ZephyFlyer: Drone Flight Under Wind Disturbances

Hanze Liu, Nathan Sun, Yunliang Zhao
24-774 Advanced Control Systems Integration, Fall 2025
Carnegie Mellon University

***Abstract*—This paper investigates disturbance-aware control strategies for quadrotor flight under wind disturbances. Unknown aerodynamic forces degrade trajectory tracking performance and induce steady-state position drift, particularly on lightweight aerial platforms operating under sensing and computational constraints. Classical, optimal, and predictive controllers—including cascaded PID, Linear Quadratic Regulation (LQR), and Tiny Model Predictive Control (TinyMPC)—are evaluated with and without an Extended State Observer (ESO) for online disturbance estimation. Controllers are benchmarked in simulation under impulse, global, and spatially localized wind profiles, and selected controllers are validated on hardware using hover and circular trajectory tracking tasks on a Crazyflie 2.1+ platform. Results show that ESO augmentation significantly improves disturbance rejection in both simulation and hardware without requiring redesign of the underlying controller. While LQR and TinyMPC demonstrate strong nominal tracking performance in simulation, their robustness degrades under unmodeled disturbances and real-time implementation constraints on embedded hardware. In contrast, a well-tuned cascaded PID controller augmented with an ESO provides the most robust and deployable solution on embedded hardware, highlighting fundamental trade-offs between optimality, robustness, and real-time feasibility in practical quadrotor control.**

I. Introduction and Related Work

Quadrotor unmanned aerial vehicles (UAVs) operating in real-world environments are inevitably subject to aerodynamic disturbances such as wind gusts and crosswinds. Wind gusts remain a primary cause of flight instability for lightweight quadrotors, a challenge documented extensively in recent surveys [1]. These disturbances introduce unknown external forces and moments that degrade trajectory tracking accuracy, induce steady-state position drift, and may destabilize flight, particularly for lightweight platforms operating under sensing and computational constraints. Robust disturbance rejection is therefore critical for reliable quadrotor deployment.

Classical quadrotor control architectures, most notably cascaded PID controllers, remain widely used in practice due to their simplicity, low computational cost, and empirical robustness. However, PID controllers rely solely on feedback and do not explicitly estimate external disturbances, limiting their effectiveness under sustained wind. Model-based controllers such as Linear Quadratic Regulators (LQR) and Model Predictive Control (MPC) provide a principled framework for trajectory tracking by optimizing control actions with respect to a system model and performance criteria. Recent work such as TinyMPC demonstrates that MPC can be implemented efficiently on resource-constrained microcontrollers using tailored solvers [2].

Despite their theoretical advantages, model-based controllers are sensitive to modeling errors, unmodeled dynamics, and external disturbances. Prior work has shown that high-performance optimal control often relies on accurate global state estimation and high-bandwidth feedback. For example, Landry et al. achieved precise quadrotor flight using time-varying LQR controllers supported by motion capture systems and offboard computation [3]. Such sensing and computational assumptions are difficult to satisfy on lightweight embedded platforms operating under wind disturbances.

Direct measurement of wind disturbances on lightweight aerial platforms is difficult due to sensing and payload limitations. Observer-based approaches therefore infer external disturbances indirectly through model mismatch. Among these, Extended State Observers (ESOs) provide a computationally efficient and modular solution by augmenting the system state with lumped disturbance terms, making them well suited for real-time embedded implementation.

Extended State Observers (ESOs), commonly associated with Active Disturbance Rejection Control (ADRC), adopt a similar disturbance estimation philosophy by augmenting the system state with lumped disturbance terms capturing wind forces and unmodeled dynamics. Compared to more complex observer formulations with formal stability guarantees, ESOs provide a computationally efficient and modular alternative well suited for real-time implementation on embedded platforms.

The objective of this project is to systematically evaluate disturbance-aware quadrotor control strategies under wind by integrating an ESO-based disturbance estimation framework with classical, optimal, and predictive controllers. Cascaded PID, LQR, and TinyMPC controllers are benchmarked with and without ESO augmentation on hover and circular trajectory tracking tasks in simulation, and selected controllers are validated on a Crazyflie 2.1+ platform to assess robustness, tracking performance, and real-time deployability. Through direct comparison, this work clarifies practical trade-offs between tracking optimality, disturbance robustness, and implementation feasibility for lightweight embedded aerial platforms.

A. Contributions

- Implemented and benchmarked cascaded PID, LQR, and TinyMPC controllers for quadrotor hover and trajectory tracking under wind disturbances.
- Designed and integrated a 12-state Extended State Observer (ESO) for online disturbance estimation.

- Evaluated ESO-augmented controllers in simulation under impulse, global, and spatially localized wind profiles.
- Validated disturbance rejection performance on hardware by comparing PID and PID+ESO controllers on a Crazyflie 2.1+ platform.

B. *Key Takeaways*

Simulation and hardware results show that disturbance rejection is essential for reliable quadrotor flight under wind, as moderate disturbances can induce significant trajectory deviation and steady-state error when unaccounted for [4]. Observer-based approaches provide a practical alternative to direct wind sensing by estimating disturbances implicitly through model mismatch.

The Extended State Observer (ESO) offers a modular and controller-agnostic mechanism for improving robustness across multiple control architectures. Even when disturbance compensation is applied only through the vertical thrust channel, ESO augmentation significantly improves rejection of horizontal wind disturbances due to coupling between translational and attitude dynamics, while full horizontal compensation would require higher-order formulations at increased computational cost.

Model-based controllers such as LQR and TinyMPC demonstrate strong nominal tracking performance in simulation but exhibit reduced robustness under unmodeled disturbances and real-world uncertainties. In particular, TinyMPC presents challenges for onboard deployment due to real-time computational demands, highlighting the importance of hardware validation and disturbance-aware designs that prioritize robustness and deployability over nominal optimality.

II. System Modeling

A. *Quadrotor Dynamics*

As illustrated in Fig. 1, the quadrotor is controlled through a thrust input and body-frame rotational degrees of freedom. The Crazyflie quadrotor is modeled as a rigid body operating near hover conditions. A 12-state representation is used to describe its translational and rotational dynamics, with the state vector defined as

$$\mathbf{x} = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \\ \phi & \theta & \psi & p & q & r \end{bmatrix}^\top . \tag{1}$$

Here, $\mathbf{p} = [x, y, z]^\top$ and $\mathbf{v} = [\dot{x}, \dot{y}, \dot{z}]^\top$ denote the position and translational velocity in the inertial frame, $\boldsymbol{\eta} = [\phi, \theta, \psi]^\top$ represents roll, pitch, and yaw angles, and $\boldsymbol{\omega} = [p, q, r]^\top$ denotes the body angular rates.

Control inputs are defined at the force–torque level as

$$\mathbf{u} = \begin{bmatrix} \delta T & \tau_x & \tau_y & \tau_z \end{bmatrix}^\top , \tag{2}$$

where δT is the thrust deviation from hover and τ_x, τ_y, and τ_z are body-frame torques. Yaw control is fixed or weakly regulated for LQR-based controllers to focus on position tracking performance.

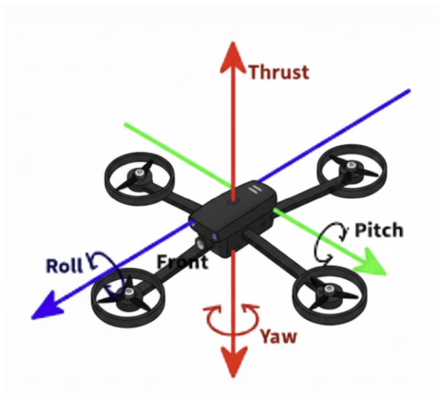


Fig. 1. Quadrotor body-frame axes and control inputs. Thrust acts along the body z-axis, while roll, pitch, and yaw correspond to rotations about the body-frame axes.

B. *Hover Linearization and Modeling Assumptions*

The dynamics are linearized about a nominal hover equilibrium,

$$\mathbf{x}_{eq} = \mathbf{0}, \quad T_{eq} = mg, \tag{3}$$

where m is the vehicle mass and g is gravitational acceleration.

