DP4: Cat-Pilots Race for Gold and Home-Baked Sweets

Myers, Andrew W.* and Brown, Luke C.†
Department of Aerospace Engineering, University of Illinois at Urbana-Champaign

This paper seeks to create a controller and an observer for a four-rotor drone as it flies through an obstacle course. The drone must reposition itself using data from observing the position of two of its rotors. The goal of the mission is to fly through a set of 14 rings as quickly as possible while avoiding collisions with other drones. Through repeated tests, our drone was able to complete the course in an average of xxxx seconds. Additionally, the fastest time that our drone completed the course was xxxx seconds.

I. Nomenclature

A, B	=	Matrices used to describe the state-space system model
C, D	=	Matrices used to describe the state-space observer model
$I_{i,j}, 0_{i,j}$	=	Identity and zero matrices of size (i, j)
K, L	=	Controller and observer gain matrices, respectively
Q_c, R_c	=	LQR weight matrices for the controller
Q_o, R_o	=	LQR weight matrices for the observer
f, g	=	Functions denoting the equations of motion and sensor models, respectively
f_z	=	Thrust force applied to the drone (N)
m, n, o	=	State, input, and sensor measurement vectors, respectively
p_x, p_y, p_z	=	Components of the position of the drone (m)
t	=	Variable denoting the measurement of time (s)
v_x, v_y, v_z	=	Components of the velocity of the drone (m/s)
τ_x, τ_y, τ_z	=	Components of the applied torque to the drone (N $\cdot$ m)
ψ, θ, ϕ	=	Yaw, pitch, and roll angles of the spacecraft (rad)
$\omega_x, \omega_y, \omega_z$	=	Components of the angular velocity of the drone (rad/s)

II. Introduction

The drone described in this paper is tasked with flying through a course of hoops in a predetermined order and direction. The drone is in a quadrotor configuration, in which it has four propellers evenly spaced in a square pattern. A cat pilot is on board the drone, intending to race other drones through the course while avoiding collisions with other drones and the loops themselves.

The drone is tracked by a system that allows for measurement of two position markers on the drone. Our observer is designed to estimate the position of the center of the drone. We then track the position values and use our controller to have the drone follow a path through the course by minimizing the state error.

Our drone and our report will be considered successful if they meet the following criteria. First, the drone must make it through the entire course while avoiding potential collisions with other drones, the hoops, and the ground. Second, we must outline how well our observer and controller are working with a sample size of at least 100 trials. Third, we must show how fast our drone completes the race and how fast the controller runs with a sample size of at least 100 trials.

*AE 353, Spring 2025, Junior in Aerospace Engineering, Department of Aerospace Engineering, University of Illinois at Urbana-Champaign.
†AE 353, Spring 2025, Junior in Aerospace Engineering, Department of Aerospace Engineering, University of Illinois at Urbana-Champaign.

III. Theory

A. Equations of Motion

We are tasked with finding a way to apply three body torques and a thrust force to a drone to control 12 state variables, represented as

$$m = \begin{bmatrix} p_x & p_y & p_z & \psi & \theta & \phi & v_x & v_y & v_z & \omega_x & \omega_y & \omega_z \end{bmatrix}^T \quad \text{and} \quad n = \begin{bmatrix} \tau_x & \tau_y & \tau_z & f_z \end{bmatrix}^T. \tag{1}$$

Each variable is governed by the nonlinear equations of motion [1] of the following form:

$$\dot{m} = f\left(p_x, p_y, p_z, \psi, \theta, \phi, v_x, v_y, v_z, \omega_x, \omega_y, \omega_z, \tau_x, \tau_y, \tau_z, f_z\right). \tag{2}$$

The only external force acting on the drone is gravity, but the rotors can generate a body force and a torque that modify the drone's motion.

B. State-Space Model and Controllability

To begin making a linear-state feedback controller, a state-space model must be obtained by linearizing about an equilibrium point. That is, an equilibrium point must be obtained to satisfy $\dot{m} = 0 = f(m_e, n_e)$. A valid selection of m_e and n_e that fulfills this requirement is

$$m_e = \begin{bmatrix} 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \quad \text{and} \quad n_e = \begin{bmatrix} 0 & 0 & 0 & 4.905 \end{bmatrix}^T. \tag{3}$$

Using this equilibrium point, the drone will find equilibrium by hovering at $p_z = 4$m, which is the average height of all of the rings. It is now possible to derive the A and B matrices [2], by computing

$$A = \left.\frac{\partial f}{\partial m}\right|_{(m_e, n_e)} \quad \text{and} \quad B = \left.\frac{\partial f}{\partial n}\right|_{(m_e, n_e)}. \tag{4}$$

These matrices are evaluated in the project code [3]. Additionally, the controllability matrix is computed in the project code [3] and has a full rank of 12, indicating that the linearized system is controllable.

