Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 3044520, 17 pages
https://doi.org/10.1155/2022/3044520

Research Article

A Novel Framework of Modelling, Control, and Simulation for Autonomous Quadrotor UAVs Utilizing Arduino Mega

Hoang T. Tran,[1,2] Dong L. T. Tran,[1] Vinh Q. Nguyen,[3] Hai T. Do,[4] and Minh T. Nguyen[4]

[1]*Center of Electrical Engineering, Duy Tan University, Danang, 550000, Vietnam*
[2]*Faculty of Electrical-Electronic Engineering, Duy Tan University, Danang, 550000, Vietnam*
[3]*Academy of Military Science and Technology, Hanoi 100000, Vietnam*
[4]*Thai Nguyen University of Technology, Thainguyen, 240000, Vietnam*

Correspondence should be addressed to Minh T. Nguyen; tuanminh.nguyen@okstate.edu

Received 2 March 2022; Revised 4 August 2022; Accepted 8 August 2022; Published 19 August 2022

Academic Editor: Fuliang Li

In recent decades, there has been a constant increase in the use of unmanned aerial vehicles (UAVs). There has also been a huge growth in the number of control algorithms to support the many applications embodied by the vehicles, including challenges and open issues to develop. This paper focuses on three major classes of control methods applied to quadrotors in order to create an open-source model based on the Arduino Mega that allows for the derivation and design of quadrotor control strategies. We consider the perspective classes, including linear, nonlinear, and intelligent methods representing in details with applications in developing an open-source controller for the quadrotor using the Arduino Mega and the BNO055 9 DOF sensor. We propose Proportional Integral Derivative (PID), backstepping integrator, and model predictive control (MPC) to track a generated Lissajous curve for surveillance. Simulations in the Matlab–Simulink environment with 3D visualization of a developed quadrotor model using CAD software, with robustness and performance discussion, are provided. Our experimental work is developed with an extensive illustration of the hardware and algorithm design and by demonstrating the effectiveness of the proposed architectures. The results show promise in practical and in intelligent applications.

1. Introduction

In recent years, unmanned aerial vehicles (UAVs) have been facilitating many different applications in both military and civilian fields [1, 2]. The domain has received countless research for academic work. Different kinds of tasks, such as monitoring, detecting and collecting data, relaying data, and carrying goods, are suitable for UAVs, especially quadrotor UAVs [3–6]. The use of UAVs has become a sort of indispensable task in many sectors, leading to a massive investment of time, money, and technology to develop algorithms and strategies for sophisticated systems of control [7, 8].

Quadrotor UAVs are good platforms for control systems research because of their nonlinear nature and underactuated designs, which make them suitable for the synthesis and analysis of control algorithms. A quadrotor is a complete manoeuvrable underactuated UAV with six degrees of freedom (DOF) [9]. It is energized by four rotors with fixed pitch propellers that rotate at different controlled speeds. In addition, the quadrotor dynamics are highly nonlinear regardless of the uncertainties of modelling, identification, and design parameterization, and despite neglecting the disturbance of the environment and noise of actuators and sensors. To address the control challenge, diverse approaches have been considered in the literature and research about the control philosophy of quadrotors [10].

Advanced control theory contributed significantly to overcoming the difficulties in a variety of designs of autonomous vehicles. Adaptive designs and platforms were more developed for the aspects of specific tasks and applications. Early works about quadrotor development focused just on a reliable control [11]. Conversely, recent papers present a deeper sight of tasks and control performance such as 3D mapping [12], simultaneously localization and mapping

(SLAM) [13], trajectory generation, tracking [14–17], sparse and swarm control [18, 19], augmented and hybrid systems, and autonomous landing [20, 21].

Generally, the control of quadrotors was streamed in three main axes linear, nonlinear, and intelligent methods. For the linear methods, PID controller was the essential quadrotor control approach [22]. Miranda-Colorado and Aguilar presented a novel robust PID control methodology for controlling a quadrotor UAV with a procedure for reducing the power [23]. LQR for stability and tracking was proven functional and robust to perturbation in [24]. Another type of linear control was represented in H infinite, which assured robustness to the uncertainties of modelling [25]. Nonlinear approaches were alternated between many topologies, and a model inversion strategy was applied for a quadrotor with a suspended load in [26]. The hybrid control philosophy was opted by creating two nested control loops. Inner loop for load position control and an outer loop for quadrotor general position control, feedback linearization was opted by many researchers as its simplest approach, although it can show instability in many cases due to noninvertibility. In [27], the simulation proved the applicability of the dynamic inversion model for the control. The backstepping method was used in a variety of papers as it is not that complicated in implementation on hardware. Bhargavapuri et al. proceeded with a flip manoeuvre of a quadrotor using the backstepping approach with a gain update [28]. The backstepping method was also used by Koksal et al. to guarantee UAV's trajectory tracking [29]. The sliding mode was subjected by many researchers and showed a robust control and tracking, as demonstrated in [30]. Another approach was based on using diagonal recurrent neural network (DRNN) for updating PID gains in [22].

The majority of quadrotors designs take advantage of the extended Kalman filter (EKF) for optimal full state vector estimation from measurements, especially in autonomous navigation under disturbance or in the presence of white Gaussian noise, as shown in [31, 32]. Guo et al. experimented with an EKF observer to enhance PID based H infinity control of multiple quadrotors [20], an evolutionary algorithm with principles of KF to tune unknown noises was optimized in [33], and other researchers have used virtual sensing based on EKF observer to develop a quadrotor controller [22]. Trajectory tracking and obstacle avoidance were tackled by many quadrotor projects. Pérez-Alcocer and Moreno-Valenzuela considered a model-based optimization of trajectory. The performance of the proposed method was compared with respect to three different classical known trajectory tracking controllers [15]. Tracking of trajectory subject to uncertain inertial and mass parameters was considered in [14]. Other researchers opted for a trajectory generation first using quadratic programming with linear constraints. Then, the control of tracking was decided in a closed-loop scheme with double integrator Lyapunov control, following minimal SNAP (4th derivative) and minimal JERK (3rd derivative) [16]. Backstepping also was developed for trajectory tracking with avoidance of singularities by emphasizing the quaternion model [17]. The path following problem is defined when implementing a controller that tracks trajectory without time referencing. Those controllers discuss more the feasibility of the realization of stable following along the generated path. Thanh et al. demonstrated the realizability of optimal trajectory tracking of nonlinear dynamical systems [34].

Trajectory optimization had another research area where researchers worked on control of multiple and swarm of quadrotors. The problem of distributed finite-time formation tracking control with collision avoidance was investigated for multiple quadrotor UAVs subject to external disturbances by Huang et al. focus was on mounting a challenge of disturbance estimation with a sliding mode control (SMC) to follow the desired trajectory generated by optimal control problem (OCP) as they proved Lyapunov stability [33]. Borkar et al. exploited Lissajous curves for preplanning of trajectories for reconfigurable formations of quadrotors for surveillance applications [19]. Another group of researchers tackled the zone search optimization of a quadrotor's flock by adhering to a bioinspired methodology and considering the interaction between quadcopters to avoid obstacles and collision using PID control applied to the swarm algorithm. The authors proved the advantage of multiobject navigation optimization of a dynamic model with other techniques [18]. Others regarded the multiple quadrotor cooperative control as subject to aerodynamics drag and position deviations. The paper written by Yue et al. suggested a leader drone to be followed based on adaptive SMC [35]. The same leader-followers philosophy was maintained by Liu et al. in [36], where they opted for visually serving formation tracking control for quadrotor UAV team. The authors studied the following patterns, cooperative work, and a sudden change of leader process.

Navigation in an unknown environment for quadrotors was studied and well-referred as guidance, navigation, and control (GNC) algorithms. GNC got some research contributions, vision-based localization was used as well as an option for 3D positioning [36], building a 3D maps using SLAM was also simulated by Jiang et al. for a quadrotor in Gazebo data set combined by robot operating system (ROS) [12], and SLAM combined with state estimation through inertial navigation system (INS) was also researched, for bioinspired autonomous landing using a Kinect camera [13]. Another study was about the stereovision for a quadrotor to avoid single and multiple obstacles defined in the ellipsoidal bounding box. The stereotype vision was about detecting and localizing obstacles, and an ellipsoid bounding was generated to envelop the obstacles. The cone of possible contact was then calculated with consideration of minimal distance to the ellipsoid to avoid contact. Convex optimization later was designed to generate optimal trajectory [37].

Recently, the neural network (NN) methods have been used in quadrotors for identification and obstacle avoidance. Structure from motion (SFM) with SLAM enabled by reinforcement learning-based approaches considering a CNN scheme for obstacle avoidance was studied in [13], and DRNN for updating PID gains strategy was implemented successfully in [22].