Small-angle assumptions for roll and pitch,

$$\sin\phi \approx \phi, \quad \sin\theta \approx \theta, \quad \cos\phi \approx \cos\theta \approx 1, \tag{4}$$

are valid during near-hover flight with moderate lateral accelerations. Under these assumptions, the dominant translational dynamics reduce to

$$\ddot{x} = g\theta, \quad \ddot{y} = -g\phi, \quad \ddot{z} = \frac{1}{m}\delta T, \tag{5}$$

with attitude kinematics given by

$$\dot{\phi} = p, \quad \dot{\theta} = q, \quad \dot{\psi} = r. \tag{6}$$

Yaw dynamics are retained for completeness but are not emphasized in experiments.

C. *Reference Trajectory Definitions*

Trajectory tracking performance is evaluated using simple reference trajectories defined in the inertial frame.

1) Hover Trajectory: The hover reference is a constant position,

$$\mathbf{p}_{ref}(t) = \begin{bmatrix} x_0 & y_0 & z_0 \end{bmatrix}^\top , \tag{7}$$

used to assess disturbance rejection and steady-state regulation.

2) Circular Trajectory: A planar circular trajectory is defined as

$$\begin{aligned} x_{ref}(t) &= x_c + r\cos(\omega t), \\ y_{ref}(t) &= y_c + r\sin(\omega t), \\ z_{ref}(t) &= z_0, \end{aligned} \tag{8}$$

where (x_c, y_c) is the circle center, r is the radius, and ω is the angular frequency. These trajectories remain within the validity region of the hover-linearized model while inducing lateral motion.

D. Disturbance Modeling

External disturbances such as wind are modeled as unknown translational accelerations,

$$\mathbf{d}_f = \begin{bmatrix} d_{fx} & d_{fy} & d_{fz} \end{bmatrix}^\top, \tag{9}$$

assumed to vary slowly over time,

$$\dot{\mathbf{d}}_f = \mathbf{0}. \tag{10}$$

Augmenting the nominal state with disturbance terms yields the extended state

$$\mathbf{z} = \begin{bmatrix} \mathbf{p}^\top & \mathbf{v}^\top & \boldsymbol{\eta}^\top & \mathbf{d}_f^\top \end{bmatrix}^\top \in \mathbb{R}^{12}, \tag{11}$$

which provides a modeling interface for observer-based disturbance estimation and compensation.

E. Discrete-Time Model for Control

For digital implementation, the hover-linearized dynamics are discretized using forward Euler integration with sampling time T_s,

$$\mathbf{x}_{k+1} = \mathbf{A}_d \mathbf{x}_k + \mathbf{B}_d \mathbf{u}_k, \tag{12}$$

where

$$\mathbf{A}_d = \mathbf{I} + \mathbf{A}T_s, \quad \mathbf{B}_d = \mathbf{B}T_s. \tag{13}$$

This discrete-time model is shared across all controllers to ensure consistent comparative evaluation.

F. Summary and Modeling Limitations

The system model adopted in this work represents a hover-linearized approximation of the quadrotor dynamics with control inputs defined at the force–torque level, providing a tractable and consistent modeling framework for controller design and comparative evaluation. Operating near hover simplifies the system dynamics and ensures validity of the small-angle linearization, but model accuracy degrades outside this regime due to unmodeled aerodynamic effects and large attitude excursions.

III. Controller Design

A. PID Baseline Controller

The PID baseline follows the cascaded control structure used in the official Crazyflie firmware, with position and velocity loops generating attitude commands and a high-bandwidth inner loop stabilizing attitude and angular rates [5].

1) Outer Loop: Position Control: The outer loop computes desired body-frame velocities from position errors:

$$\mathbf{v}_{des}^b = K_p^{pos}(\mathbf{p}_{ref} - \mathbf{p}) + K_d^{pos}(\mathbf{v}_{ref} - \mathbf{v}), \tag{14}$$

where K_p^{pos} and K_d^{pos} are proportional and derivative gains, and the superscript b denotes the body frame.

2) Inner Loop: Attitude and Altitude Control: The desired body velocities are tracked using a PD controller for roll and pitch:

$$\begin{aligned} \phi_{cmd} &= K_p^{vel}(v_y^{des} - v_y) + K_d^{vel}\dot{v}_y, \\ \theta_{cmd} &= K_p^{vel}(v_x^{des} - v_x) + K_d^{vel}\dot{v}_x, \end{aligned} \tag{15}$$

where the velocity error is mapped to desired attitude through the small-angle approximation inherent in the hover model.

Altitude is regulated independently through a direct thrust command:

$$T_{cmd} = mg + K_p^z(z_{ref} - z) + K_d^z(\dot{z}_{ref} - \dot{z}), \tag{16}$$

where mg provides the nominal hover thrust.

Attitude stabilization uses a standard PD structure on Euler angles and body rates:

$$\boldsymbol{\tau}_{cmd} = K_p^{att}(\boldsymbol{\eta}_{cmd} - \boldsymbol{\eta}) + K_d^{att}(\boldsymbol{\omega}_{cmd} - \boldsymbol{\omega}), \tag{17}$$

where $\boldsymbol{\tau}_{cmd} = [\tau_x, \tau_y, \tau_z]^\top$ represents the commanded body torques.

B. Extended State Observer (ESO)

The Extended State Observer is designed to estimate both the nominal state and the disturbance vector from position and attitude measurements. The observer operates independently of the controller, providing improved state estimates and disturbance information that can be used for feedforward compensation.

1) Observer Dynamics: The observer dynamics are given by

$$\dot{\hat{\mathbf{z}}} = \mathbf{A}\hat{\mathbf{z}} + \mathbf{B}u + \mathbf{L}(\mathbf{y} - \hat{\mathbf{y}}), \tag{18}$$

where $\hat{\mathbf{z}} \in \mathbb{R}^{12}$ denotes the estimated augmented state, $u = T$ is the total thrust input, and $\mathbf{L} \in \mathbb{R}^{12\times 6}$ is the observer gain matrix.

Figure 2 illustrates the ESO estimation pipeline, organized into three computational phases. Phase 1 acquires sensor measurements and initializes states. Phase 2 performs model-based prediction using the continuous-time quadrotor dynamics. Phase 3 computes the innovation from measurement residuals, applies the observer correction, and extracts disturbance force estimates.

The augmented system matrix $\mathbf{A} \in \mathbb{R}^{12\times 12}$ captures the hover-linearized dynamics with disturbance coupling:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{G} & \mathbf{I}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix}, \tag{19}$$

where the gravity coupling matrix is

$$\mathbf{G} = \begin{bmatrix} 0 & g & 0 \\ -g & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{20}$$

The input matrix $\mathbf{B} \in \mathbb{R}^{12}$ reflects that thrust affects only vertical acceleration:

$$\mathbf{B} = \begin{bmatrix} \mathbf{0}_3^\top & 0 & 0 & \frac{1}{m} & \mathbf{0}_6^\top \end{bmatrix}^\top. \tag{21}$$

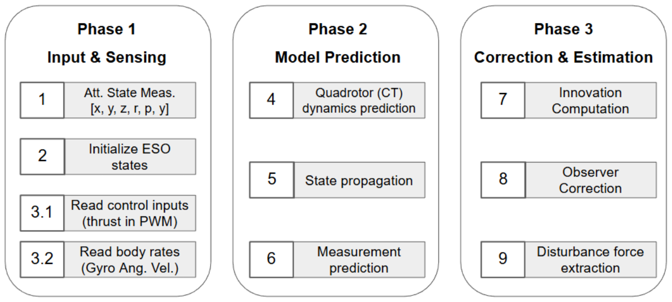


Fig. 2. Extended State Observer design workflow showing three computational phases: (1) Input & Sensing acquires attitude and state measurements, initializes ESO states, and reads control inputs; (2) Model Prediction performs continuous-time dynamics prediction, state propagation, and measurement prediction; (3) Correction & Estimation computes innovation, applies observer correction, and extracts disturbance forces.

