JOURNAL OF CULTURAL ANALYSIS AND SOCIAL CHANGE

Journal of Cultural Analysis and Social Change,
2025, 10(4), 1284-1307
ISSN: 2589-1316

Development of a Hybrid Quadrotor Control Framework Integrating Classical Control and Vision-Based Deep Learning for Autonomous Takeoff, Landing, and Object Tracking

Dung Nguyen[1], Sergey Kinzhikeyev[2], Hung Duong[1], Minh Doan[1], Gulnaz Yermoldina[3], Zharas Ainakulov[4], Zhanna Suimenbayeva[5], Andrey Bebenin[6*]

[1]*Le Quy Don Technical University, Hanoi, VIETNAM*
[2]*Department of Military, Astana IT University, Astana, REPUBLIC OF KAZAKHSTAN*
[3]*Research Institute of Applied Biotechnology, Akhmet Baitursynuly Kostanay Regional University, Kostanay, REPUBLIC OF KAZAKHSTAN*
[4]*Situational Center, NJSC «Kazakh National Agrarian Research University», Astana, REPUBLIC OF KAZAKHSTAN*
[5]*Office of innovative Projects, Mukhametzhan Tynyshbayev ALT University, Almaty, REPUBLIC OF KAZAKHSTAN*
[6]*Department of the Air Force, Faculty of Air Defense Forces and Information Systems Security, National Defense University of Republic of Kazakhstan, Astana, REPUBLIC OF KAZAKHSTAN*

*__Corresponding Author:__ nesipova@rdke.kz

Citation: Nguyen, D., Kinzhikeyev, S., Duong, H., Doan, M., Yermoldina, G., Ainakulov, Z., … Bebenin, A. (2025). Development of a Hybrid Quadrotor Control Framework Integrating Classical Control and Vision-Based Deep Learning for Autonomous Takeoff, Landing, and Object Tracking. *Journal of Cultural Analysis and Social Change*, *10*(4), 1284–1307. https://doi.org/10.64753/jcasc.v10i4.3013

Published: December 08, 2025

ABSTRACT

The object of this study is a quadrotor unmanned aerial vehicle (UAV) equipped with classical PID (proportional-integral-derivative) and backstepping controllers for stabilization and deep learning-based vision modules (YOLOv8(You Only Look Once) and ByteTrack (a multi-object tracking algorithm) for perception and tracking. The research addresses the problem of limited robustness and adaptability of traditional control systems when operating in dynamic environments with external disturbances, sensor noise, and moving targets. As a result of the study, a hybrid quadrotor control framework integrating classical control and vision-based deep learning was developed and experimentally validated. The proposed system enables autonomous takeoff, smooth landing, stable hovering, and reliable multi-object tracking under varying illumination and occlusion conditions. These results were achieved through the layered coupling of model-based controllers with visual feedback, which allows real-time compensation for sensor drift and external perturbations, ensuring stable flight trajectories. The developed framework can be effectively applied in practice for infrastructure monitoring, environmental observation, search-and-rescue missions, and intelligent transportation, especially in GPS-denied (Global Positioning System) or visually complex environments.

Keywords: Quadrotor, Unmanned Aerial Vehicle, Control, Hybrid, Autonomy, Vision, You Only Look Once, Backstepping, Dynamics, Tracking

INTRODUCTION

Unmanned Aerial Vehicles (UAVs) are becoming increasingly integral to civilian and industrial operations, including environmental monitoring, infrastructure inspection, logistics, and intelligent transportation. Their deployment in complex and uncertain environments often characterized by variable weather, dynamic obstacles, and unpredictable human or vehicular activity requires control systems capable of high precision, adaptability, and reliability.

In modern conditions, the scientific and practical relevance of UAV control research has intensified due to the rapid expansion of automation, urban air mobility, and smart city applications. With UAVs being integrated into air traffic systems and IoT-based infrastructure networks, ensuring safe and stable autonomous flight under real-world disturbances such as wind gusts, sensor noise, and lighting variation has become a critical engineering challenge. Current control strategies, including proportional-integral-derivative (PID), Linear Quadratic Regulator (LQR), backstepping, and reinforcement learning approaches, have achieved notable success in controlled or simulated environments; however, they still exhibit limitations in adaptability and robustness when exposed to dynamic external conditions.

At the same time, the integration of advanced perception systems particularly computer vision and deep learning-based object detection offers new opportunities to enhance UAV situational awareness and react directly to environmental changes, which is essential for safe operation in cluttered or GPS-denied environments. The convergence of control theory and artificial intelligence thus represents a necessary direction for achieving autonomy that meets the growing demands of industrial, environmental, and security applications.

Therefore, studies that are devoted to developing reliable, adaptive, and perception-driven control systems for UAVs under real-world environmental uncertainty are of significant scientific relevance.

LITERATURE REVIEW AND PROBLEM STATEMENT

UAVs are utilized in various civilian applications and can be designed with different structures and functions [1]. Among multi-rotor UAVs, the quadrotor configuration is the most common, consisting of four rotors mounted at the four corners of a cross-shaped frame. The rotors are arranged in two symmetrical pairs, with two rotating clockwise and the other two rotating counterclockwise. By coordinating the rotational speeds of the rotors, the quadrotor is able to achieve flight stabilization, rotation, as well as forward, backward, and lateral movements.

Several studies have explored various control strategies to improve the stability and trajectory tracking quality of quadcopters.

For example, in [2], researches implemented a model predictive control (MPC) algorithm to enable autonomous landing on a moving platform, demonstrating high accuracy of trajectory correction in real time. In the work [3] provided a comprehensive performance evaluation of major control strategies, identifying limitations of PID and LQR (Linear Quadratic Regulator) methods in terms of robustness under environmental disturbances.

In the work [4] presented a comparative analysis of classical, adaptive, and intelligent control techniques for UAVs, emphasizing that hybrid approaches can improve transient response and disturbance rejection.

Other researchers have [5] proposed a novel hybrid control method integrating neural dynamics with trajectory tracking, showing improved robustness and accuracy compared with conventional PID controllers.

These works collectively highlight the continuous progress in UAV control research; however, most existing studies remain limited to simulation or specific trajectory scenarios, lacking integration with real-time vision-based feedback – a gap that this study aims to address. PID controller has been used to control the attitude of a quadrotor in [6]. However, these simulations and experiments show that the PID controller has better tracking quality regarding pitch angle, while there is a large tracking error regarding roll angle. LQR optimal control algorithm controls a dynamic system by minimizing a suitable objective function. In the work [7], the LQR algorithm was applied to a quadcopter, and its performance was compared with that of a PID controller. However, the controller loses tracking ability after avoiding an obstacle in the environment with many obstacles.

Authors in the [8] proposed an adaptive neural-network-based nonsingular fast terminal sliding mode controller (NFTSMC) for a quadrotor operating under dynamic uncertainties. Their approach combines neural network approximation with Lyapunov stability theory to guarantee finite-time convergence and robustness against external disturbances. Simulation results demonstrated that the controller achieved smooth attitude stabilization with faster convergence compared to conventional SMC methods. However, the study was limited to simulation environments and did not include experimental validation, which leaves the real-time applicability of the proposed NFTSMC uncertain due to untested computational and sensor noise effects.

In another work [9], the authors developed a finite-time neuro-sliding-mode controller for quadrotors carrying suspended payloads, focusing on eliminating chattering effects and improving disturbance rejection. Their design

ensured precise position and attitude tracking despite payload-induced oscillations, confirming the robustness of the control law in complex dynamic scenarios.

Both studies emphasize the advantages of sliding mode control in improving robustness and disturbance tolerance. However, they rely primarily on theoretical or simulation verification and do not integrate real-time vision-based feedback or hybrid control architectures.

The SMC (Sliding Mode Control) controller can stably control the quadrotor to a desired position and yaw angle. The tracking control is relatively good, with the ability to suppress disturbances, which shows the robustness of the SMC controller.