C. Observer Dynamics and Observability

Sensors provide a noisy measurement of the position in space of two markers. The first marker is at the center of the left rotor, and the second is at the center of the right rotor. From the observed locations of these two markers, we obtain a state estimate for our drone.

With the sensor model defined, it must be translated into state-space form with linearization. This process demands knowledge of what the sensors should read when the system is at equilibrium. That is, the output is defined as

$$y = o - g(m_e, n_e), \tag{5}$$

where $o = g(m, n)$ calculates the estimated x, y, and z positions of the left and right rotors.

We can now compute the C and D matrices using the equations [2]

$$C = \left.\frac{\partial g}{\partial m}\right|_{(m_e, n_e)} \quad \text{and} \quad D = \left.\frac{\partial g}{\partial n}\right|_{(m_e, n_e)}, \tag{6}$$

which are showcased in the project code [3]. The observability matrix is also computed in the project code [3] and has a full rank of 12, indicating that the state-space model is observable.

D. Linear State Feedback and Observer Design

The LQR method was used to derive values for the K and L matrices used in an observer system model [4]. For the gain matrix K, the Q_c and R_c matrices are selected to be

$$Q_c = \text{diag}[20, 20, 100, 10, 10, 10, 1, 1, 1, 0.1, 0.1, 0.1] \qquad \text{and} \qquad R_c = \text{diag}[1000, 1000, 1000, 500]. \tag{7}$$

These values for Q_c and R_c allow the drone to enter a stable hover, ready to start the race. For the observer gain matrix L, the Q_o and R_o matrices evaluate to be

$$Q_o = 10I_{6,6} \qquad \text{and} \qquad R_o = 100I_{12,12}. \tag{8}$$

The K and L matrices are numerically evaluated and displayed in the project code [3] using the values determined in Section V.

E. Closed-Loop

All three of the matrices necessary to determine the stability of the closed-loop system are computed in the project code [3]. The eigenvalues of the $A - BK$ matrix all contain negative real quantities, denoting its asymptotic stability. Furthermore, the eigenvalues of the $A - LC$ matrix are also all negative, demonstrating observer asymptotic stability. The quantities of these eigenvalues are specified in the theory section of the project code [3]. Finally, the blocked system stability matrix is calculated in the project code [3], indicating that the full system is asymptotically stable about the selected equilibrium point.

F. Tracking

To guide the drone along the race, the drone must be able to track the center of the upcoming hoop. The variables p_x, p_y, and p_z can be tracked by our system, as they are not present in the equations of motion of the drone. Therefore, by changing the desired position of the drone to the center of the next hoop, we can travel from hoop to hoop and complete the race.

G. Path-finding

To ensure that the drone does not collide with other drones or the sides of the hoop, the drone must find a suitable path to travel along. This section, along with the tracking section, will be further explored next week after Dr. Bretl goes over these topics more in depth in lecture.

IV. Experimental Methods

A. Tuning K and L

In order to tune our K and L values, we will adjust our Q_c, R_c, Q_o, and R_o matrices according to observed drone behavior. For example, if we notice the drone flipping over often, we will increase the first three values in R_c to produce smaller output torques.

B. Verifying Proper Controller and Observer Implementation

To verify that the controller works as intended, we will track the error between the actual state and the desired state throughout an entire simulation. If the error remains small, defined as being less than 5% for the majority of time steps, then the controller will be considered successfully implemented.

Similarly, to verify that the observer works as intended, we will track the error between the state estimate provided by the position sensors and the actual state throughout an entire simulation. The same 5% error benchmark will be used to define a successful observer.

C. Tuning the Potential Function

To tune the coefficients in our potential function, in which we obtain the desired state, we will iterate through k_{att} values. For each value, we will check average time and success rate on the same seed. These will serve as valuable data points to help us choose our values to fit our desired characteristics.

