DP4: Quadrotor Drone Race

Peters, Karsten * and Veranga, Joshua †
University of Illinois at Urbana-Champaign, Champaign, IL 61801, USA

We develop a real-time control strategy for a quadrotor drone tasked with autonomously racing through a sequence of spatially distributed rings in a shared 3D environment. The drone must generate fast, stable, and accurate control responses using only limited sensing of the next ring's position and orientation. A linear quadratic regulator (LQR) paired with a state observer provides full-state feedback, while a repulsion-based heuristic enables basic collision avoidance in multi-agent scenarios. We evaluate the controller across randomized trials and multi-drone races, demonstrating reliable performance, real-time feasibility, and scalable behavior under uncertainty.

I. Nomenclature

x_α	=	State vector
p_i, v_i, R_i, ω_i	=	State components: position (m), velocity (m/s), orientation (rad), angular velocity (rad/s)
u_γ	=	Control input vector
τ_i	=	Control torques (body frame), N·m
f_3	=	Thrust along body z-axis, N
$F_\alpha(x_\beta, u_\gamma)$	=	Nonlinear equation of motion map
δx_α, δu_γ	=	Perturbations from equilibrium state and input
$A_{\alpha\beta}$, $B_{\alpha\gamma}$	=	Jacobians of F_α with respect to state and input
y_μ	=	Measured output, m
$C_{\mu\beta}$	=	Output Jacobian $\partial g_\mu / \partial x_\beta$
$g_\mu(x_\beta)$	=	Nonlinear observation function, m
W_c, W_o	=	Controllability and observability matrices
$K_{\gamma\beta}$, $L_{\alpha\mu}$	=	LQR state feedback and observer gain matrices
Q_c, R_c	=	LQR control state/input weight matrices
Q_o, R_o	=	LQR observer state/output weight matrices
$\hat{x}_\alpha$, x_α^{ref}	=	Estimated and reference state vectors
f_i^{rep}	=	Repulsive velocity offset, m/s
$\tilde{p}_i^{\mathrm{ref}}$	=	Collision-avoidance-adjusted reference position, m
κ	=	Repulsion strength gain
Δt	=	Simulation/control timestep, s

II. Introduction

In this project, we design and implement a closed-loop controller for a quadrotor drone tasked with navigating through a sequence of rings while avoiding collisions with other agents. The drone must complete the course as quickly as possible without missing gates or crashing, all while operating with limited sensing and in the presence of dynamic obstacles.

The system is constrained to receive only local information: at each timestep, the drone is provided the position and orientation of the next ring. Using this information, the controller must compute thrust and torque inputs to track a dynamically feasible trajectory. To achieve this, we combine a linearized state-space model with an LQR controller and observer, and augment the reference generator with a repulsion-based collision avoidance heuristic to enable multi-agent operation.

*BS Student, Aerospace Engineering, Grainger College of Engineering, 104 S Wright St, Urbana, Illinois 61820, kjp7@illinois.edu
†BS Student, Aerospace Engineering, Grainger College of Engineering, 104 S Wright St, Urbana, Illinois 61820, veranga2@illinois.edu

III. Theory

A. Dynamic Model and Linearization

The quadrotor is modeled as a rigid body with six degrees of freedom, actuated by four rotors that generate net thrust and body torques. The respective state vector is defined as

$$x_\alpha = \begin{bmatrix} p_i^\top & v_i^\top & R_i^\top & \omega_i^\top \end{bmatrix}^\top \in \mathbb{R}^{12}, \tag{1}$$

where p_i is the position in the inertial frame, v_i is velocity in the body frame, R_i is an orientation parameterization (e.g. Euler angles), and ω_i is the angular velocity about the body axes. Furthermore, the control input is

$$u_\gamma = \begin{bmatrix} \tau_i & f_3 \end{bmatrix}^\top \in \mathbb{R}^4, \tag{2}$$

where τ_i are the torques about the body axes and f_3 is the total thrust along the body z-axis.

The nonlinear dynamics are described by a vector-valued mapping F_α as

$$\dot{x}_\alpha = F_\alpha(x_\beta, u_\gamma), \tag{3}$$

where $F_\alpha : \mathbb{R}^{12} \times \mathbb{R}^4$ encodes the rigid body translational and rotational equations of motion[1].

