (m/s)
v_y	=	linear velocity along body-fixed y-axis (m/s)
v_z	=	linear velocity along body-fixed z-axis (m/s)
w_x	=	angular velocity along body-fixed x-axis (rad/s)
w_y	=	angular velocity along body-fixed y-axis (rad/s)
w_z	=	angular velocity along body-fixed z-axis (rad/s)
τ_x	=	net torque about the body-fixed x-axis (N·m)
τ_y	=	net torque about the body-fixed y-axis (N·m)
τ_z	=	net torque about the body-fixed z-axis (N·m)
f_z	=	net force along the body-fixed z-axis (N)
x	=	state
u	=	input
A	=	state matrix
B	=	input matrix
K	=	optimal gain matrix
Q	=	weights of state variables
R	=	weights and input variables

II. Introduction

This paper will outline the theory, design, implementation, and experimentation of a drone controller. The aim of the controller is to enable a single drone to fly through a series of rings and avoid any possible collisions. These collisions could occur either with the rings or with other drones attempting to complete the course at the same time. Note that these competing drones are being operated independently. Controller design will utilize reference tracking and linear quadratic regulators to optimize the drone's speed and the controller's runtime. Experimental analysis will discuss drone performance and its ability to complete the course, with a focus on failure diagnostics.

III. Theory

The objective of this exploration is to make the drone possess the ability to guide itself through rings efficiently in order to complete the course. The motion of the drone's path is detailed by the state, x, while the input, u, details the torques and the net force produced by the four rotors on the drone.

A. Dynamic Model

1. Equations of Motion

The motion of the drone is detailed by ordinary differential equations of the form as shown:

$$f(p_x, p_y, p_z, \psi, \theta, \phi, v_x, v_y, v_z\omega_x, \omega_y, \omega_z, \tau_x, \tau_y, \tau_z, f_z) = \begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \\ \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \\ \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \\ \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} \tag{1}$$

Due to the complex nature of the differential equations of motion, the symbolic description of the above equations can be viewed in the code associated with this project*. From the function f, correspondingly, the state and the input of the system can be described as such:

$$x = \begin{bmatrix} p_x - p_{x_e} \\ p_y - p_{y_e} \\ p_z - p_{z_e} \\ \psi - \psi_e \\ \theta - \theta_e \\ \phi - \phi_e \\ v_x - v_{x_e} \\ v_y - v_{y_e} \\ v_z - v_{z_e} \\ w_x - w_{x_e} \\ w_y - w_{y_e} \\ w_z - w_{z_e} \end{bmatrix} \qquad u = \begin{bmatrix} \tau_x - \tau_{x_e} \\ \tau_y - \tau_{y_e} \\ \tau_z - \tau_{z_e} \\ f_z - f_{z_e} \end{bmatrix} \tag{2}$$

2. Equilibrium Points

1) The net force, f_z, was set to 4.905 $\frac{m}{s^2}$. As it is acting in the z direction, this force produces lift for the drone to counteract gravity in the same direction.
2) All the other equilibrium points indicated in Equation (2) were set to 0 in order to ensure that $f_{eq} = 0$ as it is a criteria that dictates if the set of equilibrium conditions entail a valid equilibrium point.

3. Linearization

The state, x, and the input, u, need to be linearized in order to ensure a valid controller and find the corresponding state and input matrices A and B. The system for this process shown in Equation (3). The Jacobian of the function f is then evaluated with respect to x and u at the equilibrium points detailed in Section 3.A.2 of this report to obtain the matrices A and B. The entries of these matrices can be viewed in the project code associated with this report.

$$\dot{x} = Ax + Bu \tag{3}$$

*https://github.com/varshakrishnakumar/AE-353-Design-Project-4

4. Controllability

In order to check for the controllability of the system, the controllability matrix, W was calculated. The procedure of evaluating this matrix is detailed in the project code associated with this report. W is reliant on the A and B matrices, and was verified to have a full rank of 12.