Different from the previous work, in this paper, we focus more on practical approaches in controlling the UAVs. Our

methods can be considered as a combination of different techniques to support the UAVs. We present the mathematic models, consider all the motion of the UAVs, apply advanced techniques, formulate and design controllers, build experimental work, and collect/adjust data from real UAVs. Noise and disturbance are considered by the high and low sampling times of white Gaussian noise to simulate the sensor's noise and wind gusts. The existing papers consider or implement some points separately, such as PID control [8, 22], 3D mapping [12], obstacle avoidance [13, 20], tracking trajectories [14–17], and surveillance optimization [19, 21]. Compared to existing work, we do not focus on only one or two charateristics but almost major problems to be able to support UAVs in real applications. In short, our contributions are addressed as follows.

(1) Detailed derivation of the quadrotor modelling and development of three methods of control with all schemes, algorithms, and simulations in the presence of wind disturbance and actuators and sensors noises

(2) Implementation and realization of quadrotor model with some algorithms

(3) Development of a 3D simulation environment to visualize the quadrotor flight results to ease the perception of control effectiveness

(4) Codes, Simulink models, and programs are publicly available for researchers to be exploited as a benchmark for the studies

The rest of the paper is organized as follows. The dynamics of UAVs are modelled in Section 2 with both math models and illustrations. Simulation processes are presented in Section 3. Some developments are provided with details in Section 4. Finally, conclusions and future developments are presented in Section 5.

Table 1: Symbol definition.

Symbol	Definition
ϕ	Roll
θ	Pitch
ψ	Yaw
J_{TP}	Inertia
B	Thrust coef
D	Drag coef
M	Mass
$I_{3\times3}$	Identity matrix
T_i	Motor torque i^{th}
$\ddot{T}^E$	Acceleration
F^E	Forces
$R\Theta$	Rotation matrix
$T\Theta$	Transformation matrix
Ω	Rotor speed
l	Lever
g	Gravity
I	Inertia matrix
$0_{3\times3}$	3×3 zero matrix
$\dot{V}^B$	Linear accel
$I_{xx,}\, I_{yy}, I_{zz}$	Diagonal inertia

2. Modelling of Dynamics

In this section, the mathematical model of the quadrotor is developed utilizing the Newton Euler formalism. Hubb forces or ground effects are not considered due to the validation of the model by many researchers. Used symbols are defined in Table 1.

2.1. Reference Frame and Coordinate System Transformation Matrix. To describe the movements of the UAVs, the following two reference frames are used as follows.

Inertial reference frame $\{E\}$ is a frame of reference without acceleration in which Newton's laws are satisfied. The coordinate system associated with this frame has the origin of coordinates (symbolized by OE) attached to an object that is fixed relative to Earth. The position of the centre of gravity of the drone in the inertial coordinate system is represented by vector $\xi^E = (x, y, z)$. The vector $\Theta^E = (\phi, \theta, \psi)$ describes the direction of the device. The vector Φ is used to describe the position and orientation of the device in the inertial coordinate system: $\Phi = [\xi^E \Theta^E]^T$.

The reference system attached to the drone [38] is a fixed frame of reference which moves along with the drone. The origin OB of the coordinate system attached to the object is taken to coincide with the centre of gravity of the device. The coordinate axes are denoted by x_B, y_B, z_B. The vector $v^B = [u \quad v \quad w]^T$ and $\omega^B = [p \quad q \quad r]^T$ describe the linear and angular velocity vectors of the device, respectively. The dimensional convention of reference systems is shown in Figure 1.

Rotation matrix ${}^E_B R(\phi, \psi, \theta)$ converts from an object–mounted coordinate system to inertial coordinate system [39]:

$$\frac{E}{B}R(f,\theta,\Psi) = R_Z(\Psi)R_Y(\theta)R(\phi) = \begin{bmatrix} \cos\theta\cos\Psi & \sin\phi\sin\theta\cos\Psi - \cos\phi\sin\Psi & \cos\phi\sin\theta\cos\Psi + \sin\phi\sin\Psi \\ \cos\theta\sin\Psi & \sin\phi\sin\theta\sin\Psi + \cos\phi\cos\Psi & \cos\phi\sin\theta\sin\Psi - \sin\phi\cos\Psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}. \quad (1)$$

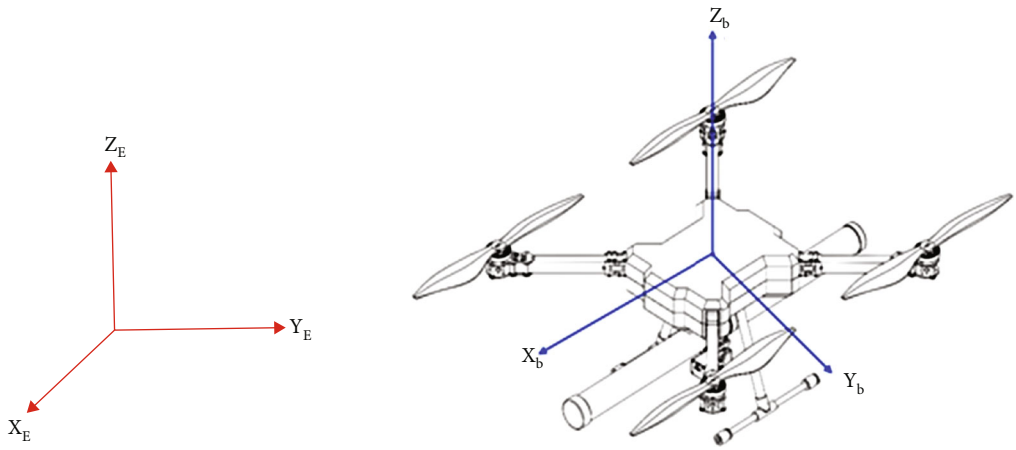


Figure 1: Inertial reference frame and reference system attached to the drone.

Note: the coordinate transformation matrix is orthogonal, so ${}^{E}_{B}R^{-1} = {}^{B}_{E}R = {}^{E}_{B}R^{T}$.

2.2. Description of Device Movement. The drone is designed with two pairs of reversible propellers, of which two propellers rotating on the same side are arranged opposite each other. By adjusting the speeds of these four propellers reasonably, the drone will make the movements and navigation as required. The thrust of propellers is calculated as follows.

$$F_i = c_t \omega_i^2, \tag{2}$$

where c_t is the thrust coefficient of the propellers (Ns^2); ω_i is the angular velocity of the i^{th} propeller (rad/s). The movements of the UAVs are divided into the following motions.

2.2.1. Suspension Motion (Keeping Balance at a Certain Height) or Height Rise/Fall. To perform suspension motion, the vehicle must control the propellers to rotate at the same speed in order to create a thrust balanced with gravity and, at the same time, suppresses the moments causing rotation. Increasing or decreasing the rotational speed creates height-based up and down movements. Note that the propellers must rotate at the same speed so that the torque generated by four propellers cancels each other out, thus keeping the direction of the vehicle unchanged.

The combined thrust generated by the four propellers on the device-mounted frame of reference:

$$U_1 = F_1 + F_2 + F_3 + F_4 = c_t\left(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2\right). \tag{3}$$

2.2.2. Roll Motion (Rotation around the Ox_b Axis). To perform roll rotation around the Ox_b axis, it is necessary to adjust the speeds of the propellers 2 and 4 while the velocities of the propellers 1 and 3 remain constant. If the speed of propeller 2 (left) is higher than that of propeller 4 (right), the device will rotate in a clockwise direction around the x_b axis, otherwise, it reverses the rotation. This rotation occurs because of the moment difference between two sides of the propellers of the device. Equation (4) represents the value of the total moment on the reference frame mounted on the device.

$$U_2 = -\tau_1 + \tau_2 + \tau_3 - \tau_4 = c_t l \cos(45)\left(-\omega_1^2 + \omega_2^2 + \omega_3^2 - \omega_4^2\right). \tag{4}$$

Along with the rotation, the device will move accordingly. The device moves to the right with clockwise rotation (shown in Figure 2) and to the left with counter-clockwise rotation.