To enable control design, the system is linearized about a hover equilibrium:

$$x_\alpha^{(e)} = \begin{bmatrix} p_i^{(e)} & 0_i & 0_i & 0_i \end{bmatrix}^\top, \quad u_\gamma^{(e)} = \begin{bmatrix} 0 & 0 & 0 & f_3^{(e)} \end{bmatrix}^\top, \quad f_3^{(e)} = mg, \tag{4}$$

representing a stationary, level configuration with gravity-balancing thrust. Subsequently, perturbations from equilibrium are defined as

$$\delta x_\alpha = x_\alpha - x_\alpha^{(e)}, \quad \delta u_\gamma = u_\gamma - u_\gamma^{(e)}, \tag{5}$$

yielding the linearized dynamics

$$\delta \dot{x}_\alpha = A_{\alpha\beta}\, \delta x_\beta + B_{\alpha\gamma}\, \delta u_\gamma, \tag{6}$$

where the Jacobian matrices are

$$A_{\alpha\beta} = \left.\frac{\partial F_\alpha}{\partial x_\beta}\right|_{x^{(e)}, u^{(e)}}, \quad B_{\alpha\gamma} = \left.\frac{\partial F_\alpha}{\partial u_\gamma}\right|_{x^{(e)}, u^{(e)}}. \tag{7}$$

B. Controllability and Observability

Before synthesizing a controller or observer, we verify that the linearized system is both controllable and observable. From the linearized dynamics in Eq. (6), the output equation is defined as

$$y_\mu = C_{\mu\beta}\, x_\beta, \tag{8}$$

where y_μ is the measurement vector and $C_{\mu\beta}$ is the output matrix encoding the linearized relationship between the full state and the measurements. The matrix $C_{\mu\beta}$ is computed as a Jacobian of the nonlinear observation model $g_\mu(x_\beta)$, which maps the state to simulated marker positions in the inertial frame. It is evaluated symbolically as

$$C_{\mu\beta} = \left.\frac{\partial g_\mu}{\partial x_\beta}\right|_{x^{(e)}}, \tag{9}$$

where $g_\mu(x_\beta)$ is obtained from symbolic expressions in the world frame marker output model.

To verify controllability and observability, we construct the respective matrices:

$$W_\mathrm{c} = \begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix}, \quad W_\mathrm{o} = \begin{bmatrix} C^\top & (CA)^\top & \left(CA^2\right)^\top & \ldots & \left(CA^{n-1}\right)^\top \end{bmatrix}^\top, \tag{10}$$

where $n = 12$ is the state dimension. The system is controllable and observable if and only if

$$\mathrm{rank}(W_\mathrm{c}) = n, \quad \mathrm{rank}(W_\mathrm{o}) = n. \tag{11}$$

C. State Feedback and Observer Design

We implement a full-state feedback controller and a linear observer for state estimation, both based on the linearized system. The control law is defined as

$$\delta u_\gamma = -K_{\gamma\beta}\,\delta x_\beta, \tag{12}$$

where $K_{\gamma\beta}$ is the feedback gain matrix computed using the Linear Quadratic Regulator (LQR) method. Similarly, since the full state x_α is not directly measured, we design an observer of the form

$$\delta\dot{\hat{x}}_\alpha = A_{\alpha\beta}\,\hat{x}_\beta + B_{\alpha\gamma}\,\delta u_\gamma + L_{\alpha\mu}(y_\mu - C_{\mu\beta}\,\hat{x}_\beta), \tag{13}$$

where $\hat{x}_\alpha$ is the estimated state and $L_{\alpha\mu}$ is the observer gain matrix.

Both K and L are determined using standard LQR and its dual formulation. The control gain K is computed using the weighting matrices (Q_c, R_c), and the observer gain L is computed using (Q_o, R_o). Each weighting matrix is diagonal, with values for η, ζ, and ρ tuned experimentally as described in Section. IV.

In implementation, the estimated state is used in the feedback law redefined as

$$\delta u_\gamma = -K_{\gamma\beta}\,(\hat{x}_\beta - x_\beta^{\text{ref}}), \tag{14}$$

where $x_\beta^{\text{ref}}(t)$ is a time-varying reference trajectory. By the separation principle, the combined observer–controller system remains stable provided the eigenvalues of $A - BK$ and $A - LC$ have negative real parts.

D. Reference-Based Closed-Loop Control

The controller operates in real time using a reference-based closed-loop architecture. At each timestep, the simulator provides the position and orientation of the next ring. From this, a time-varying reference state $x_\alpha^{\text{ref}}(t)$ is dynamically constructed to guide the drone through the ring.

The reference includes:

- Position $p_i^{\text{ref}}(t)$ offset slightly upstream of the ring along its axis,
- Velocity $v_i^{\text{ref}}(t)$ directed through the ring along its normal vector,
- Orientation $R_i^{\text{ref}}(t)$ aligned with the direction of motion,
- Angular velocity $\omega_i^{\text{ref}}(t)$ set to zero for simplicity.