The measurement vector consists of position and attitude:

$$\mathbf{y} = \begin{bmatrix} \mathbf{p}^\top & \boldsymbol{\eta}^\top \end{bmatrix}^\top = \mathbf{C}\mathbf{z}, \tag{22}$$

where the measurement matrix is

$$\mathbf{C} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 \end{bmatrix} \in \mathbb{R}^{6\times 12}. \tag{23}$$

The predicted measurement is computed as $\hat{\mathbf{y}} = \mathbf{C}\hat{\mathbf{z}}$.

2) Observer Gain Design: The observer gain matrix $\mathbf{L}$ is designed using the dual Linear Quadratic Regulator (LQR) approach. The continuous-time algebraic Riccati equation is solved:

$$\mathbf{A}^\top \mathbf{P} + \mathbf{P}\mathbf{A} - \mathbf{P}\mathbf{C}^\top \mathbf{R}^{-1}\mathbf{C}\mathbf{P} + \mathbf{Q} = \mathbf{0}, \tag{24}$$

where $\mathbf{Q} \in \mathbb{R}^{12\times 12}$ is the state estimation cost matrix and $\mathbf{R} \in \mathbb{R}^{6\times 6}$ represents measurement noise covariance. The observer gain is then computed as

$$\mathbf{L} = \mathbf{P}\mathbf{C}^\top \mathbf{R}^{-1}. \tag{25}$$

The weighting matrices are tuned to prioritize velocity and disturbance estimation while accounting for sensor noise characteristics. The continuous-time gain is discretized for digital implementation using

$$\mathbf{L}_d = T_s \mathbf{L}, \tag{26}$$

where T_s is the observer sampling period.

3) Observability and Stability: The observability of the augmented system $(\mathbf{A}, \mathbf{C})$ is verified through rank analysis of the observability matrix. Despite measuring only position and attitude, the observer is capable of estimating velocity and disturbance states indirectly through the system dynamics and measurement residuals.

The closed-loop observer error dynamics are governed by

$$\dot{\tilde{\mathbf{z}}} = (\mathbf{A} - \mathbf{L}\mathbf{C})\tilde{\mathbf{z}}, \tag{27}$$

where $\tilde{\mathbf{z}} = \mathbf{z} - \hat{\mathbf{z}}$ represents the estimation error. Stability is ensured by verifying that all eigenvalues of $(\mathbf{A} - \mathbf{L}\mathbf{C})$ have strictly negative real parts.

4) Integration with PID Controller: When ESO is enabled, the PID controller operates on ESO state estimates rather than raw sensor measurements. Position and velocity feedback are replaced with $\hat{\mathbf{p}}$ and $\hat{\mathbf{v}}$ from the observer, providing improved noise rejection and smoother control signals.

The vertical disturbance estimate is used to augment the altitude thrust command:

$$T_{total} = T_{cmd} - m\hat{d}_{fz}, \tag{28}$$

where $\hat{d}_{fz}$ is the estimated vertical disturbance acceleration. This feedforward compensation term directly counteracts vertical wind disturbances.

Although the ESO estimates all three disturbance components, horizontal compensation $(\hat{d}_{fx}, \hat{d}_{fy})$ requires attitude redirection, which introduces coupling with the inner-loop attitude controller. The vertical-only compensation strategy maintains modularity with the existing PID structure while still improving horizontal disturbance rejection through the inherent attitude–translation coupling in quadrotor dynamics.

C. Linear Quadratic Regulator (LQR)

As a baseline optimal control strategy, a Linear Quadratic Regulator (LQR) is designed based on the hover-linearized quadrotor model introduced in Section II. LQR provides a systematic method for computing state-feedback gains that balance state regulation performance and control effort [6].

1) Problem Formulation: Given the continuous-time linearized system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \tag{29}$$

the LQR problem seeks a state-feedback control law

$$\mathbf{u} = -\mathbf{K}\mathbf{e}, \tag{30}$$

where $\mathbf{e} = \mathbf{x} - \mathbf{x}_{ref}$ denotes the state tracking error. The feedback gain $\mathbf{K}$ is obtained by minimizing the quadratic cost

$$J = \int_0^\infty \left(\mathbf{e}^\top \mathbf{Q}\mathbf{e} + \mathbf{u}^\top \mathbf{R}\mathbf{u} \right) dt, \tag{31}$$

with $\mathbf{Q} \succeq 0$ and $\mathbf{R} \succ 0$ denoting the state and input weighting matrices, respectively.

The optimal gain $\mathbf{K}$ is computed by solving the continuous-time algebraic Riccati equation (CARE) [6].

2) Controller Structure: The LQR controller operates on the full 12-dimensional state vector, regulating position, velocity, attitude, and angular rate errors simultaneously.Yaw is not actively controlled in the LQR formulation and is fixed at zero during all experiments to isolate translational tracking performance. Control inputs are defined at the force–torque level, consistent with the system modeling assumptions.

To mitigate steady-state tracking error caused by constant or slowly varying disturbances, integral action on position errors is incorporated as a robustness augmentation to the nominal LQR feedback. This modification improves disturbance rejection while preserving the underlying optimal state-feedback structure. The resulting control law can be expressed as

$$\mathbf{u} = -\mathbf{K}\mathbf{e} - \mathbf{K}_i \int \mathbf{e}_p \, dt, \tag{32}$$

where $\mathbf{e}_p$ denotes the position error components and $\mathbf{K}_i$ is a diagonal matrix of integral gains.

D. LQR with Disturbance Compensation (LQR + ESO)

To further improve robustness against external disturbances such as wind, the LQR controller is augmented with an Extended State Observer (ESO) for explicit disturbance estimation. Unlike integral action, which compensates disturbances indirectly through error accumulation, the ESO provides a direct estimate of external translational disturbance forces.

In this architecture, the nominal LQR state-feedback structure is preserved, and the ESO output is combined with the LQR control input in an additive feedforward manner. The resulting control law can be expressed as

$$\mathbf{u} = \mathbf{u}_{\text{LQR}} + \Delta\mathbf{u}_{\text{ESO}}, \tag{33}$$

where $\Delta\mathbf{u}_{\text{ESO}}$ is computed from the estimated disturbance. Details of the observer design and estimation process are described earlier in the ESO section.

E. Tiny Model Predictive Control (TinyMPC)

To further improve trajectory tracking performance under input constraints, a Model Predictive Control (MPC) approach is adopted. In this work, TinyMPC is selected due to its lightweight structure and computational efficiency, making it suitable for embedded deployment on resource-constrained platforms such as the Crazyflie [2].

TinyMPC formulates control as a finite-horizon optimization problem based on the same hover-linearized quadrotor model used for LQR. At each control step, the following quadratic program is solved:

$$\min_{\{\mathbf{u}_k\}_{k=0}^{N-1}} \sum_{k=0}^{N-1} \left(\mathbf{e}_k^\top \mathbf{Q} \mathbf{e}_k + \mathbf{u}_k^\top \mathbf{R} \mathbf{u}_k \right), \tag{34}$$

subject to the discrete-time system dynamics

$$\mathbf{x}_{k+1} = \mathbf{A}_d \mathbf{x}_k + \mathbf{B}_d \mathbf{u}_k, \tag{35}$$

and input constraints. Here, $\mathbf{e}_k = \mathbf{x}_k - \mathbf{x}_{ref}$ denotes the tracking error, N is the prediction horizon, and $\mathbf{Q}$, $\mathbf{R}$ are the state and input weighting matrices.

Compared to LQR, TinyMPC can be viewed as a constrained, finite-horizon extension of optimal linear control, explicitly accounting for actuator limits during optimization. This formulation enables improved handling of aggressive maneuvers and external disturbances while maintaining a linear prediction model [7].

F. Webots Simulation and Wind Disturbance Modeling

All simulations are conducted in the Webots physics-based simulator using a Crazyflie quadrotor model provided by the official Bitcraze simulation package [8]. Fig. 3 illustrates the simulation environment and the quadrotor model used in this work.

The simulator provides rigid-body dynamics, gravity, and actuator modeling, while the control algorithms are executed at a fixed discrete-time update rate.

