Quaternion-based Attitude Sliding Mode Control and State Estimation for Aggressive Maneuver Stabilization of a Minidrone

Khanh Ky Do
École polytechnique fédérale de Lausanne

Antoine Carrel
École polytechnique fédérale de Lausanne

Aloïs Millot
École polytechnique fédérale de Lausanne

***Abstract*—This project implements a Sliding Mode Control strategy for attitude and altitude stabilization, as well as trajectory tracking, of a Parrot Minidrone in Simulink. Extended Kalman Filters are used to provide accurate state estimation from the sensor data. Using quaternion-based representation, the controller avoids singularities at high angles and ensures accurate 3D orientations. A nonlinear algorithm enhances robustness against model uncertainties and disturbances. A set of simulation results shows adequate orientation stabilization in uncertain environments, validating the Sliding Mode Control approach.**

***Index Terms*—Sliding Mode Control, Extended Kalman Filter, Quaternion, Parrot Minidrone, Simulink, MATLAB.**

Link

Project video : https://youtu.be/z0z447DR1LI.

I. Introduction

Unmanned Aerial Vehicles are increasingly used in fields like search and rescue, exploration, and robotics competitions, demanding agile control systems capable of managing aggressive maneuvers and rapid dynamic changes. This project explores the model-based design of advanced nonlinear control and estimation strategies for a minidrone using Simulink, aiming to achieve stabilization from harsh initial conditions.

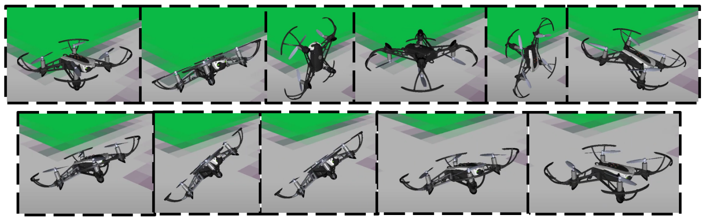


Fig. 1. Minidrone flight sequence of a backflip in Simulation Environment.

To meet these objectives, we implemented Sliding Mode Controllers (SMCs) managing quaternion-based attitude control, altitude stabilization based on [2] and [7], and Extended Kalman Filters (EKFs) estimating both attitude and altitude to enhance robustness. The full system is developed and validated in a high-fidelity simulation based on the Parrot Minidrone Hover Model, demonstrating the strength of sliding mode control and estimation in agile flights.

II. Minidrone dynamics

A. Equations of motions

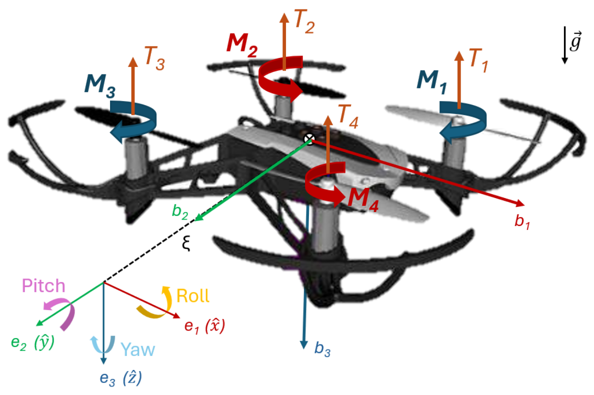


Fig. 2. Parrot Mambo minidrone model and reference frames.

Knowing drone´s moment of inertia tensor $\mathbf{J}$ and its mass m, the high-level drone dynamics are modeled as follows:

$$\begin{aligned} \dot{\boldsymbol{\xi}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= g\mathbf{e}_3 + \frac{1}{m}\mathbf{q}^* \otimes f\mathbf{e}_3 \otimes \mathbf{q} \\ \dot{\mathbf{q}} &= \frac{1}{2}\mathbf{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \\ \dot{\boldsymbol{\omega}} &= \mathbf{J}^{-1}\left(-\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} + \boldsymbol{\tau}\right) \end{aligned} \tag{1}$$

where $\boldsymbol{\xi} = [x, y, z]^T$ is the position vector, $\mathbf{v}$ is the velocity vector, $\mathbf{q}$ is the quaternion, $\mathbf{q}^*$ the conjugate of $\mathbf{q}$, ω is the angular rate, f the total thrust and τ the moments. The operation $\otimes$ denotes a quaternion product.

B. Control inputs

Control inputs that we seek to compute are f and τ. To pass from the control inputs to motor commands, we use the baseline motor mixing algorithm of the basic hover model in the Parrot Minidrone Simulink Support Package. Therefore, the moments and forces and the actuator speeds are related as follows:

$$\begin{bmatrix} f \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} b(M_1^2 + M_2^2 + M_3^2 + M_4^2) \\ Lb(M_1^2 + M_2^2 - M_3^2 - M_4^2) \\ Lb(M_1^2 - M_2^2 - M_3^2 + M_4^2) \\ d(-M_1^2 + M_2^2 - M_3^2 + M_4^2) \end{bmatrix}, \tag{2}$$

where b and d are respectively the thrust and drag coefficients and L is the lever arm. The thrust on the motor i is approximated quadratically with its motor speed M_i with the following relation:

$$T_i = bM_i^2. \tag{3}$$

III. State Estimation

A. Extended Kalman Filter

We employ an Extended Kalman Filter (EKF) for attitude and altitude estimation due to its capability to manage nonlinear system dynamics through local linear approximations. The EKF works in two main steps: a prediction step, which uses motion model inputs to estimate the system's next state, and a correction step, which refines this estimate using measurements from other sensors (such as ultrasonic and barometric data). By linearizing the motion and sensor models around the current estimate, the EKF is well-suited for flight, offering improved accuracy and robustness when fusing multiple sensor types during dynamic maneuvers.

