DP4: Drone Race

Emily Lory* and Conan Zhang†
University of Illinois at Urbana-Champaign

This report details the design of a control system that navigates a quadrotor through an obstacle course through implemented target tracking and collision avoidance. The theory behind the model is described, a state-space model is defined, and the system requirements, with their verification methods, are specified. To ensure that an optimal controller is built, a closed-loop and asymptotically state feedback system is linearized between the system and the drone. The drone simulation behavior is then analyzed through experimentation and post-processing analysis. Failures are identified, diagnosed, and eliminated to enable the drone to pass through all seven gates while achieving the fastest race time.

I. Nomenclature

A, B	=	Coefficient matrices describing the state space model
C, D	=	Coefficient matrices describing the observer
d_i	=	Scalar distance between drone estimated position and *i*th gate (m)
e_{max}	=	Error norm value between drone desired and estimated position (m)
f	=	Controller equations of motion
f_z	=	Net force along the body-fixed z axis (N)
g	=	Sensor model
$h_{attract}, h_{repel}$	=	Attractive and repulsive force utilized for gradient descent
K	=	Controller gain matrix
k_{des}	=	Adjust step size of gradient descent
L	=	Observer gain matrix
m, m_e	=	Original and equilibrium state variables
m_i	=	Marker coordinates in the x, y, and z
n, n_e	=	Original and equilibrium control inputs
p_x, p_y, p_z	=	Position in the x, y,and z (m)
$\dot{p_x}, \dot{p_y}, \dot{p_z}$	=	Velocity in the x,y,and z (m/s)
$\hat{p}, p_{des}$	=	Estimated and desired positions in x, y, and z (m)
ϕ	=	Roll angle(rad)
ψ	=	Yaw angle (rad)
Q_c, R_c	=	Matrices for controller LQR
Q_o, R_o	=	Matrices for observer LQR
τ_x, τ_y, τ_z	=	Net torque about the body-fixes x,y,and z-axis (Nm)
θ	=	Pitch angle (rad)
u_{des}	=	Desired output of the system
v_x, v_y, v_z	=	Linear velocity along the body-fixed x,y,and z-axis (m/s)
$\dot{v_x}, \dot{v_y}, \dot{v_z}$	=	Linear acceleration along the body-fixed x,y, and z-axis (m/s^2)
w_x, w_y, w_z	=	Angular velocity about body-fixed axes (rad/s)
$\dot{w_x}, \dot{w_y}, \dot{w_z}$	=	Angular acceleration about body fixed axes (rad/s^2)
W_c	=	Controllability matrix
W_o	=	Observability matrix
x_{des}	=	Desired state of the system
y	=	Output of the sensor model

*Undergraduate Student, Aerospace Engineering at University of Illinois at Urbana-Champaign
†Undergraduate Student, Aerospace Engineering at University of Illinois at Urbana-Champaign

II. Introduction

Quadrotor drones are pivotal for the future of aerospace technology due to their versatility and ubiquitous nature in today's industries. These unmanned air vehicles hold essential roles in both commercial sectors and government operations due to their autonomous nature and human-centered design. For example, Autel Robotics' EVO Max drone series used in aerial photography and farm surveillance, enables scene reconstruction and agricultural land surveying in hazardous environments [1]. On the other hand, Skydio's X2D drone, utilized in the US Army's Short Range Reconnaissance program, enables a broad range of stealth and recon missions [2]. Given the importance that quadrotor drones like the EVO Max and X2E play in both commercial and military contexts, it is becoming increasingly crucial to enhance their operational abilities through precise navigation and robust control systems. Recognizing these specializations, drone racing has emerged to the forefront of pushing the boundaries of high-speed and agile flight.

This report details the design and implementation of an advanced control system for a quadrotor drone with four rotors powered by an electric motor tasked with a challenging scenario—an obstacle course containing 7 hoops of varying heights. The drone produces three net torques, (τ_x, τ_y, τ_z), and a net force, (f_z). Sensors on the drone provide estimated position measurements, (p_x, p_y, p_z), for stable movement through each hoop. The drone is intended to complete the obstacle course as fast as possible while avoiding collisions with the hoops themselves and other competing drones. In the report, a controller and observer are designed and implemented to accomplish the goal of having the shortest race time without a collision. To do this, trajectory tracking will be implemented to ensure that the drone can navigate through the rings of the course. Next, equilibrium points will be identified and the equations of motion will be linearized about them. Then the state space model and the observer will be defined while also confirming controllability and observability. Finally, the controller will be rigorously tested to find the drone's performance limits, eliminating any areas of failure toward completing the obstacle course.