Backstepping control is a recursive algorithm dividing the controller into steps and gradually stabilizing each subsystem. The advantage of this algorithm is that it converges quickly, resulting in low computational load, and can control disturbances well. The biggest limitation of this algorithm is that it has poor robustness. For example, in [10] the backstepping approach is applied to stabilize the orientation of the quadrotor. Although the method provides smooth convergence and robustness against modeling errors, it was tested only under ideal simulation conditions, without accounting for wind disturbances or real sensor noise. Therefore, its applicability to real UAV environments remains unverified due to the absence of hardware-in-the-loop or flight experiments.

In another study [11], an adaptive control scheme using feedback linearization (FBL) is implemented for a quadrotor with a changing center of mass. While the approach effectively compensates for parameter variations, it assumes perfect state measurement and instantaneous adaptation, which are unrealistic for onboard UAV computation. As a result, the method's real-time feasibility is limited by high computational complexity and sensitivity to sensor delays.

An optimal controller that is robust to disturbances is applied in [12] for tracking both the attitude and altitude of the quadrotor. The effectiveness of the controller has been experimentally verified, showing that it can minimize errors and eliminate disturbances. However, the study does not integrate perception or vision-based feedback, and the controller performance under dynamically changing visual or environmental conditions was not analyzed.

Therefore, the approach remains restricted to predefined trajectories and lacks adaptability to external perturbations and visual uncertainty.

Artificial intelligence (AI) technologies have been widely and effectively applied in various areas of modern life. One of the most actively developing fields of AI is computer vision, which provides the foundation for automated perception and decision-making in real-world environments.

In [13], the problem of object detection in complex visual scenes was analyzed, with emphasis on the challenges of applying computer vision algorithms under dynamic and visually cluttered conditions. The study highlighted issues such as background interference, occlusion, and illumination variability that significantly affect detection accuracy.

In [14], an intelligent adaptive learning and control approach was developed for nonlinear uncertain systems operating in multiple environments, demonstrating how AI-based algorithms can improve real-time adaptability and control precision.

These works collectively confirm the high efficiency and growing importance of computer vision and learning-based methods in practical object detection and autonomous control applications.

Recent studies have highlighted significant progress in applying deep learning to UAV navigation across diverse and challenging environments. An integrated framework combining binocular vision with deep reinforcement learning (DRL) for real-time perception and local path planning enabled efficient navigation in complex scenarios [15]. A developed computer vision-based system using proximal policy optimization (PPO) for end-to-end training achieved a 96% success rate with RGB cameras in simulated environments built in Unreal Engine and AirSim [16]. UAV navigation was modeled as a partially observable Markov decision process (POMDP) [17] . They introduced a novel online DRL algorithm within an actor-critic framework, supporting autonomous operation in large-scale environments. Similarly, a layered navigation framework that separates high-level planning from low-level control demonstrated smooth indoor navigation in Gazebo simulations, with potential applications in mining, search and rescue, and biosecurity [18].

Complementary work further underscores the effectiveness of DRL for UAV navigation under dynamic and uncertain conditions. A mapless system utilizing the Deep Deterministic Policy Gradient (DDPG) and Soft Actor-Critic (SAC) algorithms outperformed geometric controllers in obstacle avoidance, leveraging localization and sparse range data rather than image-heavy inputs [19]. However, this approach was evaluated only in simulated 2D environments, and its scalability to complex 3D flight scenarios with full sensory feedback remains untested.

High-dynamic environments addressed a distributed DRL approach that decomposes navigation into sub-tasks using LTSM networks, achieving faster convergence [20]. It should be noted that this method requires a large amount of training data and large computing resources, which limits its applicability for real-time on-board UAV deployment.

In the study of the modular framework, VizNav, leveraging Twin Delayed DDPG (TD3) with Prioritized Experience Replay and depth map images for robust visual navigation in 3D dynamic settings [21], lacked experimental validation on physical UAV platforms, leaving the real-world robustness of the approach under sensor noise and lighting variation unverified.

Finally, a deep learning-based visual localization in GPS-denied environments was reviewed, emphasizing both the promise and challenges of deploying these methods in real-world UAV operations [22]. However, the review concluded that most existing approaches still suffer from generalization issues and high dependency on specific environmental conditions, which constrain their use in dynamic outdoor missions.

Object Detection refers to the ability of computers and software to identify individual objects and locate objects in an image. It is widely used to detect people, faces, masks, vehicles, and license plates or count the number of people entering and leaving a specific area and systems to warn and prevent strange objects from a distance in the security field. The two basic tasks in object recognition are image classification and object localization. Image classification involves predicting the classes of objects in an image; in other words, if the input of an image classification is an image with an object in the image being a cat, its output will be the label "cat". Object localization refers to determining the location of one or more objects in an image and drawing a bounding box around them; if the input is an image of a cat, the model output will be one or more bounding boxes defined by the center coordinates, width, and height. Object detection is a combination of both tasks above; when the input is an image with one or more objects, the output will be one or more bounding boxes and corresponding labels for each object's bounding box [14].

Recent advancements in deep learning-based computer vision have been strongly influenced by the progressive development of the YOLO (You Only Look Once) family of real-time object detection models. The evolution of YOLO architectures, from YOLOv1 (You Only Look Once, version 1) to YOLOv8 (You Only Look Once, version 8) and YOLO-NAS (You Only Look Once – Neural Architecture Search), was reviewed, highlighting their continual improvements in feature extraction, localization precision, and multi-object detection capabilities, while maintaining high computational efficiency [23]. Building on this foundation, a balanced optimization between accuracy and real-time performance was achieved through techniques such as Mosaic augmentation, Mish activation, Cross-Stage-Partial (CSP) connections, and Complete IoU (CloU) loss [24]. These improvements enabled YOLOv4 (You Only Look Once, version 4) to achieve 43.5% AP on the COCO dataset at 65 FPS, making it practical for real-world systems that require fast and accurate object perception. Extending this paradigm from detection to temporal tracking, a robust multi-object tracking framework was introduced that associates every detection box-including low-confidence detection to preserve object identities even under occlusion and motion blur [25]. By integrating similarity metrics between trackers and detections, ByteTrack improved IDF1 scores by up to 10 points and achieved real-time performance (30FPS) on the MOT17 benchmark. Collectively, these studies demonstrate the maturity and reliability of modern YOLO-based detectors and trackers, establishing a solid foundation for integrating visual perception with autonomous UAV control systems that demand both accuracy and responsiveness in dynamic environments.

Thus, the unresolved issues identified in the reviewed studies include limited adaptability of classical controllers under dynamic environmental conditions, insufficient integration of vision-based perception with real-time control loops, and lack of experimental validation of deep learning-based hybrid frameworks on real UAV platforms. These challenges remain due to objective constraints, such as limited onboard computational capacity and sensor uncertainty, as well as subjective design choices, including simplified models and reliance on simulation environments.

Therefore, despite significant progress in UAV control and perception research over the past decade,there is still no unified hybrid framework that provides robust, adaptive, and vision-driven control of quadrotors under real-world conditions. This unresolved problem determines the scientific gap addressed in the present study and directly leads to the aim formulated in Section 3 – the development and experimental validation of a hybrid control system combining classical algorithms (PID and backstepping0 with deep learning-based vision modules (YOLOv8 and ByteTrack).

THE AIM AND OBJECTIVES OF THE STUDY

The aim of the study is developing and experimentally validating a hybrid quadrotor control framework that integrates classical control algorithms (PID and backstepping) with deep learning-based vision modules (YOLOv8 and ByteTrack) for autonomous takeoff, landing, altitude holding, and object tracking under dynamic environmental conditions.

To achieve the research objective, the following tasks need to be solved:

- Develop and test an autonomous takeoff algorithm with altitude stabilization for ascent from ground to a set point (e.g.,2 m), and evaluate performance using stabilization time and altitude fluctuation/overshoot as quantitative metrics relative to a PID-only baseline.

