Quadcopter Control For Aggressive Maneuvers

Vezhish Ali
Electrical Engineering
Habib University
Karachi, Pakistan
va0760@st.habib.edu.pk

Muhammad Abdullah
Electrical Engineering
Habib University
Karachi, Pakistan
ma07534@st.habib.edu.pk

Syed Muhammad Muslim Hussain
Electrical Engineering
Habib University
Karachi, Pakistan
sh07730@st.habib.edu.pk

Shahjahan Sangrasi
Computer Engineering
Habib University
Karachi, Pakistan
ss07620@st.habib.edu.pk

***Abstract*—This paper presents a robust control framework for tracking aggressive trajectories using a quadrotor UAV. Classical controllers, such as PID and LQR, struggle to manage aggressive maneuvers involving large angular displacements due to limitations like singularities and inability to handle nonlinearity. To address these challenges, we propose a hybrid control algorithm that integrates Geometric Tracking Control (GTC) on $SE(3)$ for altitude and position tracking and Quaternion-Based Attitude Control for orientation tracking.**

Two maneuvers—a 360-degree flip and a helical trajectory—are used as test cases to demonstrate the effectiveness of the proposed method. The flip maneuver involves large rotational displacements, while the helical trajectory combines smooth circular motion with progressive ascent. The proposed controller guarantees stability and precision by leveraging nonlinear control methods, with thrust and torque computed directly in the body frame. Simulation results validate the approach, showing accurate trajectory tracking, reduced position and velocity errors, and robust performance under aggressive conditions. The hybrid controller effectively avoids singularities and ensures global stability, making it suitable for real-world UAV applications requiring high performance.

***Index Terms*—Quadrotor, Control, Aggressive Maneuvers, Geometric Tracking Control, Quaternion Control**

I. Introduction

In recent years, quadrotor Unmanned Aerial Vehicles (UAVs) have garnered substantial interest due to their versatility, simplicity, and ability to perform complex aerial maneuvers. A quadrotor UAV consists of two pairs of counter-rotating rotors and propellers positioned at the vertices of a square frame. Unlike traditional helicopters, quadrotors eliminate the need for complex mechanical linkages such as swash plates or teeter hinges, making them a more mechanically straightforward alternative. This structural simplicity, coupled with their ability to perform Vertical Take-Off and Landing (VTOL), makes quadrotors ideal for a wide array of applications, including surveillance, mobile sensor networks, and educational purposes. Their cost-effectiveness and agility have resulted in increased adoption in both university-level research projects [1] [2] [3] [4] and commercial products [5] [6] [7].

Despite the widespread implementation of quadrotor UAVs, challenges persist in designing robust and high-performance control systems, particularly for complex and aggressive maneuvers like single flips, double flips, or circular-wave trajectories. Traditional linear controllers, such as Proportional-Derivative (PD) controllers or Linear Quadratic Regulators (LQR), are commonly used to stabilize quadrotors. [8] [9]. However, these methods are inherently limited as they rely on linearized dynamics around equilibrium points, which restricts their applicability to small perturbations. As a result, linear controllers struggle to handle highly nonlinear systems like quadrotors, especially during extreme maneuvers or under external disturbances.

The fundamental limitation of linear controllers arises from their inability to represent the global dynamics of systems that evolve on nonlinear manifolds. In the case of quadrotor UAVs, the configuration space is described on the Special Euclidean Group SE(3), which combines rotational and translational dynamics. Approaches relying on Euler angles, such as Newton-Euler formulations [10], are intuitive but suffer from singularities like "gimbal lock," where the loss of one degree of freedom occurs due to axis alignment. These formulations also incur high computational costs because of the extensive trigonometric calculations involved. Additionally, converting between Euler angles and other representations like Direction Cosine Matrices (DCM) introduces further complexity and computational overhead.

To overcome these limitations, nonlinear control strategies such as Geometric Control and Quaternion-Based Control have emerged as promising solutions. Geometric control techniques are designed to operate directly on nonlinear manifolds, providing coordinate-free formulations that avoid singularities and achieve almost global asymptotic stability [11] [12] [13]. These methods are particularly well-suited for dynamic systems like quadrotors, where rotational and translational dynamics must be simultaneously controlled for trajectory tracking. Geometric controllers ensure robust performance during aggressive maneuvers, such as recovering from inverted

orientations or tracking complex trajectories.

Similarly, quaternion-based control offers an elegant alternative to Euler angles and DCMs by representing rotations in a compact, singularity-free form. Quaternions describe orientations using a four-dimensional unit vector, which reduces computational cost and avoids singularities inherent to Euler angle-based methods. Unlike DCMs, which involve solving nine coupled differential equations, quaternion dynamics can be represented using only four coupled differential equations, simplifying the overall system. While earlier works have utilized quaternions for attitude stabilization, many approaches still rely on converting quaternion errors back to Euler angles for regulation, reintroducing the associated nonlinearities and singularities [14] [15].

The motivation for this project stems from the need for advanced nonlinear controllers that can address the limitations of linear strategies and provide superior performance during complex maneuvers. This report focuses on two controllers: **Geometric Tracking Control on SE(3)** and **Full Quaternion-Based Attitude Control**. These controllers will be implemented and simulated in MATLAB to test their performance against benchmark linear controllers, such as PID-based hover controllers, under challenging trajectories including single flips, and helical-wave paths. Additionally, the project explores the integration of geometric and quaternion-based controllers to evaluate their combined effectiveness.

The remainder of the report is organized as follows: Section II introduces the mathematical formulations of the controllers, Section III describes the simulation setup and test trajectories, Section IV presents the results and performance comparison, and Section V concludes the report with key findings and future directions.

II. Related Work

A. Classic Control Approach: Euler Angles-Based PID Controller for Quadcopter UAV Dynamics and Trajectory Tracking

1) Introduction: Quadcopter Unmanned Aerial Vehicles (UAVs) require precise control mechanisms for stability and accurate trajectory tracking. This document presents the design and implementation of an Euler Angles-Based PID Controller, aimed at ensuring reliable attitude and position control for a quadcopter UAV. The controller leverages Proportional-Integral-Derivative (PID) control strategies to manage both translational and rotational dynamics.

2) Translational Dynamics ($\dot{v}$): Translational dynamics ($\dot{v}$) include gravity compensation and thrust:

$$\dot{v} = -g \cdot e_3 + \frac{u_1}{m}$$

3) Rotational Dynamics ($\dot{\omega}$): Rotational dynamics ($\dot{\omega}$):

$$\dot{\omega} = J^{-1} \cdot (\tau - \omega \times (J \cdot \omega))$$

This is correct for rigid-body rotational dynamics.