5. LQR: Controller

The way the LQR controller is tuned is by adjusting the diagonal entries in the Q and r matrices. These matrices are detailed below. Adjusting these relative to each other allows for different weights to be placed on controlling each state. The values for these matrices were picked through trial and error in accordance with an efficient system.

$$Q = diag(23, 15, 5, 10, 1.05, 1.05, 5.16487, 1, 1, 1, 1, 1) \qquad R = diag(200, 200, 200, 2) \tag{4}$$

The gain matrix for the controller system, K, can be evaluated using the relations $K = R^{-1}B^T P$, where P was solved for using a feature from the SciPy library in Python (*linalg.solve_continuous_are*()). This process is detailed in the project code, as well as the values present in the gain matrix K.

B. Sensor Model

1. Equations of Motion

The sensor system provides noisy measurements of the position of the drone (given by p_x, p_y, p_z), the yaw of the drone (given by ψ), the position of the next ring's center of which the drone has to maneuver through, as well as the position of the other drones in the course.[†] The equations of motion of the sensor system is shown below:

$$y = g(p_x, p_y, p_z, \psi) \tag{5}$$

2. Linearization

The sensor system detailed in Equation (4) can be linearized in relation to the state, x, using the C matrix.

$$y = Cx \tag{6}$$

This system follows a linear relation to the state, and therefore, does not require separate equilibrium conditions. Correspondingly, the C matrix has the shape 4x12 and is described as:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 & 0 & 0 \end{bmatrix} \tag{7}$$

3. Observability

The system's observability can now be verified with the C matrix. This can be done with the observability matrix W_o. In this case, $n = 12$, resulting in a 12x12 W_o matrix. The rank was calculated to be 12, which is equal to the horizontal shape of C.

4. LQR: Observer

Values for the cost matrices Q_o and R_o, which will have sizes 4x4 and 12x12 respectively, also need to be initialized. The matrices are detailed below.

$$Q_o = 500 I_4 \tag{8}$$

$$R_o = 0.015 I_{12} \tag{9}$$

The gain matrix for the observer system, L, can be evaluated using the feature from the SciPy library in Python (*linalg.solve_continuous_are*()), as previously carried out for the controller system. This process is detailed in the project code, as well as the values present in the gain matrix L.

[†] https://tbretl.github.io/ae353-sp22/projectsthe-system

C. Tracking

In this project model, the tracking method needs to be introduced. Since the objective of this project is for the drone to maneuver around the rings to complete the course, a *desired state*, x_{des} needs to be set in relation to the state estimate. This model can be detailed below:

$$u = -K(\hat{x} - x_{des}) \tag{10}$$

In this case, x_{des} refers to the center of the next ring on the course through which the drone has to maneuver through. $\hat{x}$ refers to the current state of the system. Primarily, this incentive will be used for position tracking, in order to ensure that the drone stays on track for the entirety of the course. The method for this tracking is shown below.

$$p_{des} = \hat{p} + r\frac{p_{ring} - \hat{p}}{||p_{ring} - \hat{p}||} \tag{11}$$

As noted in Equation (9), p_{des} refers to a position in the path between the initial position, $\hat{p}$, and the ending position, p_{ring}. On that path, r refers to the distance between p_{des} and $\hat{p}$. By tracking the position of the drone in this manner, the drone will be able to travel through the rings in an efficient way.

In addition to position tracking, in an attempt to lower the drone's completion time, velocity tracking was implemented. The method used was linear regression using gradient. descent[‡]. The method for this tracking is shown in the equation below.

$$w_{i+1} = w_i - \frac{\eta}{n}\hat{x}^T(\hat{x}w_i - x_{des}) \tag{12}$$

Here, the current measured state, $\hat{x}$ is treated as the input, while x_{des} is treated as the target (i.e. training "labels"). n refers to the number of iterations in the implemented code, η refers to the learning rate of the model, and i is the current iteration in $i \epsilon n$. Finally, w is the optimized goal velocity that the drone needs to achieve in the next step in the drone's path. This method was initially implemented in the code for the purpose of excelling in the race, but was excluded due to a higher risk of exceeding the required computation time of 0.01 seconds.