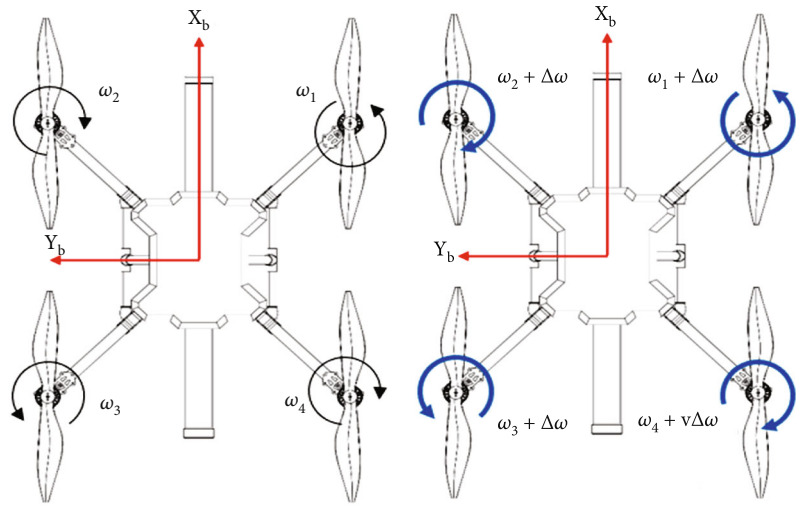
2.2.3. Pitch Motion (Rotation around the y_b Axis). To perform a pitch rotation around the y_b axis, it is necessary to adjust the speeds of the propellers 1 and 3 while the velocities of the propellers 2 and 4 are kept constant. If the speed of propeller 1 (front) is higher than that of propeller 3 (rear), the device will rotate right around the y_b axis, otherwise, it will rotate left. This rotation occurs because of the moment difference between the two sides of the propellers of the device. Equation (5) represents the value of the moment on the reference frame mounted on the device:

$$U_3 = -\tau_1 - \tau_2 + \tau_3 + \tau_4 = c_t l \cos(45)\left(-\omega_1^2 - \omega_2^2 + \omega_3^2 + \omega_4^2\right). \tag{5}$$

Along with the rotation, the device will move accordingly. The device moves forward with a right rotation (Figure 2) and backward with a left rotation around the y_b axis.

2.2.4. Yaw Motion (Rotation around the z_b Axis). To make yaw motion clockwise on the z_b axis, it is necessary to adjust the speeds of the propellers 2, 4 to increase by $\Delta\omega$, and the speeds of the propellers 1, 3 to decrease by $\Delta\omega$. Increasing and decreasing by the same amount of $\Delta\omega$ ensure the same height of the device when making yaw movement.

To make yaw movement counterclockwise on the z_b axis, it is necessary to adjust the speeds of the propellers 2, 4 to decrease by a $\Delta\omega$, and the speeds of the propellers 1, 3 to increase by $\Delta\omega$.



(a) Suspension (b) Height rise

FIGURE 2: Movements of the device along the z_b axis.

Equation (6) describes the relationship between the total torque around the yaw axis with the speeds of the four propellers on the frame of reference attached to the object, where c_q is the coefficient of the ratio of the moment to the rotational velocity.

$$U_4 = c_q\left(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2\right). \tag{6}$$

From (3) to (6), the force and torque generated from the rotational speed of the propellers are summed up as follows:

$$\begin{cases} U_1 = c_t\left(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2\right), \\ U_2 = c_t l \cos\,(45)\left(-\omega_1^2 + \omega_2^2 + \omega_3^2 - \omega_4^2\right), \\ U_3 = c_t l \cos\,(45)\left(-\omega_1^2 - \omega_2^2 + \omega_3^2 + \omega_4^2\right), \\ U_4 = c_q\left(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2\right). \end{cases} \tag{7}$$

2.3. Kinematic Model. The kinematic model describes the device's motion in reference frames through velocity equations, thereby determining the position and orientation of the object.

2.3.1. Linear Velocity. The relationship between the linear velocity of the device in the inertial frame of reference and the reference frame on the body is as follows.

$$\dot{\xi}^E = {}^E_B R(\phi, \psi, \theta) \times v^B. \tag{8}$$

2.3.2. Angular Velocity. The matrix converting from the angular velocity $\dot{\Theta}^E = (\dot{\phi}, \dot{\theta}, \dot{\psi})$ in the inertial reference frame to the angular velocity $\omega^B = (p, q, r)$ in the reference frame on the object is as follows.

$$\omega^B = {}^B_E T \dot{\Theta}^E. \tag{9}$$

$${}^B_E T = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix}. \tag{10}$$

From (8) to (10), the kinematic model of the device is as follows [1]:

$$\begin{cases} \dot{x} = w[\sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta] - v[\cos\varphi\sin\psi - \cos\varphi\sin\psi\sin\theta] + u[\cos\psi\cos\theta], \\ \dot{y} = v[\cos\phi\cos\psi + \sin\varphi\sin\psi\sin\theta] - w[\cos\psi\sin\varphi - \cos\psi\sin\psi\sin\theta] + u[\cos\theta\sin\psi], \\ \dot{z} = w[\cos\phi\cos - \theta] - u[\sin\theta] + v[\cos\theta\sin\varphi], \\ \dot{\phi} = p + r[\cos\phi\tan\theta] + q[\sin\varphi\tan\theta], \\ \dot{\theta} = q\cos\phi - r\sin\phi, \\ \dot{\psi} = q\dfrac{\cos\phi}{\sin\theta} + r\dfrac{\cos\phi}{\cos\theta}. \end{cases} \tag{11}$$

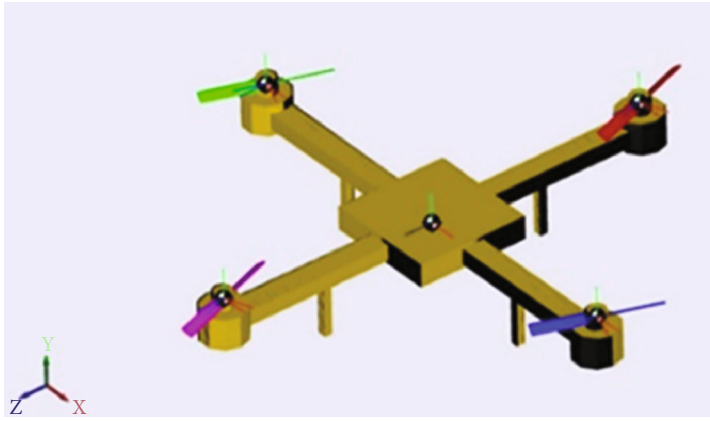


Figure 3: Euler 3D simulation of the quadrotor in Matlab Simmechanic environment.

3. Simulation

In this section, PID and backstepping integrator will be derived and demonstrated to control the quadrotor model in Simulink subjected to trajectory tracking with noise and disturbance consideration.

3.1. Trajectory Design. A 3D quadrotor model software was used for real-time visualization, as shown in Figure 3. The known validated mathematical model was considered, and disturbance and noise were added as Gaussian white noises to the model. Therefore, a low feeding rate Gaussian noise was added as a perturbation to reproduce gust wind effects on Cartesian positions. On the other side, a very high feeding rate Gaussian noise was added to emulate the noise of sensors readings and to simulate the unmodeled dynamics.

The 3D visualization model uses a multibody configuration developed in Simmechanic, with all parts attached in the same environment with 9 DOF. The drawings are imported from Creo Parametric to constitute the complete quadrotor. The 3D visualization is controlled online by the control part of Simulink that executes mathematical simulation of each control theory in the final synthesis of the Simmechanic model.

A path tracking challenge has been raised. Lissajous curves, which describe complex harmonic motion, were chosen as a base trajectory to be followed with variable altitude for surveillance. We generate the desired trajectory in Matlab as Algorithm 1 as follows.

Figure 4 illustrates the 3D desired trajectory to be followed in time reference. The horizontal motion of the quadrotor was designed to cover a large surface of surveillance, and the altitude was planned to follow an exponential convergence to the desired height.

Another depth of challenge to the trajectory tracking was given by assigning different initial positions for the quadrotor and the desired trajectory at time zero.

Nested loops are found to be a practical design strategy to execute control approaches, as shown in Figure 5.

3.2. Calculation of Desired Roll and Pitch Angles

3.2.1. First Method. The first method is based on the calculation of those desired angles ϕ_d and θ_d from the desired position while considering an instantaneous small ψ angle. However, this function generates desired angles with more discontinuities than tolerated for the needed derivative contributions.

$$u_x = \cos\phi \sin\theta \cos\psi + \sin\phi sin\psi. \tag{12}$$

$$u_y = \cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi. \tag{13}$$

$$\begin{cases} \psi << 1, \\ \phi = \arcsin\left(-u_y\right), \\ \theta = \arcsin\left(u_x/\cos\phi\right). \end{cases} \tag{14}$$

3.2.2. Second Method. A second solution which is widely used came trivially by solving the equations analytically:

$$\phi_d(t) = \frac{\arcsin\left(\sin\psi(t)u_x - \cos\psi_d(t)\right)}{u_x^2(t) + u_y^2(t)}. \tag{15}$$

$$\theta_d(t) = \frac{\arcsin\left(u_x(t) - \sin\psi_d(t)\sin\phi_d(t)\right)}{\cos\psi_d(t)\cos\phi_d(t)}. \tag{16}$$

Nonetheless, this method is related just to the dynamics considered and limited toward more added consideration of unmodeled dynamics.

3.2.3. Third Method. A third smoother method with a more expensive calculation cost is based on direct nonlinear solving of the angles system.