III. Theory

Before any experimentation or simulations can be conducted, the equations necessary for the dynamic and sensor model must be theoretically defined. Utilizing governing equations of dynamics including Coordinate Transformations, Newton's Laws, and Euler's Equations, the two models can be derived, and the linearization process can then begin.

A. Controller Design

To build an optimal controller, the dynamical system of equations for the drone are placed into a linear state-space model consisting of ordinary differential equations (ODEs). Through linearization, the complex equations of motion can be simplified, enabling a more thorough analysis of the system's behavior under varying conditions. The result is a state-space model simplifying the original complex system.

$$\dot{x} = Ax + Bu \tag{1}$$

For the state space model, the column vectors m and n represent the states and inputs

$$m = \begin{bmatrix} p_x & p_y & p_z & v_x & v_y & v_z & \phi & \theta & \psi & \omega_x & \omega_y & \omega_z \end{bmatrix}^T \qquad n = \begin{bmatrix} \tau_x & \tau_y & \tau_z & f_z \end{bmatrix}^T \tag{2}$$

where the state and input can thus be defined respectively as:

$$x = m - m_e \qquad u = n - n_e \tag{3}$$

Given the equations of motion in the function $f(m, n)$ from the dynamic model the controller can now be designed. Equilibrium points are first defined and used to linearize the system where $f(m_e, n_e) = 0$. The chosen equilibrium points for the dynamic model were all set to a value of zero aside from $f_{ze} = 4.905$. The equilibrium force along the drone-fixed z-axis of 4.905 Newtons signifies the force output needed by the rotors to counter the force of gravity. Values of zero for all state and input variables except for this net force were chosen to stabilize the system around an airborne drone while keeping the controller controllable. The next step is to compute the Jacobian of the dynamic model with respect to the state variables and control inputs. The coefficient matrices A and B are defined as:

$$A = \frac{\partial f}{\partial m}|_{(m_e, n_e)} = (12\ x\ 12\ matrix) \qquad B = \frac{\partial f}{\partial n}|_{(m_e, n_e)} = (12\ x\ 4\ matrix) \tag{4}$$

The result is a linear approximation of the right-hand side of the first-order differential equations. Plugging in the equilibrium points found earlier, a linearized state-space model is derived where A and B represent the coefficient matrices of the controller state-space model.

B. Observer Design

Due to the separation principle, the observer can be defined and confirmed separately from the controller. If both the controller and the observer are defined and confirmed separately then they are also valid when combined. The state-space model for an observer can thus be defined as:

$$y = Cx + Du \tag{5}$$

Using the sensor model equations derived, the linear output can be defined as y where m_i representing marker coordinates is a function of specific states and inputs. Here, the sensor provides an estimated position in space of four markers, one above each rotor. With this position, the drone can thus orient itself correctly to pass through each ring.

$$y = m_i - g(p_x e, p_y e, p_z e, \phi_e, \theta_e, \psi_e) \qquad m_i = g(p_x, p_y, p_z, \phi, \theta, \psi) \qquad \begin{bmatrix} m_{ix} \\ m_{iy} \\ m_{iz} \end{bmatrix} = g(p_x, p_y, p_z, \phi, \theta, \psi) \tag{6}$$

The equations for computing the C and D matrices can then be derived by taking the Jacobian concerning the sensor model:

$$C = \frac{\partial g}{\partial m}|_{(m_e, n_e)} = (12\ x\ 12\ matrix) \qquad D = \frac{\partial g}{\partial n}|_{(m_e, n_e)} = (12\ x\ 4\ matrix) \tag{7}$$

Because the sensor model is independent of the control inputs (net torques and net force along the body fixed axes), the resulting D matrix is entirely composed of values of zero. For this reason, the state space model can be rewritten as:

$$y = Cx \tag{8}$$

C. Controllability and Observability

An ideal control system should be controllable and observable meaning the controller should be able to achieve desired states through the control inputs and should be able to determine an estimated state through its built-in observer. Thus, when computing the controllability matrix for the state-space models, if the rank of the matrices equates to the number of states, then the system if both controllable and observable. To test for controllability for the controller, the controllability matrix W_c can be derived from A and B. W_c is a $n \times (n \times m)$ matrix where n is the number of columns of A and m is the number of columns of B. Each column of W_c is obtained through matrix multiplication $A^i \times B$ where i is the order of the columns between 0 and n-1. The controllability matrix is defined as:

$$W_c = [B\ AB\ \ldots\ A^{n-1}B] = (12\ x\ 48\ matrix) \tag{9}$$

Similarly for observability for the observer, observability matrix W_o can be derived from A and C. W_o is a $(n \times m) \times n$ matrix where n is the number of rows of A and m is the number of rows of C. Each row of W_o is obtained through matrix multiplication $C \times A^i$ where i is the order of the columns between 0 and n-1. The observability matrix is defined as:

$$W_o = [C^T\ A^T C^T\ \ldots\ A^{T^{n-1}} C^T]^T = (144\ x\ 12\ matrix) \tag{10}$$

The rank of the controllability matrix and observability matrix are both equated to be 12 which equals the number of states in the system. The system is thus controllable and observable.