We need therefore the following variables:

- $\mathbf{x}_k \in \mathbb{R}^n$: state vector at time step k,
- $\mathbf{u}_k \in \mathbb{R}^m$: control input vector,
- $\mathbf{c}_k \in \mathbb{R}^p$: measurement (observation) vector,
- $\hat{\mathbf{x}}_{k|k} \in \mathbb{R}^n$: estimated state after measurement update,
- $\mathbf{P}_{k|k} \in \mathbb{R}^{n\times n}$: estimation error covariance matrix,
- $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k)$: process noise,
- $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k)$: measurement noise,
- $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$: nonlinear state transition function,
- $\mathbf{F}_k \in \mathbb{R}^{n\times n}$: Jacobian of f with respect to $\mathbf{x}_k$,
- $h : \mathbb{R}^n \to \mathbb{R}^p$: nonlinear measurement function,
- $\mathbf{H}_k \in \mathbb{R}^{p\times n}$: Jacobian of h with respect to $\mathbf{x}_k$,
- $\tilde{\mathbf{y}}_k \in \mathbb{R}^p$: innovation (measurement residual),
- $\mathbf{S}_k \in \mathbb{R}^{p\times p}$: innovation covariance,
- $\mathbf{K}_k \in \mathbb{R}^{n\times p}$: Kalman gain.

The generalized state-space model is formulated as follows:

$$\begin{aligned} \mathbf{x}_k &= f(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \mathbf{w}_{k-1}, \\ \mathbf{c}_k &= h(\mathbf{x}_k) + \mathbf{v}_k. \end{aligned} \tag{4}$$

The state vector $\mathbf{x}_k$ can then be estimated using Algorithm 1.

Algorithm 1 Extended Kalman Filter (EKF)

```
1:  Initialization: $\hat{\mathbf{x}}_0 \in \mathbb{R}^n, \quad \mathbf{P}_0 \in \mathbb{R}^{n\times n}$

2:  1. Prediction (a priori):
3:  $\hat{\mathbf{x}}_{k|k-1} \leftarrow f(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_{k-1})$
4:  $\mathbf{F}_{k-1} \leftarrow \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_{k-1}}$
5:  $\mathbf{P}_{k|k-1} \leftarrow \mathbf{F}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1}$

6:  2. Update (correction):
7:  $\tilde{\mathbf{y}}_k \leftarrow \mathbf{c}_k - h(\hat{\mathbf{x}}_{k|k-1})$
8:  $\mathbf{H}_k \leftarrow \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k|k-1}}$
9:  $\mathbf{S}_k \leftarrow \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k$
10: $\mathbf{K}_k \leftarrow \mathbf{P}_{k|k-1}\mathbf{H}_k^T\mathbf{S}_k^{-1}$
11: $\hat{\mathbf{x}}_{k|k} \leftarrow \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\tilde{\mathbf{y}}_k$
12: $\mathbf{P}_{k|k} \leftarrow (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1}$
```

B. Altitude estimation

An Extended Kalman Filter (EKF) is implemented to estimate vertical position and velocity by fusing inertial acceleration data from an IMU with sonar-based altitude measurements.

Similarly to [1], we can define, the state vector as:

$$\mathbf{x}_k = \begin{bmatrix} z_k & \dot{z}_k \end{bmatrix}^\top, \tag{5}$$

where z_k is the altitude and $\dot{z}_k$ is the vertical velocity.

The control input is defined as :

$$\mathbf{u}_k = \ddot{z}_k, \tag{6}$$

with $\ddot{z}_k$ representing the vertical acceleration in the earth frame, computed by rotating the IMU acceleration vector using the estimated orientation and subtracting gravity :

$$\ddot{z}_k = \left(R_{\text{be}}^\top \cdot \mathbf{a}_{\text{imu}}\right)_z - g. \tag{7}$$

The discrete-time state update equations and the Jacobian with respect to the state are :

$$\mathbf{x}_{k+1} = \begin{bmatrix} z_k + \dot{z}_k T_s \\ \dot{z}_k + \ddot{z}_k T_s - bias \end{bmatrix}; \quad F_k = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix}. \tag{8}$$

The sonar-based measurement model observes only altitude:

$$h(\mathbf{x}_k) = z_k, \quad H_k = \begin{bmatrix} 1 & 0 \end{bmatrix}. \tag{9}$$

The covariance matrices are then tuned in order to have the best representation of the real state.

The system is nonlinear due to the computation of $\ddot{z}_k$, which depends on roll, pitch, and yaw angles via the direction cosine matrix. Therefore, an Extended Kalman Filter (EKF) is required to handle the nonlinear input. The updated estimation improves tracking of vertical position and velocity, reducing oscillations and the initial transient bump, leading to more stable and accurate performance, especially in the early stages.

C. Quaternion-based attitude estimation

We implemented an Extended Kalman Filter (EKF) attitude estimator using quaternion representation. This approach was selected to overcome the limitations associated with Euler angle representations, particularly the issue of gimbal lock, which can compromise estimation accuracy during aggressive or large-angle maneuvers.

We define the state vector as $\mathbf{x} \triangleq \mathbf{q} = \begin{bmatrix} q_\omega & q_x & q_y & q_z \end{bmatrix}^T$ and the control vector as $\mathbf{u} \triangleq \boldsymbol{\omega} = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T$, where $\boldsymbol{\omega}$ denotes the angular velocity measured by the gyroscope.