The benefits of this third method can be hidden behind the simplification and assumptions made while modelling. In complicated modelling cases where we consider more accurate modelling with no neglecting of minimal nonlinear contribution, this method is more reliable and precise. *Fsolve* has been used in Simulink as an interpreted function due to

```
1. t =0 : 0.2 : 30∗pi;
2. a =5; a =5; B =10; b =4; C =8; c =5; coef =0.07;
3. x = A∗cos(a∗coef∗t); y = B∗sin(b∗coef∗t);
4. z = C∗(1-exp(-5∗coef∗t));
5. plot3(x,y,z,'b')
6. Grid on
```

Algorithm 1: Desired trajectory generation in Matlab.

the fact that *fsolve* is not among the functions supported for code-generation in Embedded MATLAB function blocks.

3.3. Quadrotor PID Control. A design of multiple PID controllers was approached for each position's state. A tuning of six PIDs parameters was performed in the Matlab-Simulink environment. The objective of this PIDs strategy is to stabilize the attitude based on inner looping that controls angles (orientations) and their derivatives and then to minimize linear position errors as a second-order stable dynamics applied in the outer loop to ensure path following. From a trivial perspective, the PID correction made for attitude should be much quicker than the outer loops PID corrections attributed for translation positions.

As shown in Figure 6, the simulation used several blocks. PIDs are used for each linear and rotational position, a manual tuning was proceeded to get desired responses, and parameters for each PID can be done separately. For tracking a trajectory, a tracking error vector e of six components e_i is formulated with the objective to converge them exponentially to zero. Each of these tracking errors is assigned for one of the six-position states. For each control, we want the state x_i to follow the desired x_{id}, we define

$$e_i = x_{id} - x_i. \tag{17}$$

The derivation of this error e_i is calculated as

$$\dot{e}_1 = \dot{\phi}_d - p. \tag{18}$$

To formulate an exponential stable error that converges to zero, we may consider this dynamic:

$$\ddot{x}_{id} - \ddot{x}_i + K_d \dot{e}_i + K_p e_i + K_i \int_0^t e_i(\tau) d\tau = 0. \tag{19}$$

$$\ddot{x}_i = \ddot{x}_{id} + K_d \dot{e}_i + K_p e_i + K_i \int_0^t e_i(\tau) d\tau. \tag{20}$$

This methodology will be followed to generate all control inputs for each PID loop.

3.3.1. PID Control of Quadrotor Results. The PID control has been applied for all linear and angular positions. The simulation demonstrated very satisfying following dynamics as shown in Figure 7, the quadrotor trajectory in red, following the blue desired reference while starting initially from different spots. Figure 7 shows the result of the simulation using the validated quadrotor model with PID control strategy to follow the Lissajous desired trajectory with an acceptable error considering the different initial positions.

Linear position tracking showed minimal error, and wind gust simulation effect thru the low-rate Gaussian noise is clear on quadrotor position, with a quick correction to track. The angle tracking was also very efficient, observing the noise introduced on the sensor by the high-rate white Gaussian noise. Obtained results of the simulation were satisfactory regarding the simple synthesis approach of PID control.

3.4. Quadrotor Control by Integrator Backstepping. Integrator backstepping (IBS) is a nonlinear approach that contributes to the integral benefits of the backstepping design. This approach may ensure the asymptotic stability with attenuation of steady-state errors with the help of the integral effect. Its robustness was simulated for tracking in the presence of external disturbance and internal noise. Control design will be based on four IBS controllers. In fact, the four controllers result from the under-actuation nature of the quadrotor, which is based on just four actuators' capabilities. The derivation of those controllers is similar for both attitude angles and altitude. X and Y control will be done systematically as we ensure a validated tracking of desired ϕ_d and θ_d angles. Thus, only one control will be derived in this paper. Once we calculate Ui, we apply them to the next block to calculate the voltages and control rotation of each motor.

3.4.1. Altitude. The first step in the IBS control strategy is to define the tracking error for each rotation. Let us consider the error for ϕ angle around X_b axes by derivation of this error:

$$e_1 = \phi_d - \phi, \tag{21}$$

$$\dot{e}_1 = \dot{\phi}_d - p, \tag{22}$$

where p is the angular rate around X_b. We need to control the dynamics of p to converge to $\dot{\phi}_d$. For that, the desired tracking dynamic is calculated as

$$p_d = c_1 e_1 + \dot{\phi}_d + \alpha_1 \int_0^t e_1(\tau) d\tau. \tag{23}$$

Hence, $\alpha_1 \int_0^t e_1(\tau) d_\tau$ represents the integral contribution to cancelling the steady error. Let $\dot{e}_2$ consider:

$$\dot{e}_2 = p_d - p. \tag{24}$$

By derivation of (24) and using of (22) and (23) and ϕ dynamics, we can get:

$$\dot{e}_2 = c_1 \left(\dot{\phi}_d - p \right) + \ddot{\phi}_d + \alpha_1 e_1 - \dot{\theta}\psi a_1 - \dot{\phi} a_2 \Omega_r - b_1 U_2, \tag{25}$$

where an appearance of a control input U_2 is shown. By assuming a stable dynamic:

$$\dot{e}_2 + c_2 e_2 + e_1 = 0. \tag{26}$$

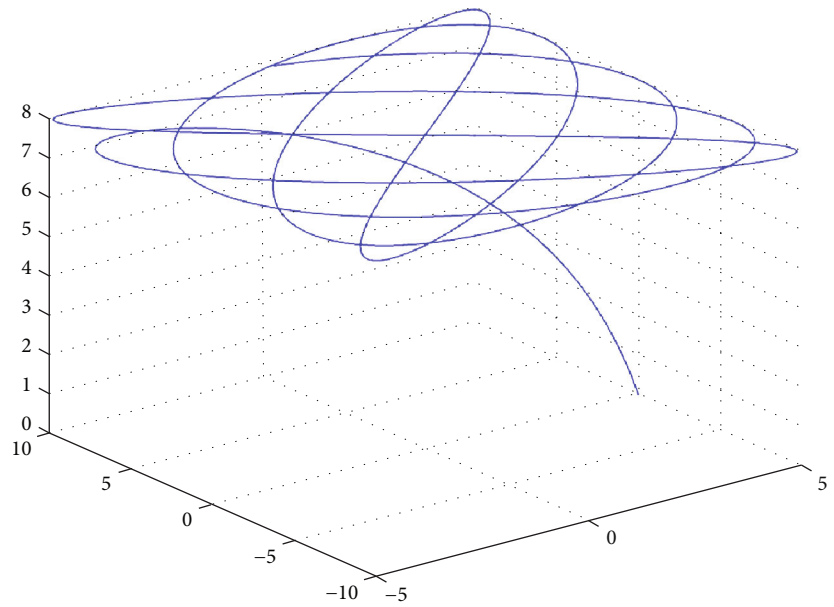


Figure 4: Desired trajectory to be tracked by quadrotor for surveillance aim.

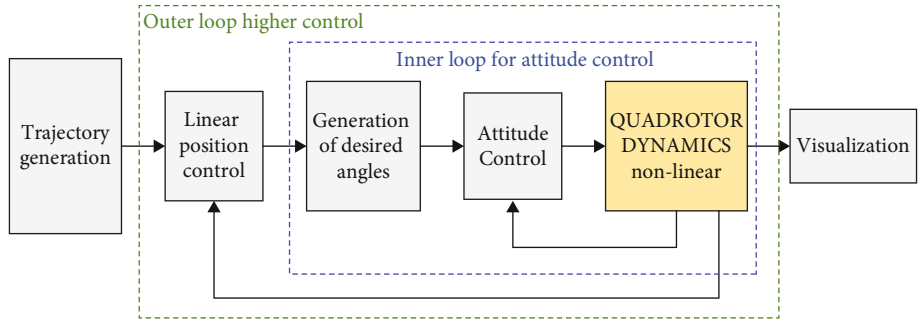


Figure 5: Nested control loops for a quadrotor.