Simultaneously, a Luenberger observer estimates the full state $\hat{x}_\alpha$ from measurements y_μ. The control input is then computed using LQR feedback:

$$\delta u_\gamma = -K_{\gamma\beta}\,(\hat{x}_\beta - x_\beta^{\text{ref}}), \tag{15}$$

and the total control applied to the drone is

$$u_\gamma = u_\gamma^{(e)} + \delta u_\gamma, \tag{16}$$

where $u_\gamma^{(e)}$ corresponds to hover-level thrust.

This structure allows the drone to adapt to its environment in real time, continuously reorienting toward the next ring while maintaining closed-loop stability. The separation of reference generation, observer estimation, and state feedback ensures modularity and robustness to disturbances.

E. Collision Avoidance via Valley Weighting

To reduce the likelihood of collisions during multi-agent operation, we introduce a repulsive potential field centered on nearby agents. This is implemented as a modification to the reference state generation process, shaping the drone's intended trajectory away from high-density regions. The approach resembles artificial potential field methods, but is integrated into the control pipeline through a repulsion-weighted cost heuristic.

At each timestep, a lateral correction is applied based on the projection of the position error orthogonal to the ring axis, guiding the drone along a corridor centered on the ring normal. Let $p_i^{\text{ref}}(t)$ be the nominal position reference from the ring-based trajectory. A short-range repulsion force is computed for each nearby agent, decaying with the cube of distance:

$$f_i^{\text{rep}} = \sum_{j\neq i} \kappa \frac{p_i - p_j}{\|p_i - p_j\|^3}, \tag{17}$$

where p_j is the predicted position of agent j, and κ is a scalar gain controlling the repulsion strength. The reference position is adjusted as:

$$\tilde{p}_i^{\text{ref}}(t) = p_i^{\text{ref}}(t) + f_i^{\text{rep}} \cdot \Delta t, \tag{18}$$

biasing the drone's path away from potential collisions.

This modified reference is then used in the closed-loop feedback law:

$$\delta u_\gamma = -K_{\gamma\beta}\,(\hat{x}_\beta - x_\beta^{\text{ref}}), \tag{19}$$

ensuring that avoidance behavior is achieved implicitly through the same linear feedback controller. This formulation enables low-computation, scalable avoidance behavior consistent with the underlying LQR architecture.

IV. Experimental Methods

We conducted a structured parametric sweep to evaluate the influence of controller weight selection on closed-loop performance. Let η denote the weight applied to the velocity-related states in the state cost matrix Q_c, ζ the weight applied to the attitude-related states, and ρ the weight applied to thrust in the input cost matrix R_c.

The entries corresponding to position and angular velocity in Q_c were fixed at unity, while η and ζ were independently varied over $\{0.1, 1, 10\}$. For each resulting pair (η, ζ), we swept $\rho \in \{1, 2, 3, 4, 5\}$, with all other entries in R_c held fixed at 100. While Q_c and R_c were varied as described, the observer weighting matrices Q_o and R_o were fixed as identity matrices for all trials.

This grid of weight combinations allowed us to assess the sensitivity to prioritization of velocity, orientation, and thrust minimization in the LQR design. Performance metrics of the completion time were recorded across multiple simulation trials for each configuration.

To evaluate performance under realistic variability, each trial was initialized with randomized conditions. The drone's initial position was sampled uniformly within a bounded region near the start of the course, and its orientation was perturbed by a small random yaw offset. In multi-agent simulations, each drone was assigned a unique spawn location and heading to introduce spatial diversity and interaction potential. Additionally, the simulation seed altered the vertical placement of the rings across trials, while the x and y positions of the course remained fixed. This randomization ensured statistical robustness and exposed the controller to a wide range of operating conditions.