IV. Experimental Methods

A. Requirements

For this project, success will be determined quantitatively by setting requirements to ensure the stability of the drone.

1) Out of 1000 simulations conducted, the drone must be successful in completing the course for at least 80% of the simulations. Completing the course is defined as the drone flying through every ring with no collisions, and landing in the final ring.
2) The mean course completion time of these simulations must be at most 30 seconds.
3) Computations should take no longer than 0.01 seconds. If 10 or more computations in a single trial take longer than 0.01 seconds, the drone is not successful.

In addition to the performance requirements set, equilibrium points were set to optimize the model of the drone. As noted in equation (1) with stability as a priority, p_{x_e}, p_{y_e}, p_{z_e}, ψ_e, θ_e, ϕ_e, v_{x_e}, v_{y_e}, v_{z_e}, w_{x_e}, w_{y_e}, w_{z_e}, τ_{x_e}, τ_{y_e}, and τ_{z_e} shall be set to 0. This is based on the ideal state of the drone using minimal torque to stabilize itself over the duration of the race.

B. Verification

A Jupyter Notebook using a Pybullet environment was utilized as code interface for simulating and testing the Drone. The instructions for compilation of the project code is detailed in the GitHub repository associated with this report. * Instead of verifying random controller gain matrices (K), controller gain matrices will be found by simulating various Q_c and R_c matrices and their subsequent gain matrices and verifying that they fulfill the requirements. These Q_c and R_c matrices will be found by iterating through Q_c components based on the behavior (detailed in the Requirements subsection) desired. After obtaining a list of possible gain matrices, the right gain matrices can be found by testing to

[‡]https://courses.grainger.illinois.edu/ece449/sp2022/_site/

*https://github.com/varshakrishnakumar/AE-353-Design-Project-4

see if most of the simulations result in the drone achieving an average completion time of 30 seconds. The observer gain matrix L was kept constant for testing purposes. The implementation of this can be viewed in the code associated with the report. Finally, the controller design will be verified and evaluated by comparing actual and desired position of the drone (in accordance with the controller), and by comparing the actual and estimated position of the drone (in accordance with he observer design) by implementing a histogram detailing the error.

To verify that requirements 1 and 2 are met, the Pybullet simulation will be run 1000 times on randomized courses. The simulation reports whether or not the drone finished the course and the time at which the drone finished. The total number of runs in which the drone finishes will be counted. After all 1000 simulations are complete, the percentage of runs in which the drone finished will be calculated. If this is at least 80%, requirement 1 is met. All of the completion times will be averaged and if this number is 30 seconds or less, requirement 2 is met. To verify that requirement 3 is met, every implementation of the controller at each step will be timed. For an individual run, if there are fewer than 10 computations with runtimes exceeding 0.01 seconds, requirement 3 is met.

V. Results

After 1000 trials, the controller was deemed successful. All 3 requirements were verified to have been met. For 1000 trials, the completion rate was 87.7% with an average completion time of 17.32 seconds. Zero trials had 10 or more computations that exceeded 0.01 seconds.