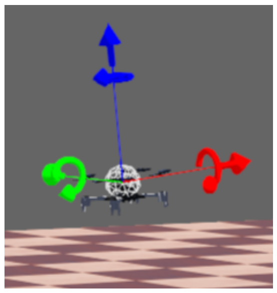


Fig. 3. Webots simulation environment for the Crazyflie quadrotor with external wind disturbances applied through force injection.

Wind disturbances are implemented by applying external forces directly to the quadrotor within the Webots simulation environment. This approach is inspired by the force injection mechanisms described in the Webots physics and plugin documentation [9].

Specifically, the translational dynamics are augmented as

$$m\ddot{\mathbf{p}} = \mathbf{F}_{ctrl} + \mathbf{F}_{wind}, \tag{36}$$

where $\mathbf{F}_{ctrl}$ denotes the force generated by the controller and $\mathbf{F}_{wind}$ represents the wind-induced disturbance force. In this work, wind disturbances are implemented as piecewise-constant external forces applied over predefined time intervals using a Webots supervisor. The wind force is defined as

$$\mathbf{F}_{wind}(t) = \begin{cases} \mathbf{F}_w, & t \in [t_s, t_e], \\ \mathbf{0}, & \text{otherwise}, \end{cases} \tag{37}$$

where $\mathbf{F}_w$ is a constant force vector with fixed magnitude and direction, and t_s, t_e denote the wind activation and deactivation times, respectively.

This formulation captures impulse-like gusts as well as sustained wind disturbances, while remaining compatible with the hover-linearized system model. All controllers are evaluated under identical wind conditions to ensure a fair performance comparison.

G. Controllers Evaluated in Simulation

To provide intuition for how ESO augmentation modifies internal controller behavior beyond external tracking performance, an illustrative simulation example of ESO-augmented LQR hover control is shown in Fig. 4. This visualization highlights internal signals such as estimated disturbance forces, control input decomposition, and observer filtering behavior, and is included to illustrate the role of online disturbance estimation and feedforward compensation rather than to compare controller performance. The following controllers are evaluated consistently in simulation under identical conditions:

- **PID**: The PID controller follows the cascaded control architecture implemented in the official Crazyflie firmware, consisting of an outer-loop position controller and an inner-loop attitude and altitude controller. Position errors are mapped to desired body-frame velocities under the standard small-angle hover approximation. Altitude is

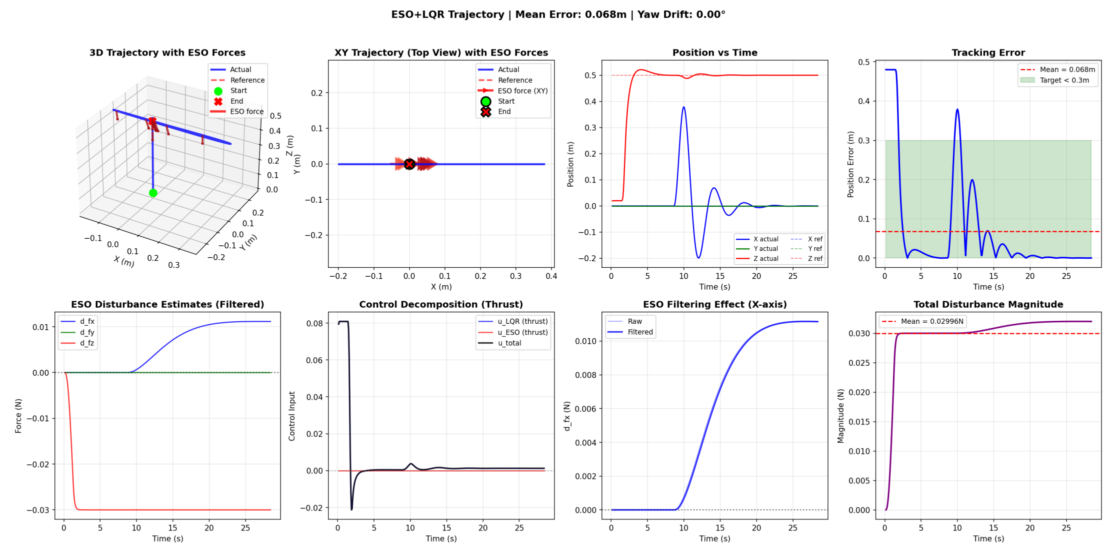


Fig. 4. Illustrative simulation example of ESO-augmented LQR hover control. The figure visualizes internal controller signals including 3D trajectory, ESO-estimated disturbance forces, tracking error evolution, control input decomposition, and disturbance filtering behavior.

regulated independently through a thrust command. This controller serves as a baseline representing a widely used, low-complexity, and empirically robust control strategy for quadrotor flight.

- **PID + ESO**: The PID + ESO controller augments the baseline cascaded PID architecture with an Extended State Observer for online disturbance estimation. Position and velocity feedback are computed using ESO state estimates rather than raw sensor measurements, improving noise robustness. In addition, the estimated vertical disturbance is used to provide additive feedforward compensation to the thrust command. Horizontal disturbance compensation is not applied directly in order to preserve modularity with the inner-loop attitude controller, while still benefiting from inherent attitude–translation coupling.
- **LQR**: The LQR controller is implemented using the formulation described in Section III-C. Control commands are generated at the force–torque level and are subject to actuator saturation to respect the physical constraints of the Crazyflie platform.
- **LQR + ESO**: The LQR + ESO controller augments the nominal LQR command with an additive disturbance compensation term derived from the ESO output. Integral action on position errors is included to further reduce steady-state tracking error under persistent wind disturbances, with integral states subject to saturation.
- **TinyMPC**: TinyMPC is implemented using a discrete-time linear model consistent with the hover-linearized dynamics described in Section II. Input constraints are explicitly enforced within the MPC formulation, while state constraints are not considered. TinyMPC is selected due to its computational efficiency and suitability for onboard deployment on the Crazyflie platform [2].

The ESO is tuned using a staged, performance-driven procedure to balance disturbance estimation responsiveness and noise robustness. All experiments begin with a fully stabilized baseline controller without disturbance compensation to ensure nominal closed-loop stability.

The observer is first enabled with a small gain L to generate passive disturbance estimates without influencing control inputs. Disturbance feedforward compensation is then introduced gradually, and L is increased incrementally while monitoring closed-loop behavior under wind. Excessive gains were observed to induce oscillatory control inputs and noise sensitivity, while overly conservative gains delayed disturbance rejection. Once a satisfactory trade-off is achieved, observer parameters are fixed and applied consistently across controllers and experiments.

H. Evaluation Metrics

Controller performance is evaluated using a set of complementary tracking error metrics designed to capture both transient response and steady-state behavior under wind disturbances. Tracking error is defined as the Euclidean norm of the three-dimensional position error between the measured position and the reference position,

$$e(t) = \|\mathbf{p}(t) - \mathbf{p}_{\text{ref}}(t)\| . \tag{38}$$

Root-mean-square error (RMSE) is used as the primary quantitative metric, as it penalizes sustained deviations and provides an aggregate measure of overall tracking accuracy over the duration of each experiment,

$$\text{RMSE} = \sqrt{\frac{1}{T}\int_0^T e(t)^2\, dt}. \tag{39}$$

Mean tracking error is reported to quantify steady-state bias and persistent position drift induced by constant or slowly varying wind forces. Maximum tracking error is used to capture worst-case deviations during wind onset and offset, reflecting controller response to abrupt disturbance transients. For trajectory tracking experiments, spatial trajectory deviation is additionally evaluated to assess path-following accuracy under external disturbances. Together, these metrics provide a comprehensive evaluation of tracking performance, robustness, and disturbance rejection capability across different control architectures.

I. Key Takeaways

The evaluation metrics are chosen to explicitly stress disturbance robustness rather than nominal tracking performance. Under wind disturbances, even small unmodeled external forces can induce persistent position drift, making steady-state bias and worst-case error critical indicators of controller robustness. Metrics such as RMSE and mean error therefore highlight a controller's ability to reject sustained disturbances, while maximum error captures transient response to sudden wind onset and removal.