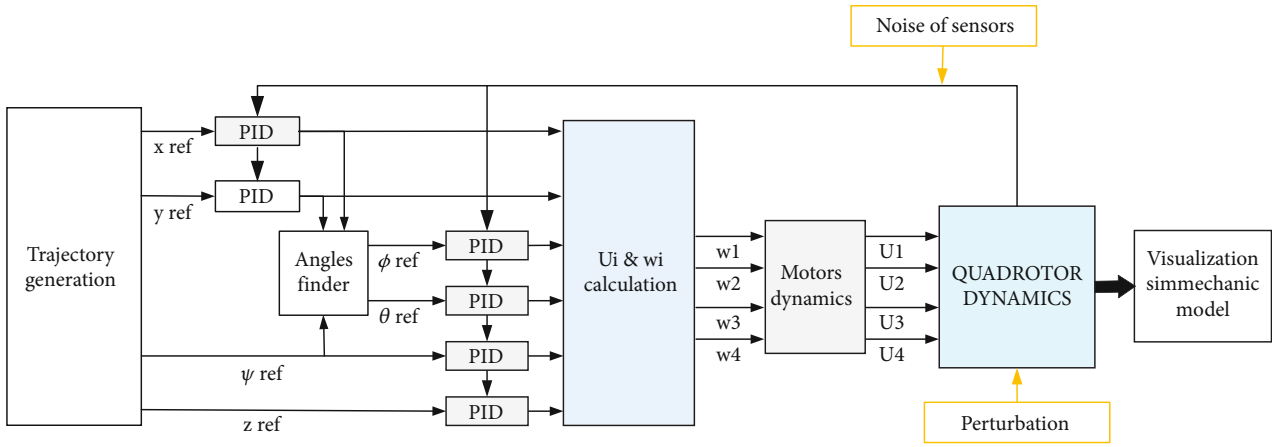
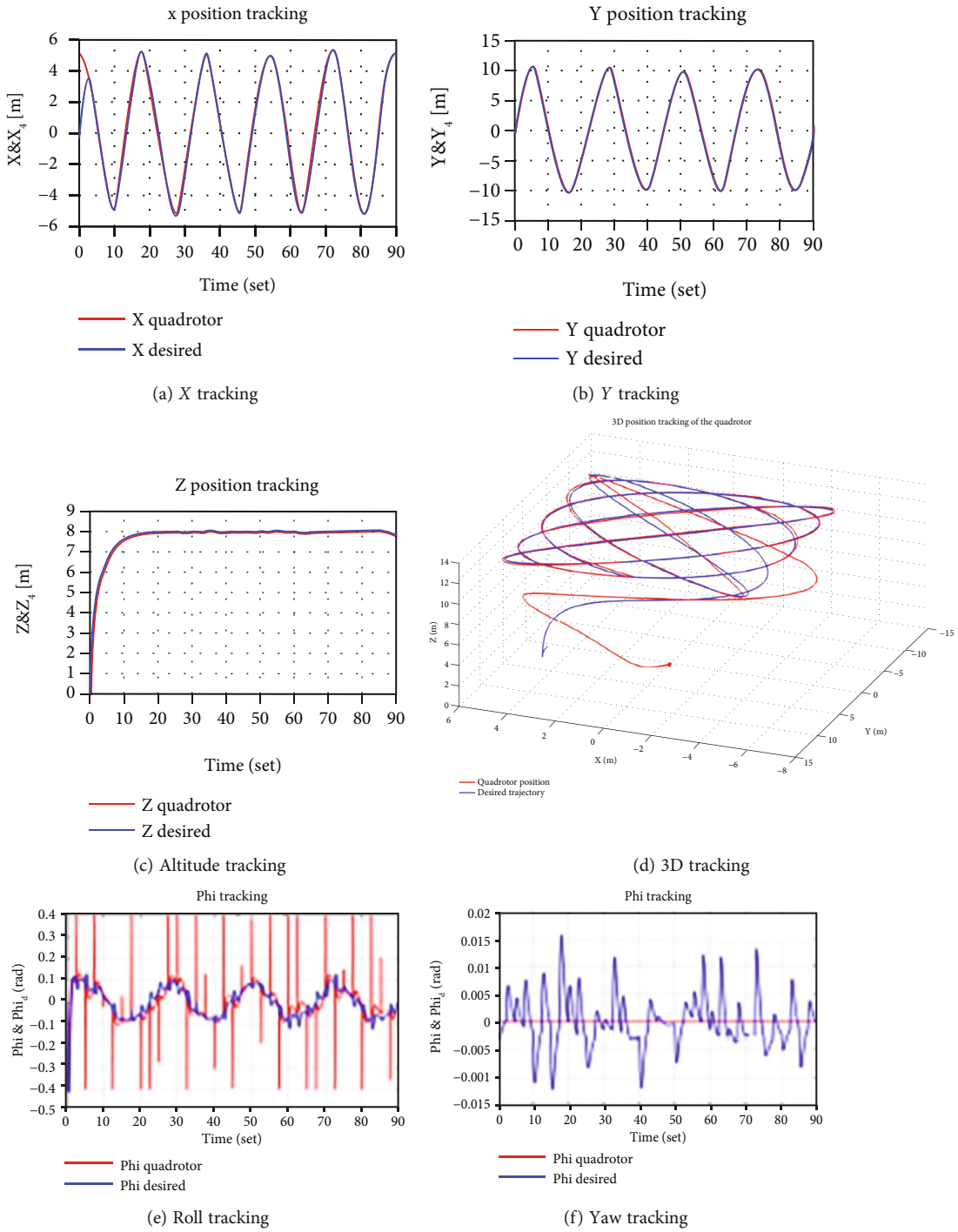


Figure 6: PID control of quadrotor design.



(a) X tracking

(b) Y tracking

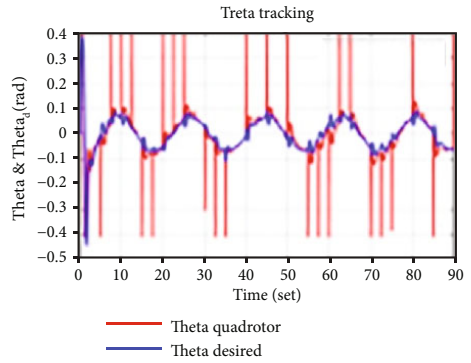
(c) Altitude tracking

(d) 3D tracking

(e) Roll tracking

(f) Yaw tracking

Figure 7: Continued.



(g) Pitch tracking

Figure 7: PID trajectory tracking.

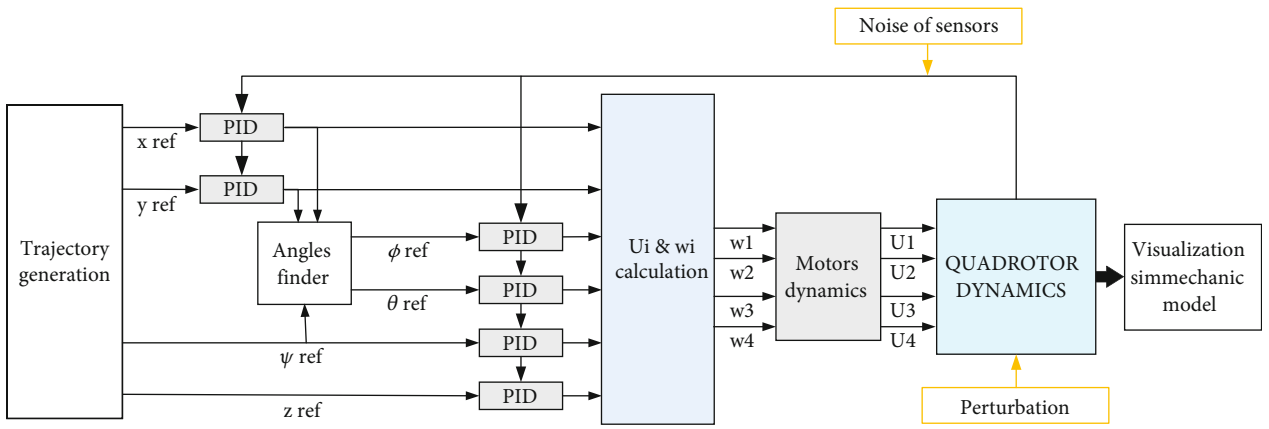


Figure 8: IBS control design of quadrotor.

1. Require coefficients of correction
2. Read trajectory states
3. Read/call for the sensor's reading/state estimations
4. Calculate errors, their derivatives & their summations
5. Calculate *Ui* by (28) to (30) & (33).
6. Calculate each rotor input

Algorithm 2: Algorithm integrator backstepping.

With $c_2 > 0$, a control input that satisfies this assumption can be calculated as follows:

$$U_2 = \frac{1}{I_{xx}}\left[\left(1 - c_1^2 + \alpha_1\right)e_1 + (c_1 + c_2)e_2 - c_1\alpha_1\int_0^t e_1(\tau)d\tau + \ddot{\phi}_d - \frac{\dot{\phi}\left(I_{yy} - I_{zz}\right)}{I_{xx}} - \dot{\theta}\Omega_r\frac{J_{TP}}{I_{xx}}\right]. \tag{27}$$

Systematically, the other control inputs derived from θ and ψ errors are given by