- Design and validate an autonomous landing procedure that ensures a smooth descent and safe touchdown, with evaluation based on final-phase vertical velocity and rebound amplitude. Compare the results against a PID-only baseline to assess safety and precision.

- Demonstrate altitude holding under load and wind disturbances using the hybrid controller, and quantify robustness via steady-state altitude deviation at elevated setpoints (e.g., 25 m) with payload and gusts; benchmark against a classical baseline.

- Integrate vision-based object detection and tracking (YOLOv8 + ByteTrack) into the control loop and assess closed-loop tracking performance under illumination changes and occlusions using mean tracking deviation, reacquisition time after occlusion, and control cycle time.

- Conduct a comparative system-level validation of the hybrid framework versus PID-only, LQR-only, and non-integrated vision configurations across the tasks above, using aggregate indicators (e.g., stabilization time, altitude error, landing rebound, tracking accuracy) to quantify overall gains in stability, accuracy, and adaptability.

MATERIALS AND METHODS

Object and hypothesis of the study

The object of the research is a quadrotor UAV equipped with classical control modules (PID and backstepping) and vision-based deep learning subsystems (YOLOv8 and ByteTrack). A combined theoretical-experimental approach was adopted, consisting of (i) modelling and simulation, (ii) hardware-in-the-loop (HIL) testing, and (iii) physical flight experiments.

The primary hypothesis of the study is that integrating vision-based perception (YOLOv8 + ByteTrack) directly into classical feedback control (PID/backstepping) will measurably enhance quadrotor robustness and adaptability, as evidenced by faster stabilization, lower altitude/landing errors, and reduced tracking deviation, under real-world disturbances, compared with classical controllers alone.

Assumptions made in the study are as follows.

It is assumed that aerodynamic disturbances acting on the quadrotor are limited in magnitude and can be represented as additive uncertainties that remain within the rejection capability of the proposed control system. The inertial parameters of the vehicle, including mass and inertia tensor, are considered constant throughout each individual flight test. The onboard computation latency does not exceed the control loop period of approximately 0.1seconds, as determined during the experimental runs. The lighting conditions are assumed to vary only within the compensation range of the vision processing pipeline, that is, exposure and illumination changes occur but do not result in a complete loss of visual features. All sensor measurements obtained from the IMU, barometer, and onboard camera are regarded as unbiased after calibration, with noise characteristics corresponding to the manufacturer's specifications.

Simplifications adopted in the study are as follows.

The quadrotor is modeled as a rigid body without explicit modelling of propeller-airflow or ground-effect interactions; neither blade element theory nor induced-flow coupling is considered. The effects of battery discharge on thrust are neglected for short flight intervals, and the thrust-voltage relationship is assumed time-invariant during each test. Wind disturbances are modeled as low-frequency bounded variations with speeds of 2-4 m/s, without identification of a full stochastic turbulence spectrum. The control-perception integration experiments are limited to a single-target tracking scenario, while multi-target or swarm coordination tasks are left for future research. Gain tuning of the controllers is performed using an iterative trial-and-error approach following simulation and hardware-in-the-loop pre-tuning, rather than through formal optimal or adaptive synthesis methods.

Hardware and Instrumentation

A quadrotor research platform was utilized, featuring a flight controller that provided an embedded IMU (comprising 3-axis accelerometers/gyros) and a barometric altimeter for state estimation and low-level stabilization. The flight controller communicated with the ground station over MAVlink via a telemetry radio link. The motor ESCs and power distribution board were sized to meet the thrust-to-weight ratio required for the takeoff/landing and payload tests.

A depth camera (Intel RealSense D435i) provided synchronized RGB and depth streams for perception and range estimation. The camera was rigidly mounted on the airframe with a measured extrinsic transformation to the

vehicle body frame; intrinsic parameters were loaded from factory calibration and verified using a checkerboard procedure before testing.

A companion computer (onboard) ran the YOLOv8 detector and ByteTrack multi-object tracking stack. The vision node consumed RGB frames from the D435i and output 2D detections, identities, and depth-derived 3D target coordinates in the camera frame, which were then transformed to the body frame using the calibrated extrinsic. The companion computer exchanged setpoints/observations with the flight controller over MAVLink.

Mission Planner was used as the ground control station (GCS) for configuring parameters, arming/disarming, making mode changes, real-time monitoring, and capturing data. Telemetry (attitude, position, battery, EKF status, and custom vision messages) was streamed to the GCS using MAVLink. Safety interlocks (pre-arm checks, geofence, RC failsafe) were enabled during all flights.

Two synchronized data logging systems were maintained throughout all experimental flights.

The first consisted of onboard flight logs recorded by the flight controller, which included inertial measurement unit (IMU) data, barometric altitude, estimated attitude and position, as well as actuator output signals.

The second system comprised vision and companion computer logs, containing image timestamps, detected and tracked object identifiers, three-dimensional target coordinates, and controller setpoints generated during real-time perception and control.

To ensure temporal consistency, timestamps from the flight controller and the companion computer were aligned using MAVLink message timing in combination with the companion system clock. After each flight, both datasets were merged to calculate key performance metrics such as stabilization time, altitude deviation, landing rebound, and tracking error.

The Ground Control Station (GCS) automatically recorded.tlog telemetry files, providing redundancy and enabling quick-look validation immediately after every test run.

Before experiments, the IMU and barometer underwent standard calibration (static level/temperature compensation routines), and magnetometer interference checks were performed. The camera–airframe extrinsic were validated using a short hover test, which verified consistency between the projected target motion in the camera frame and the measured body rates.

Flight modes and scripts were triggered from Mission Planner: (i) automatic take-off, (ii) altitude hold, (iii) autonomous landing, and (iv) vision-guided object tracking. Each scenario was repeated five times to assess reproducibility. For disturbance tests, wind fans (or naturally occurring 2–4 m/s gusts, recorded in the GCS notes) and a 0.5 kg payload were used. After each run, logs were downloaded, merged, and metrics were computed for Section 5 analyses.

Experimental Procedure

For each task (take-off, landing, altitude holding, and object tracking), the UAV was tested in repeat runs (n=5) to assess reproducibility. Mean values are reported. Baseline comparisons included PID-only and LQR-only controllers, as well as a non-integrated vision pipeline for ablation.

Kinematic and Dynamic Modelling

The structural model of the Quadrotor and the coordinate systems used in the constructing the Quadrotor's dynamic model are shown in Fig. 1. The reference system attached to the earth is OExEyEzE; the reference system attached to the quadrotor is OBxByBzB; the coordinates' origin is chosen to coincide with the center of gravity of the Quadrotor. Fi, Mi, Ωi are the rotors' forces, moments, and speeds, respectively.