For the prediction step we use the Euler–Rodrigues formula, that define the predicted orientation quaternion $\hat{\mathbf{q}}_{k|k-1}$ as:

$$\hat{\mathbf{q}}_{k|k-1} = \left[\cos\left(\frac{\|\boldsymbol{\omega}|\Delta t}{2}\right)\mathbf{I}_4 + \frac{2}{\|\boldsymbol{\omega}\|}\sin\left(\frac{\|\boldsymbol{\omega}\|\Delta t}{2}\right)\boldsymbol{\Omega}_{k-1}\right]\mathbf{q}_{k-1}, \tag{10}$$

with $\mathbf{I_4}$ representing an identity matrix of order 4, and the quaternion rate matrix $\boldsymbol{\Omega}_{k-1}$:

$$\boldsymbol{\Omega}_{k-1} = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix}. \tag{11}$$

After linearizing at time t_k, making it a first-order EKF using a Taylor expansion, the predicted quaternion $\hat{\mathbf{q}}_{k|k-1}$ becomes:

$$\hat{\mathbf{q}}_{k|k-1} = \left(\mathbf{I}_4 + \frac{\Delta t}{2}\boldsymbol{\Omega}_{k-1}\right)\mathbf{q}_{k-1}. \tag{12}$$

With this, we can now compute $\mathbf{F}_{k-1}$:

$$\mathbf{F}_{k-1} = \mathbf{I}_4 + \frac{\Delta t}{2}\boldsymbol{\Omega}_{k-1}. \tag{13}$$

Next we compute the Process Noise Covariance Matrix :

$$\boldsymbol{Q}_{k-1} = \mathbf{W}_{k-1}\boldsymbol{\Sigma}_\omega\mathbf{W}_{k-1}^T. \tag{14}$$

Assuming equal gyroscope noise σ_ω across all axes, the Jacobian $\mathbf{W_k}$ and the noise values $\boldsymbol{\Sigma}_\omega$ are defined :

$$\mathbf{W}_{k-1} = \left.\frac{\partial f}{\partial \omega}\right|_{\hat{\mathbf{q}}_{k-1|k-1},\omega_{k-1}}, \tag{15}$$

$$\boldsymbol{\Sigma}_\omega = \sigma_\omega^2\mathbf{I}_3. \tag{16}$$

The predicted error state covariance $\mathbf{P}_{k|k-1}$ how estimation confidence evolves by propagating prior uncertainty and process noise through the linearized system.

The first step in the correction phase begins with the measurement vector $\mathbf{c}_k$, which uses the normalized accelerometer data. $\mathbf{a} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^T$:

$$\mathbf{c}_k = \frac{1}{\|\mathbf{a}\|}\begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^T. \tag{17}$$

These normalized readings are then used to relate the measured acceleration to the predicted state, we use the quaternion rotation matrix $\mathbf{C}(\mathbf{q})$. This matrix transforms a vector $\mathbf{x}$ from the body frame to the global frame, resulting in the transformed vector $\mathbf{x}'$.

Since we are using an North East Down (NED) coordinate system, the unit gravity vector is $\mathbf{g}_{dir} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$. In this context, only the direction matters, not the magnitude. To correct the prediction model, we use the measurement function $\mathbf{h}(\hat{\mathbf{q}}_{k|k-1})$ along with its Jacobian $\mathbf{H}_k$.

$$\mathbf{h}(\hat{\mathbf{q}}_{k|k-1}) = \mathbf{C}(\hat{\mathbf{q_k}})^T\mathbf{g}_{dir} = 2\begin{bmatrix} \hat{q}_x\hat{q}_z - \hat{q}_\omega\hat{q}_y \\ \hat{q}_\omega\hat{q}_x + \hat{q}_y\hat{q}_z \\ \frac{1}{2} - \hat{q}_x^2 - \hat{q}_y^2 \end{bmatrix}, \tag{18}$$

$$\mathbf{H}_k = 2\begin{bmatrix} -\hat{q}_y & \hat{q}_z & -\hat{q}_\omega & \hat{q}_x \\ \hat{q}_x & \hat{q}_\omega & \hat{q}_z & \hat{q}_y \\ \hat{q}_\omega & -\hat{q}_x & -\hat{q}_y & \hat{q}_z \end{bmatrix}. \tag{19}$$

Assuming identical noise characteristics σ_a across all axes, we define the noise covariance matrix $\mathbf{R} = \sigma_a^2\mathbf{I}_3$ to compute $\mathbf{S}_k$ and $\mathbf{K}_k$.

After the update step, the corrected quaternion and its associated covariance are obtained.

The EKF can diverge with an inaccurate model, partly because quaternion components are not independent due to the unit norm constraint. A simple fix is to normalize the updated quaternion.

IV. Control Method

A. Overall architecture

Our lateral controller is composed of a cascaded multi-loop structure that seperates lateral position from attitude control. More precisely, we employ a unique position controller that feeds the attitude controller with a certain pitch and roll trajectory. Our altitude controller, however, is a single loop controller. Figure 3 shows a block diagram representation of our controller.

As introduced earlier, the controller computes f and τ, corresponding to the total thrust and rotational moments.

B. Lateral position control

From this point onward, the term *Position* will refer exclusively to the lateral position components, excluding altitude. The position controller is based on the basic hover model and features a cascaded control structure. This structure consists of an outer proportional (P) loop that functions as the position tracker, and an inner proportional-integral (PI) loop that acts as the velocity tracker. The general continuous-time control law for a PI controller is given by:

$$\gamma = K_{p_\gamma}\left(\begin{bmatrix} v_{d_x} - v_x \\ v_{d_y} - v_y \end{bmatrix} + K_{i_\gamma}\int\left(\begin{bmatrix} v_{d_x} - v_x \\ v_{d_y} - v_y \end{bmatrix}\right)dt\right), \tag{20}$$

where γ is the vector containing the pitch and roll command.