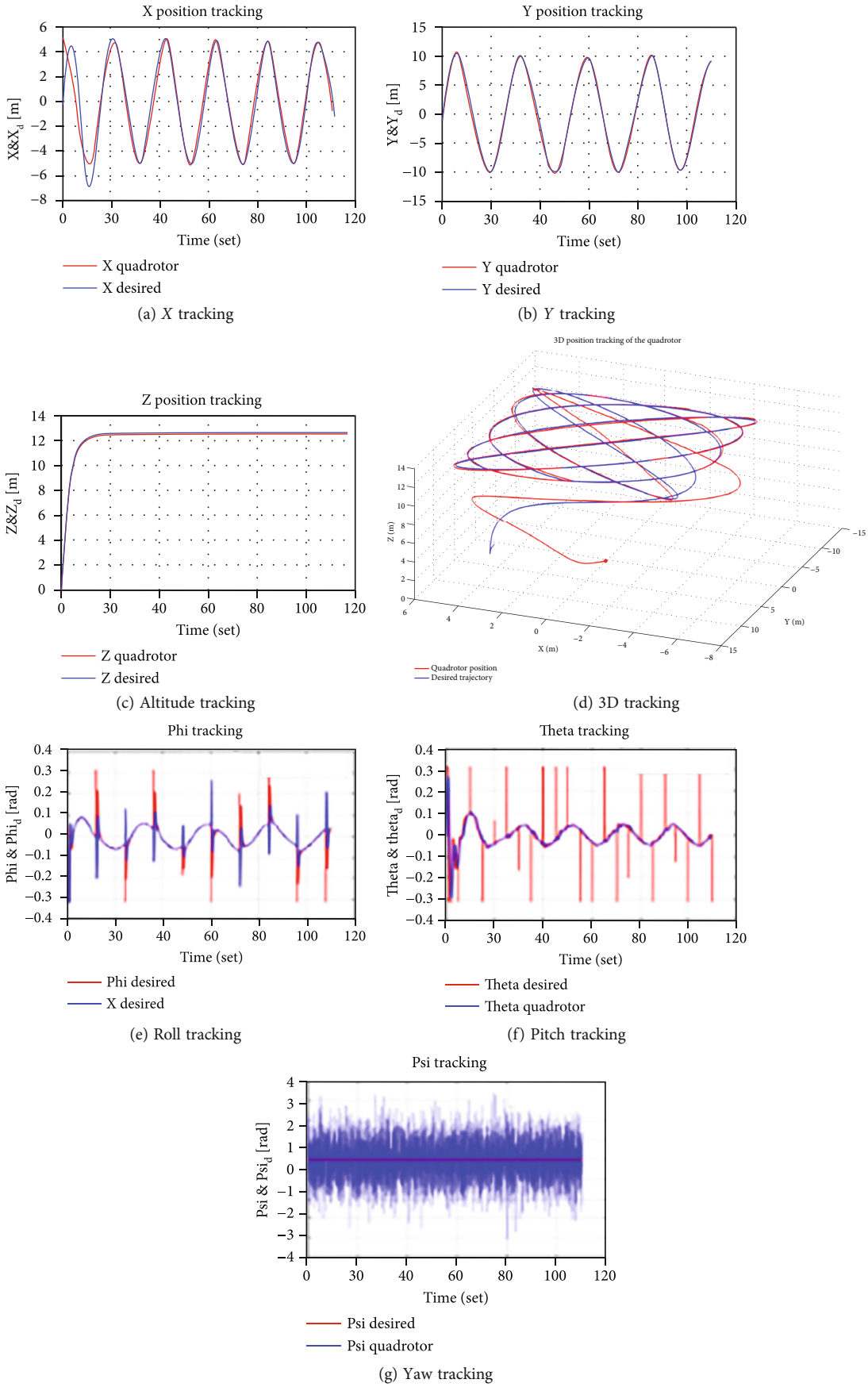
$$U_3 = \frac{1}{I_{yy}}\left[\left(1 - c_3^2 + \alpha_2\right)e_3 + (c_3 + c_4)e_4 - c_3\alpha_2\int_0^t e_3(\tau)d\tau + \ddot{\theta}_d - \frac{\dot{\phi}\dot{\psi}(I_{zz} - I_{xx})}{I_{yy}} - \frac{\dot{\phi}J_{TP}}{I_{yy}\Omega_r}\right], \tag{28}$$

$$U_4 = \frac{1}{I_{zz}}\left[\left(1 - c_5^2 + \alpha_3\right)e_5 + (c_5 + c_6)e_6 - c_5\alpha_3\int_0^t e_5(\tau)d\tau\right] \quad (c_3, c_4, c_5, c_6, \alpha_2, \alpha_3) > 0. \tag{29}$$

3.4.2. Altitude and Planar Position Control. Similarly, an altitude tracking error is defined as

$$e_7 = z_d - z, \tag{30}$$

$$e_8 = c_7e_7 + \dot{z}_d + \alpha_4\int_0^t e_4(\tau)d_\tau. \tag{31}$$



(a) X tracking

(b) Y tracking

(c) Altitude tracking

(d) 3D tracking

(e) Roll tracking

(f) Pitch tracking

(g) Yaw tracking

Figure 9: IBS trajectory tracking.

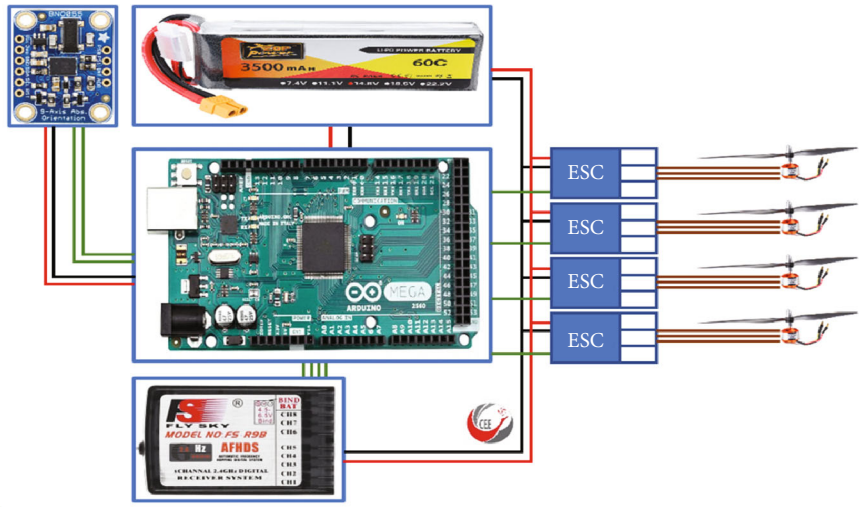


Figure 10: Hardware architecture.

1. A. Include needed libraries;
2. B. Define PID coefficients values for roll, yaw & Pitch;
3. C. Declaration of variables:
4. + channels;
5. + counters;
6. + control Ui;
7. + battery level;
8. + calibration of gyro reading;
9. + PID intermediate variables, ...
10. D. Setup of the program:
11. + define microcontroller as I2C master for communication;
12. + define inputs & outputs with readiness led signs;
13. + communicate w/GYRO and calibrate readings with a function. (during calibration, silent the ESC by zero Ui);
14. + enable "PCICR" & define masks register for pins;
15. + keep zero Ui command while waiting;
16. + define state of launch.
17. E. the loop:
18. + call FCT of gyro reading;
19. + filter the noise of reading and transfer to deg/sec;
20. + define SATRT/STOP sequence by joystick, (using moor machine);
21. + reset all PID for new cycle for roll yaw pitch;
22. + recalculate the setpoint from channels;
23. + calculate the PID FCT;
24. + define led warning and state-dependent;
25. + limit max throttle (Ui);
26. + calculate signals of ESCs;
27. + compensate the control by battery discharge ranging of ESCs signal [min-max];
28. + keep motors always with zero Ui in case of waiting for the state;
29. + wait to complete 4 ms, (250 hz).

Algorithm 3: Algorithm of PID control.

Its error dynamics can be tracked as

$$e_8 = c_7 e_7 + \dot{z}_d + \alpha_4 \int_0^t e_4(\tau) d_\tau - \dot{z}. \tag{32}$$

By derivation of this error:

$$U_1 = \left(\frac{m}{\cos\phi \cos\theta}\right)\left[g + \left(1 - c_7^2 + \alpha_4\right)e_7 + \left(c_7 + c_8\right) - c_7\alpha_4 \int_0^t e_4(\tau) d\tau\right]. \tag{33}$$

For both X and Y control, a simple stable second-order tracking dynamics had been defined by a PID, then desired angles were generated by direct calculation continuously to energize the IBS block, shown in Figure 8, in order to calculate the Ui control inputs using equations (28) to (30) and (33).

3.4.3. Integrator Backstepping Algorithm. During the simulation process, twelve coefficients were defined of IBS to the eight errors dynamics and their integral actions (ϕ, θ, ψ, and z). Errors should be calculated first at each step of simulation based on desired states and the measured outputs and their derivatives. Then, Ui control inputs should be calculated to excite the other blocks prior quadrotor nonlinear model block. Algorithm 2 can summarize the IBS method is illustrated as follows.

3.4.4. Quadrotor Control by Backstepping Integrator Result. The simulation of IBS control showed an excellent stabilization and very good tracking of the desired trajectory despite the initial position difference, refer to Figure 9. The quadrotor tries quickly to catch the desired trajectory and follow it with a tolerated error. IBS was validated as an approach for quadrotor control to track the desired Lissajous curves. The rejection of perturbation was well attributed relatively to introduced noise and initial conditions.

At initial tracking, the quadrotor hardly caught the desired trajectory, but once done, a very good tracking was observed. Desired angle tracking was also excellent despite all noise and disturbance. During implementation, coefficients should be wisely chosen to not excite the Ui to saturation threshold. Nonrealizable control inputs will disastrously deteriorate the performance of IBS.