4) Position Control (u_1): The position control force u_1 is computed as:

$$u_1 = -\text{diag}(k_p)\cdot\text{pos\_err} - \text{diag}(k_d)\cdot\text{vel\_err} + m\cdot\text{acc\_des} + m\cdot g\cdot e_3.$$

This formulation is correct for a PID-like position controller. The use of gravitational compensation and desired acceleration is reasonable.

5) Desired Euler Angles (Attitude Control Inputs): The desired roll (ϕ_{des}) and pitch (θ_{des}) are computed as:

$$\phi_{des} = \frac{1}{g} \cdot (u_1(1) \cdot \sin(\text{yaw\_des}) - u_1(2) \cdot \cos(\text{yaw\_des}))$$
$$\theta_{des} = \frac{1}{g} \cdot (u_1(1) \cdot \cos(\text{yaw\_des}) + u_1(2) \cdot \sin(\text{yaw\_des}))$$

These equations align with standard quadcopter dynamics where pitch and roll are functions of the desired force vector in the inertial frame and the yaw angle.

The desired yaw (ψ_{des}) is set directly as the desired trajectory's yaw component.

6) Attitude Control (Torque, τ): The torque is computed using a PD formulation:

$$\tau = -k_p \cdot \text{angle\_err} - k_d \cdot \omega_{\text{err}}$$

This formulation is typical for quadcopter attitude control.

Equations of Motion

7) Potential Issues: Euler Angle Representation

- Euler angles can lead to singularity issues (e.g., gimbal lock). If the desired trajectory or dynamics approach such configurations, the controller may fail.

Decoupling Assumption

- The roll and pitch equations assume decoupling between translational and rotational dynamics, which is an approximation. For aggressive maneuvers, this assumption can lead to inaccuracies.

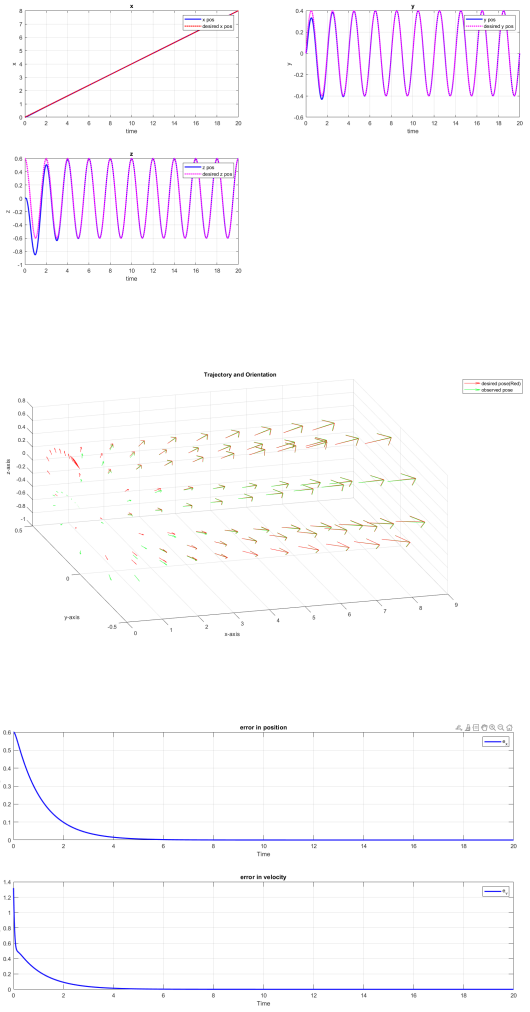
Angular Velocity Feedback

- The code assumes angular velocity error directly corresponds to feedback gains, but real systems often involve noise in angular velocity measurements, which might need filtering.

Control Saturation

- There's no explicit saturation handling for control inputs (e.g., maximum thrust or torque limits). This could lead to unrealistic simulations.

8) Simulation Results: The following are the simulation results



B. Geometric Tracking

An approach to controlling aggressive maneuvers is based on **closed-loop control**. The first step is to design a **reference trajectory** for the maneuver, which is then given to a **nonlinear controller** to achieve precise reference tracking. Compared to **open-loop control**, this closed-loop approach offers the advantage of generality, as it can be applied to a wide variety of aggressive maneuvers. While open-loop strategies can be tailored for specific maneuvers, they fail to account for real-world uncertainties and disturbances. Closed-loop control, on the other hand, provides robustness and adaptability to external disturbances.

The main framework for the implementation of geometric tracking control is based on the seminal work by **Lee et al.**, titled *"Geometric Tracking Control of a Quadrotor UAV on $SE(3)$"* [16]. This approach leverages the mathematical structure of the **Special Euclidean Group** $SE(3)$ and the **Special Orthogonal Group** $SO(3)$ to provide a globally defined, singularity-free formulation for attitude and position control. Geometric control ensures precise tracking of desired trajectories by directly formulating the control laws on these manifolds.

1) Quadrotor Dynamics: The dynamics of a quadrotor UAV can be described using the **Newton-Euler equations**, assuming a rigid body model with symmetric structure. The state of the quadrotor is defined by its **position** $x \in \mathbb{R}^3$, **orientation** $R \in SO(3)$, **linear velocity** $v \in \mathbb{R}^3$, and **angular velocity** $\Omega \in \mathbb{R}^3$ in the body-fixed frame. The translational and rotational dynamics are given as:

$$m\ddot{x} = -FRe_3 + mge_3 \tag{1}$$

$$J\dot{\Omega} = -\Omega \times (J\Omega) + M \tag{2}$$

where:

- m is the mass of the quadrotor.
- F is the total thrust produced by the rotors.
- $R \in SO(3)$ is the rotation matrix representing the attitude.
- $e_3 = [0; 0; 1]$ is the vertical unit vector.
- g is the gravitational acceleration.
- J is the inertia matrix.
- $M \in \mathbb{R}^3$ is the moment vector.

The force F acts along the third body axis Re_3, and the torques M determine the rotational motion of the UAV.

2) Control Errors on $SE(3)$: The geometric tracking control approach introduces errors for position, velocity, attitude, and angular velocity that are directly defined on the manifold $SE(3)$. The errors are represented as follows:

1) **Position Error**:
$$e_x = x - x_d,$$
where x_d is the desired position.
2) **Velocity Error**:
$$e_v = \dot{x} - \dot{x}_d,$$
where $\dot{x}_d$ is the desired velocity.
3) **Attitude Error**: The error e_R between the current orientation R and the desired orientation R_d is defined using the **Lie algebra** of $SO(3)$ as:
$$e_R = \frac{1}{2}\text{vee}(R_d^T R - R^T R_d) \tag{3}$$
where **vee** extracts the vector from a skew-symmetric matrix.
4) **Angular Velocity Error**:
$$e_\Omega = \Omega - R^T R_d \Omega_d, \tag{4}$$
where Ω_d is the desired angular velocity.