The P control law tracking the position is :

$$\begin{bmatrix} v_{d_x} \\ v_{d_y} \end{bmatrix} = K_{p_v}\left(\begin{bmatrix} x_d - x \\ y_d - y \end{bmatrix}\right). \tag{21}$$

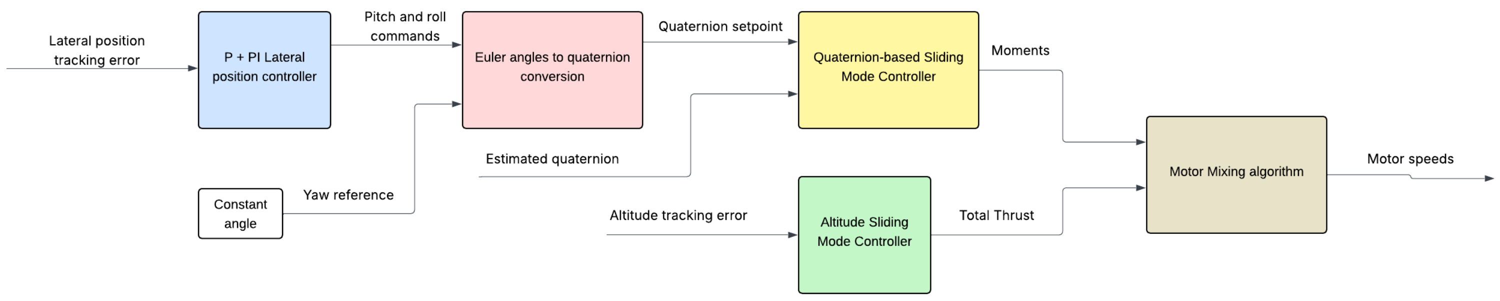

Lateral position tracking error
P + PI Lateral position controller
Pitch and roll commands
Euler angles to quaternion conversion
Quaternion setpoint
Quaternion-based Sliding Mode Controller
Moments
Estimated quaternion
Constant angle
Yaw reference
Altitude tracking error
Altitude Sliding Mode Controller
Total Thrust
Motor Mixing algorithm
Motor speeds

Fig. 3. Block diagram representing the overall architecture of the flight controller.

The parameters K_{p_γ}, K_{i_γ} and K_{p_v} are tuned to achieve optimal performance. For practical implementation, these control laws are discretized using the forward Euler method.

The pitch and roll commands are converted into quaternion setpoints, which are then fed to the attitude controller. A fixed value is used for the reference heading. **Although the orientation setpoints are represented using Euler angles for ease of interpretation, the controller's function is limited to stabilization rather than generating complex orientation trajectories.**

C. Altitude control

Sliding Mode Control is a robust control method ideal for drones operating in uncertain and dynamic environments. It works by forcing the system to reach and stay on a predefined sliding surface, ensuring stability and tracking despite disturbances like wind or model inaccuracies. Let us define a sliding surface:

$$s_z = \dot{z} - \dot{z}_d + \lambda_z (z - z_d), \tag{22}$$

where $\lambda_z > 0$ is a design parameter that defines the slope of the sliding surface s_z. This surface incorporates the tracking error $e_z = z - z_d$ and its derivative to guide the system dynamics. We can then relate the time derivative of the sliding surface to a hyperbolic tangent law as follows:

$$\dot{s_z} = \ddot{z} - \ddot{z_d} + \lambda_z (\dot{z} - \dot{z_d}) \overset{!}{=} -K_z \tanh(s_z), \tag{23}$$

where $K_z > 0$ is a control gain that enforces the sliding condition, and $\tanh(s_z)$ serves as a smooth approximation of the discontinuous sign function. This helps to reduce *chattering*, which refers to high-frequency oscillations caused by abrupt switching in control signals. Chattering is a common issue in SMC and can excite high-frequency dynamics or cause mechanical wear in the actuators.

The control objective is to drive s_z to zero, ensuring that the altitude z tracks the desired altitude z_d. To do this, we consider the translational dynamics of the drone along the z-axis in Euler angles representation which is governed by:

$$m\ddot{z} = -f \cos\phi \cos\theta + mg, \tag{24}$$

where f is the total thrust generated by the propellers, m is the mass of the drone, g is gravitational acceleration, and ϕ, θ are the roll and pitch angles, respectively.

Solving for f, the control law based on sliding mode theory is given by:

$$f = \frac{m}{\cos\phi \cos\theta} \left(-\ddot{z}_d + g + \lambda_z (\dot{z} - \dot{z}_d) + K_z \tanh(s_z) \right). \tag{25}$$

This control law ensures finite-time convergence to the sliding surface and robustness against bounded disturbances and model uncertainties. By default, we set $\ddot{z}_d = 0 \; m.s^{-2}$.

D. Quaternion-based sliding mode attitude control

Since we are using a quaternion-based controller, first let us define the tracking errors of the quaternions and the angular velocities:

$$\mathbf{q_e} = \mathbf{q_d^*} \otimes \mathbf{q}, \quad \omega_\mathbf{e} = \omega - \omega_\mathbf{d}, \tag{26}$$

where the subscript d denotes the desired states. The desired angular velocity ω_d is calculated using a simple proportional controller taking the attitude error as input, for a more precise tracking. In tracking mode, a sliding surface is defined using a line characterized by a linear equation with a negative slope, relating each error state to its time derivative. This error state vector is constrained to follow the surface, guiding the system toward the origin where all tracking errors converge to zero.

Similarly to the SMC alitude controller, let us define a sliding surface for the attitude:

$$\mathbf{s_q} = \omega_\mathbf{e} + \mathbf{\Lambda_q} sgn(q_{w_e}) \vec{\mathbf{q_e}}, \tag{27}$$

where $\mathbf{\Lambda_q}$ is a diagonal matrix containing the tunable slopes of the sliding surface. $\vec{\mathbf{q_e}}$ only contains the imaginary parts. $sgn(q_{w_e})$ is the sign of the scalar part of the quaternion error that forces the shortest path to the desired states (setpoints).