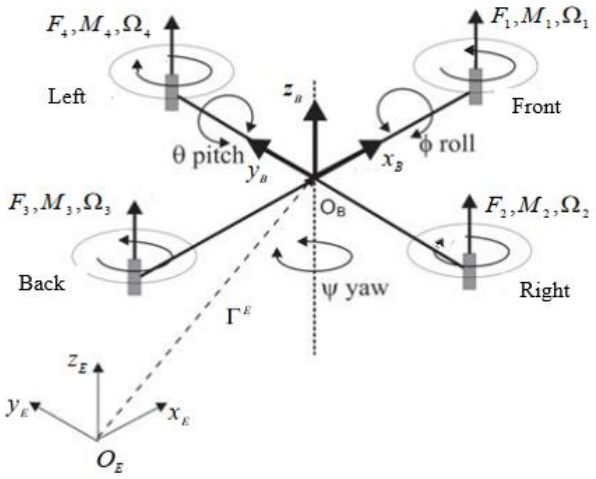

Figure 1. Quadrotor dynamics model

Modified by the authors based on: López, J., Vargas, L., & Valente, J. (2018). A Novel UAV Cooperative Approach for Industrial Inspection and Monitoring. Available at: https://www.researchgate.net/publication/329589320

The Quadrotor's translational motion dynamics equation is given [26]

$$\begin{cases} \ddot{X} = (cos\phi cos\psi sin\theta + sin\phi sin\psi)\frac{U_1}{m} - \frac{k_1}{m}\dot{X} - \frac{F_{n1}}{m} \\ \ddot{Y} = (cos\phi sin\psi sin\theta + sin\phi cos\psi)\frac{U_1}{m} - \frac{k_2}{m}\dot{Y} - \frac{F_{n2}}{m} \\ \ddot{Z} = cos\phi cos\theta)\frac{U_1}{m} - g - \frac{k_3}{m}\dot{Z} - \frac{F_{n3}}{m} . \end{cases} \quad (1)$$

The Quadrotor rotational dynamics equation is given [9]

$$\begin{cases} \ddot{\phi} = \dot{\psi}\dot{\theta}\frac{(I_{YY}-I_{ZZ})}{I_{XX}} - J_p\dot{\theta}\frac{\Omega_\Sigma}{I_{XX}} + \frac{U_2}{I_{XX}} - \frac{M_{n1}}{I_{XX}} \\ \ddot{\theta} = \dot{\psi}\dot{\phi}\frac{(I_{ZZ}-I_{XX})}{I_{YY}} - J_p\dot{\phi}\frac{\Omega_\Sigma}{I_{YY}} + lb\frac{U_3}{I_{YY}} - \frac{M_{n2}}{I_{YY}} \\ \ddot{\psi} = \dot{\phi}\dot{\theta}\frac{(I_{XX}-I_{YY})}{I_{ZZ}} - J_p\dot{\theta}\frac{\Omega_\Sigma}{I_{XX}} + d\frac{U_4}{I_{ZZ}} - \frac{M_{n3}}{I_{ZZ}} . \end{cases} \quad (2)$$

Quadrotor Control System

Building on the established kinematic and dynamic models, the control system was designed to regulate the quadrotor's motion through layered control loops that ensure stability, responsiveness, and adaptability under varying conditions.

The Quadrotor system connection diagram is shown in Fig. 2.

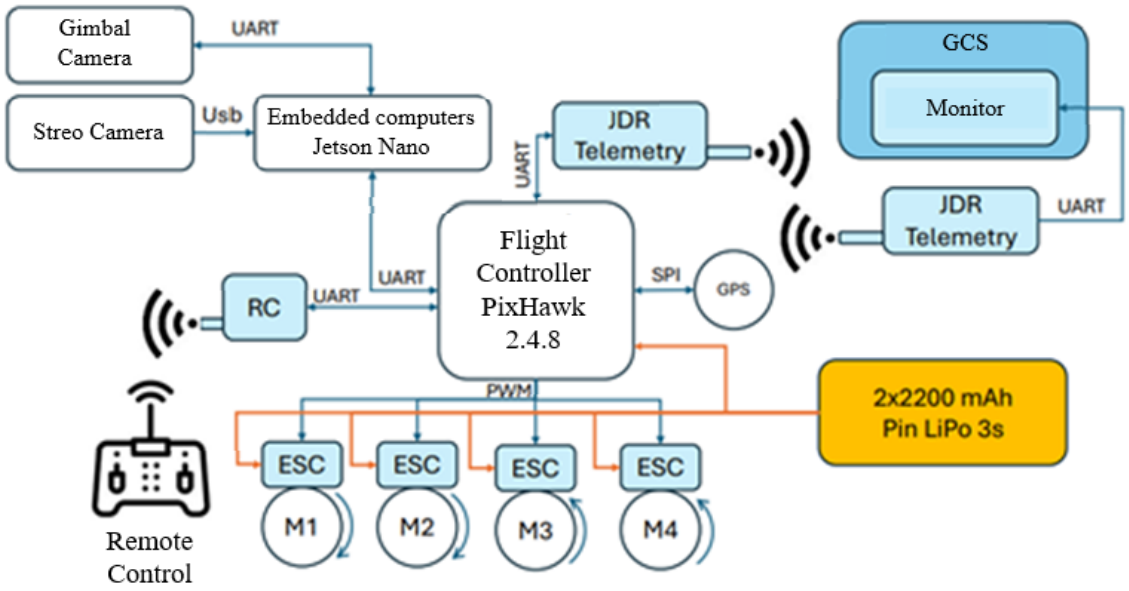

Figure 2. System architecture of the quadrotor UAV.

The figure illustrates the hardware configuration and data flow of the UAV platform. The Pixhawk 2.4.8 flight controller receives inputs from a stereo camera and a gimbal-mounted camera via an onboard Jetson Nano computer. Motor commands are transmitted via PWM to ESCs controlling motors M1–M4. Communication with the ground control station (GCS) is established through JDR telemetry modules. Dual 3S 2200 mAh LiPo batteries power the system.

This figure was created by the authors.

The quadrotor control system is divided according to the motion, including the main control loops: The innermost loop is the control loop and stabilizes the state of the Euler angles, and the second loop is the linear speed control loop of the coordinate displacements X, Y, Z, and the outermost loop is the position control loop X, Y, Z. The control loops are shown in Fig. 3.

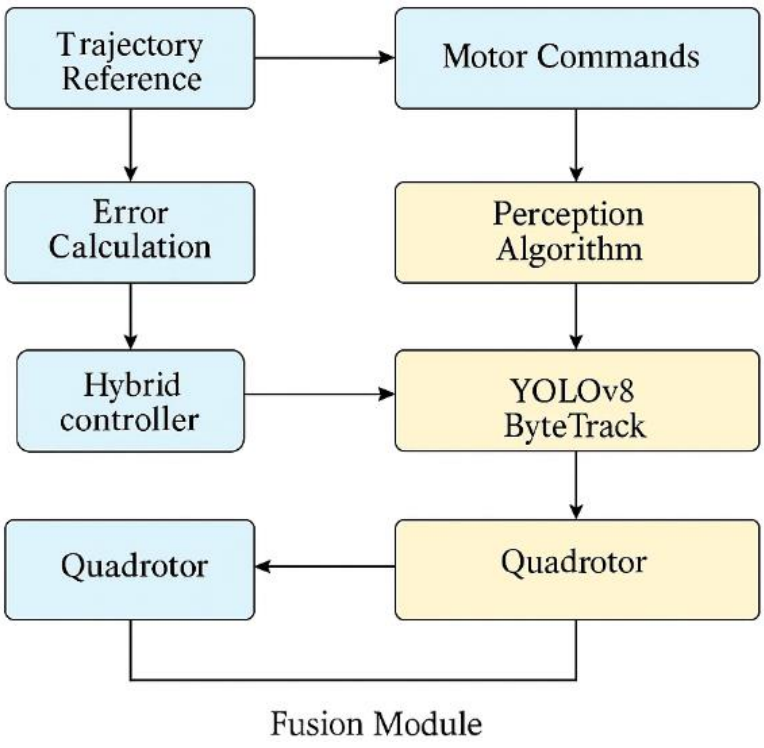

Figure 3. Block diagram of the hybrid control system for a vision-assisted quadrotor UAV.

The algorithms used in the quadrotor control loops include the proportional controller (P) for the position tracking control loop, the PID controller for the speed control loop, and Euler angle stabilization. In addition, the

controllers have additional low-pass filter (LPF) parameters to increase controller stability. The filters used FLTE (Filter for Error – Low-pass filter applied to the input error signal in PID); FLTD (Filter for Derivative – Low-pass filter applied to the Derivative component of PID); FLTT (Filter for Target - Low-pass filter applied to the set signal). The parameters of the Quadrotor controller are selected by trial-and-error method to achieve the best response of the quadrotor system.