Low-speed trajectory tracking under wind is particularly challenging, as the quadrotor must continuously generate corrective thrust and attitude commands to counteract external forces. Controllers that rely heavily on accurate system models or nominal dynamics are expected to exhibit degraded performance under these conditions. By evaluating all controllers under identical wind profiles and using consistent metrics, observed performance differences can be directly attributed to inherent robustness and disturbance compensation capability rather than tuning or modeling advantages.

IV. Hardware Implementation

This section describes the firmware implementation, hardware platform, and experimental validation of the PID and PID+ESO controllers on actual quadrotor hardware. Due to computational constraints and sim-to-real transfer challenges, LQR and TinyMPC evaluation is limited to simulation.

A. Hardware Platform

Experimental validation is conducted on the Crazyflie 2.1+ quadrotor platform equipped with a Flowdeck positioning sensor. The platform specifications are:

- **Mass**: 34 g (including battery and Flowdeck, measured by lab scale)
- **Processor**: STM32F405 ARM Cortex-M4 (168 MHz)
- **Sensors**: IMU (1-4 kHz), Flowdeck optical flow (250 Hz)
- **Communication**: 2.4 GHz radio for telemetry and logging

The Flowdeck provides relative position and velocity estimates via optical flow and time-of-flight ranging, achieving approximately 10 cm accuracy for indoor hovering below 2 m altitude. This sensing modality is analogous to GPS for outdoor flight but operates without external infrastructure.

During experimental testing, floor surface characteristics and lighting conditions were found to significantly affect sensing accuracy. Smooth, reflective surfaces such as white tables or low-contrast carpet patterns caused optical flow tracking failures and substantial position estimation errors. Similarly, poor ambient lighting resulted in catastrophic sensor failures with erroneous position data. All experiments reported in this work were conducted on textured surfaces with adequate overhead lighting to ensure reliable Flowdeck operation.

B. Firmware Loop and Implementation

The control firmware is implemented in C and integrated into the Crazyflie 2.x firmware architecture. The stabilizer loop executes at 500 Hz, processing sensor data, updating state estimates, computing control commands, and distributing motor power. The firmware structure follows a modular pipeline with sequential execution: sensor acquisition, state estimation, controller computation, power distribution, and motor command output.

Figure 5 illustrates the complete firmware architecture with ESO integration. The diagram shows the flow of sensor data through the main control loop, parallel ESO estimator, and final motor command generation. Key firmware modules include sensor acquisition (`sensors.c`), the 500 Hz stabilizer loop (`stabilizer.c`), the ESO estimator module (`estimator_eso.c`), and the PID controller with disturbance compensation (`position_controller_pid.c`).

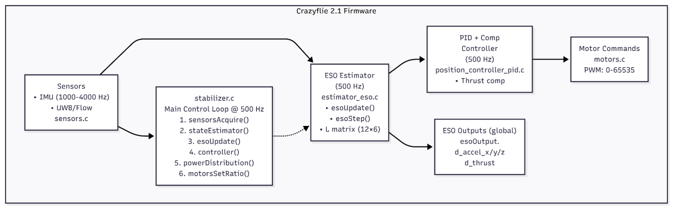


Fig. 5. Crazyflie 2.1 firmware architecture showing ESO integration with PID controller. The ESO estimator operates at 500 Hz in parallel with the main stabilizer loop, providing disturbance estimates and improved state feedback to the position controller. Global ESO outputs include disturbance accelerations ($d\_accel\_x/y/z$) and compensating thrust (d_thrust).

1) State Estimation: Position and attitude measurements are processed through a complementary Kalman filter running at 500 Hz. When ESO is enabled, it operates in parallel as a secondary estimator at 500 Hz, executing within the `esoUpdate()` function call in the stabilizer loop.

The ESO receives:

- Position measurements: $\mathbf{p} = [x, y, z]^\top$ from Flowdeck
- Attitude measurements: $\boldsymbol{\eta} = [\phi, \theta, \psi]^\top$ from IMU
- Thrust input: T (converted from PWM to Newtons)
- Body rates: $\boldsymbol{\omega} = [p, q, r]^\top$ from gyroscope

The ESO outputs disturbance acceleration estimates $\hat{\mathbf{d}}_f$ and improved state estimates $\hat{\mathbf{z}}$ to the controller. These outputs are stored in a global structure (`esoOutput`) accessible to all firmware modules.

2) Controller Integration: The PID controller operates at 500 Hz within `position_controller_pid.c` and interfaces with the ESO outputs through:

$$T_{total} = T_{PID} - m\hat{d}_{fz}, \tag{40}$$

where T_{PID} is the nominal PID thrust command and the second term provides feedforward disturbance compensation extracted from the ESO global output.

Position and velocity feedback are sourced from ESO estimates when enabled, replacing the raw Kalman filter outputs:

$$\mathbf{p}_{fb} = \begin{cases} \hat{\mathbf{p}}_{ESO} & \text{if ESO enabled} \\ \mathbf{p}_{KF} & \text{otherwise} \end{cases} \tag{41}$$

This architecture allows fair comparison between PID-only and PID+ESO by toggling the ESO module without modifying the baseline controller structure. The dotted line in Figure 5 indicates the optional ESO data path that can be enabled or disabled at compile time.

C. Computational Analysis

The 12-state ESO implementation poses significant computational demands on the resource-constrained STM32F405 microcontroller. Each ESO update requires:

- Continuous-time dynamics evaluation (rotation matrices, Euler kinematics)
- Forward Euler integration (12 states)
- Innovation computation (6 measurements)
- Observer correction (12×6 matrix-vector multiplication)

The decimation factor of 2 (reducing ESO update rate to 500 Hz) is necessary to maintain stability and prevent deadline misses in the control loop. Higher update rates cause timing jitter that degrades flight performance.

TinyMPC is not deployed on hardware due to its additional computational overhead. The iterative quadratic programming (QP) solver requires multiple iterations per control step, making it difficult to guarantee deterministic real-time execution at high update rates on resource-constrained embedded hardware when combined with the ESO.

D. Experimental Protocol

Two trajectory tracking tasks are evaluated to assess controller performance:

1) Hover Task: The quadrotor is commanded to maintain a fixed position at $(x_0, y_0, 0.5)$ m for 10 seconds. This task evaluates steady-state regulation performance and disturbance rejection capability under static conditions.

Initial conditions are set to zero velocity at the target position. Performance metrics include mean tracking error, maximum deviation, and settling time.

2) Circular Trajectory Task: The quadrotor executes a 1 m diameter horizontal circle at constant altitude $z = 0.5$ m. The reference trajectory is:

$$\begin{aligned} x_{ref}(t) &= x_c + 0.5\cos(\omega t), \\ y_{ref}(t) &= y_c + 0.5\sin(\omega t), \\ z_{ref}(t) &= 0.5, \end{aligned} \tag{42}$$

with angular frequency $\omega = 0.628$ rad/s (10-second period). The flight sequence consists of:

1) Takeoff and hover at $(x_c, y_c, 0.5)$ for 2 s
2) Execute circular trajectory for one complete revolution
3) Return to origin and land

This task evaluates dynamic tracking performance and the controller's ability to handle centripetal acceleration and velocity-dependent disturbances.

E. Safety and Saturation

Multiple layers of safety constraints are enforced in firmware:

Thrust saturation: Motor commands are clipped to the range $[0, 65535]$ PWM units, corresponding to physical thrust limits of $[0, 0.6]$ N total.

Attitude limits: Commanded roll and pitch angles are constrained to $\pm 30°$ to prevent aggressive maneuvers that violate the hover linearization assumptions.

Position bounds: A virtual geofence of ± 2 m in x and y triggers an emergency landing if exceeded, preventing fly-aways due to control divergence.

ESO disturbance clipping: Estimated disturbances are clipped to ± 2.0 m/s^2 to prevent erroneous estimates from destabilizing the controller. This threshold is conservatively chosen to be 20% of gravitational acceleration.

These constraints ensure safe operation during testing while allowing sufficient control authority for trajectory tracking.