This allows us to obtain the analytical expression for the rate of the auxiliary variable s_q while relating it to a custom reaching law:

$$\dot{\mathbf{s_q}} = \dot{\omega_\mathbf{e}} + \mathbf{\Lambda_q} sgn(q_{w_e}) \dot{\vec{q_e}} \overset{!}{=} -\mathbf{K_q} \tanh\left(\frac{\mathbf{s_q}}{\beta}\right) - \mathbf{K_s s_q}, \tag{28}$$

where the diagonal matrix $\mathbf{K_q}$ is the tunable rate at which the states converge to the sliding surface, according to [2]. A proportional term to the auxiliary variable $\mathbf{K_s s_q}$ is added here to increase the aggressiveness of the convergence to the surface. In addition to the chattering-free control law produced by the hyperbolic tangent function, a new parameter β is introduced to manually move the boundary layer under which the control effort is smoothly reduced. Finally, the final derived control law obtained from the equations of motions (1) is:

$$\tau = \mathbf{J}\dot{\omega}_d + \omega \times \mathbf{J}\omega - \mathbf{J}\mathbf{\Lambda_q} sgn(q_{\omega_e})\vec{q_e} - \mathbf{J}\mathbf{K_q}\tanh(\mathbf{s_q}) - \mathbf{J}\mathbf{K_s}\mathbf{s_q}. \quad (29)$$

V. Experiment Setups

A. Test goals

Before attempting complex maneuvers such as a 360° flip, we aim to validate our controller under challenging initial conditions, including free-fall and an inverted orientation. These tests enable us to fine-tune the control parameters and assess the performance of the flight control system prior to executing more aggressive maneuvers.

B. Physical properties and hardware

The minidrone model we are using for the experiments is the Parrot Mambo Minidrone with the physical parameters of Table I. This model uses a MPU6050 6DOF IMU sensor whose data are fed into the state estimators. It is important to keep in mind, the sensor model in the simulator does not include the change of the gyroscope bias over time which can occur in real life.

TABLE I
Physical Properties of the minidrone Mambo

Parameter	Value	Units
Mass (m)	0.063	kg
Thrust coefficient (b)	0.0107	N·s^2
Thrust drag coefficient (d)	$0.782 \cdot 10^{-3}$	N·m·s^2
Moment of inertia (J_{xx})	$0.5829 \cdot 10^{-4}$	kg·m^2
Moment of inertia (J_{yy})	$0.7169 \cdot 10^{-4}$	kg·m^2
Moment of inertia (J_{zz})	$1.0000 \cdot 10^{-4}$	kg·m^2
Lever arm (L)	0.0624	m
Vertical acceleration bias ($bias$)	-0.098	m·s^{-1}

C. Free-fall

In the free-fall test, the drone is positioned upright at a height of approximately 3 meters above the ground and then released. For the first 0.43 seconds after release, only idle thrust is applied. After this delay, control inputs from the flight control system are activated to recover. The position setpoint is set at $\xi_\mathbf{d} = [0, 0, -2]^T m$. This test is important to assess the responsiveness of the altitude controller facing situations with high acceleration, as well as a sudden loss of lift or control input.

D. 180° Flip maneuver

In this second scenario, the drone is placed in an inverted orientation (in the roll direction) to simulate the midpoint of a full flip. It is held statically at the origin and at a height of 1.146 meters and at for 3 seconds before being released, without any externally applied disturbances. To give the position controller a headstart in which direction to flip, the user-defined position setpoint is $\xi_\mathbf{d} = [-0.2, -0.2, -1.1]^T m$. Recall that we use an NED reference frame. This setup allows us to evaluate whether the attitude controller can respond rapidly and accurately to the pitch and roll commands issued by the position controller, enabling a quick and efficient recovery to the upright position. Furthermore, it involves nonlinear control behavior and can expose instability in the controller or estimator.

VI. 360° flip maneuver trajectory planning

For complex maneuvers such as the 360° flip, a state-machine-like approach is required to distinguish between the maneuvering and stabilization phases. During the maneuver phase, the position controller must be temporarily disabled to allow execution of the user-defined attitude trajectory. Once the maneuver is complete, the position controller is reactivated to stabilize the drone and resume nominal flight.

First, the trajectory of the scalar part of the quaternion during the flip is defined as a smooth, time-dependent sigmoid function, as proposed in [5], and is given by:

$$q_{d_\omega}(t) = \frac{2}{1 + e^{-v_m(t - \frac{t_m}{2})}} - 1, \quad (30)$$

where v_m is the flip speed and t_m determines the time when the flip occurs.

If we want to do the flip around the e_2 axis, we set the second imaginary part to be:

$$q_{d_y}(t) = \sqrt{(1 - q_{d_\omega}(t)^2)}. \quad (31)$$

The other two elements in the desired quaternion are zeros. A flip in the e_1 axis will require switching q_{d_y} with q_{d_x}. Using quaternion in the attitude control, the drone is able to do a flip about the e_2 axis without suffering singularities.

To execute this flip using quaternion-based control, the desired quaternion trajectory $\mathbf{q_d(t)}$ must be converted into a corresponding angular velocity command $\omega_d(t)$. This is necessary because the SMC attitude controller typically tracks both the angular velocity and quaternion simultaneously.

To compute the angular velocity, we take the analytical time derivative of the desired quaternion $\mathbf{\dot{q}_d(t)}$ and apply the following relationship:

$$\omega_\mathbf{d} = 2 \cdot \mathrm{vec}(\dot{q}_d \otimes q_d^*), \quad (32)$$

where $\mathrm{vec}(\cdot)$ extracts the vector (imaginary) part of the quaternion. This expression ensures that the computed angular velocity corresponds to the body-frame rotational velocity required to track the quaternion trajectory.