The position tracking showed minimal tracking error. The wind gust effect by low-rate Gaussian noise is observable on quadrotor position, with the quick correction to track. Angle tracking was also excellent and better than PID attitude results, all observing the noise effect introduced on the sensor by the high-frequency Gaussian noise.

4. Developments of the Quadrotor Model

In this section, the discussion will be made on building a quadrotor model. The hardware/software architecture for the quadrotor UAV will be detailed. The developed platform was made for research and educational purposes. All UAVs are equipped with a flight control unit (FCU) to ease the exploitation of the inertial measurement unit (IMU) [33] and other sensors.

4.1. Hardware Design. A basic design for a low-cost quadrotor solution is based on:

(i) *DIY Mechanical Frame CEE-v22*. It is used to fix all components; a smaller and lighter body frame ensures better stability

(ii) SunnySky V3058 high-efficiency brush-less motor

(iii) *Electronic Speed Controller (ESC) 25-35 amp*. They use the energy from the battery to supply the motors with variated voltage to control the motors' speed via PWM generated from the CPU/RC receiver (it is important to use fast-reacting ESCs within the max current possible)

(iv) *Propellers 8045* ($304 \times 114\,mm$). The chosen propellers should generate enough thrust at a relatively low RPM

(v) *Processing Unit*. Such as Arduino Mega 2560 which is a microcontroller board based on the AT-mega2560 at 16 MHz crystal oscillator. It has 54 digital input/output pins where we essentially use 15 of them as PWM connections in parallel with UARTS (serial communication ports). It has a mission to execute the flight control program and manage the communication between electronic parts

(vi) *IMU*. BNO055 (9 DOF sensor) is a system in package (SiP), integrating a triaxial 14-bit accelerometer, a triaxial 16-bit gyroscope with a range of ±2000 degrees per second, a triaxial geomagnetic sensor, and a 32-bit cortex M0+ microcontroller running Bosch Sensortec sensor fusion software, in a single package

(vii) *The Battery*. As an energy source, a LiPo battery of 2000 to 5000 mAh with a high C factor of 30+ ensures an endurance of more than ten minutes and a good reaction of motors

(viii) *Transmitter and Receiver That Will Be Used by Pin Change Interrupt Mode*. A need for at least four channels to control (height, pitch, yaw, and roll).

1. Calculate error between gyro reading & set point;
2. Calculate integration contribution for RYP w/max value;
3. Calculate: PID = min ≤ PIDi + p ∗ error + d ∗ (error − exError) ≤ maxforRYP;
4. Update ex-error;
5. Send back values.

Algorithm 4: PID calling FCT.

1. Once PCI is called by pin voltage change.
2. **If** (last value == 0) **then**
3. If (new reading == 1) **then**
4. Last value =1;
5. Start timer;
6. **Else**
7. **If** (new reading == 0) **then**
8. Last value =0;
9. Receiver = time − Timer;
10. Send back the time of the pulse

Algorithm 5: ISR: interrupt subroutine.

Figure 11: Self-developed drone.

Figure 12: The designed quadrotor during flight.

The complete drone is approximately 1.2 kg. An extra power distribution card is added for a better setup.

4.1.1. Architecture. The connection between parts is shown in Figure 10.

It is important to mount the gyro sensor with orientation as mentioned in the sensor datasheet. For the microcontroller, it should be set the way it is easy to access to periphery and sensors.

The battery should be fixed but never connected till a confirmed use.

A tension divider is crucial to watch the battery level from being discharged during the flight, once batteries are discharged, the rotation of motors are directly affected, and stability may be altered as well.

For the ESCs, it is sufficient to switch two cables to change the rotation direction of the propeller.

The receiver channel outputs are connected to digital pins to read the PWM generated by the receiver with PCI mode.

4.2. Software Architecture. The control code is developed in an IDE environment in C language, and it is available to be downloaded. The program should make the calibration of the sensors first, then ensure a routine of a refresh rate of 250 Hz loop of the control, which makes a 4 ms time for each loop, since the ESCs are 1000 to 2000 microsecond controllable, only 2 ms are available time to read the sensor data and calculate the control outputs.

Refresh rate is limited by the max refresh rate of RC controller, by the min time of full program (reading, corrections, and sending PWM) processing time.

Pin change interrupts routine is the way we interrupt the running program to execute a special task as reading the new 2.4 GHz receiver signal on the PWM ports.

It is always judicious to make a gyro calibration in a static state to have a good reference. For the magnetometer, a calibration for the soft and hard iron distortion is crucial. The GYRO is connected just by CLK and SDA, where care should be made about getting two correct bytes from the

same time sample and for the correct axis. For both the accelerometer and gyroscope, we use a complementary filter to smooth the reading data via the I2C protocol.

The gyro and the accelerometer each provide two bytes of data for each axe in every iteration.

A PID correction should be set for possible rotation. The values of the PID command are calculated based on the length of the square signals of ESCs.

It is trivial that the PID of roll and pitch is similar. However, the yaw angle and height PID's may be chosen as a slow response for better attitude control. The PID output should always keep the motor running as it does not exceed max of 2000 microseconds (1800 ms).

Use the proportional to rapid answer the error, the derivative as a dumper, and the integral to kill the static error. The battery voltage drop-down should be considered and compensated.

4.2.1. The PID Based Control Algorithm. Algorithms 3, 4, and 5 with functions of controlling are addressed as follows.

4.3. Experiments of the Quadrotor Model Flight. Obtained experimental results with the designed platform showed that the hovering mode was stable, but we observed a little drift from the position due to gyro drift and level correction by the accelerometer. No GPS was mounted to correct the global position. The aim of the hovering mode experiment was to prove the stability of the design without pilot control and under normal conditions. Figure 11 shows the quadrotor on the ground and then during hovering flight.

Figure 12. demonstrates the control of the quadrotor during flight with limited tilt angles. The quadrotor was agile and responsive to the transmitted control inputs.

The test results show that our approaches proved more effective in a number of aspects such as:

(i) A 9 DOF IMU is potentially better than a 6 DOF IMU in the article [16] because it can use sensor fusion (mixing the data from different sensors) in order to improve the quality of the final output

(ii) We have added to the model both low feeding rate and high-frequency Gaussian white noise instead of only high feeding rate as in the test of the article [40]. This increased the accuracy of the test results

(iii) The multi-PID control algorithm deployed on Arduino Mega proves to be effective and simple, making it easier and faster to approach research on UAV technology than AR Drone 2.0 Quadcopter Embedded Coder in the article [24]. Our methodology allows users to deploy customizations and perform tests quickly and directly on the UAVs

4.4. Practical Depth of PID Implementation. Implementation of such a method on microcontrollers is the easiest compared to other approaches. Criteria of running speed of the control loop and input saturation should be considered, and this latter justifies the use of saturation over the control Ui used for simulation. A control loop frequency at less than 50 Hz may induce enough delay to destabilize the system and may create too much oscillation that deteriorates even IMU measurement quality.

Although the high-rate frequency of IMU readings, an appropriate loop of execution should be wisely chosen to not waste data from motion measurement.

An integral action should be considered to kill the steady-state error. Nevertheless, a reset of integral contribution should be integrated with the process so as not to diverge the summation of errors. The fast response actuation of the electronic speed controller (ESC) over the brushless motors is a key factor in stabilization. A delay from the execution loop or from ESC response will tremendously break down the attitude performance. In the case of implementation, accelerometer data should be used to stabilize the quadrotor horizontally and define the zero level due to the electronic drift of the gyro.

5. Conclusions and Future Developments

This paper presents three methods of quadrotor control: PID, MPC, and IBS, which are all derived and simulated on the validated nonlinear model of the quadrotor. The performance and robustness are studied based on the trajectory tracking and the implementation of curves used for surveillance. Simulation considered adding white noise at a low sample rate to contribute to the disturbance effect of wind gusts and at a high sample rate for actuator noise despite setting the quadrotor far away from the desired initial position to simulate and analyze the response time. The experimental work includes an application of developing an open-source quadrotor model based on the Arduino Mega microcontroller. This model is stable and can be considered a development study model. All the results show promising points to support UAVs in practical applications.

In future work, the proposed algorithm would be exploited with other control methods to be able to provide more results, including comparison and correctness for further developments. Deep learning and reinforcement learning techniques could be integrated into the algorithm to provide more effectiveness for the UAVs.

Data Availability

All the data used to support the findings of this study are included within the article as shown in the references.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors would like to thank Thai Nguyen University of Technology (TNUT), Ministry of Education and Training (MOET), Vietnam, for the support.

References

[1] T. Q. Duong, K. J. Kim, Z. Kaleem, M.-P. Bui, and N.-S. Vo, "UAV caching in 6G networks: a survey on models, techniques, and applications," *Physical Communication*, vol. 51, article 101532, 2022.

[2] M. T. Nguyen, C. V. Nguyen, H. T. Do et al., "UAV-assisted data collection in wireless sensor networks: a comprehensive survey," *Electronics*, vol. 10, no. 21, p. 2603, 2021.