3) Geometric Control Law: The geometric control law generates the thrust F and moment M required to stabilize the UAV and track the desired trajectory. The control inputs are designed as follows:

$$F = mge_3^T Re_3 - k_x e_x - k_v e_v + m\ddot{x}_d \tag{4}$$

$$M = -k_R e_R - k_\Omega e_\Omega + \Omega \times (J\Omega) \tag{5}$$

Here:

- k_x, k_v: Control gains for position and velocity errors.
- k_R, k_Ω: Control gains for attitude and angular velocity errors.
- $\ddot{x}_d$: Desired acceleration.

The thrust F acts along the body-fixed z-axis, and the control moment M stabilizes the attitude using the error metrics defined on $SO(3)$.

4) Advantages of Geometric Tracking Control: Geometric tracking control offers the following advantages over traditional control methods:

- **Global Stability**: By operating on $SO(3)$ and $SE(3)$, the controller avoids singularities and ensures global stability.
- **Robust Tracking**: The control law naturally handles large rotations and aggressive maneuvers.
- **Precision and Robustness**: The nonlinear formulation accounts for real-world uncertainties and disturbances.

[16]

C. Quaternions

One of the most used technique to model the dynamics is the Newton Euler equations which gives the linear and angular dynamics and for the angular dynamics, it employs Euler Angle representation to represent the rotations. Most of the dynamics implementations use Euler angles as they are easy to interpret and understand and it is one of the most common techniques when working with rotations. While Newton Euler this is considered fundamental, it has three drawbacks due to the Euler Angles [17].

1) Gimbal Locking [18]
2) Computation Complexity due to sines and cosines.
3) Computational Complexity due to the calculations of Jacobians.

a) : These drawbacks due to the Euler Angles also reflect in the attitude control of the UAVs whihc are given as:

1) **Singularities in Euler Angles:**
   - Traditional attitude representations, such as Euler angles, are prone to singularities as mentioned above. This issue arises when two rotational axes align, leading to a loss of a degree of freedom. Such singularities can cause control failures during aggressive rotations or large angular displacements.
2) **Computational Complexity of Direction Cosine Matrices (DCM):**
   - While DCMs (rotation matrices) avoid singularities, they require 9 parameters to represent a 3D orientation. This redundancy increases computational overhead, especially in real-time UAV applications.
3) **Precision During Large Rotations:**
   - For aggressive maneuvers involving large angular displacements, maintaining smooth and accurate control is challenging, particularly with noisy sensor data.

b) : A way to avoid these drawbacks is to use the quaternions approach. A quaternion is used to represent rotations in 3 dimensions but instead of 3 components it has 4 and its norm should always be 1. The quaternion approach addresses the challenges in attitude control as follows

1) **Quaternion Representation:**
   - Quaternions provide a singularity-free representation of orientation with only 4 parameters, reducing computational complexity compared to DCMs.
2) **Dynamic Model in Quaternion Space:**
   - By working directly in quaternion space, the control avoids the need for conversions to other representations, ensuring precision and efficiency.
3) **Feedback Linearization for Smooth Control:**
   - The paper proposes a feedback control law that directly manipulates the quaternion dynamics to achieve desired orientation and angular velocity.

1) Quaternion Mathematics: A quaternion is a hyper complex number of rank 4, which can be represented in many ways. The most common representation is

$$\mathbf{q} = q_0 + q_1\, i + q_2\, j + q_3\, k \tag{6}$$

$$\mathbf{q} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T \tag{7}$$

The quaternion units from q_1 to q_3 are called the vector part of the quaternion, while q_0 is the scalar part. The multiplication is given as

$$\mathbf{p} \otimes \mathbf{q} = \begin{bmatrix} p_0q_0 - p_1q_1 - p_2q_2 - p_3q_3 \\ p_0q_1 + p_1q_0 + p_2q_3 - p_3q_2 \\ p_0q_2 - p_1q_3 + p_2q_0 + p_3q_1 \\ p_0q_3 + p_1q_2 - p_2q_1 + p_3q_0 \end{bmatrix} \tag{8}$$

The norm of the quaternion is similar to the norm of a vector

$$\text{Norm}(\mathbf{q}) = \|\mathbf{q}\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \tag{9}$$

The conjugate and the inverse of the quaternions are given as

$$\text{Conj}(\mathbf{q}) = \mathbf{q}^* = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \end{bmatrix}^T \tag{10}$$

$$\text{Inv}(\mathbf{q}) = \mathbf{q}^{-1} = \frac{\mathbf{q}^*}{\|\mathbf{q}\|^2} \tag{11}$$

The derivative of the quaternion $\dot{\mathbf{q}}$ given the quaternion $\mathbf{q}$ and the angular velocity vector ω is given as

$$\dot{\mathbf{q}}_\omega(\mathbf{q}, \boldsymbol{\omega}) = \frac{1}{2}\, \mathbf{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \tag{12}$$

One of the significant transformations when using quaternions are the Euler to quaternion and quaternion to Euler transformations.

The quaternion $\mathbf{q}$, obtained from the Euler angles ϕ, θ, ψ (roll, pitch, yaw), is given by:

$$\mathbf{q} = \begin{bmatrix} \cos(\phi/2)\cos(\theta/2)\cos(\psi/2) + \sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \sin(\phi/2)\cos(\theta/2)\cos(\psi/2) - \cos(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\sin(\theta/2)\cos(\psi/2) + \sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\cos(\theta/2)\sin(\psi/2) - \sin(\phi/2)\sin(\theta/2)\cos(\psi/2) \end{bmatrix} \tag{13}$$

The Euler angles ϕ, θ, ψ, obtained from a quaternion $\mathbf{q} = [q_0, q_1, q_2, q_3]^T$, are given by:

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \text{atan2}\left(2(q_0q_1 + q_2q_3), q_0^2 - q_1^2 - q_2^2 + q_3^2\right) \\ \text{asin}\left(2(q_0q_2 - q_3q_1)\right) \\ \text{atan2}\left(2(q_0q_3 + q_1q_2), q_0^2 + q_1^2 - q_2^2 - q_3^2\right) \end{bmatrix} \tag{14}$$