D. Testing Success for the Best Model

Using all of the data we have already collected in this section, we will choose the model with the highest success rates to further test and enter into the competition. We will track the success rate, average time, and fastest completion time for 500 simulations with the highest performing controller and observer system. Each simulation will have randomized

initial conditions as defined by the drone code while having randomized ring heights, chosen at random from a list of possible heights of $p_z = 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6$ meters.

V. Results and Discussion

A. Tuned K and L Results

From print statements and visual observation, we realized that our original values for R_c were two low. Our controller was producing very high output torques at the onset of every simulation, and thus was very unstable and would always flip over. From this observation, we decided to increase our R_c values to use $R_c = [800.0, 800.0, 800.0, 500.0]$ to produce smaller output torques and therefore a more stable controller.

B. Verified Controller and Observer Implementation

First, we need to evaluate our controller and observer design. To do this, we look at histograms of error between actual state and desired state for our controller, and estimated state and actual state for our observer. The histograms are shown below for data from 100 simulations in Fig. (1), and Fig. (2) respectively. Fig. (1) shows that the majority of error between the actual and the desired state accumulated at each time step for 100 trials falls below 2%. Similarly, we see in Fig. (2) that the majority of the error between the estimated and actual state for every timestep also falls under 2%. Given that our controller and observer both function to minimize these errors respectively, this data shows that each is functioning properly and keeping error under two percent in most cases.

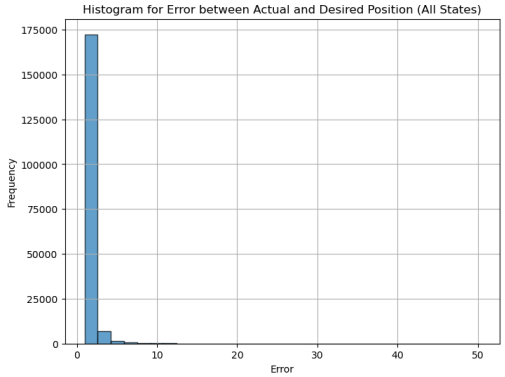

Fig. 1 This figure shows a histogram of error between the actual position and desired position for 100 simulations with fixed rings.

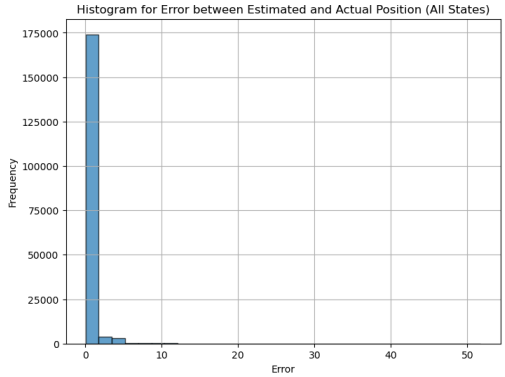

Fig. 2 This figure shows a histogram of error between the estimated position and actual position for 100 simulations with fixed rings.

C. Potential Function Success Metrics

With k_{rep} fixed at a value of 150, we systematically increased k_{att} by 10 for our verified controller and observer to find the value of k_{att} that would yield the highest success rate for our drone. With success being defined as fully completing the race and landing back at the starting point, we tested our drone with randomized initial conditions given by the drone code.

Table (1) demonstrates the success rate and the average time to complete the course of our drone, keeping the heights of the rings fixed in the first set of values provided by our seed. This method successfully finds the values of k_{att} that allow for high success rates from any set of initial conditions. The values $k_{att} = 210, 220, 230,$ and 240 all performed exceptionally well, completing the course 100% of the time. The difference in average completion times for these values was less than a second, proving that all of these values are viable options. However, despite not completing the track for all 100 initial conditions, $k_{att} = 250$ had the fastest average completion time. Figure (3) plots all 100 trajectories for this value, demonstrating the drone's high success rate while also documenting the two instances of failure.

Table (2) demonstrates the success rate and the average time to complete the course of our drone as well. However, for these trials, the ring heights were changed every simulation according to the first 100 values of our seed. This table showcases the drone's ability to adapt to different course layouts. Our controller and observer performed worse in

these trials, with the greatest success coming from $k_{att} = 240$ with a success rate of 51% and an average time of 92.44 seconds. The trajectories from these simulations is showcased in Fig (4), where it is evident that a majority of failures arise from the set of back and forth rings in the center of the track.