Highly aggressive maneuvers like this introduce significant delays in state estimation, making it challenging to rely on

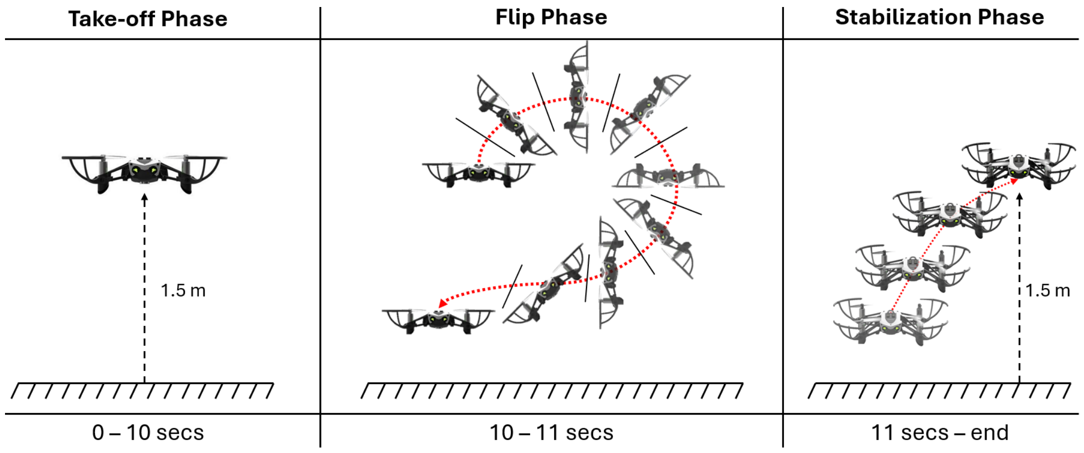
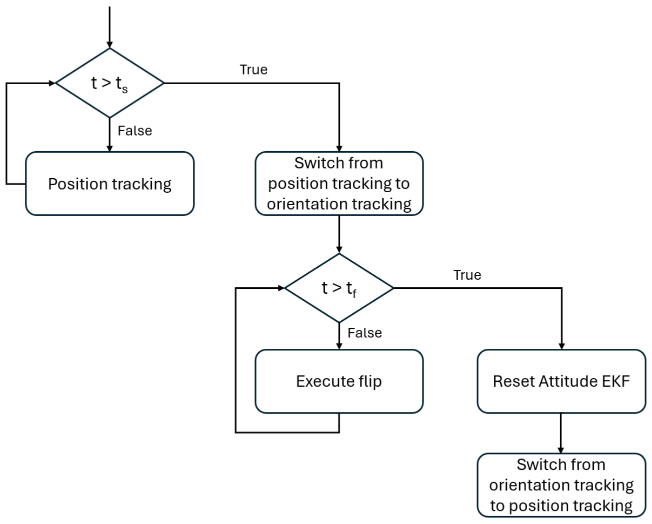


Fig. 4. Flight sequence of the 360° flip maneuver.

the attitude commands from the position controller during the stabilization phase. As a result, the estimation system must be reconfigured after a certain period to prevent unexpected biases in attitude estimation caused by gyroscope drift. To address this, the covariance matrix $\mathbf{P}$ in the attitude state estimator, as well as the previously estimated quaternion, must be reset after the flip.



t > t_s
True
False
Position tracking
Switch from position tracking to orientation tracking
t > t_f
True
False
Execute flip
Reset Attitude EKF
Switch from orientation tracking to position tracking

Fig. 5. Tracking Mode Flow During Aerial Flip.

The switching between control phases is governed by the duration of the maneuver to ensure effective stabilization throughout the flip. Two key time-dependent parameters are adjusted: t_s, the time at which the position controller is disabled and the quaternion trajectory planner begins generating attitude setpoints; and t_f, the time at which the position controller is re-enabled to stabilize the vehicle after the flip as shown in Figures 4 and 5.

VII. Simulation Results

A. Free fall

Figure 6 illustrates the results of the free-fall test, where a significant overshoot was expected due to the high acceleration accumulated during the descent. The lateral position tracking is slightly affected by the large thrust input, which results from the reactive nature of the SMC altitude controller. Nevertheless, the drone successfully stabilizes and resumes trajectory tracking with only minimal overshoot, demonstrating the effectiveness of the altitude controller in handling aggressive vertical maneuvers.

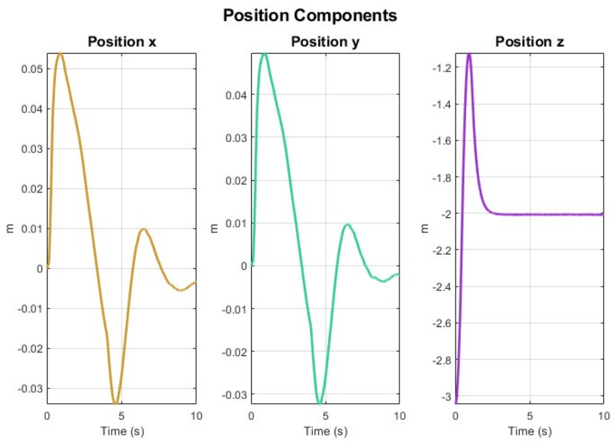


Fig. 6. Position components and height estimations during free-fall test.

B. 180° flip maneuver

Figures 7 - 8 show the estimated states of the straight angle flip. During the first three seconds, the attitude state estimator begins detecting the inverted orientation, starting from its initial condition of $\mathbf{q_0} = [1,\ 0,\ 0,\ 0]^T$, which represents an upright orientation with the non-inertial b_3 axis pointing downwards. Due to the shortest-path logic used by the attitude controller and the absence of a fixed reference

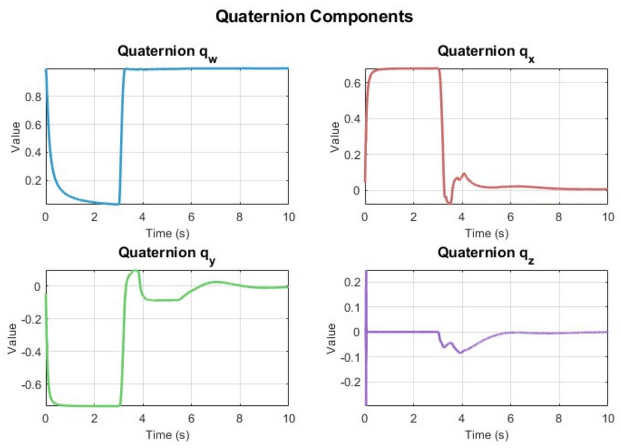


Fig. 7. Quaternion components estimations during the 180° flip maneuver.