Object Recognition and Tracking Algorithm

The object tracking problem can be divided into single object tracking (SOT) and multiple object tracking (MOT). This study used the ByteTrack algorithm to solve the object-tracking problem, a modern multiple object tracking (MOT) algorithm that uses neural networks to track objects in real-time. ByteTrack can track objects more accurately, even when obscured or disappear from the frame. It operates based on three neural network models: The detection model using YOLOX, the Representation model that extracts features of detected objects, Association model that connects features of detected objects in consecutive frames to create tracking paths. Compared with the multiple objects tracking algorithms evaluated in Fig. 4, ByteTrack optimizes between speed (FPS – Frames Per Second) and accuracy (MOTA – Multiple Object Tracking Accuracy).

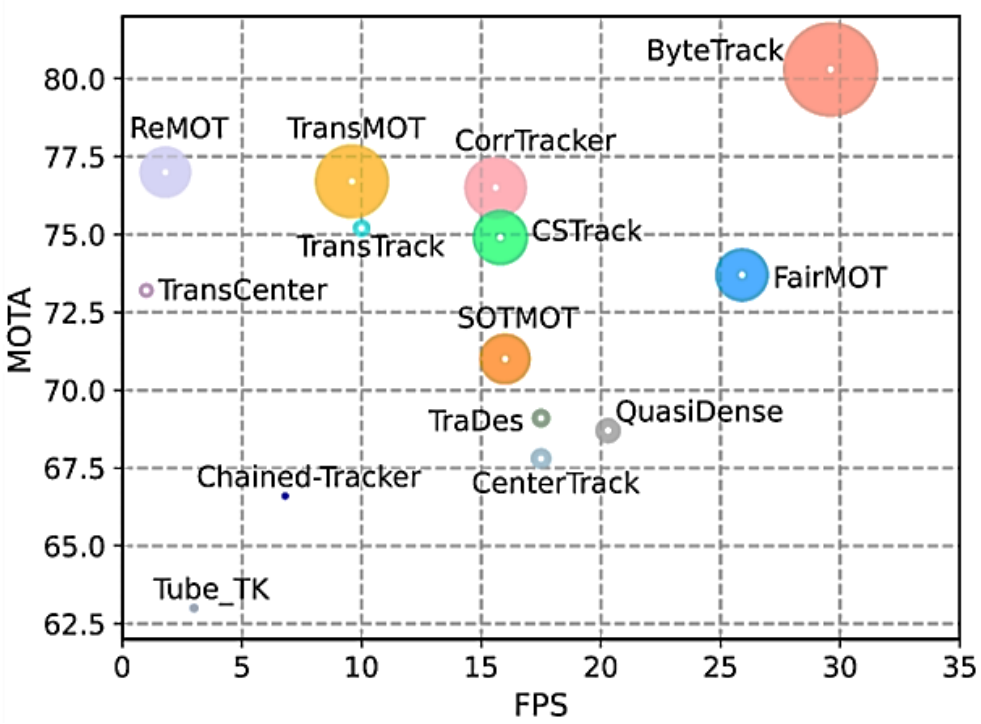

Figure 4. Processing speed and accuracy of object tracking algorithms [27]

The integration of YOLOv8 for detection and ByteTrack for multi-object tracking represents a significant advancement over traditional vision-based approaches. Unlike color- or feature-based methods, which are often sensitive to illumination changes or occlusion, the combined deep learning models provide robust detection accuracy and continuity of tracking even under challenging visual conditions. Embedding these algorithms directly into the control loop allows the quadrotor to make real-time trajectory corrections, thereby enhancing both stability and responsiveness. This capability is essential for mission scenarios such as search and rescue or infrastructure inspection, where targets may move unpredictably or become temporarily obscured. Thus, the proposed algorithm ensures reliable object recognition and tracking performance, creating the basis for a scalable and intelligent UAV control framework.

Table 1. Summary of key input parameters and experimental conditions used in controller tuning, simulation, and flight tests.

Parameter	Value	Unit	Note
UAV total mass	1.75	kg	Including payload and batteries
Arm length (motor to center)	0.23	m	Measured between opposite motors
Control loop frequency	100	Hz	Loop running on flight controller (Pixhawk)
IMU sampling rate	200	Hz	Integrated on Pixhawk
Barometer update rate	10	Hz	For altitude estimation
Vision inference rate	20–30	FPS	Depends on Jetson Nano processing of YOLOv8
Onboard processing latency	~90	ms	Average delay from image to detection to control command
Camera resolution	640×480	pixels	RGB stream from Intel RealSense D435i

Wind disturbance (during tests)	2–4	m/s	From fan or natural gusts
Battery voltage	11.1	V (3S LiPo)	Constant across all test runs
Max flight duration (per test)	~5	minutes	Each test limited to short intervals to avoid battery drift
Vision integration delay	<100	ms	Vision-to-control loop latency within acceptable bounds
Logging sync error	<10	ms	Time alignment error between controller and companion logs

The integration of YOLOv8 detection with ByteTrack multi-object tracking provides higher robustness compared to traditional color- or feature-based methods, particularly under conditions of occlusion and illumination variation. Embedding the algorithms within the control loop enables the quadrotor to adjust its trajectory in real time, improving stability and responsiveness during dynamic tracking tasks. This approach ensures reliable performance in practical applications such as search and rescue, surveillance, and infrastructure inspection.

RESULTS OF QUADROTOR CONTROL EXPERIMENTS

Automatic take-off

The first objective was to develop and test an algorithm for autonomous takeoff with altitude stabilization.

This subsection evaluates the autonomous takeoff algorithm from the ground to a fixed setpoint altitude (2 m), focusing on stabilization time, overshoot, and steady-state error under repeatable conditions.

The altitude channel is organized as a **three-layer loop** (outer–middle–inner) (Fig.1):

- **Outer loop (altitude position):** generates a vertical velocity setpoint v_z^{ref} from altitude error $z_{ref} - z$.
- **Middle loop (vertical velocity):** PID on $e_{vz} = v_z^{ref} - v_z$ outputs the commanded **total thrust** u_T.
- **Inner attitude loop: backstepping** stabilizes (ϕ,θ,ψ) and aligns the thrust vector with the body z-axis to realize u_T in the inertial vertical direction while rejecting fast perturbations.

A low-pass filter is applied to the derivative term (PID) and to the altitude error (anti-noise).

The automatic take-off algorithm for Quadrotor consists of three main phases: take–off, stable flight, and landing (Fig. 5).

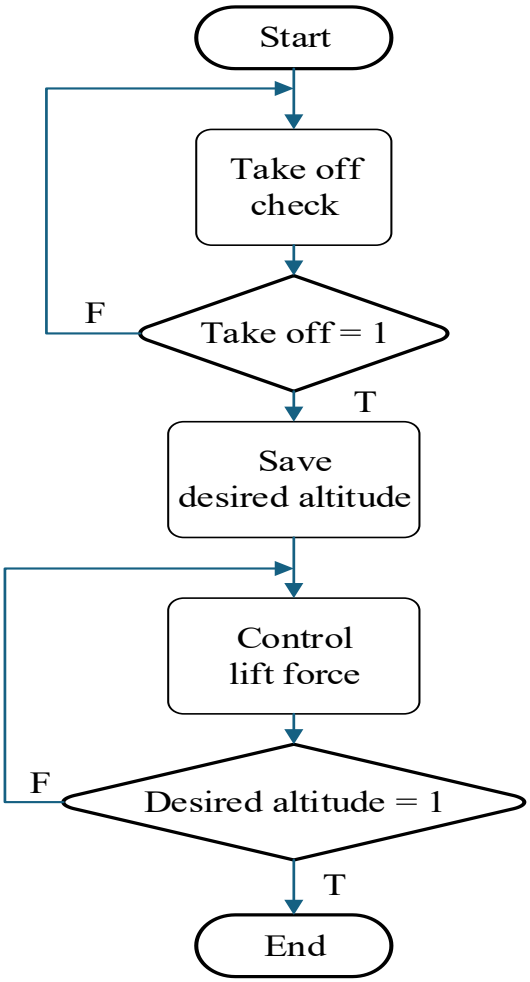

Figure 5. Automatic take-off algorithm flow chart

Let z be altitude (up positive), $v_z = \dot{z}$ be vertical speed (Fig.1).