D. LQR Method

The Linear Quadratic Regulator (LQR) method is applied to optimize the controller and observer design. This method balances effort and error with weighted matrices Q and R. Q is a symmetric matrix where the diagonal elements correspond to the system's state variables m. R is also a symmetric matrix where the diagonal elements correspond to

each control input n. The magnitude of the diagonal element indicates which variable yields greater importance when selecting a gains matrix. The LQR method allows for prioritizing certain weights of specific variables in the state-space models, while also holding the system to eigenvalues with negative real parts. Negative real part eigenvalues suggest that the system will be asymptotically stable returning to equilibrium.

$$\begin{aligned} \underset{u_{[t_0,\infty)}}{\text{minimize}} \quad & \int_{t_0}^{\infty} \Big(x(t)^T Q x(t) + u(t)^T R u(t) \Big)\, dt \\ \text{subject to} \quad & \dot{x}(t) = Ax(t) + Bu(t), \quad x(t_0) = x_0 \end{aligned}$$

The problem above illustrates the infinite-horizon time-invariant LQR problem. Solving the continuous-time algebraic Riccati equation for P and the gain matrix equation yields a gains matrix K that minimizes the cost function and achieves desirable controller conditions.

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \qquad K = R_c^{-1} B^T P_c \tag{11}$$

For the linear state feedback controller, state feedback is defined below.

$$u = u_{des} - K(\hat{x} - x_{des}) \tag{12}$$

The weights of Q_c were chosen to prioritize aligning the drone's orientation and angular velocity with equilibrium. The weights of R_c were chosen to ensure that the rotors could initiate enough torque control on the drone orientation without being too aggressive.

$$Q_{\mathrm{c}} = \mathrm{diag}\,(q_{1c}, q_{2c}, ..., q_{12c}) \qquad R_{\mathrm{c}} = \mathrm{diag}\,(r_{1c}, r_{2c}, r_{3c}, r_{4c}, r_{5c}, r_{6c}) \tag{13}$$

Implementing the weights into the LQR function yielded a gains matrix K for the controller design:

$$K = \Big[4 \times 12 \Big] \tag{14}$$

For an observer, state-space form can be represented below as

$$\dot{\hat{x}} = A\hat{x} + Bu - L(C\hat{x} - y) \qquad L = P_o C^T Q_o \tag{15}$$

where L represents the observer gains. Similar to finding K for the controller, an analogous LQR method can be implemented to find L. The weights of Qo were chosen to minimize the uncertainties/disturbances in the dynamics of the system. The weights of Ro were chosen to minimize the measurement noise in the sensor measurements.

$$Q_{\mathrm{o}} = \mathrm{diag}\,(q_{1o}, q_{2o}, ..., q_{12o}) \qquad R_{\mathrm{o}} = \mathrm{diag}\,(r_{1o}, r_{2o}, ..., r_{12o}) \tag{16}$$

Implementing the weights into the LQR function yielded a gains matrix L for the observer design:

$$L = \Big[12 \times 12 \Big] \tag{17}$$

Testing the associated matrices $A - BK$ and $A - LC$, the eigenvalues all result in negative real parts which suggest that the gains derived theoretically produce an asymptotically stable closed loop system separately for the controller and the observer. Thus, due to the separation principle, the combined observer and feedback system is stable.

E. Collision Avoidance and Tracking

To enable the drone to precisely navigate through each ring, a tracker must be implemented within the observer. The observer equation as seen in Eq. 15 solves for the time derivative of the state estimate $\dot{\hat{x}}$ in terms of the state estimate $\hat{x}$. The state-feedback equation in Eq. 12 implements desired state feedback x_{des} and desired state u_{des} into the system where each desired position corresponds to the center of a hoop in the obstacle course. The variables are defined below

$$x_{des} = \bar{m}_e - n_e \qquad u_{des} = \bar{n}_e - n_e \tag{18}$$

where $\bar{m}_e$ describes the new equilibrium point for the state of the system and $\bar{n}_e$ describes the new equilibrium point for the control inputs. Here, $\bar{m}_e$ will be set to the position coordinates of the next hoop with all other variables remaining unchanged to maintain drone orientation and stability. $\bar{n}_e$ will remain equal to the initial equilibrium control inputs.