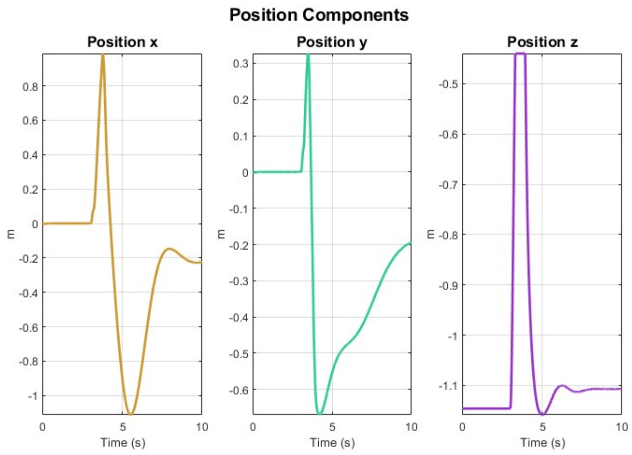


Fig. 8. Position components and height estimations during the 180° flip maneuver.

in the estimator, the drone recovers in a diagonal direction. This explains why the quaternion at the moment of release is $q = [0, 0.707, -0.707, 0]^T$, rather than $q = [0, 1, 0, 0]$, which corresponds to a 180° rotation about the roll axis.

Regarding position stabilization, the drone requires a moderate lateral distance to recover, which is important in constrained environments. The altitude drop is approximately 0.7 meters, mitigated by the aggressiveness of the altitude controller.

C. 360° back-flip maneuver

Figures 9–11 present the estimated states during a 360° flip flight about the pitch axis from 5 seconds to 20 seconds. The take-off phase is not showed here, as we focus on the flip behaviour. The 3D visualizer and the 2D plots both show a successful flip. The flip is clearly indicated by the prominent peaks in Figure 9, occurring around the 10-second mark. The subsequent steady-state values reflect the system's stabilization, although slight overshoots are observable. The duration of the flip is under half of a second, allowing the

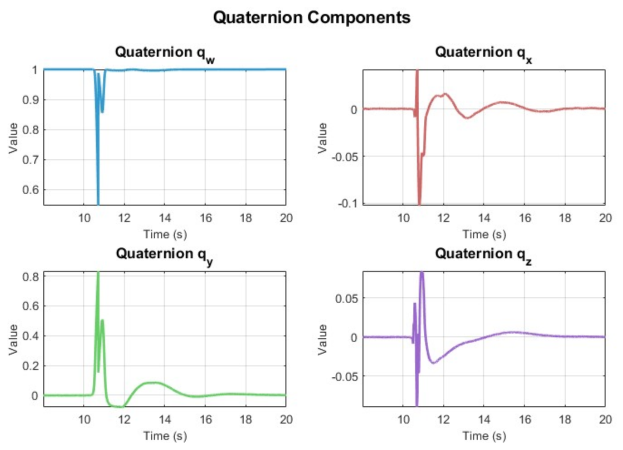


Fig. 9. Quaternion components estimations during the 360° flip maneuver in the e_2 (y) axis.

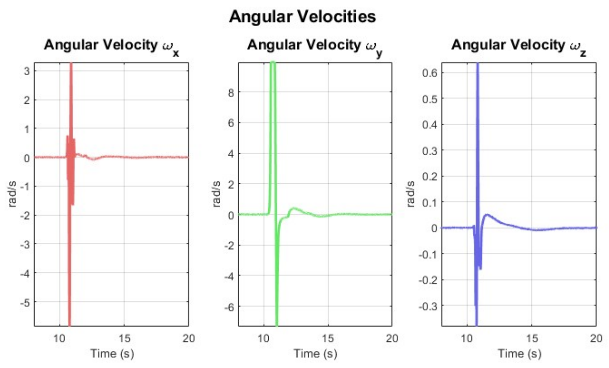


Fig. 10. Angular velocity components estimations during the 360° flip maneuver in the e_2 (y) axis.

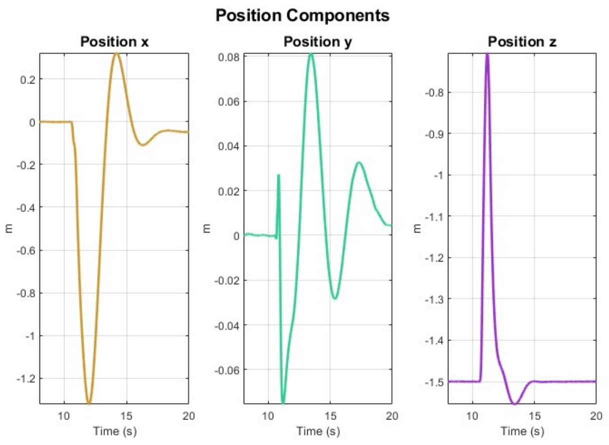


Fig. 11. Position components and height components estimations during the 360° flip maneuver e_2 (y) axis.

miniature aerial vehicle to recover without significant loss in altitude.

During the maneuver, the drone reaches high angular velocities, particularly around the e_2-axis. Despite the aggressive attitude trajectory, the position control system successfully stabilizes the drone within a reasonable range of motion. However, the swinging motion induced by such aggressive dynamics suggests that improved tuning or a more robust position control strategy may be necessary to further reduce positional deviations. For the rest of the simulation time, the smooth quaternion transitions and bounded position deviations indicate good control performance during the flip.