2) Quadrotor Modeling Using Quaternions: In order to model the drone dynamics using quaternions we start with standard base assumptions. The structure and propellers are assumed rigid and symmetrical, the frame center is assumed to coincide with the center of gravity, the axes are set according to figure 1 and only differential forces from propeller rotation are considered to effect the drone rotation. If we use the Newton-Euler equations for translational and rotational dynamics, then the dynamics of the system can be represented using forces F, torques τ, and angular velocities ω. The equations are given as follows:

The force-torque relationship is described as:

$$\begin{bmatrix} F \\ \tau \end{bmatrix} = \begin{bmatrix} m & 0 \\ 0 & I_{cm} \end{bmatrix} \begin{bmatrix} a_{cm} \\ \dot{\omega} \end{bmatrix} + \begin{bmatrix} 0 \\ \omega \times (I_{cm} \cdot \omega) \end{bmatrix} \tag{15}$$

where:

- F represents the external force acting on the system.
- τ is the torque acting on the system.
- m is the mass of the system.
- I_{cm} is the inertia matrix about the center of mass.
- a_{cm} is the linear acceleration of the center of mass.
- $\dot{\omega}$ is the angular acceleration.
- $\omega \times (I_{cm} \cdot \omega)$ represents the gyroscopic torque caused by rotational motion.

The angular velocity ω is defined as:

$$\omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{16}$$

Here, $\omega_x, \omega_y, \omega_z$ are the components of the angular velocity vector along the x-, y-, and z-axes, respectively.

Then we combine the right hand quaternion derivative from (12) with the rotation dynamics from (19), that results in an equation system describing the entire rotation dynamics of the quadcopter in quaternion form [17]

$$\dot{\mathbf{q}} = -\frac{1}{2}\mathbf{q} \otimes \boldsymbol{\omega}_{\text{quat}} \tag{17}$$

$$\boldsymbol{\omega}_{\text{quat}} = [0; \boldsymbol{\omega}] \tag{18}$$

$$\dot{\boldsymbol{\omega}} = J^{-1}\left(\boldsymbol{\tau} - \boldsymbol{\omega} \times (J\boldsymbol{\omega})\right) \tag{19}$$

III. Implementation of Drone Control

A. Test Trajectories

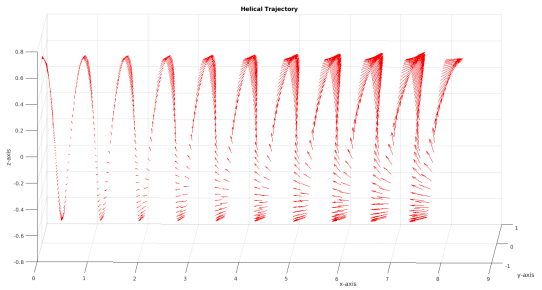


Fig. 1. Helical Trajectory

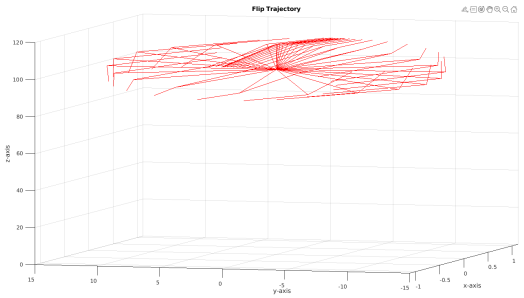


Fig. 2. Flip Maneuver

IV. System Level Description

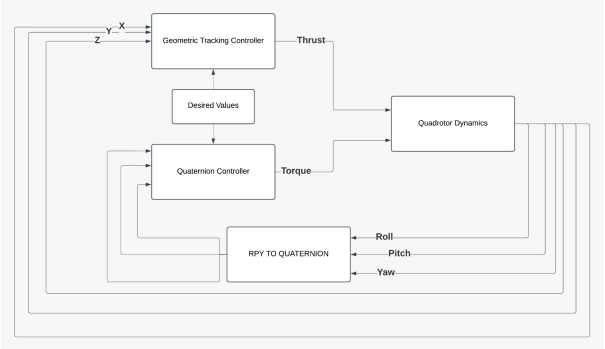


Fig. 3. Architecture of the Proposed Algorithm

The world and robot frames are assigned as shown in figure 4 [19]–[21].

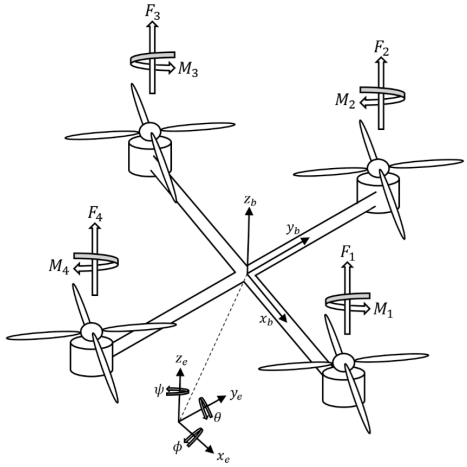


Fig. 4. Global and Robot frame assignment

Where $\{A\}$ is the global or world frame and $\{B\}$ is the robot frame whose origin is placed at the center of gravity of the drone. In the **global frame**, the linear position is ξ and the angular position is η in which the authors in [20] uses X-Y-Z Euler angles, whereas the authors in [21] have used Z-X-Y Euler angles. and a vector q which contains the linear and angular position of the quadcopter.

$$\xi = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \eta = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}, q = \begin{bmatrix} \xi \\ \eta \end{bmatrix}$$

In the **robot frame** the linear and angular velocities are given by

$$V_B = \begin{bmatrix} v_{x,B} \\ v_{y,B} \\ v_{z,B} \end{bmatrix}, v = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

We are using $X-Y-Z$ Euler angles, the rotation matrix from frame $\{A\}$ to $\{B\}$ will change.

$$R_{X-Y-Z} = \begin{bmatrix} C_\psi C_\theta & C_\psi S_\theta S_\phi - S_\psi C_\phi & C_\psi S_\theta C_\phi + S_\psi S_\phi \\ S_\psi C_\theta & S_\psi S_\theta S_\phi + C_\psi C_\phi & S_\psi S_\theta C_\phi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix}$$

The linear velocity in global frame is given as

$$v = \dot{\xi}$$

We also have a transformation matrix for angular velocities from global frame to robot frame, which is given by W_η. (this matrix is only invertable if $\theta \neq (2k-1)\phi/2, (k \in Z)$) [20].

$$v = W_\eta \dot{\eta}$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -S_\theta \\ 0 & C_\phi & C_\theta S_\phi \\ 0 & -S_\phi & C_\theta C_\phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

Assuming that the quadcopter is symmetric, the inertia matrix is given as

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

The off diagonal entries are zero and $I_{xx} = I_{yy}$.