For the tracker, the gradient descent optimization algorithm is used to implement collision avoidance with hoops and competing drones in the race while navigating through the course.

$$p_{des} = \hat{p} - k_{\text{des}} \nabla h(\hat{p}) \qquad \nabla h(\hat{p}) = \nabla h_{\text{attract}}(\hat{p}) + \nabla h_{\text{repel}}(\hat{p}) \tag{19}$$

In Eq. 19, p_{des} corresponds to the x, y, z position of the hoop which is defined within the first three rows of x_{des}. This process of finding the desired position involves calculating a potential field based on the drone's current position, target position, and positions of other drones. The algorithm then updates the drone's desired position by moving against the gradient of this field.

$$\nabla h_{\text{attract}}(\hat{p}) = k_{\text{attract}} \frac{\hat{p} - p_{\text{des}}}{\|\hat{p} - p_{\text{des}}\|} \qquad \nabla h_{\text{repel}}(\hat{p}) = -k_{\text{repel}} \sum_i \left(\frac{1}{d_i(\hat{p})^2} \right) \nabla d_i(\hat{p}) \tag{20}$$

Attractive force h_{attract} is designed to pull the drone towards the goal, while repel force h_{repel} is designed to push the drone away from obstacles. The result when integrated into the state feedback equation is control input u, designed to minimize the error to this desired state.

IV. Experimental Methods

A. Requirements

Through mathematical analysis, it was established that the controller and observer were both controllable and observable respectively. To evaluate the impact of the created controller, five requirements must be met by five verifications. The following section details the requirements, parameters, verifications, and trial procedures for the experiment.

- The drone shall complete 100 out of 100 simulations without a collision with the obstacle course.
- The drone shall have an average race time of 30 seconds or less after 100 simulations.
- The drone shall have an average root mean square error of 0.1 meters or less between the estimated position and actual position of the drone after 100 simulations.
- The drone shall have an average root mean square error of 0.5 meters or less between the actual position and the desired position of the drone after 100 simulations.
- The drone shall complete 100 out of 100 simulations without a collision with another drone.

B. Initial Conditions

All simulations were conducted with the drone initially at rest and under Earth's gravity. The maximum simulation time was set to 30 seconds, and if the race time exceeded the maximum simulation time, the simulation ended and concluded the race. The sensor noise was set to 0.01 meters. For all of the requirements but the fifth, it was assumed that the simulation was conducted with only one drone in the race. The success of a singular drone was prioritized before pursuing a controller that factored in multiple drones.

C. Verification

The verification of the requirements stated above ensures that the quadrotor is meeting the needs of stakeholder drone company *Tiny Whoop* and its consumers. Below is the analytical description of how each requirement will be verified.

- To verify the first requirement, 100 simulations were ran through the Conda interface where the number of complete and incomplete races were recorded. Once collided with a part of the obstacle course, the quadrotor would either deflect towards the ground or off course. This essentially resulted in an incomplete race. Thus, the desired number of complete races was set to 100 to ensure that under no condition would the quadrotor collide with any part of the obstacle course. It was assumed that the only collisions occurring during these simulations were between the quadrotor and the obstacle course, as there was only one quadrotor participating in this set of 100 simulations.
- To verify the second requirement, the last time step recorded in each of the 100 simulations were displayed in a histogram and averaged to find the completion time of each race. Choosing the average simulation time to be a limit of 30 seconds ensures that the drone can navigate through the course in an effective amount of time, while

achieving stability in the air. A limit of 30 seconds also constrains the control system to desirably achieve a quick race-time, an indication of a well-designed controller.

- To verify the third requirement, the 100 simulations previously used will be observed by the built-in observer in the controller generating a history of the estimated state values $\hat{x}$ along with the actual state x. The estimated states will then be compared to the actual state containing the true position of the drone p_x, p_y, and p_z. After gathering all position data, these quantities will then be calculated for their RMSE values, plotted into a histogram, and averaged. A visual and mathematical observation will then be made to analyze if the average RMSE value is less than 0.1 meters. Additionally, a visual comparison plot will be made with the last simulation to compare how the state estimate vs. actual state varies over time. This way the histogram graphs can also be visually verified through a sample trial to pinpoint how accurate the sensor is in its recordings.
- To verify the fourth requirement, the actual state for each position p_x, p_y, p_z will also be compared to p_{des}. Similar to the third requirement, after gathering all position data, these quantities will then be calculated for their RMSE values, plotted into a histogram, and averaged. A visual and mathematical observation will then be made to analyze if the average RMSE value is less than 0.5 meters. Another visual comparison plot will be made with the last simulation compare how the actual state vs. desired state vary over time. This way, the histogram graphs can also be visually verified through a sample trial to pinpoint how accurate the controller is in navigating towards each gate while stabilizing the drone. Through analysis, RMSE was chosen as an indicative measurement of success as it consolidates error magnitudes across various data points into a single metric of predictive accuracy according to the specified formula:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \hat{x}_i)^2}{n}} \tag{21}$$