It is important to note that the estimated quaternions can be inaccurate, particularly during and immediately after the flip. This explains why the trajectory defined in Section VI does not appear to be followed. The reason for this is that the attitude EKF is reset after the flip, introducing a sudden discontinuity in the state estimation.

D. 360° roll-flip maneuver

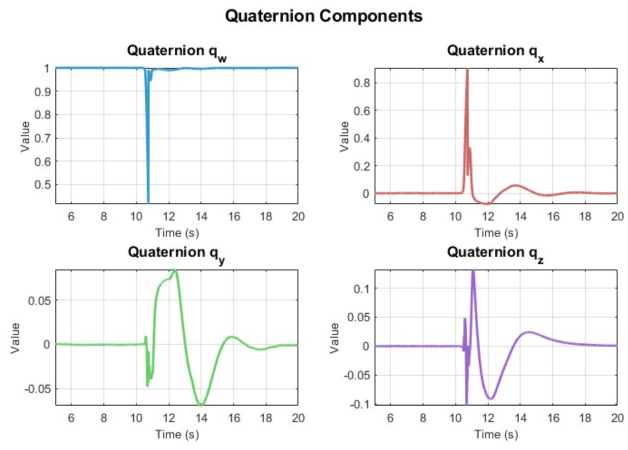


Fig. 12. Quaternion components estimations during the 360° flip maneuver in the e_1 (x) axis.

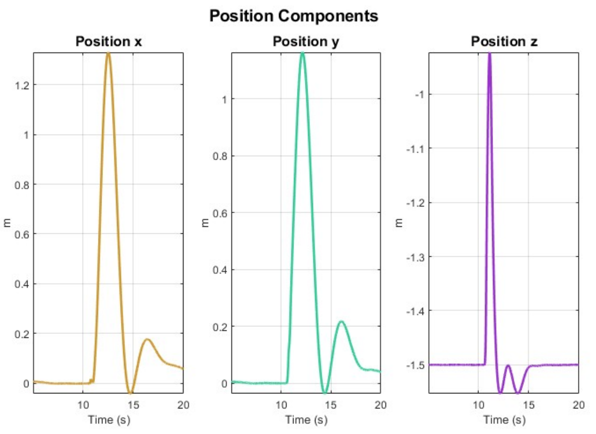


Fig. 13. Position components and height estimations during the 360° flip maneuver e_1 (x) axis.

Figures 12–13 present the estimated states during a 360° flip about the roll axis. Similar to the backflip, the flip duration remains under half of a second. However, the recovery performance shows better results with an altitude drop of approximately 0.6 meters. Position overshoots are still present but remain within a reasonable range.

VIII. Conclusion

This project validates the effectiveness of quaternion-based Sliding Mode Control combined with Extended Kalman Filters to stabilize a minidrone under aggressive conditions. The quadcopter successfully recovered after a free-fall, an upside down position, and 360° flips, demonstrating robust performance despite rapid dynamics and model uncertainties.

Simulation results confirmed stable tracking and reliable estimation throughout all maneuvers. However the drone exhibited sometimes some lateral positional drift, particularly due to large thrust inputs and estimator delays. While this deviation remained within acceptable bounds, it highlights an area for further refinement.

To address this, one approach would be to enhance the lateral position controller by retuning the current PID gains or replacing it with a more robust control strategy such as a Sliding Mode Controller or a Linear Quadratic Regulator (LQR). These methods would yield greater precision and improved disturbance rejection under aggressive flight conditions.

Acknowledgment

We would like to express our sincere gratitude to Professor Josie Hughes for her guidance and supervision throughout the course of this research. We are also grateful to Doctoral Assistant Guanran Pei for his support and helpful insights.

We extend our special thanks to Dr. Res Joehr and Dr. Roberto G. Valenti from MathWorks for their technical expertise and valuable collaboration.

Finally, we acknowledge the École Polytechnique Fédérale de Lausanne (EPFL) for providing the resources and environment that made this work possible.

References

[1] Asmaa Taame, Ibtissam Lachkar, Abdelmajid Abouloifa, Ismail Mouchrif, “UAV Altitude Estimation Using Kalman Filter and Extended Kalman Filter”, January 2024, in Lecture Notes in Electrical Engineering, DOI: 10.1007/978-981-97-0126-1_72

[2] A. Yazdanshenas et R. Faieghi, ”Quaternion-Based Sliding Mode Control for Six Degrees of Freedom Flight Control of Quadrotors”, March 16 2024, arXiv: arXiv:2403.10934. doi: 10.48550/arXiv.2403.10934.

[3] Stevens, Brian L., Frank L. Lewis. Aircraft Control and Simulation, 2nd Edition. Hoboken, NJ: John Wiley & Sons, 2003.

[4] Guo, S., Wu, J., Wang, Z., & Qian, J. (2017). Novel MARG-sensor orientation estimation algorithm using fast Kalman filter. Manuscript published September 24, 2017.

[5] P. Antal, T. Peni, et R. Toth, ”Backflipping With Miniature Quadcopters by Gaussian-Process-Based Control and Planning”, IEEE Transactions on Control Systems Technology, vol. 32, n° 1, p. 3-14, janv. 2024, doi: 10.1109/TCST.2023.3297744.

[6] Zhang, F. (1997). Quaternions and matrices of quaternions. Linear Algebra and Its Applications, 251, 21–57. doi : 10.1016/0024-3795(95)00543-9

[7] N. P. Nguyen, N. X. Mung, H. L. N. N. Thanh, T. T. Huynh, N. T. Lam and S. K. Hong, ”Adaptive Sliding Mode Control for Attitude and Altitude System of a Quadcopter UAV via Neural Network,” in IEEE Access, vol. 9, pp. 40076-40085, 2021, doi: 10.1109/ACCESS.2021.3064883.