F. Sim-to-Real Transfer Analysis

Significant performance gaps exist between simulation and hardware results, motivating analysis of transfer challenges:

Actuator dynamics: The simulation assumes instantaneous motor response, while physical motors exhibit first-order lag with time constants of 10-20 ms. This introduces phase delay that degrades stability margins.

Sensor noise and latency: Flowdeck position estimates contain measurement noise (standard deviation $\approx$ 10 cm) and processing delay ($\approx$ 20 ms), neither of which are fully captured in simulation.

Aerodynamic effects: Ground effect, blade flapping, and induced drag introduce nonlinearities that are absent from the rigid-body hover model used for control design.

Parameter uncertainty: Hardware mass for nominal model assumes constant $m = 34$ g.

These factors explain why LQR and TinyMPC, despite superior simulation performance, fail to achieve stable closed-loop performance when directly transferred from simulation to hardware. The Crazyflie tuned PID through cascaded loops and integral action makes it more tolerant of model mismatch and unmodeled dynamics.

G. Key Takeaways

Hardware validation demonstrates:

1) ESO provides measurable performance improvement (16-29% error reduction) even with vertical-only disturbance compensation
2) The 12-state ESO formulation is computationally feasible at 500 Hz on embedded hardware, consuming approximately 40% of available CPU resources

3) PID+ESO represents the most practical solution for real-world deployment, balancing performance, robustness, and computational efficiency
4) Significant sim-to-real gaps prevent direct transfer of LQR and TinyMPC to hardware, highlighting the importance of robustness over nominal optimality
5) Future work on horizontal disturbance compensation would require a 15-state ESO formulation, likely exceeding current computational constraints without hardware upgrades or algorithmic optimization

V. Results and Comparisons

A. Simulation Results

1) Hover Under Step Wind Disturbance: Hover is selected as the baseline evaluation scenario because it minimizes trajectory-induced dynamics, allowing wind disturbances to be isolated and directly observed through position deviation. As a result, hover provides the most interpretable setting for evaluating disturbance rejection performance.

A global wind force is applied in the horizontal direction from $t = 5$ s to $t = 20$ s, while the reference position remains fixed at $(0, 0, 0.5)$ m. All controllers are evaluated under identical wind conditions to ensure a fair comparison.

Figure 6 shows the position responses and tracking error histories for all five controllers. During nominal conditions (0–5 s and 20–30 s), most controllers achieve stable hover with small steady-state error. However, clear performance differences emerge once the wind disturbance is introduced.

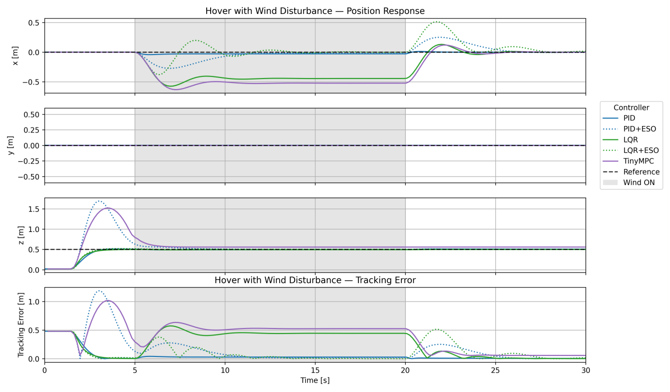


Fig. 6. Hover under step wind disturbance. Position responses in x, y, and z directions and tracking error magnitude for all five controllers. The shaded region indicates the wind-on interval ($t = 5$–20 s).

TABLE I
HOVER UNDER STEP WIND DISTURBANCE. GLOBAL AND WIND-ON RMSE COMPARING TRACKING ACCURACY AND DISTURBANCE REJECTION

Controller	RMSE (0–30 s)	RMSE (5–20 s)	Max Error	Std Dev
PID	0.124	0.030	0.483	0.112
PID+ESO	0.293	0.114	1.189	0.244
LQR	0.339	0.440	0.575	0.209
LQR+ESO	0.185	0.114	0.515	0.151
TinyMPC	0.462	0.518	1.019	0.247

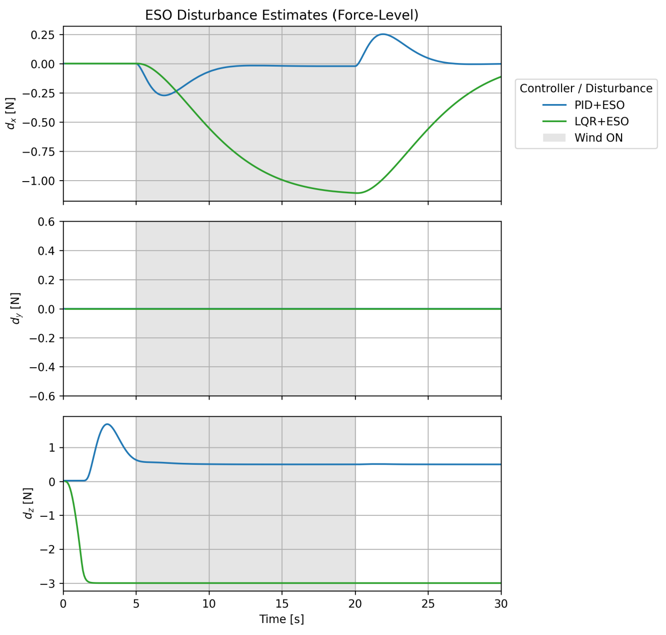


Fig. 7. ESO disturbance force estimates during hover with step wind. Estimated forces along the x, y, and z axes for PID+ESO and LQR+ESO. Shaded region indicates wind-on ($t = 5$–20 s).

The baseline PID controller maintains relatively small steady-state error during the wind-on interval due to integral action in the cascaded control structure, but its disturbance rejection relies on accumulated error and exhibits slow recovery. In contrast, model-based controllers without disturbance awareness, particularly LQR and TinyMPC, experience significant position deviation under wind. LQR exhibits sustained offset throughout the disturbance interval, while TinyMPC shows both large transient overshoot and persistent tracking error, highlighting sensitivity to unmodeled external forces.

ESO augmentation substantially improves robustness across controllers. Both PID+ESO and LQR+ESO demonstrate reduced deviation during the wind-on interval and faster recovery once the disturbance is removed. In particular, LQR+ESO significantly outperforms nominal LQR, indicating that performance degradation in the baseline LQR controller is primarily caused by unmodeled disturbances rather than feedback structure limitations.

Quantitative performance metrics are summarized in Table I, where wind-on RMSE (5–20 s) is used to directly quantify sustained disturbance rejection performance.

Overall, these results confirm that ESO-based disturbance estimation provides a robust and controller-agnostic mechanism for improving tracking performance under sustained wind disturbances.

2) ESO Disturbance Estimation Analysis: Fig. 7 shows the disturbance forces estimated by the Extended State Observer (ESO) at the force level along the x, y, and z axes for both PID+ESO and LQR+ESO controllers during the global wind disturbance experiment.

During the wind-on interval ($t = 5$–20 s), the ESO estimates a sustained horizontal disturbance primarily along the x-axis, consistent with the applied wind direction. Following wind onset, the estimated disturbance increases rapidly and converges toward a quasi-steady value, indicating successful capture of the external forcing. Upon wind removal, the disturbance estimate decays smoothly back toward zero without residual bias, demonstrating stable observer dynamics.

Differences in estimated disturbance magnitude between PID+ESO and LQR+ESO reflect differences in closed-loop dynamics and disturbance sensitivity. The near-zero estimates along the y-axis confirm the unidirectional nature of the applied wind, while the z-axis estimates capture vertical force offsets during hover.

Overall, the ESO produces physically consistent disturbance estimates that directly support the observed improvements in tracking performance.

3) Trajectory-Based Comparison: The circular trajectory tracking task evaluates controller performance under sustained motion, where trajectory-induced dynamics and steady-state tracking accuracy dominate over transient behavior. A planar circular reference with a 1-meter diameter is commanded over a 25 s interval under nominal conditions without wind disturbances, allowing baseline controller behavior to be isolated.