1) Newton Euler Equations: Using the Newtonian Mechanics we derive the dynamical equations of the drone. The force on the drone is given as

$$m\dot{v}_{[3\times1]} = mg\vec{a_3}_{[3\times1]} + \mathbf{R}_{[3\times3]}F_{[3\times3]}$$

Where F is the force or thrust vector, $\mathbf{R}$ is the rotation matrix from world frame to robot frame, m is the mass of the drone, g is the gravity and $\vec{a_3}$ is the unit vector in the z direction. Expanding the equations:

$$m\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = -mg\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \mathbf{R}_{[3\times3]}\begin{bmatrix} 0 \\ 0 \\ f_1+f_2+f_3+f_4 \end{bmatrix}$$

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = -g\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \frac{1}{m}\mathbf{R}_{[3\times3]}\begin{bmatrix} 0 \\ 0 \\ F \end{bmatrix} \quad (20)$$

The angular equations are given as

$$I_{[3\times3]}\dot{v}_{[3\times1]} = -v_{[3\times1]} \times I_{[3\times3]}v + \tau_{[3\times1]}$$

This equation is derived in [22], where I is the inertia tensor of the drone, τ is the torque applied on the drone ($I\dot{v}$) is the inertia of the drone. Expanding the equation:

$$I_{[3\times3]}\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I_{[3\times3]}\begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (21)$$

Equation (1) is the equation for the linear dynamics and equation (2) is the angular dynamics. In these equations the F is the input for the linear dynamics so the altitude controller outputs the F. And τ is the input of the angular dynamics so the attitude controller outputs τ.

V. Control Algorithms

A. Geometric Tracking

The implementation of Geometric Tracking Control is demonstrated through two maneuvers: a **Flip Trajectory** and a **Helical Trajectory**. These maneuvers leverage the control laws discussed in Section 2.1 to achieve precise trajectory tracking for a quadrotor UAV. The code is divided into key sections for system parameter definition, trajectory generation, control law computation, and visualization.

1) Flip Trajectory: The **Flip Trajectory** involves performing a 360-degree backflip while maintaining stability and accurate tracking of desired states. The main steps of the implementation are as follows:

Algorithm 1 Quadrotor Flip Trajectory Generation and Control Law

Input: Initial conditions: x_0, v_0, R_0, w_0, Quadrotor parameters: $m, J, g, k_1, k_2, k_R, k_\Omega$
Initialize: Flip time parameters: $T_{\text{flip}}, T_{\text{start}}, T_{\text{end}}$
for $t = t_0$ to t_{end} with step Δt **do**
  if $t \leq T_{\text{start}}$ **then**
    Hover phase: $\phi(t) = 0, \theta(t) = 0, \psi(t) = 0$
  else if $T_{\text{start}} < t \leq T_{\text{end}}$ **then**
    Compute roll angle for flip:

$$\phi(t) = 2\pi \frac{t - T_{\text{start}}}{T_{\text{flip}}}, \quad \theta(t) = 0, \quad \psi(t) = 0$$

  else
    Stabilization phase: $\phi(t) = 2\pi, \theta(t) = 0, \psi(t) = 0$
  end if
  Compute desired thrust direction:

$$\mathbf{b}_3 = \frac{-k_1 e_x - k_2 e_v + m\mathbf{a}_{\text{des}} + mg\mathbf{e}_3}{\| \cdot \|}$$

  Compute control moments:

$$M = -k_R e_R - k_\Omega e_\Omega + \Omega \times (J\Omega)$$

  Update state variables: x, v, R, Ω
end for
Output: Trajectory $x(t), v(t), R(t), \Omega(t)$ =0

a) System Parameters: The system parameters define the physical properties of the UAV, including its mass, inertia matrix J, and gravity g, as well as control gains k_1, k_2, k_R, k_Ω for position, velocity, and attitude control.

b) Trajectory Function: The desired trajectory consists of a hover phase followed by a full backflip. The roll angle $\phi(t)$ progresses linearly during the flip, completing a full 360-degree rotation. Pitch and yaw remain zero.

c) Control Law: The control law for thrust and moments is implemented as follows:

- The thrust direction $\mathbf{b}_3$ is computed using the position and velocity errors:

$$\mathbf{b}_3 = \frac{-k_1 e_x - k_2 e_v + m\mathbf{a}_{\text{des}} + mg\mathbf{e}_3}{\| \cdot \|}.$$

- The moments M are calculated using the attitude and angular velocity errors:

$$M = -k_R e_R - k_\Omega e_\Omega + \Omega \times (J\Omega).$$

2) Helical Trajectory: The **Helical Trajectory** consists of a smooth circular path in the y-z plane with a progressive ascent, forming a helical shape. The trajectory is defined as:

$$x(t) = 0.4t, \quad y(t) = 0.4\sin(\pi t), \quad z(t) = 0.6\cos(\pi t).$$

a) Desired Trajectory: The position, velocity, and acceleration of the helical trajectory are derived analytically. The trajectory ensures smooth motion with tangential alignment.

Algorithm 2 Quadrotor Helical Trajectory Generation and Control Law

Input: Initial conditions: x_0, v_0, R_0, w_0, Quadrotor parameters: $m, J, g, k_1, k_2, k_R, k_\Omega$
Define Trajectory:

$$x(t) = 0.4t, \quad y(t) = 0.4\sin(\pi t), \quad z(t) = 0.6\cos(\pi t)$$

Compute velocity and acceleration:

$$v(t) = \frac{d}{dt}x(t), \quad a(t) = \frac{d^2}{dt^2}x(t)$$

for $t = t_0$ to t_{end} with step Δt **do**
  Compute desired thrust direction:

$$\mathbf{b}_3 = \frac{-k_1 e_x - k_2 e_v + m\mathbf{a}_{\text{des}} + mg\mathbf{e}_3}{\| \cdot \|}$$

  Compute desired rotation matrix:

$$R_d = [\mathbf{b}_2 \times \mathbf{b}_3, \mathbf{b}_2, \mathbf{b}_3]$$

  Compute control moments:

$$M = -k_R e_R - k_\Omega e_\Omega + \Omega \times (J\Omega)$$

  Update state variables: x, v, R, Ω
end for
Output: Trajectory $x(t), v(t), R(t), \Omega(t)$ =0

b) Control Law: The control law aligns the UAV's nose tangentially to the trajectory by computing the desired rotation matrix R_d as follows:

$$R_d = [\mathbf{b}_2 \times \mathbf{b}_3, \mathbf{b}_2, \mathbf{b}_3],$$

where $\mathbf{b}_3$ is the desired thrust direction, and $\mathbf{b}_2$ is computed to ensure orthonormality.