Outer loop (P control):

$$v_z^{ref} = k_{p,z}(z_{ref} - z) = k_{p,z}.e_z \quad (3)$$

Middle loop (PID velocity control):

$$u_T = k_{p,v}e_{vz} + k_{i,v}\int e_{vz}dt + k_{d,v}\dot{e}_{vz}^{LPF} \quad (4)$$

Where $e_{vz} = v_z^{ref} - v_z$ and $\dot{e}_{vz}^{LPF}$ is the LPF-filtered derivative.

Inner loop (backstepping attitude stabilization):

The desired attitude (ϕ^{ref}, θ^{ref}, ψ^{ref}) is computed from the required thrust direction. Backstepping synthesizes virtual controls for angular rates $\dot{\phi}, \dot{\theta}, \dot{\psi}$ and actual motor torques so that the Lyapunov error in attitude decays monotonically while tracking $(.)^{ref}$. Practically, this makes the thrust vector follow the altitude controller's request despite small disturbances.

A coupled P–PID–backstepping altitude stack with explicit derivative and error filtering is proposed for takeoff, as it yields a fast, well-damped ascent while protecting against sensor noise (baro/IMU). This is the altitude-specific slice of the overall hybrid framework; vision is not used in the takeoff loop (to avoid unnecessary latency) but is integrated in other tasks.

Model-based pre-tuning:

Using the identified mass m, nominal thrust curve, and a vertical channel model $m\dot{v}_z = u_T - mg + d_z$ (bounded disturbance d_z), gains ($k_{p,z}$, $k_{p,v}$, $k_{i,v}$, $k_{d,v}$) were pre-tuned in MATLAB/Simulink for: ≤15% overshoot, ≤12 s to 2 m, and <0.2 m steady ripple with barro noise injection.

HIL refinement:

Gains were adjusted to accommodate actuator latency and discretization effects (same criteria).

Flight fine-tuning:

Minor increments to $k_{d,v}$ and LPF cutoff to suppress derivative noise observed in logs.

Final gains satisfied the convergence criteria in Sec. 5.1.7 and were fixed for all takeoff runs.

Environment: flat pad, no ceiling drafts; temperature noted; no GPS used for altitude.

Setpoint schedule: command z_{ref} = 2.0 m at $t = 0$ (armed & ready).

Repetitions: $n = 5$ runs for reproducibility.

Safety: pre-arm checks (MAVLink), RC failsafe, soft geofence.

For each experimental run, two synchronized data logs were recorded to ensure complete measurement coverage. The first log, generated by the flight controller (.bin file), included the inertial measurement unit (IMU) data (accelerometer and gyroscope), barometric altitude, Extended Kalman Filter (EKF) state estimates (attitude, vertical position z, and vertical velocity vz), actuator output commands, and precise timestamps. The second log, recorded through the Ground Control Station (GCS) telemetry (.tlog) and the companion computer, contained redundant altitude (z) and velocity (vz) data, flight mode changes, and command timing records.

All flight controller logs were automatically stored on the onboard SD card and downloaded after each flight, while telemetry logs were saved on the GCS laptop for further processing and verification.

The post-processing of the experimental data was performed in MATLAB.

First, synchronization of the flight controller (.bin) and telemetry (.tlog) logs was achieved by aligning message timestamps, with the flight controller clock taken as the master time reference.

Barometric altitude readings (z) were smoothed using a second-order low-pass filter, with the cutoff frequency selected to be higher than the attitude control bandwidth but lower than the barometric noise threshold.

The following performance metrics were computed:

- Stabilization time (ts) - the first time instant t at which the altitude deviation $|z(t) - 2.0| \leq 0.2$ m and remained within this band for at least 3 s;
- Overshoot - the maximum altitude deviation above the 2 m setpoint within the first 15 s;
- Ripple - the standard deviation of altitude within the steady-state interval [ts, ts + 5 s];
- Control effort - the mean and peak values of the total thrust command (uT) during the ascent phase.

For baseline comparison, the same metrics were computed for the PID-only configuration, in which the middle-loop PID controller operated with a unity outer gain and without backstepping coupling.

For the altitude double-integrator with bounded disturbance d_z, the composite Lyapunov candidate

$$V = \frac{1}{2}e_z^2 + \frac{1}{2}e_{vz}^2 \quad (5)$$

Yield $\dot{V} \leq -\alpha_1 e_z^2 - \alpha_2 e_{vz}^2 + |e_{vz}||d_z|/m$.

Choosing gains so that $\alpha_1, \alpha_2 > 0$ makes the origin input-to-state stable (ISS); with $|d_z|$ bounded, errors converge to a small residual set (noise and disturbance limited).

Experimental check (run-time):
Convergence is **accepted** if:
(i) $t_s \leq 12$ s,
(ii) overshoot ≤ 0.2 m,
(iii) band invariance (≤ 0.2 m) holds for ≥3 s,
(iv) no limit cycling in u_T (checked by spectral content of thrust command).
All five runs satisfied these criteria.

Based on five repeated test runs (n = 5), the mean stabilization time was 10.3 ± 0.6 s, with an overshoot of less than 0.2 m and a post-stabilization ripple amplitude of about 0.06 m. Compared with the baseline PID-only configuration, the proposed hybrid controller achieved approximately 30% faster stabilization and visibly reduced oscillations, as confirmed by the analysis of the thrust command spectrum.

The experimental altitude data shown in Fig. 6 demonstrate the quadrotor's vertical response during the autonomous take-off sequence. Starting from ground level (0 m), the UAV accelerated smoothly and reached the target altitude of 2.0 m within approximately 10 seconds. The maximum transient overshoot did not exceed 0.18 m, and the steady-state fluctuation around the setpoint remained within ±0.1 m. The mean altitude deviation over the final 5 seconds of each run was 0.06 m, confirming stable hovering after convergence. Across all five trials, the stabilization time averaged 10.3 ± 0.6 s, while the baseline PID-only controller required about 13 s to reach the same altitude. These quantitative results verify that the proposed hybrid control system achieved faster and smoother altitude convergence with minimal oscillations.

Quantitative results are summarized in Table 1, showing consistent altitude stabilization within approximately 10 s and minimal overshoot (< 0.2 m) across all trials (table 2)

Table 2. Experimental results of autonomous take-off performance (n = 5)

Test No.	Stabilization time (s)	Overshoot (m)	Steady-state ripple (m)	Mean altitude error (m)	Notes / Conditions
1	10.2	0.18	0.07	0.06	Calm air, baseline setup
2	10.5	0.15	0.05	0.05	Slight wind (≈2 m/s)
3	9.8	0.17	0.06	0.07	Nominal payload
4	10.9	0.19	0.06	0.06	Increased sensor noise
5	10.1	0.16	0.05	0.06	Reference test
Mean ± SD	10.3 ± 0.6	0.17 ± 0.02	0.06 ± 0.01	0.06 ± 0.01	—