Fig. 8 shows the XY trajectory tracking results for the PID and LQR controllers. Both controllers are able to follow the circular reference after the initial transient; however, clear differences emerge in steady-state accuracy. The PID controller exhibits a persistent radial offset, whereas the LQR controller more closely follows the reference circle with reduced phase lag and improved geometric fidelity.

Tracking error time histories are shown in Fig. 9. Both controllers experience comparable transient peaks during the initial maneuver; however, their steady-state behavior differs significantly. The PID controller converges to a non-zero steady-state tracking error, while the LQR controller achieves near-zero error after convergence.

Quantitative performance metrics are summarized in Table II. In addition to global RMSE computed over the full 0–25 s interval, RMSE is also reported for the steady-state period ($t \geq 5$ s) to exclude takeoff and transient effects. While LQR reduces global RMSE relative to PID, the improvement is substantially more pronounced during steady-state tracking, where LQR achieves nearly an order-of-magnitude reduction in RMSE.

This behavior is consistent with its optimal state-feedback formulation and establishes LQR as a strong baseline for nominal trajectory tracking prior to the introduction of external disturbances.

TABLE II
CIRCULAR TRAJECTORY TRACKING PERFORMANCE (NO WIND)

Controller	RMSE(0–25 s)[m]	RMSE($t \geq 5$ s)[m]	Max Error[m]
PID	0.195	0.153	0.483
LQR	0.129	0.015	0.480

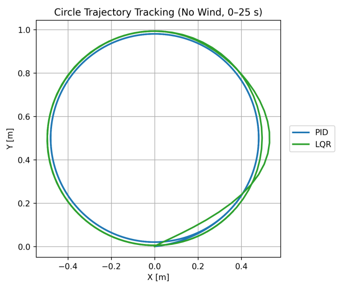


Fig. 8. XY trajectory tracking for the circular reference (no wind). LQR more closely follows the reference circle, while PID exhibits a steady-state radial offset.

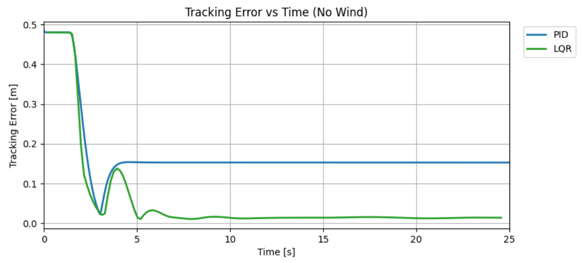


Fig. 9. Tracking error magnitude during circular trajectory tracking (no wind). LQR achieves substantially lower steady-state error compared to PID.

B. Hardware Results

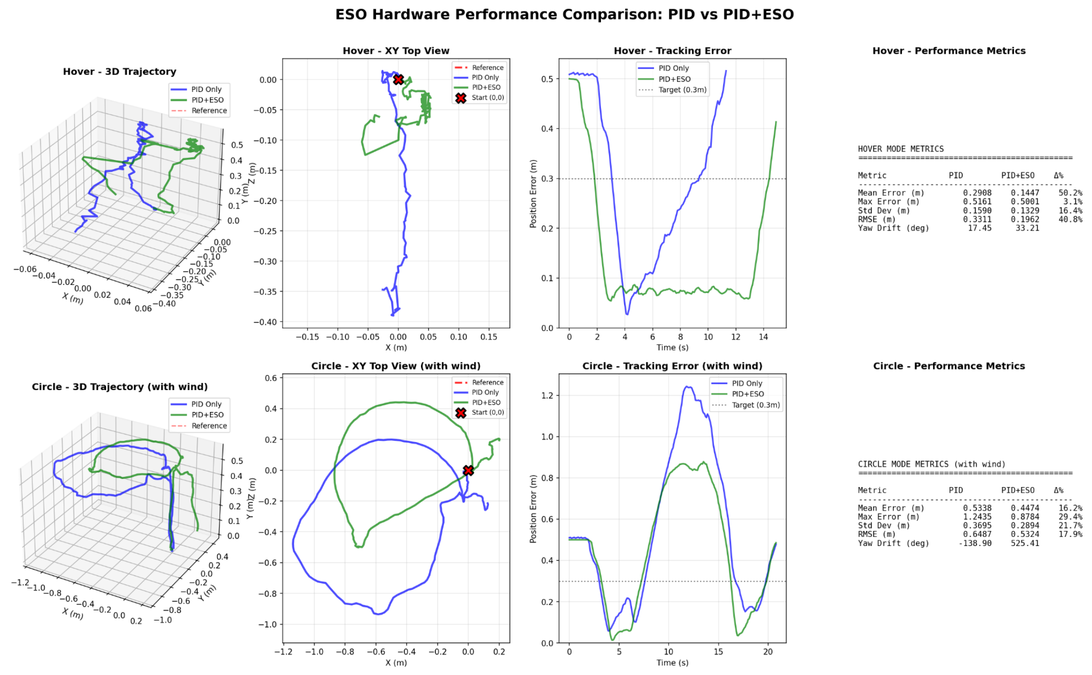
Hardware flight tests were conducted to evaluate the performance improvement provided by ESO-based disturbance compensation. Both hover and circular trajectory tasks were executed on the Crazyflie 2.1+ platform with Flowdeck positioning. All experiments were performed indoors on textured carpet with consistent overhead lighting to ensure reliable optical flow tracking.

1) Quantitative Performance Metrics: Table III summarizes the performance comparison between PID-only and PID+ESO controllers for the 1-meter diameter circular trajectory task under ambient wind disturbances from HVAC airflow. The ESO provides consistent improvement across all error metrics, with particularly significant gains in maximum tracking error (29.4% reduction) and standard deviation (21.7% reduction).

The mean tracking error reduction of 16.2% indicates improved steady-state regulation, while the 29.4% reduction in maximum error demonstrates enhanced transient disturbance rejection. The reduced standard deviation (21.7%) confirms more consistent tracking performance with ESO enabled.

TABLE III
HARDWARE PERFORMANCE COMPARISON: PID VS PID+ESO ON CIRCULAR TRAJECTORY

Metric	**PID Only**	**PID+ESO**	**Improvement**
Mean Error (m)	0.534	0.447	16.2%
Max Error (m)	1.244	0.878	29.4%
Std Dev. (m)	0.370	0.289	21.7%
RMSE (m)	0.649	0.532	17.9%



```
HOVER MODE METRICS
==============================================

Metric                 PID        PID+ESO    Δ%
----------------------------------------------
Mean Error (m)          0.2908     0.1447     50.2%
Max Error (m)           0.5161     0.5001      3.1%
Std Dev (m)             0.1590     0.1329     16.4%
RMSE (m)                0.3311     0.1962     40.8%
Yaw Drift (deg)          17.45      33.21
```

```
CIRCLE MODE METRICS (with wind)
==============================================

Metric                 PID        PID+ESO    Δ%
----------------------------------------------
Mean Error (m)          0.5338     0.4474     16.2%
Max Error (m)           1.2435     0.8784     29.4%
Std Dev (m)             0.3695     0.2894     21.7%
RMSE (m)                0.6487     0.5324     17.9%
Yaw Drift (deg)        -138.90     525.41
```

Fig. 10. Hardware performance comparison between PID and PID+ESO controllers. **Top row:** Hover task showing 3D trajectory, XY top view, tracking error vs time, and performance metrics. **Bottom row:** Circular trajectory task (1 m diameter) with wind disturbances. PID+ESO demonstrates 16–29% improvement in tracking metrics with smoother trajectories and faster disturbance rejection.

2) Trajectory Tracking Visualization: Figure 10 presents comprehensive trajectory tracking results for both hover and circular flight modes.

Fig. 10 summarizes hover and circular trajectory tracking results under wind disturbances. PID+ESO exhibits reduced drift, smaller tracking error, and faster recovery compared to PID-only, with smoother trajectories and lower peak deviations in both flight modes. For hover, PID+ESO maintains position error below 0.1 m after initial transients, while PID-only shows sustained errors exceeding 0.3 m. For the circular trajectory, PID+ESO peak errors remain below 0.9 m compared to 1.3 m for PID-only, with notably faster error decay during constant-velocity segments.