B. Quaternion-Based Attitude Control

The quaternion-based attitude controller employs the quaternion representation to track a reference trajectory while stabilizing angular velocity. This approach ensures a singularity-free attitude control solution, suitable for high-precision applications such as quadrotor UAVs.

1) Control Law: The control law minimizes the orientation error using a proportional controller on the quaternion error and angular velocity. The torque command $\boldsymbol{\tau}$ is given as:

$$\boldsymbol{\tau} = -K_q \mathbf{q}_{\text{err,vec}} - K_\omega \boldsymbol{\omega},$$

where $\mathbf{q}_{\text{err,vec}}$ is the vector part of the quaternion error, and K_q, K_ω are proportional gains for orientation and angular velocity control, respectively.

2) System Dynamics: The quaternion kinematics and angular velocity dynamics evolve as follows:

$$\dot{\mathbf{q}} = -\frac{1}{2}\mathbf{q} \otimes \boldsymbol{\omega}_{\text{quat}}, \quad \boldsymbol{\omega}_{\text{quat}} = [0; \boldsymbol{\omega}], \tag{22}$$

$$\dot{\boldsymbol{\omega}} = J^{-1}\left(\boldsymbol{\tau} - \boldsymbol{\omega} \times (J\boldsymbol{\omega})\right). \tag{23}$$

The quaternion is normalized at each time step to ensure numerical stability.

Algorithm 3 Quaternion-Based Attitude Control Law

0: **Input:** Initial conditions $\mathbf{q}_0, \boldsymbol{\omega}_0$, Quadrotor parameters J, K_q, K_ω, and reference trajectory $\mathbf{q}_{\text{ref}}(t), \boldsymbol{\omega}_{\text{ref}}(t)$

0: **Initialize:** Time step Δt

0: **for** $t = t_0$ to t_{end} with step Δt **do**

0: Compute quaternion error:

$$\mathbf{q}_{\text{err}} = \mathbf{q}_{\text{ref}} \otimes \mathbf{q}^*$$

0: Extract vector part of quaternion error:

$$\mathbf{q}_{\text{err,vec}} = \begin{bmatrix} q_1^{\text{err}} \\ q_2^{\text{err}} \\ q_3^{\text{err}} \end{bmatrix}$$

0: Compute control torque:

$$\boldsymbol{\tau} = -K_q \mathbf{q}_{\text{err,vec}} - K_\omega \boldsymbol{\omega}$$

0: Update angular velocity:

$$\dot{\boldsymbol{\omega}} = J^{-1} \left(\boldsymbol{\tau} - \boldsymbol{\omega} \times (J\boldsymbol{\omega}) \right)$$

0: Integrate angular velocity:

$$\boldsymbol{\omega}(t + \Delta t) = \boldsymbol{\omega}(t) + \Delta t \cdot \dot{\boldsymbol{\omega}}$$

0: Update quaternion kinematics:

$$\dot{\mathbf{q}} = -\frac{1}{2} \mathbf{q} \otimes \boldsymbol{\omega}_{\text{quat}}, \quad \boldsymbol{\omega}_{\text{quat}} = [0; \boldsymbol{\omega}]$$

0: Normalize quaternion:

$$\mathbf{q}(t + \Delta t) = \frac{\mathbf{q}(t) + \Delta t \cdot \dot{\mathbf{q}}}{\|\mathbf{q}(t) + \Delta t \cdot \dot{\mathbf{q}}\|}$$

0: **end for**

0: **Output:** Quaternion trajectory $\mathbf{q}(t)$, angular velocity $\boldsymbol{\omega}(t)$, and torque $\boldsymbol{\tau}(t)$ =0

3) Trajectory Tracking: The controller is tested on the following reference trajectories:

- **Step Trajectory:** Discrete steps in roll (ϕ), pitch (θ), and yaw (ψ).
- **Flip Trajectory:** Smooth 360° roll over a defined duration.
- **Circular Trajectory:** Constant angular velocity resulting in rotation in the xy-plane.

4) Simulation Results: The simulation evaluates the controller's performance through the following metrics:

1) **Angular Velocity:** Evolution of $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]$.
2) **Control Torque:** Torque components τ_x, τ_y, τ_z over time.
3) **Orientation Tracking:** Actual Euler angles (ϕ, θ, ψ) compared against the reference trajectory.

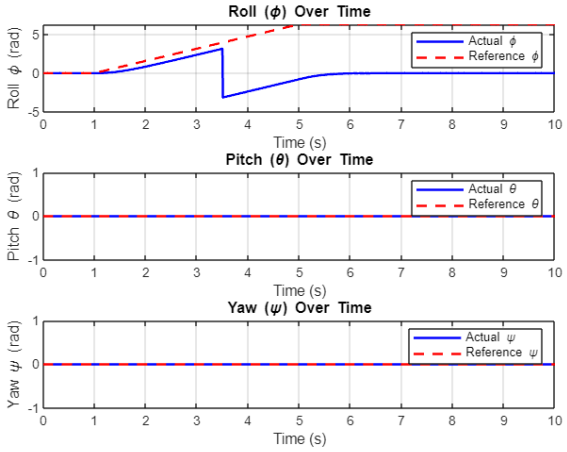


Fig. 5. Roll, Pitch, and Yaw over Time

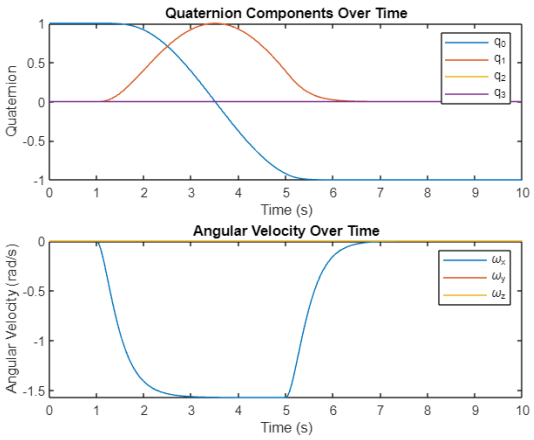


Fig. 6. Quaternion & Angular Velocity

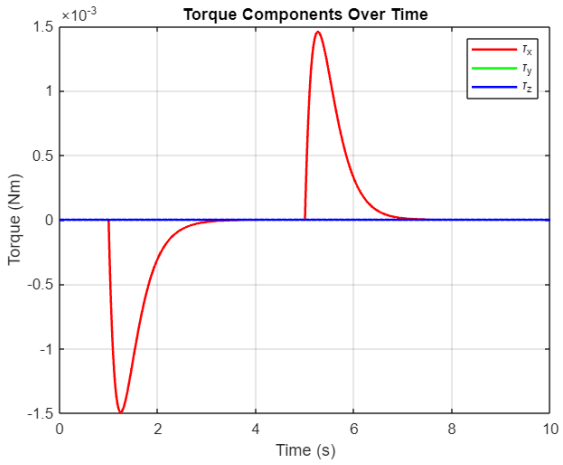


Fig. 7. Torque Components

VI. Proposed Algorithm

The proposed algorithm is the combination of the both the *Geometric Tracking Control* and the *Full Quaternion based*