Table 1 Success count for varying k_{att} with $k_{rep} = 150$ for 100 simulations with fixed rings

k_{att}	180	190	200	210	220	230	240	250	260	270
Success Count	37	51	82	100	100	100	100	98	95	91
Average Time (s)	84.82	84.98	83.59	83.31	83.34	82.57	82.38	81.99	82.02	82.14

Table 2 Success count for varying k_{att} with $k_{rep} = 150$ for 100 simulations with variable rings

k_{att}	180	190	200	210	220	230	240	250	260	270
Success Count	31	32	35	37	44	35	51	36	34	39
Average Time (s)	115.14	104.95	97.23	101.84	95.76	92.53	92.44	91.60	92.75	95.34

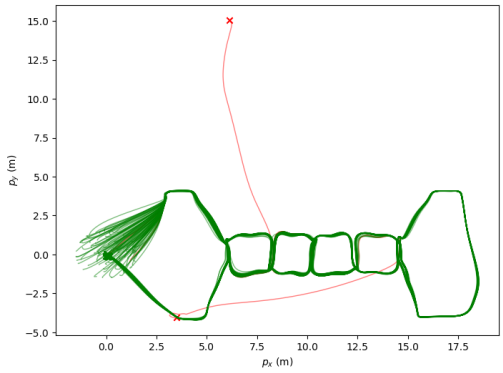

Fig. 3 Drone trajectories with green lines represent successful missions while red lines represent unsuccessful missions for 100 simulations with fixed rings.

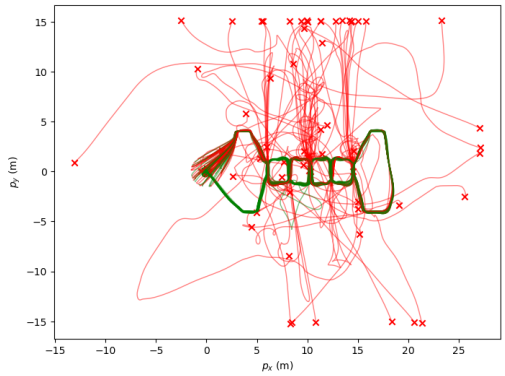

Fig. 4 Drone trajectories with green lines represent successful missions while red lines represent unsuccessful missions for 100 simulations with variable rings.

D. Best Model Success

Given that $k_{att} = 240$ performed very well for both fixed and variable ring heights, we decided to pursue this value for our final design. Performing the same tests as before but with a larger sample size, we found that our drone had a 41.4% success rate with an average completion time of 93.02 seconds. The fastest run our drone had was 77.08 seconds, which has become a target for many others seeking gold in the upcoming race as noted in the appendix.

VI. Conclusion

Appendix

Some of our brave cat-pilots have reached out to share their praise for our design. Meowx Verstappen has been quoted as saying:

> *"If you thought racing cars was difficult, you should try piloting a drone. If I didn't have Luke and Andrew's design to help me out, I would've flown right into the side of the rings!"*
>
> – Meowx Verstappen

A cat-pilot in training has reached out as well:

"Meowx's record of 77.08 seconds will not stand once I get to race with the best of them. I'm going to push their design to the extreme, no more error limiting when I'm in control!"

– Tail Furnhardt

While we do not advise Mr. Furnhardt to adjust our well tested design, we wish him and all other cat-pilots luck when racing using our controller and observer system.

Acknowledgments

Authors Luke Brown and Andrew Myers thank Dr. Timothy Bretl for his excellent instruction throughout the entire semester. Without his instruction and GitHub containing course notes, examples, and the basis of the simulation code for this project, this project would not have been possible to execute. We would also like to thank the teaching staff for their continued feedback on our reports, helping us to become better technical writers.

References

[1] Bretl, T., "Design Project 4 (racing drone)," https://tbretl.github.io/ae353-sp25/projects/04-drone.html, April 2025.

[2] Bretl, T., "State Estimation Reference Page," https://tbretl.github.io/ae353-sp25/reference/state-estimation.html, April 2025.

[3] Brown, L., and Myers, A., "DP4 Google Colab Code," https://colab.research.google.com/drive/1ptkvjp0LXlxvjdcoT6_YPu2EI0D9iNFV?usp=sharing, April 2025.

[4] Bretl, T., "Optimal Observers Reference Page," https://tbretl.github.io/ae353-sp25/reference/optimal-observers.html, April 2025.