  where x_i denotes the positions of the drone p_x, p_y, p_z for both p_{des} and p_{actual}. Moreover, to facilitate the assessment of the controller-observer system's effectiveness, the mean of the distribution will be marked in each histogram.
- To verify the fifth requirement, a new set of 100 simulations will be conducted with another controller in parallel to simulate race conditions. The simulation was performed with another controller designed to only navigate through the first two gates. With this limited drone, the race competition could be simulated to a controlled, competitive environment. This setup also helps mimic real-world scenarios where drones may face early obstructions but need to continue to navigate their environment effectively. To quantify the number of drone on drone collisions, the number of completions and failures of the original drone were recorded and analyzed, where completions represented the number of times the original drone was able to make it through the entire obstacle course.

D. Trials

The drone's behavior was tested using the PyBullet physics engine in the Meshcat visualization engine through the Jupyter Notebook environment. The simulation generated the drone fitted with four rotors and an obstacle course containing 7 hoops in a spiral-like pattern. The drone operates under six degrees of freedom through directional torques and vertical force control inputs. Keyboard commands, visualizations, and animations were turned off to speed up each simulation in the 100 trials. The simulation recorded the drone's position, orientation, velocity, angular velocity, gate information, position markers, and simulation time. This data could then be displayed in graphs through Python package *matplotlib.pyplot*.

V. Results and Discussion

A. Failures

Throughout testing, multiple failures were identified leading to adjustments in the controller design.

- Initially, the drone appeared unable to lift off from the ground. The quadrotor would stay airborne for approximately half a second and then drop to the floor. Upon inspection of the controller, the error came from insufficient tuning of the controller's gain matrices. The weights of the Q matrix needed to be significantly smaller than the weights of the R matrix to see any change in the duration of the flight time for the quadrotor. This change no longer resulted in a failure, but was a significant error dealt with throughout the controller development.

- Additionally, the controller design did not complete the obstacle course for all 100 simulations. The quadrotor was unable to pass through the last two gates, occasionally spiraling out or hitting the edge of the sixth gate. Upon trial and error, the constants $k_{attract}$ and e_{max} value were relaxed to accommodate a less aggressive torque control on the desired gate position. This change forced the drone to deviate slightly from the exact desired position in exchange for a perfect completion rate of the course.
- Furthermore, in testing the drone with a competing drone, the drone's flight path consistently ran into the other drone regardless of the value of k_{repel}. To account for this change, k_{repel} was reduced to a value of 1 millimeter while the desired position of the gate p_{des} was raised by 1 meter. In this manner, the drone could target a location other than the center of the gate to account for potential collisions with competing drones during takeoff and navigation.
- Finally, while the drone did meet all requirements, its speed was much slower than desired for ideal racing. To resolve this, slight adjustments to the Q and R matrices and $k_{attract}$, k_{repel}, k_{des}, and e_{max} constants were made to increase the speed of the drone and optimize race performance. These numerical values are displayed in the Ideal Parameters subsection under Section V.

B. Histogram Data

Displayed below illustrates the results from the 100 simulations as defined in Section IV. In Figure 1, the histogram data for the course completion time, RMSE between estimated vs. actual position, and RMSE between desired vs. actual position distributions were recorded.