[3] H. T. Do, H. T. Hua, M. T. Nguyen et al., "Formation control algorithms for multiple-UAVs: a comprehensive survey," *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems*, vol. 8, no. 27, article e3, 2021.

[4] H. T. Do, L. H. Truong, M. T. Nguyen et al., "Energy-efficient unmanned aerial vehicle (UAV) surveillance utilizing artificial intelligence (AI)," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 8615367, 11 pages, 2021.

[5] L. Zhang, H. Zhao, S. Hou et al., "A survey on 5G millimeter wave communications for UAV-assisted wireless networks," *IEEE Access*, vol. 7, pp. 117460–117504, 2019.

[6] M. T. Nguyen, L. H. Truong, and T. T. H. Le, "Video surveillance processing algorithms utilizing artificial intelligent (AI) for unmanned autonomous vehicles (UAVs)," *MethodsX*, vol. 8, article 101472, 2021.

[7] H. T. Do, H. T. Hua, H. T. T. Nguyen, M. T. Nguyen, and H. T. Tran, "Cooperative tracking framework for multiple unmanned aerial vehicles (UAVs)," in *International Conference on Engineering Research and Applications*, pp. 276–285, Cham, 2022.

[8] O. Bouaiss, R. Mechgoug, and R. Ajgou, "Modeling, control and simulation of quadrotor UAV," in *2020 1st International Conference on Communications, Control Systems and Signal Processing (CCSSP)*, pp. 340–345, El Oued, Algeria, 2020.

[9] O. Mofid, S. Mobayen, and W.-K. Wong, "Adaptive terminal sliding mode control for attitude and position tracking control of quadrotor UAVs in the existence of external disturbance," *IEEE Access*, vol. 9, pp. 3428–3440, 2021.

[10] S. Musa, "Techniques for quadcopter modelling and design: a review," *Journal of unmanned system Technology*, vol. 5, no. 3, pp. 66–75, 2018.

[11] D. H. Lyon, "A military perspective on small unmanned aerial vehicles," *IEEE Instrumentation & Measurement Magazine*, vol. 7, no. 3, pp. 27–31, 2004.

[12] Z. Jiang, J. Zhu, Z. Lin, Z. Li, and R. Guo, "3D mapping of outdoor environments by scan matching and motion averaging," *Neurocomputing*, vol. 372, pp. 17–32, 2020.

[13] X. Dai, Y. Mao, T. Huang, N. Qin, D. Huang, and Y. Li, "Automatic obstacle avoidance of quadrotor UAV via CNN-based learning," *Neurocomputing*, vol. 402, pp. 346–358, 2020.

[14] Y. Zou and B. Zhu, "Adaptive trajectory tracking controller for quadrotor systems subject to parametric uncertainties," *Journal of the Franklin Institute*, vol. 354, no. 15, pp. 6724–6746, 2017.

[15] R. Pérez-Alcocer and J. Moreno-Valenzuela, "A novel Lyapunov-based trajectory tracking controller for a quadrotor: experimental analysis by using two motion tasks," *Mechatronics*, vol. 61, pp. 58–68, 2019.

[16] O. Mofid and S. Mobayen, "Adaptive finite-time backstepping global sliding mode tracker of quad-rotor UAVs under model uncertainty, wind perturbation, and input saturation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 58, no. 1, pp. 140–151, 2022.

[17] R. Wang and J. Liu, "Trajectory tracking control of a 6-DOF quadrotor UAV with input saturation via backstepping," *Journal of the Franklin Institute*, vol. 355, no. 7, pp. 3288–3309, 2018.

[18] L. A. Márquez-Vega, M. Aguilera-Ruiz, and L. M. Torres-Treviño, "Multi-objective optimization of a quadrotor flock performing target zone search," *Swarm and Evolutionary Computation*, vol. 60, article 100733, 2021.

[19] A. V. Borkar, S. Hangal, H. Arya, A. Sinha, and L. Vachhani, "Reconfigurable formations of quadrotors on Lissajous curves for surveillance applications," *European Journal of Control*, vol. 56, pp. 274–288, 2020.

[20] K. Guo, J. Jia, X. Yu, L. Guo, and L. Xie, "Multiple observers based anti-disturbance control for a quadrotor UAV against payload and wind disturbances," *Control Engineering Practice*, vol. 102, article 104560, 2020.

[21] N. Q. Vinh, "INS/GPS integration system using street return algorithm and compass sensor," *Procedia Computer Science*, vol. 103, pp. 475–482, 2017.

[22] M. T. Nguyen and H. R. Boveiri, "Energy-efficient sensing in robotic networks," *Measurement*, vol. 158, article 107708, 2020.

[23] R. Miranda-Colorado and L. T. Aguilar, "Robust PID control of quadrotors with power reduction analysis," *ISA Transactions*, vol. 98, pp. 47–62, 2020.

[24] L. Martins, C. Cardeira, and P. Oliveira, "Linear quadratic regulator for trajectory tracking of a quadrotor," *IFAC-PapersOnLine*, vol. 52, no. 12, pp. 176–181, 2019.

[25] J. S. Shamma and J. R. Cloutier, "Gain-scheduled missile autopilot design using linear parameter varying transformations," *Journal of Guidance, Control, and Dynamics*, vol. 16, no. 2, pp. 256–263, 1993.

[26] H. Das, "Dynamic inversion control of quadrotor with a suspended load," *IFAC-PapersOnLine*, vol. 51, no. 1, pp. 172–177, 2018.

[27] E. Altug, J. P. Ostrowski, and R. Mahony, "Control of a quadrotor helicopter using visual feedback," in *Proceedings 2002 IEEE international conference on robotics and automation (cat. No. 02CH37292)*, vol. 1, pp. 72–77, Washington, DC, USA, 2002.

[28] O. García, P. Ordaz, O. J. Santos-Sánchez, S. Salazar, and R. Lozano, "Backstepping and Robust Control for a Quadrotor in Outdoors Environments: An Experimental Approach," *IEEE Access*, vol. 7, pp. 40636–40648, 2019.

[29] N. Koksal, H. An, and B. Fidan, "Backstepping-based adaptive control of a quadrotor UAV with guaranteed tracking performance," *ISA Transactions*, vol. 105, pp. 98–110, 2020.

[30] Y. Huang, W. Liu, B. Li, Y. Yang, and B. Xiao, "Finite-time formation tracking control with collision avoidance for quadrotor UAVs," *Journal of the Franklin Institute*, vol. 357, no. 7, pp. 4034–4058, 2020.

[31] J.-J. Xiong and E.-H. Zheng, "Optimal Kalman filter for state estimation of a quadrotor UAV," *Optik*, vol. 126, no. 21, pp. 2862–2868, 2015.

[32] O. Mofid, S. Mobayen, C. Zhang, and B. Esakki, "Desired tracking of delayed quadrotor UAV under model uncertainty and wind disturbance using adaptive super-twisting terminal sliding mode control," *ISA Transactions*, vol. 123, pp. 455–471, 2022.

[33] A. Kaba and E. Kyak, "Optimizing a Kalman filter with an evolutionary algorithm for nonlinear quadrotor attitude dynamics," *Journal of Computational Science*, vol. 39, article 101051, 2020.

[34] V. C. Thanh, N. N. A. Quan, T. L. Thang Dong, T. T. Hoang, and M. T. Nguyen, "Fusion of inertial and magnetic sensors for autonomous vehicle navigation and freight in distinctive environment," in *International Conference on Engineering Research and Applications*, pp. 431–439, Cham, 2022.

[35] Y. Yue, H. Yang, F. Liu, and H. Zang, "Cooperative control for multiple quadrotors under position deviations and aerodynamic drag," *Mechanical Systems and Signal Processing*, vol. 147, article 107096, 2021.

[36] H. Liu, Y. Lyu, and W. Zhao, "Robust visual servoing formation tracking control for quadrotor UAV team," *Aerospace Science and Technology*, vol. 106, article 106061, 2020.

[37] J. Park and H. Baek, "Stereo vision based obstacle collision avoidance for a quadrotor using ellipsoidal bounding box and hierarchical clustering," *Aerospace Science and Technology*, vol. 103, article 105882, 2020.

[38] M. Bhargavapuri, S. R. Sahoo, and M. Kothari, "Robust nonlinear control of a variable-pitch quadrotor with the flip maneuver," *Control Engineering Practice*, vol. 87, pp. 26–42, 2019.

[39] A. Khalifa and M. Fanni, "Experimental implementation of a new non-redundant 6-DOF quadrotor manipulation system," *ISA Transactions*, vol. 104, pp. 345–355, 2020.

[40] M. N. Shauqee, P. Rajendran, and N. M. Suhadis, "Proportional double derivative linear quadratic regulator controller using improvised grey wolf optimization technique to control quadcopter," *Applied Sciences*, vol. 11, no. 6, p. 2699, 2021.