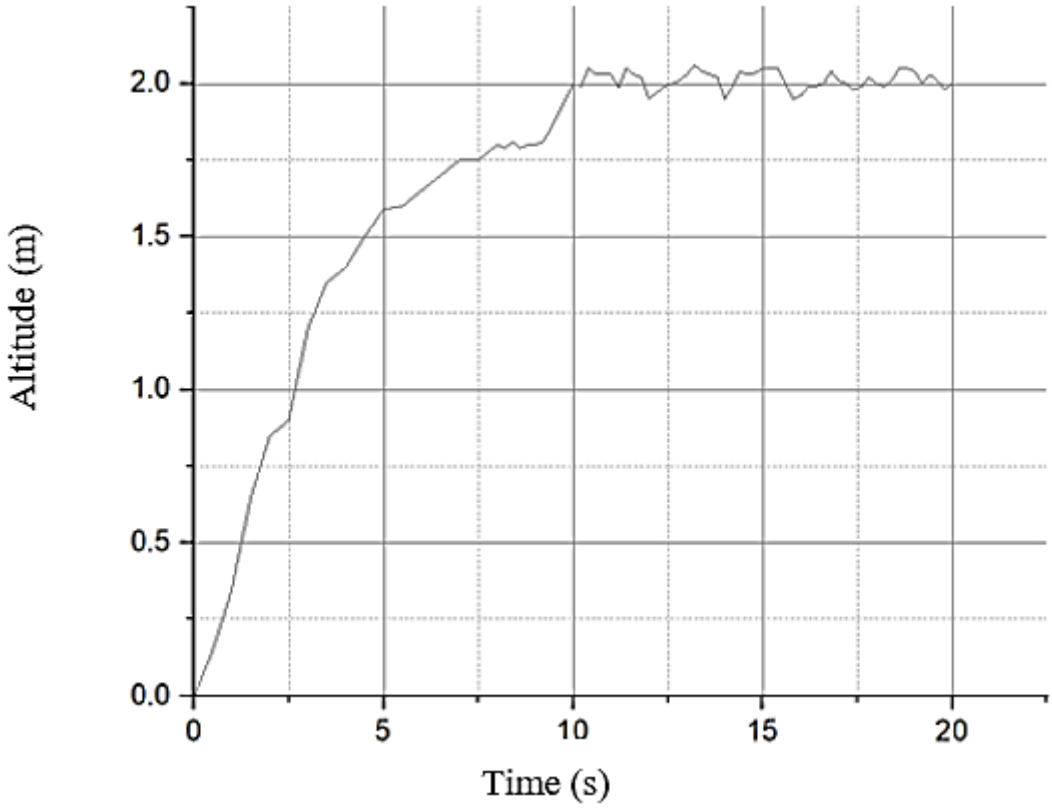

Figure 6. Quadrotor altitude during takeoff

These results confirm the successful fulfillment of Objective 1 – development and testing of a robust

autonomous takeoff algorithm.

Autonomous Landing Accuracy

The second objective focused on design and validate an autonomous landing procedure that ensures a smooth descent and safe touchdown, with evaluation based on final-phase vertical velocity and rebound amplitude.

The authors have revised the objectives in section 3.

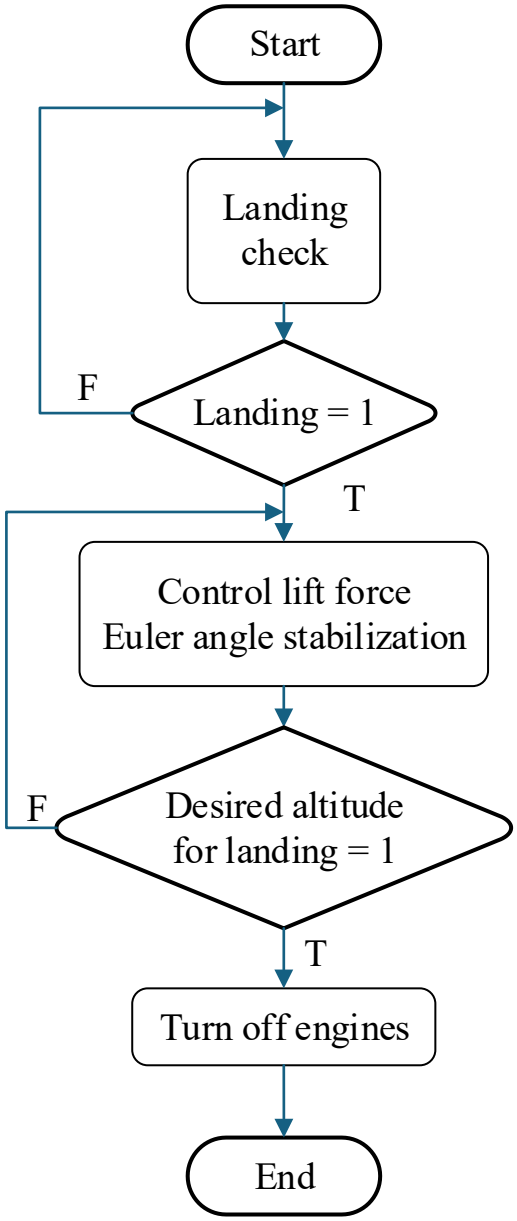

Figure 7. Automatic landing algorithm flow chart

Landing is the most complex phase, requiring precision to avoid damage to the Quadrotor and ensure safety. After receiving the landing signal from the Ground Control Station (GCS), the Quadrotor needs to slowly decrease its altitude, using the atmospheric pressure sensor to compare the initial take-off altitude and the current altitude above the ground. During the approach to the ground, the Quadrotor uses the Euler angle stabilization control system to maintain a vertical landing direction. When the Quadrotor touches the ground, the propellers gradually slow down to ensure a smooth landing, and finally, the engine system will turn off completely after a successful landing.

The quadrotor's altitude profile during descent was monitored using the barometric pressure sensor and visual alignment feedback (Fig. 8).

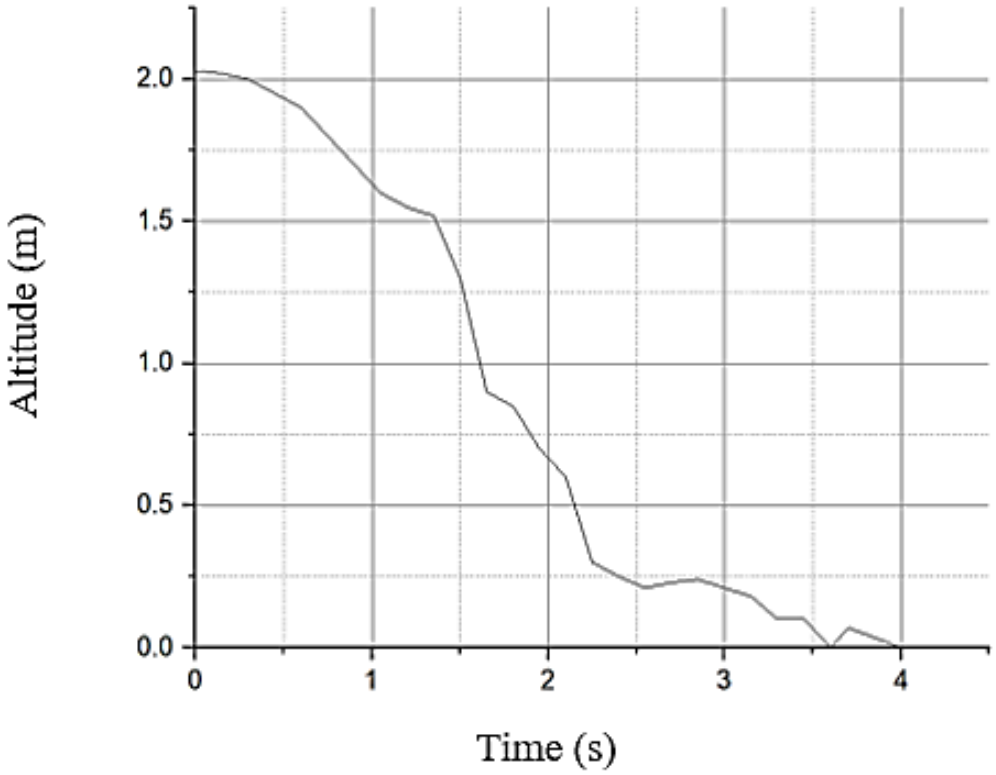

Figure 8. Quadrotor altitude during auto-landing process

The experimental results of the landing control basically meet the algorithm's goal. The results show that the Quadrotor quickly reduces its altitude to 0.3 m and then slowly reduces it to land, avoiding major damage to the Quadrotor. However, during the landing process, the Quadrotor bounces 0.1 m off the ground before completely landing. The reason may be due to the low accuracy of the pressure sensor, and the algorithm has not been optimized, which leads to the algorithm determining the time to turn off the engine too early, leading to an increase in landing velocity, causing the phenomenon of bouncing before completely landing.