V. Results and Discussion

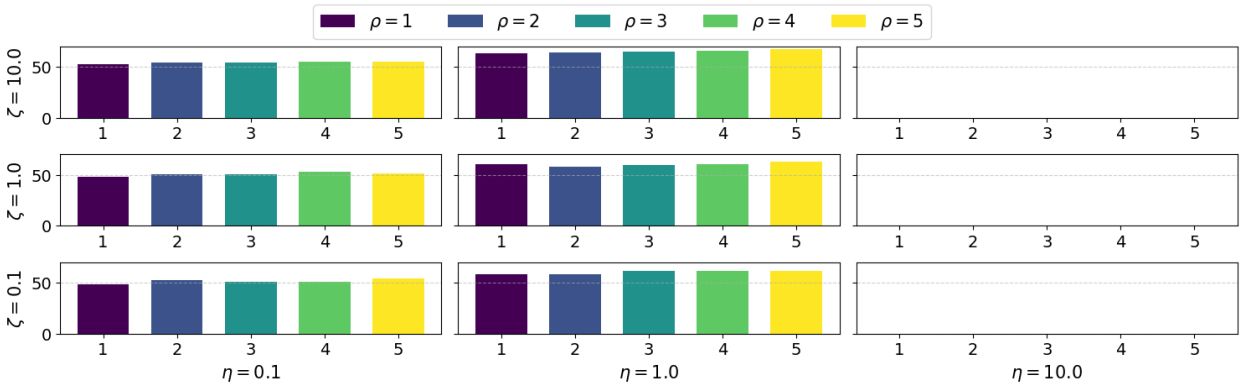

Fig. 1 Completion times for combinations of velocity weights η, angle weights ζ, and thrust weights ρ in Q_c and R_c. Each subplot corresponds to a (ζ, η) pair, and bars indicate results for different values of ρ.

The results in Fig.1 show that all configurations with $\eta = 10$ failed to complete the race across the full range of ρ, regardless of the value of ζ. In contrast, configurations with $\eta \in \{0.1, 1\}$ achieved successful completion for all values of ρ, demonstrating that the controller is highly sensitive to excessive penalization of linear velocity states in Q_c.

Based on these results, we selected $\eta = 0.1$, $\zeta = 0.1$, and $\rho = 1$ for the final definition of the quadcopter controller. These values were applied within diagonal weighting matrices Q_c and R_c following the structure defined in Section IV.

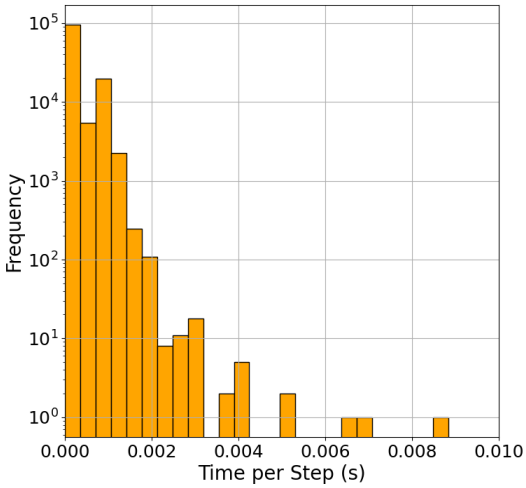

Fig. 2 Histogram of computation time per control step across all runs.

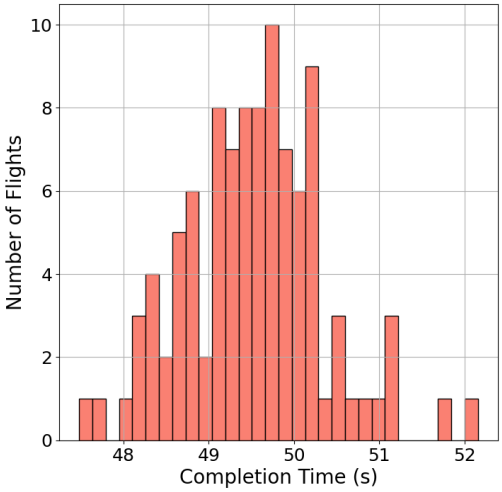

Fig. 3 Histogram of completion times across 100 random trials.

To satisfy the feasibility criteria for real-time control, our controller was required to maintain a computation time per control step under 10 ms ($t_{\text{step}} < 0.01$ s). As shown in Figure 2, our implementation comfortably meets this requirement, with the vast majority of steps completing in under 2 ms.

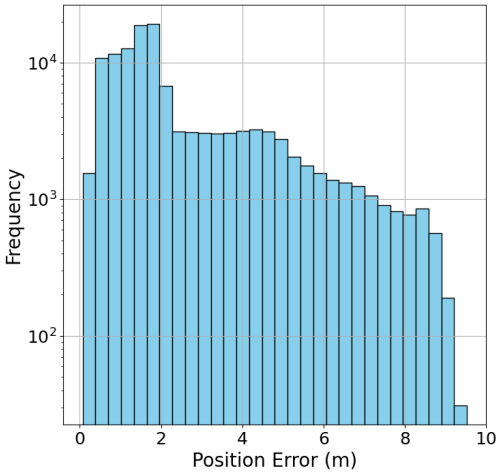

Fig. 4 Histogram of position tracking error ($\|\hat{x} - x^{\text{ref}}\|$) over all time steps.