Attitude Control. As mentioned above the classical algorithms fail when they track the aggressive maneuvers presented in the above section. Therefore combination of two robust controllers for altitude and attitude served the purpose of tracking an aggressive maneuver trajectory with great accuracy. The implementation details of the controller are given in the following section.

A. Implementation

This study was about implementing a controller that can track aggressive maneuvers with great accuracy and avoid instability. The desired trajectory is provided in the form of position and orientation, therefore the desired trajectory x_d is given as

$$x_d = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \ddot{x} & \ddot{y} & \ddot{z} & \psi \\ \theta & \phi & & & & & & & & \end{bmatrix}^T$$

Two independent controllers are implemented in the proposed algorithm which outputs F and τ, two control inputs for the drone dynamics. The altitude controller is based on geometric tracking controller. The equation for altitude controller is given as

$$F = -mg - k_x e_x - k_v e_v + m\ddot{x}_d \tag{24}$$

The attitude controller is based on the full quaternion based attitude control which converges the error to zero. The attitude controller implementation is give as

$$\tau = -K_q \begin{bmatrix} q_1^{err} \\ q_2^{err} \\ q_3^{err} \end{bmatrix} - K_\omega \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

The complete implementation of the algorithm is given as follows

Algorithm 4 Quadrotor Geometric Tracking Quaternion based control

Input: $x_t, \dot{x}_t, \ddot{x}_t$, q_{ref}, ω, $\mathbf{x}_0$, $\dot{\mathbf{x}}_0$
Initialize: Δt, m, g, I, K_x, K_v, K_q, K_ω
for $t = t_0$ to t_{end} with step Δt **do**
  $\mathbf{e}_x = \mathbf{x}_t - \mathbf{x}, \quad \mathbf{e}_v = \dot{\mathbf{x}}_t - \dot{\mathbf{x}}$
  $q_{\text{err}} = q_{\text{error}}(q_{\text{ref}}, q)$
  $\boldsymbol{\tau} = \text{att\_controller}(q_{\text{err}}, \omega, K_q, K_\omega)$
  if $4 < t < 8$ **then** $\mathbf{F} = [0; 0; 0]$
  else$\mathbf{F} = \text{alt\_controller}(\mathbf{e}_x, \mathbf{e}_v, K_x, K_v, m, g, \ddot{\mathbf{x}}_t)$
  end if
  $R = \text{Rot\_matrix}(\text{quaternion\_to\_euler}(q))$
  $\ddot{\mathbf{x}} = [0; 0; -g] + \frac{1}{m} R\mathbf{F}$
  $\dot{\omega} = I^{-1}(\boldsymbol{\tau} - \omega \times I\omega)$
  $\omega \leftarrow \omega + \Delta t \cdot \dot{\omega}$
  $\dot{q} = -0.5 \cdot \text{q\_mult}([0; \omega], q)$
  $q \leftarrow q + \Delta t \cdot \dot{q}, \quad q \leftarrow q/\|q\|$
  $\dot{\mathbf{x}} \leftarrow \dot{\mathbf{x}} + \Delta t \cdot \mathbf{a}, \quad \mathbf{x} \leftarrow \mathbf{x} + \Delta t \cdot \dot{\mathbf{x}}$
end for
Output: $\mathbf{x}(t), \omega(t), q(t)$ =0

VII. Results

The proposed algorithm was implemented in MATLAB on different aggressive manuvers. The controller gains were experimentally found which are given in Table I.

Gain	Value
K_q	0.025
K_ω	0.007
K_x	$\begin{bmatrix} 7 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 100 \end{bmatrix}$
K_v	$\begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{bmatrix}$

TABLE I
Controller Gains

A. Flip Maneuver

The controller was tested in the flip maneuver mentioned in section 3 the results as as follows

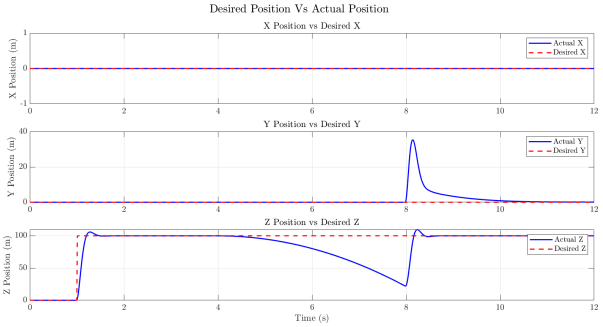


Fig. 8. Actual Vs Desired trajectory

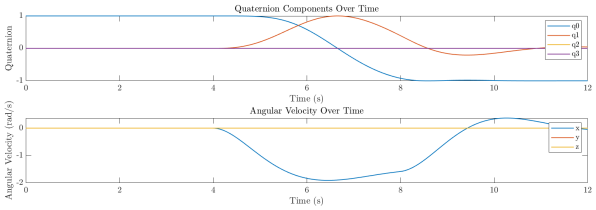


Fig. 9. Quaternions and Angular Velocities

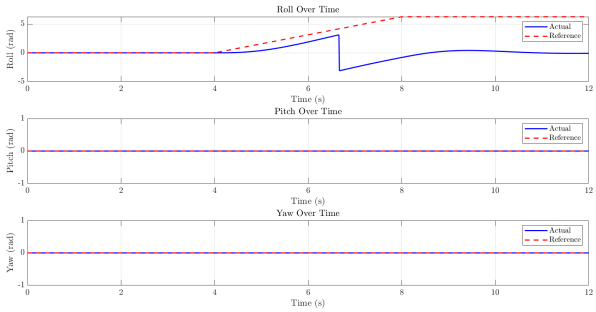


Fig. 10. Attitude Control

VIII. Conclusion

This paper presented a hybrid control framework for quadrotor UAVs, combining Geometric Tracking Control (GTC) on $SE(3)$ for position and altitude tracking and Quaternion-Based Attitude Control for precise orientation tracking. The proposed approach addresses the limitations of classical controllers like PID and LQR, which struggle with aggressive maneuvers and large angular displacements. By leveraging the mathematical rigor of $SE(3)$ and quaternions, the hybrid controller ensures global stability, singularity-free operation, and robust performance under dynamic conditions.