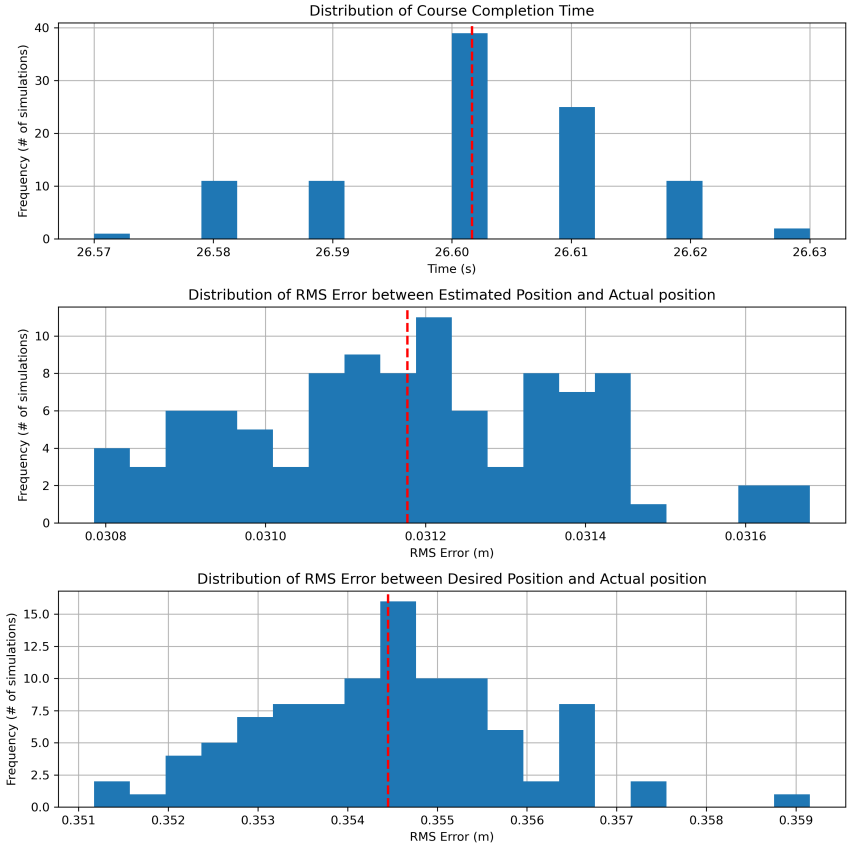

Fig. 1 Histogram data after 100 simulations for the single controller experimentation

From the *Distribution of Course Completion Time* histogram, it can be visually verified that the final controller, observer, and tracking-implementation system produced 100 successful completions of the race as the drone did not reach the simulation limit of 30 seconds. The distribution of finishing times fell between 26.57 and 26.63 seconds with an average finishing time of 26.6017 seconds. Since there were no recorded 30 second finishing times, it can be inferred

that the drone did not encounter any collisions with the obstacle course. Thus, all drones successfully reached the final ring in under 30 seconds, and the system passes the first and second requirement.

From the *Distribution of RMS Error between Estimated and Actual Position* histogram, the error between the estimated position and actual position followed a uniform distribution centered around an average RMS error of 0.031177 meters with no significant outliers. This error of about 1.22 inches suggests that the sensor was extremely accurate in sensing the drone's position as it navigated through the course. Since the average error recorded was below 0.1 meters, the third requirement is verified.

From the *Distribution of RMS Error between Desired and Actual Position* histogram, the error between the desired position and actual position also followed a uniform distribution without any significant outliers. With a mean error of 0.35445 meters, this difference can be explained by the drone needing to reach a new desired position after each pass of the gate. Thus, a RMSE error between 0.351 meters and 0.359 meters is an acceptable range for the desired vs. actual position of the drone. This error was also observed to change based on the constants chosen to solve p_{des} during gradient descent. Because the error was less than 0.5 meters, the fourth requirement is verified.

C. Plot Data

Displayed below illustrates a single data file collected from the final simulation within the 100 trials for both the *Estimated vs. Actual* and *Desired vs. Actual* x, y, z positions of the drone as defined in Section IV.

