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 ith gate (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 play essential roles in both commercial sectors and government operations due to their autonomous and robust nature. 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 with its durability and stealth capabilities has already been incorporated into the US Army's Short Range Reconnaissance program [2]. Quadrotor drones are pivotal for the future of aerospace technology due to their versatility and ubiquitous nature in today's industries. This report details a quadrotor drone with four rotors driven by an electric motor and an obstacle course containing 7 hoops of varying heights that the drone must navigate through. 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, (m_x, m_y, m_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 drones racing. 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 obstacle 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 implemented and designed 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 end 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 $f_{ze} = 4.905$. Values of zero for all state and input variables except for the net force along the body-fixed z-axis were chosen to stabilize the system around an airborne drone while keeping the controller controllable. The next step is to place the space-space model into the state feedback form. By taking the Jacobian of the dynamic model concerning the state variables and control inputs, the coefficient matrices A and B can be found.

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

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

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.

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

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:

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

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:

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

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 for the controller-observer design.

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.

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

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 parameters, requirements, verifications, and trial procedures for the experiment.

- The system shall have a controller and observer designed in such a way that at least 80 out of 100 simulations are completed without a collision with the obstacle course.
- The system shall have a controller and observer designed in such a way that after 100 runs the average race time is less than or equal to 30 seconds.
- The system shall have an observer designed in such a way that the average RMS error between the estimated position and actual position of the drone is less than 0.3 meters over 100 simulations.
- The system shall have a controller designed in such a way that the average RMS error between the actual position and the desired position of the drone is less than 0.5 meters over 100 simulations.
- The system shall have a controller and observer designed in such a way that 70 out of 100 simulations are successfully completed 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 50 seconds, and if the race time exceeded the maximum simulation time, the simulation ended and concluded the race. The sensor noise was given as 0.01. 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 with multiple drones. The drone for the race was sponsored by *Team Flying Mambas*.

C. Verification

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

- To verify the first requirement, 100 simulations were tested through the Conda interface where the number of complete races gave a measure of the number of times a collision with the obstacle course occurred. Once collided with a part of the obstacle course the quadrotor will be deflected either towards the ground or off course. This essentially leads to an incomplete race which is why the requirement is verified through the analysis of the number of complete races. It is assumed that the only collision occurring in these simulations is between the quadrotor and the obstacle course because there is only one quadrotor racing in the 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 time until failure. 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 0.3 meters. Additionally, a visual comparison plot will be made with the last simulation to compare how

the state estimate vs. actual state vary 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 and 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 multiple controllers in parallel to simulate ideal race conditions. To keep consistent with future racing conditions, the simulation was performed with five other competing drones. To quantify the number of drone on drone collisions, the number of disqualifications are recorded and analyzed.

D. Trials

The drone's behavior was tested in the Meshcat simulation 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 trails. 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 refining 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 is no longer a failure for the current controller, but it was a significant error dealt with throughout development.
- Currently the controller design does not consistently complete the obstacle course. Regularly, the quadrotor will not be able to pass through the last two gates after the longer distance it needs to travel. The drone tends to spiral out of control and lands itself on the ground.
- For future improvements, slight adjustments to the Q and R matrices will be necessary to increase the speed of the drone and optimize performance.

B. Histogram Data

C. Race Data

D. Ideal Parameters

VI. Conclusion

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," 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

B. Timeline

Table 1 Work Log

Date	Task	Person
4/16/22	Linearized Equations of Motion for Dynamic Model	Conan Zhang
4/16/22	Linearized Sensor Model	Conan Zhang
4/17/22	Verified Controllability and Observability	Emily Lory
4/17/22	Started to design Stable Controller and Observer	Both Members
4/17/22	Wrote Abstract and started Theory Section	Conan Zhang
4/17/22	Wrote Nomenclature and Introduction Sections	Emily Lory
4/18/22	Started Experimental Methods Section	Emily Lory
4/18/22	Implemented Tracking and Collision Avoidance	Both Members
4/18/22	Finished Controller design	Both Members
4/18/22	Finished Theory Section	Conan Zhang
4/19/22	Edited Experimental Methods and Results and Discussion	Emily Lory
4/19/22	Made final edits on Report for Draft Submission	Both Members