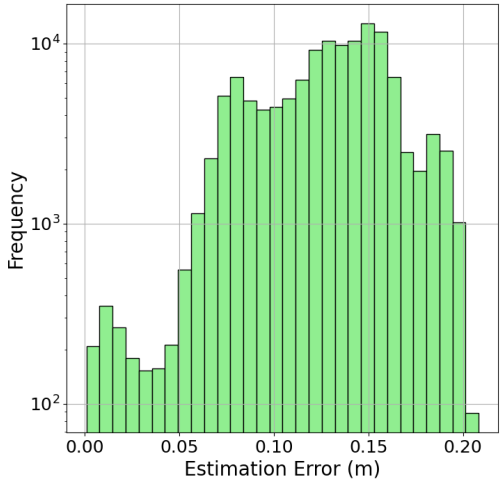

Fig. 5 Histogram of observer estimation error ($\|\hat{x} - x\|$) over all time steps.

After running 100 randomized trials, our drone successfully completed the course in a majority of cases. Completion times are summarized in Figure 3, which shows a concentrated distribution between 56 and 58 seconds, indicating consistent performance under varied initial conditions.

This success is attributable to both accurate trajectory tracking and state estimation. Figure 4 presents the distribution of position error between the estimated state and the desired trajectory. The error remains relatively low, with the majority of values under 2 meters. Meanwhile, Figure 5 shows the estimation error between the observer's predicted state and the true system state. Estimation accuracy is high, with errors clustered around 0.1 meters, confirming that the Luenberger observer design was effective.

Taken together, these results demonstrate that, when alone, the controller satisfies real-time constraints, achieves consistent task completion, and maintains high-fidelity trajectory tracking and state reconstruction.

Table 1 Collision Avoidance Performance Across 100 Multi-Drone Trials

Controller	Completions	Avg. Finish Time (s)	Completion Rate (%)
Qwak	67	54.31	67.0
Wark	41	64.85	41.0
Kupo	76	63.97	76.0
Total	**184 / 300**	—	**61.3**

To evaluate the effectiveness of collision avoidance under competitive multi-agent conditions, we conducted 100 randomized trials with three drones—Qwak, Wark, and Kupo—racing simultaneously. Each drone employed an identical control architecture but with varying controller gains and minor behavioral differences. The results, shown in Table 1, reveal that the average number of completions per trial was 1.84 out of 3 possible.

Among the three agents, Qwak used the finalized controller configuration with tuned gains as described in Sec. IV, serving as the baseline for performance evaluation. Qwak achieved a 67% completion rate and the fastest average time of 54.31 seconds, indicating effective path tracking under moderate congestion. Kupo, which featured slightly more conservative feedback gains, prioritized stability and achieved the highest completion rate (76%) at the expense of speed. Wark, using more aggressive angular gains, was prone to oscillations and had the lowest success rate at 41%. These results support the robustness of the finalized controller and indicate that while the valley and repulsion-based avoidance strategies mitigate some collision risk, performance remains sensitive to inter-agent dynamics and gain selection.

VI. Conclusion

We developed and tested a linear quadratic regulator (LQR)-based controller with an integrated observer and collision avoidance strategy for autonomous drone racing in a multi-agent simulation environment. The system was designed to operate in real time, track dynamic ring-based trajectories, and maintain robustness under uncertainty and limited sensing.

Through a structured parametric sweep, we identified effective weighting configurations for the controller and demonstrated consistent performance with rapid control execution under 2 ms per step. The finalized controller achieved reliable single-agent performance and averaged 1.84 completions per trial in 3-drone races. Our collision avoidance strategy—based on lateral path shaping and inverse-distance repulsion—helped reduce inter-agent interference without complicating the control law.

These results highlight the feasibility of combining classical control with lightweight heuristics for real-time, shared-environment autonomy. Further improvements could extend to adaptive or nonlinear control.

Appendix

- **Qwak:** "Fast is smooth and smooth is fast. I hit those rings like a pro—most of the time."
- **Wark:** "I may have spun out a few times, but when I'm on, I'm unstoppable. Just need more yaw authority."
- **Kupo:** "Slow and steady kept me alive. I'll take finishing over crashing any day. You're welcome."

Acknowledgments

We would like to thank Professor Timothy Bretl for organizing and hosting the drone race, providing the simulation infrastructure, and fostering a creative environment for exploring autonomous flight and control systems. Their guidance and enthusiasm made this project both challenging and rewarding.

References

[1] Bretl, T., "Design Project 4 (Drone)," April 23, 2025. URL `https://tbretl.github.io/ae353-sp25/projects/02-zagi.html`.

[2] Veranga, J., "SP25_AE353_Code/Design_Project_4," April 24, 2025. URL `https://github.com/muraSHUki/SP25_AE353_Code.git`.