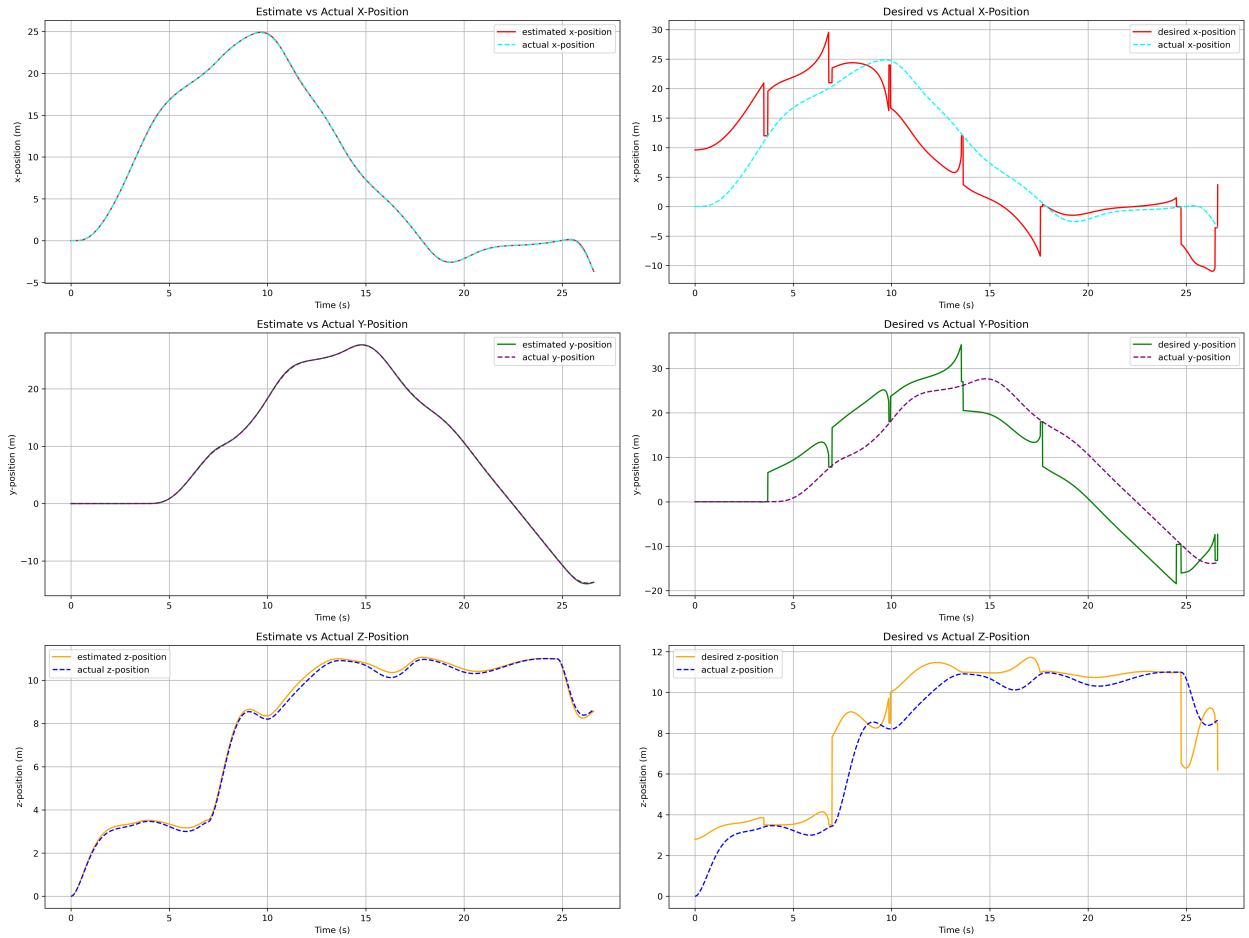

Fig. 2 Comparison plots of x, y, z position data of the drone for the final simulation

For the observer, Figure 2 illustrates a consistent overlapping between the estimated and actual x, y, z positions. While there was a slight deviation in estimation between 15-18 seconds for the *Estimated vs. Actual Z-Position* plot, this discrepancy can be attributed to the high velocity needed to propel the drone to the necessary height of the gate. Since

the visual comparison plots also suggest a low error between the estimated and actual position, the implemented sensor model facilitates an accurate state measurement of the drone position which in turn provides accurate feedback to the drone through the control inputs. Thus, the designed observer is robust.

For the controller, the desired position of each gate was followed accurately by the drone. For the *Desired vs. Actual X-Position* and *Desired vs. Actual Y-Position* plots in Figure 2, if the distance between the drone's actual and desired position of the gate was greater than 1.0 meter (e_{max}), the control system would moderate the drone's approach by gradually guiding it towards the gate. Through this gradient descent algorithm, the drone was able to successfully reach each gate p_{des} as visualized by the 7 abrupt changes in the plots finishing the course in roughly 26.6 seconds. For the *Estimated vs. Actual Z-Position* plot, there were greater magnitude jumps in the desired z-position as seen between 6-8 seconds and 24-26 seconds. This trend can be explained by the set gate heights being drastically different along with the change in the desired position of the gate by +1 meter. Overall, the controller was successfully able to guide the drone towards the desired position of the gate with an average error of 0.35445 meters across 100 simulations. Thus, the designed controller is robust.

D. Race Simulation

To verify the fifth requirement an experiment was conducted with two drones—the original drone recorded in this report and a separate drone with its own controller designed for the experiment outlined in Section IV. The simulations resulted in a 100% completion rate by the original drone. A count was kept of how many times the original drone succeeded and failed. This counter can be seen in Fig 3. confirming that the fifth requirement has been met. This experiment was meant to simulate the conditions of the race that the drone is participating in. Through this set of 100 simulations, the control system developed contains a robust collision avoidance system.