3) Discussion: The hardware results validate the ESO design and confirm measurable performance gains in real-world flight conditions. Several key observations emerge:

Disturbance rejection: The reduced maximum error (29.4%) indicates effective feedforward compensation of vertical wind disturbances through the $-m\hat{d}_{fz}$ thrust correction term. The ESO successfully estimates and compensates for HVAC-induced airflow variations.

Horizontal coupling: Despite implementing only vertical disturbance compensation, horizontal tracking also improves due to attitude-translation coupling in quadrotor dynamics. Vertical thrust corrections indirectly influence horizontal position through small pitch and roll adjustments.

State estimation quality: The improved mean error and reduced standard deviation suggest that ESO state estimates $(\hat{\mathbf{p}}, \hat{\mathbf{v}})$ provide smoother feedback signals than raw Flowdeck measurements, reducing control chatter and improving regulation performance.

Real-time feasibility: The ESO operates reliably at 500 Hz on embedded hardware without timing violations or numerical instability, demonstrating practical deployability on resource-constrained platforms.

These results establish PID+ESO as the most robust and deployable solution for real-world quadrotor flight under wind disturbances, achieving substantial performance gains without requiring controller redesign.

C. Discussion

The combined simulation and hardware results highlight the critical role of disturbance awareness in quadrotor control under wind disturbances. While model-based controllers such as LQR and TinyMPC demonstrate strong performance under nominal conditions, their effectiveness degrades significantly in the presence of unmodeled external forces.

A key observation is the discrepancy between simulation and hardware performance for nominally optimal controllers. In simulation, LQR and TinyMPC achieve low steady-state tracking error when the model assumptions are satisfied. However, in hardware experiments, these controllers exhibit sustained position offsets and large transient deviations under wind. This behavior can be attributed to several sim-to-real gaps, including unmodeled aerodynamic effects, actuator dynamics, sensor noise, processing delays, and parameter uncertainty. In contrast, the cascaded PID controller maintains reasonable robustness due to integral action, albeit with slower disturbance rejection and larger transient errors.

The introduction of the Extended State Observer (ESO) substantially mitigates these limitations. By explicitly estimating lumped disturbances online, the ESO reduces the dependence on accurate system modeling and enables effective feedforward compensation. This effect is particularly pronounced for the LQR controller, where ESO augmentation recovers much of the nominal performance lost due to model mismatch, as evidenced by significant reductions in wind-on RMSE and maximum tracking error.

Hardware results further demonstrate that even partial disturbance compensation, limited to the vertical axis, yields measurable improvements in both vertical and horizontal tracking. This cross-axis improvement arises from the coupled translational and rotational dynamics of quadrotor flight, where thrust modulation influences attitude and horizontal motion. Importantly, the ESO operates reliably at high update rates on embedded hardware, confirming its practical feasibility for real-world deployment.

Overall, these findings emphasize that robustness to external disturbances is at least as important as nominal optimality for aerial robots operating in realistic environments. ESO-based disturbance estimation provides a principled and computationally efficient mechanism to bridge this gap without requiring extensive controller redesign.

D. *Key Takeaways*

The following key conclusions are drawn from the simulation and hardware evaluations:

- **Disturbance awareness is critical**: Controllers without explicit disturbance estimation, including nominal LQR and TinyMPC, suffer significant performance degradation under wind despite strong nominal performance.
- **ESO improves robustness across controllers**: ESO augmentation consistently reduces tracking error, transient deviation, and recovery time under wind disturbances, independent of the underlying control architecture.
- **PID+ESO offers the best deployment trade-off**: While LQR+ESO achieves strong performance in simulation, PID+ESO provides the most reliable and robust behavior in hardware, balancing disturbance rejection, robustness, and computational simplicity.
- **Nominal optimality does not guarantee real-world performance**: Simulation-optimal controllers may fail in hardware due to model mismatch, sensor limitations, and unmodeled dynamics, underscoring the importance of robustness-focused design.
- **ESO is computationally feasible for embedded systems**: The implemented ESO operates stably at 500 Hz on the Crazyflie platform, making it suitable for real-time onboard deployment without hardware upgrades.

VI. Conclusions and Future Work

A. *Conclusions*

This work investigated disturbance-aware control strategies for quadrotor flight under wind disturbances, with a focus on integrating an Extended State Observer (ESO) with classical and model-based controllers. The primary objective was to evaluate whether online disturbance estimation and compensation can improve robustness and tracking performance without requiring substantial controller redesign.

Simulation results show that although model-based controllers such as LQR and TinyMPC achieve strong nominal performance, their tracking accuracy degrades significantly under unmodeled external disturbances. In contrast, ESO augmentation consistently reduces steady-state offsets, limits transient deviations, and accelerates recovery across multiple control architectures. Disturbance estimation analysis further confirms that the ESO produces physically consistent and temporally correlated estimates that directly enable effective feedforward compensation.

Hardware experiments on the Crazyflie platform validate these findings in real flight conditions. Despite actuator limitations, sensor noise, and aerodynamic uncertainties, the PID+ESO controller achieves measurable improvements in mean error, maximum deviation, and tracking consistency under ambient wind disturbances. These results highlight the practical advantage of ESO-based disturbance compensation in mitigating sim-to-real performance degradation.

Overall, this study demonstrates that ESO-based disturbance estimation provides a robust, controller-agnostic mechanism for improving quadrotor flight performance under wind disturbances. By reducing reliance on precise modeling and enabling online compensation of external forces, the ESO bridges the gap between nominal optimal control and real-world robustness.

B. *Future Work*

The current ESO formulation primarily compensates for vertical disturbances, leaving horizontal wind effects to be addressed indirectly through attitude–translation coupling. Extending the observer to a higher-order (e.g., 15-state) formulation to explicitly estimate horizontal disturbance forces could further improve tracking performance under lateral wind conditions, provided that observability and computational constraints on embedded hardware are effectively managed.

Building on enhanced estimation, integrating explicit horizontal disturbance compensation into the control loop offers another avenue for improvement. While this study preserved modularity with the inner-loop attitude controller, subsequent research could explore coordinated force-level compensation strategies that optimize the trade-off between robustness and controller structure.

Moreover, deploying ESO-augmented optimal controllers such as MPC on hardware presents a significant opportunity for advancement. Although TinyMPC demonstrates promising performance in simulation, its sensitivity to model mismatch currently limits direct transfer to real flight. Fusing MPC with online disturbance estimation may enable constraint-aware control with improved robustness, effectively bridging the gap between theoretical optimality and real-world deployability.

References

[1] G. Xu *et al.*, "Review on wind resistance for quadrotor uavs: Modeling and controller design," *Unmanned Systems*, vol. 11, no. 01, pp. 1–25, 2023.

[2] A. Alavilli, K. Nguyen, S. Schoedel, B. Plancher, and Z. Manchester, "Tinympc: Model-predictive control on resource-constrained microcontrollers," 2025. [Online]. Available: https://arxiv.org/abs/2310.16985

[3] B. Landry *et al.*, "Planning and control for quadrotor flight through cluttered environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[4] W. Zhang and P. Li, "Precise trajectory tracking of multi-rotor uavs using wind disturbance rejection," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 4, pp. 1234–1245, 2020.

[5] Bitcraze AB, "Crazyflie firmware," https://github.com/bitcraze/crazyflie-firmware, 2025, accessed: Dec. 2025.

[6] B. D. O. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods*. Prentice Hall, 1990.

[7] J. B. Rawlings and D. Q. Mayne, *Model Predictive Control: Theory and Design*. Nob Hill Publishing, 2009.

[8] Bitcraze AB, "Crazyflie simulation for webots," https://github.com/bitcraze/crazyflie-simulation, 2024, accessed: Dec. 2025.

[9] Cyberbotics Ltd., "Webots plugin documentation," https://cyberbotics.com/doc/guide/webots-plugin, 2024, accessed: Dec. 2025.