A. Completion and Computation Results

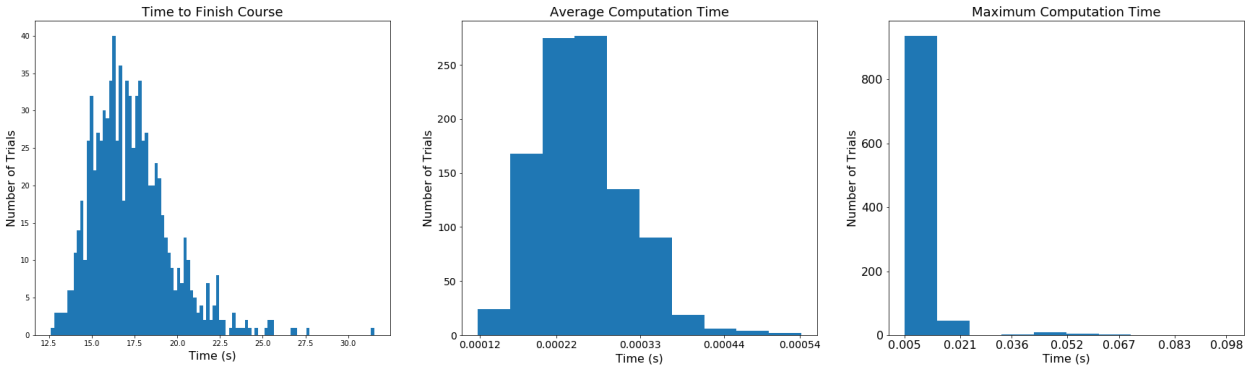

Fig. 1 Drone Completion and Computation Times

Figure 1 shows the frequency of completion times. The average completion time of 17.32 seconds and completion rate of 87.7% was achieved through iterative control design. Previous versions of the controller guided the drone slowly and clumsily with average completion times greater than 40 seconds and completion rates lower than 50%. To address this, reference tracking and collision avoidance were implemented within the controller. It was found that for desired positions further than 2.1 meters away, the drone was more likely to rotate about its axes in an undesirable way, resulting in large values of roll, pitch, and yaw. This caused an undesirable flight path and the drone was more likely to crash. By adjusting the next desired position for any rings further than 2.1 meters away to a closer location, the drone had increased stability and was able to fly along a steady path. The drone would also fail if it hit any object, such as the rings or other drones. Collision avoidance utilized the drones sensors of surrounding objects and drones. If the drone was within 0.41 meters of an object, it would correct to the opposite direction. This decreased the number of collisions which increased overall performance.

Figure 1 shows the average and maximum computation times for controller implementation. This version of the controller meets the computation time requirement as described above. However, previous iterations of the controller did not. The controller failed due to excessive computation times, which was found to be due to a robust velocity tracker. Unfortunately, the drawback of the velocity tracker was that it required many computations which increased overall runtime above the proposed limit.The velocity tracker was removed to address this failure. To compensate for the loss in performance that resulted from removing the velocity tracker, the linear quadratic regulator was further tuned to retain

speed and minimize error between actual and desired position.

B. Controller and Observer Performance

Both the controller and observer were implemented effectively. The purpose of the controller is to command torques so that the drone gets as close to its desired position as possible. The purpose of the observer is to get an accurate estimate of the drones location and orientation. The performance of the controller and observer for a single trial and aggregate performance are shown in Figure 2.

The controller consistently maintained an actual drone position within 2 meters of desired position in each direction. Previous iterations of the controller had higher error between actual and desired position, which resulted in the drone not flying through each ring, colliding with rings, or flying completely off course. To address this, reference tracking was implemented as described above. Additionally, it was found that differences in commanded and outputted torque resulted in variations in actual and desired position. Adjusting the linear quadratic regulator matrices helped to ensure that commanded torques were achievable within the actuator limits.

For the single run shown in Figure 2, estimated position was virtually the same as the actual position for the entire duration, which remained true for all trials. The histogram of RMSE for actual and estimated position is never more than 0.02, with an average around 0.01. There were very few issues regarding the observer and the observer was never determined to be a source of failure.