The effectiveness of the proposed framework was demonstrated through simulations on two aggressive maneuvers: a 360-degree flip and a helical trajectory. The controller showed precise trajectory tracking with minimal position and velocity errors, and it maintained stable attitude control during high-rotation maneuvers. Simulation results confirmed that the hybrid approach effectively balances the complementary strengths of geometric and quaternion-based control, achieving smooth and robust control of UAV dynamics.

The study highlights the potential for applying this control framework to real-world scenarios, including high-precision drone applications and tasks requiring aggressive maneuvering. Future work could focus on real-time implementation, handling external disturbances, and extending the framework to cooperative multi-UAV systems. Additionally, integrating advanced learning-based approaches with the hybrid controller could further enhance its adaptability and robustness.

IX. Future Work

While the proposed hybrid control framework combining Geometric Tracking Control (GTC) and Quaternion-Based Attitude Control demonstrated robust performance for aggressive UAV maneuvers, there are several directions for future research and improvement:

- **Real-Time Implementation:** The proposed control framework was tested in simulations, and the next logical step is to implement it on physical quadrotor systems. Real-time implementation will help evaluate its robustness under hardware constraints, sensor noise, and real-world disturbances.
- **Handling External Disturbances:** Future work could focus on extending the controller to handle external disturbances such as wind, payload variations, and collisions. This can be achieved through robust adaptive control techniques or disturbance observer-based methods.
- **Multi-UAV Systems:** Extending the framework to cooperative multi-UAV systems is an exciting direction. The control framework could be adapted for tasks such as formation flying, cooperative trajectory tracking, and collision avoidance in dynamic environments.
- **Learning-Based Control Enhancements:** Integrating learning-based approaches, such as reinforcement learning or neural networks, with the hybrid control framework could further improve its adaptability. These techniques could be used to optimize controller gains or adapt the control laws to varying conditions in real time.
- **Trajectory Optimization:** Investigating advanced trajectory planning algorithms to generate optimal reference trajectories for specific applications, such as inspection, delivery, or surveillance, could enhance the overall system efficiency.
- **Energy Efficiency:** Future work could focus on minimizing the energy consumption of the quadrotor during aggressive maneuvers by optimizing the thrust allocation and torque distribution.
- **Integration with Onboard Perception Systems:** Incorporating onboard sensors and perception systems (e.g., cameras, LiDAR) could enable autonomous navigation and environment interaction, enhancing the usability of the framework for real-world scenarios.

By addressing these areas, the hybrid control framework can be further developed into a comprehensive solution for a wide range of UAV applications, including high-precision tasks, cooperative operations, and autonomous navigation in dynamic and uncertain environments.

References

[1] M. Valenti, B. Bethke, G. Fiore, and J. How, "Indoor multi-vehicle flight testbed for fault detection, isolation, and recovery," in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, 2006.

[2] P. Pounds, R. Mahony, and P. Corke, "Modeling and control of a large quadrotor robot," *Control Engineering Practice*, vol. 18, pp. 691–699, 2010.

[3] G. Hoffmann, H. Huang, S. Waslander, and C. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, no. AIAA 2007-6461, 2007.

[4] P. Castillo, R. Lozano, and A. Dzul, "Stabilization of a mini rotorcraft with four rotors," *IEEE Control System Magazine*, pp. 45–55, 2005.

[5] "Mikrokopter," http://www.mikrokopter.de/.

[6] "Microdrone-bulgaria," http://www.microdrones-bulgaria.com/.

[7] "Dragonfly innovations," http://www.draganfly.com/.

[8] S. Bouabdalla, P. Murrieri, and R. Siegward, "Towards autonomous indoor micro vtol," *Autonomous Robots*, vol. 18, no. 2, pp. 171–183, 2005.

[9] E. Nice, "Design of a four rotor hovering vehicle," Master's thesis, Cornell University, 2004.

[10] T. Bresciani, "Modelling, identification and control of a quadrotor helicopter," Ph.D. dissertation, Lund University, 2008.

[11] V. Jurdjevic, *Geometric Control Theory*. Cambridge University, 1997.

[12] S. Bhat and D. Bernstein, "A topological obstruction to continuous global stabilization of rotational motion and the unwinding phenomenon," *Systems and Control Letters*, vol. 39, no. 1, pp. 66–73, 2000.

[13] D. Maithripala, J. Berg, and W. Dayawansa, "Almost global tracking of simple mechanical systems on a general class of lie groups," *IEEE Transactions on Automatic Control*, vol. 51, no. 1, pp. 216–225, 2006.

[14] E. Stingu and F. Lewis, "Design and implementation of a structured flight controller for a 6dof quadrotor using quaternions," in *17th Mediterranean Conference on Control and Automation*, Thessaloniki, Greece, 2009.

[15] K. A. W. J.-Y. Joshi and S.M., "Robust attitude stabilization of spacecraft using nonlinear quaternion feedback," *IEEE Transactions on Automatic Control*, vol. 40, 1995.

[16] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in *49th IEEE conference on decision and control (CDC)*. IEEE, 2010, pp. 5420–5425.

[17] E. Fresk and G. Nikolakopoulos, "Full quaternion based attitude control for a quadrotor," in *2013 European control conference (ECC)*. IEEE, 2013, pp. 3864–3869.

[18] E. G. Hemingway and O. M. O'Reilly, "Perspectives on euler angle singularities, gimbal lock, and the orthogonality of applied forces and applied moments," *Multibody system dynamics*, vol. 44, pp. 31–56, 2018.
[19] V. Kumar, "Robotics: Aerial robotics," https://www.coursera.org/learn/robotics-flight, n.d. [Online]. Available: https://www.coursera.org/learn/robotics-flight?irgwc=1&utm_medium=partners&utm_source=impact&utm_campaign=2985301&utm_content=b2c&gad_source=1&irclickid=UsGxJ5yMqxyKTgiwqJ3ZPwJeUkCROsW8q2SFx00
[20] T. Luukkonen, "Modelling and control of quadcopter," *Independent research project in applied mathematics, Espoo*, vol. 22, no. 22, 2011.
[21] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE robotics & automation magazine*, vol. 19, no. 3, pp. 20–32, 2012.
[22] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. John Wiley & Sons, 2020.