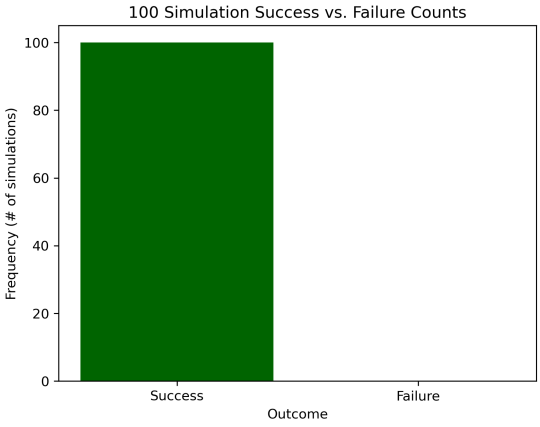

Fig. 3 Completion rate for drone in dual controller experimentation

E. Ideal Parameters

The controller and observer system were tested and analyzed to identify the conditions that optimized the performance of the drone. For the controller, the weights were determined through extensive testing, while for the observer, the weights remained the identity matrices. Q_c, R_c, Q_o, and R_o were defined as followed:

$$Q_c = \text{diag}\left(\frac{1}{100^2}, \frac{1}{100^2}, \frac{1}{10^2}, \frac{1}{105^2}, \frac{1}{105^2}, \frac{1}{105^2}, \frac{1}{50^2}, \frac{1}{100^2}, \frac{1}{100^2}, \frac{1}{50^2}, \frac{1}{50^2}, \frac{1}{50^2}\right) \tag{22}$$

$$R_c = \text{diag}\left(\frac{1}{0.065^2}, \frac{1}{0.065^2}, \frac{1}{0.065^2}, \frac{1}{16^2}\right) \tag{23}$$

$$Q_o = \text{diag}\,(1,1,1,1,1,1,1,1,1,1,1,1) \qquad R_o = \text{diag}\,(1,1,1,1,1,1,1,1,1,1,1,1) \tag{24}$$

These weights proved to work well in the designed control system, providing optimal state feedback to the drone while increasing race speed and navigation performance.

VI. Conclusion

A controller and observer intended to properly control a drone through seven hoops within an obstacle course were tested and documented in this paper. Ideal weights and conditions were determined to ensure the stability of the drone and to satisfy the completion of the course without collisions. To verify that the results of the controller were reliable and consistent, the simulation was run 100 times individually and and another 100 times with dual controller implementation with the intention of testing the controller's various qualities and performance limits. With thorough testing, the controller was able to reduce its race-time to an average of 26.60 seconds. This quick race-time along with the analysis derived, provided the paper proper data to analyze the success of the controller-observer design. For future work, further testing and design work could be put into the collision avoidance implementation to create an even more ideal control system. Extensive testing of the controller gains along with varying sensor noise for the observer could also be explored to reduce race time even further.

Acknowledgments

We would like to thank the AE353 course staff for their guidance and advice throughout this design project.

References

[1] Autel, "EVO Max Series," *Evo Max Series*, 2024.

[2] Skydio, "Skydio enters final phase of U.S. Army's short range reconnaissance tranche 2 program," *Skydio Blog RSS*, 2024.

Appendix

A. Team Reflection

As a team, we were able to accomplish the goals of the design project! We made sure to follow and respect the conditions of our team contract. Having time to work on the project during class with the assistance of the TA's and Professor Chang were very helpful.

B. Timeline

Table 1 Work Log

Date	Task	Person
4/16/24	Linearized Equations of Motion for Dynamic Model	Conan Zhang
4/16/24	Linearized Sensor Model	Conan Zhang
4/17/24	Verified Controllability and Observability	Emily Lory
4/17/24	Started Design of Stable Controller and Observer	Both Members
4/17/24	Wrote Abstract and started Theory Section	Conan Zhang
4/17/24	Wrote Nomenclature and Introduction Sections	Emily Lory
4/18/24	Started Experimental Methods Section	Emily Lory
4/18/24	Implemented Tracking and Collision Avoidance	Both Members
4/18/24	Finished Controller design	Both Members
4/18/24	Finished Theory Section	Conan Zhang
4/19/24	Edited Experimental Methods and Results and Discussion	Emily Lory
4/19/24	Made final edits on Report for Draft Submission	Both Members
4/21/24	Found desirable Q and R matrices for Controller	Conan Zhang
4/22/24	Edited Failures and Verification Sections	Emily Lory
4/23/24	Found desirable k constants for Gradient Descent	Emily Lory
4/24/24	Presented Path-Finding Algorithm	Both Members
4/25/24	Tested drone controller with another controller	Conan Zhang
4/25/24	Edited Report based on Draft feedback	Conan Zhang
4/26/24	Finished data collection for state estimate	Both Members
4/26/24	Edited Results and Discussion	Conan Zhang
4/27/24	Developed histograms for RMSE data	Both Members
4/27/24	Optimized Controller for Race Day submission	Emily Lory
4/28/24	Finished Results and Discussion Section	Conan Zhang
4/29/24	Made final changes for Race Day Submission	Emily Lory
4/29/24	Wrote Conclusion and Acknowledgements	Emily Lory
4/30/24	Made final edits on Report for Final Submission	Both Members