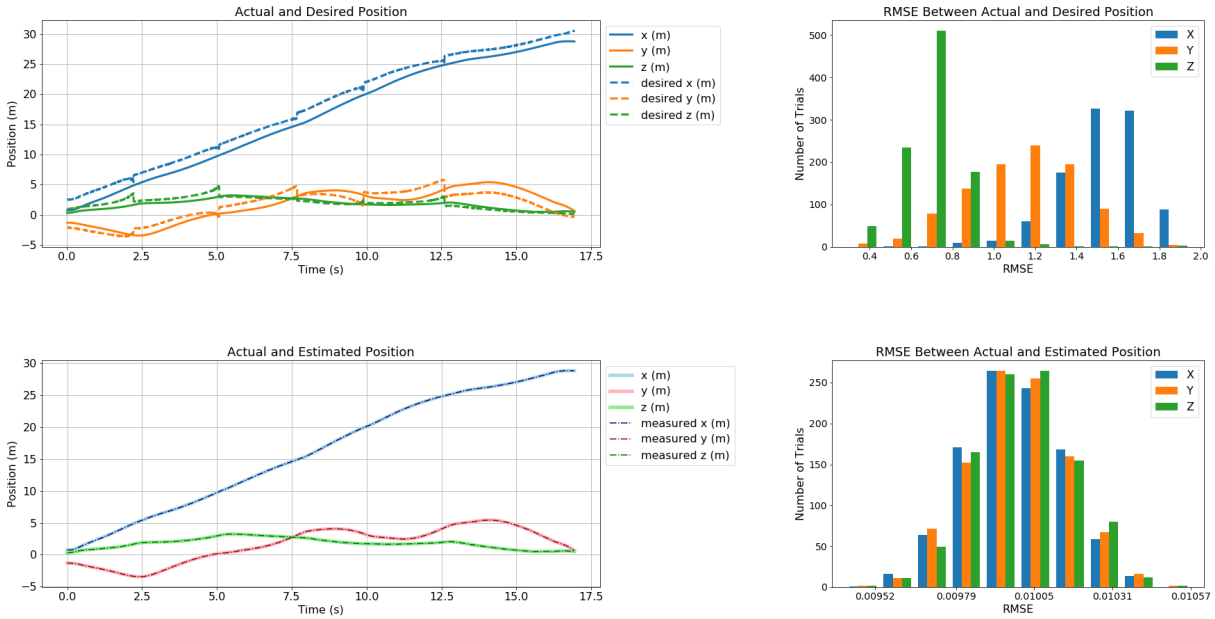

Fig. 2 Controller and Observer Performance

VI. Conclusion

In this project, the objective was to design, test, and implement a controller for a drone that receives noisy data from its camera. The drone was able to complete and maneuver around the course filled with rings, and additionally satisfy the requirements set in Section IV.A. While the completion rate was at a high 87.7%, it could have further been improved by fine tuning various test Q and R matrices in order to optimize the controller. Further, the average completion time could be maximized by optimizing the velocity tracking method using linear prediction with gradient descent to fit the run time criteria set in Section IV.A.

References

Bretl, T., "DroneDemo-Template," AE 353: Aerospace Control Systems Available: <https://github.com/tbretl/ae353-sp22/blob/main/projects/04_drone/DroneDemo-Template.ipynb>

Bretl, T., 2022. Reference. [online] AE 353: Aerospace Control Systems. Available at: <https://tbretl.github.io/ae353-sp22/reference>

"CS446/ECE449: Machine Learning (spring 2022)," CS446/ECE449 Spring 2022 | Machine Learning Available: https://courses.grainger.illinois.edu/ece449/sp2022/_site/.

Bretl, T., 2022. GitHub - tbretl/ae353-sp22: The website for AE 353 (Spring 2022) at the University of Illinois at Urbana-Champaign.. [online] GitHub. Available at: <https://github.com/tbretl/ae353-sp22>

Varsha Krishnakumar, "AE 353 Design Project 4," GitHub Available: https://github.com/varshakrishnakumar/AE-353-Design-Project-4/.

Appendix

Day	Task	Person or People
April 12	Calculated equilibrium point, state, input, ouput, and gain matrices, and started coding the basics of the controller	Grace
April 14	Drafted report of the Theory section	Varsha
April 22	Drafted report of the Experimental Methods section	Varsha
April 22	Drafted report of the Results section	Grace
May 1	Created new figures for aggregate data analysis	Grace
May 1	Implemented velocity and position tracking and collision avoidance in controller	Varsha
May 2	Edited lqr to improve performace	Varsha
May 2	Ran 1000 simulations for final data collection	Grace
May 3	Created video	Grace
May 3	Rewrote Results section	Grace
May 3	Proofread and Edits	Grace and Varsha