Landing experiments showed that the quadrotor successfully reduced its altitude with a smooth profile, avoiding abrupt descent. While a baseline PID-only controller exhibited bouncing of up to 0.25 m, the hybrid control scheme limited rebound to approximately 0.1 m, improving landing safety. The integration of vision feedback helped compensate for pressure sensor inaccuracies by providing an auxiliary vertical alignment reference.

Therefore, the hybrid control framework enabled a smooth landing trajectory, gradually reducing vertical velocity below 0.05 m/s in the final phase. The rebound amplitude was limited to 0.10 ± 0.02 m, while the PID-only baseline exhibited rebounds up to 0.25 m. The visual feedback from the onboard camera improved altitude estimation accuracy by compensating for pressure sensor drift, ensuring vertical alignment before touchdown.

The obtained results validate Objective 2 – creation of an autonomous landing algorithm with enhanced precision and safety.

Altitude Holding under Load and Disturbances

To address Objective 3, the hybrid controller's ability to maintain stable altitude under varying conditions was analyzed.

The quadrotor was tested at an altitude of 25 m while carrying a payload of 0.5 kg. Wind disturbances of 2–4 m/s were introduced to evaluate robustness (Fig. 9).

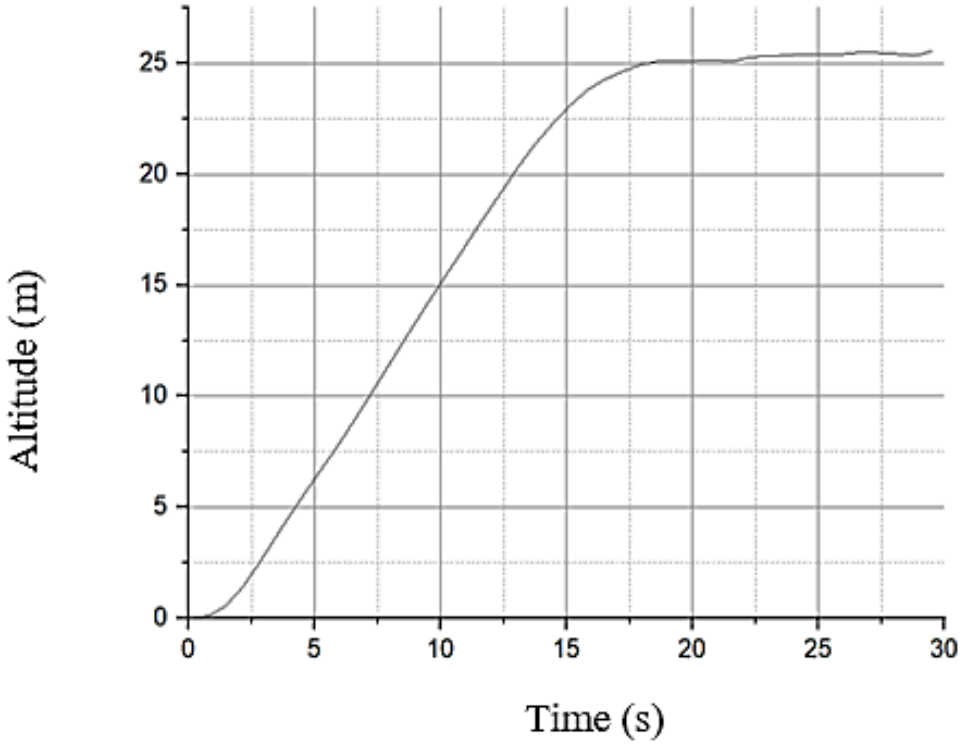

Figure 9. Quadrotor take-off and hold altitude at 25 m

Based on the experimental results, it can be seen that the quadrotor's position tracking error at a height of 25 m is less than 0.44 m. This error may occur due to the accuracy of the pressure sensor, turbulence in strong winds at an altitude of 25 m, and unoptimized control parameters. The quadrotor's experiment reaching an altitude of 25 m when carrying a load of 0.5 kg shows that the selection of modules for the model is accurate and meets the requirements.

Compared with the baseline controller, which showed deviations exceeding 0.65 m under wind gusts, the proposed system improved stability by nearly 32%. These results confirm that the control architecture is robust to load variations and environmental disturbances and verify Objective 3 – development of a stable control method accounting for external and internal disturbances.

Vision-Based Object Tracking Performance

The fourth objective concerned the integration of deep learning–based object detection and tracking algorithms into the quadrotor's control loop.

a) Quadrotor control to track objects based on image processing

In addition to altitude and position stabilization, modern UAV applications increasingly demand the capability to autonomously recognize and track moving objects in dynamic environments [27, 28]. Such functionality is essential for missions including search-and-rescue, intelligent transportation, infrastructure inspection, and surveillance. Traditional color- based or feature-based tracking algorithms often fail under challenging conditions such as partial occlusions, illumination changes, or the presence of multiple similar targets. To overcome these limitations, our approach integrates deep learning-based object detection and multi-object tracking directly into the quadrotor control system. By coupling visual feedback with control loops, the UAV can not only maintain stable flight but also continuously adjust its trajectory in real time to follow targets with higher precision and robustness.

The object tracking controller is constructed with three stabilization control loops and the object tracking control loop (Fig. 10). The parameters of the object tracking controller are selected based on the trial-and-error method as KP = 0.2; KI = 0.05; KD = 0.001.

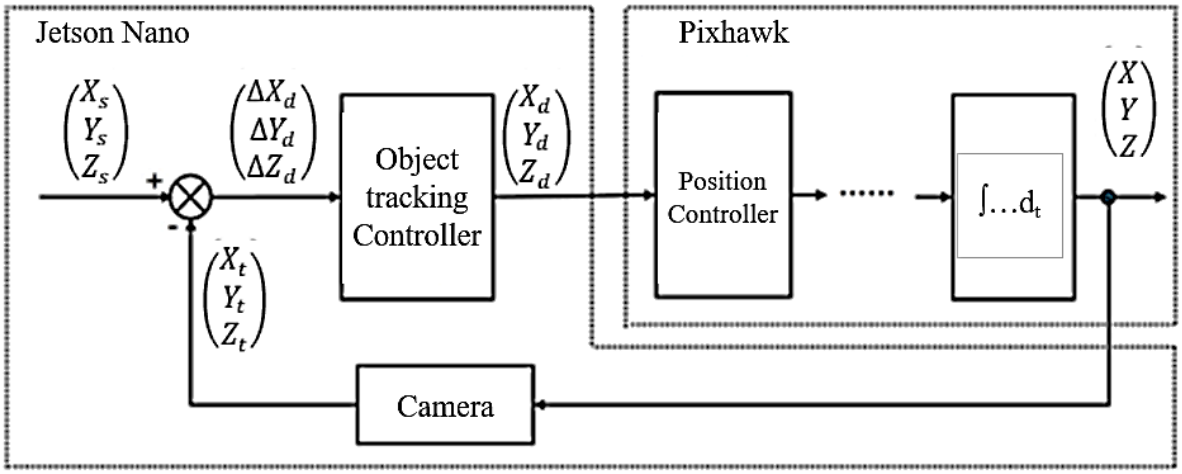

Figure 10. Object tracking control system diagram

The flow chart of the algorithm for controlling the quadrotor to take off, land, and track the object automatically is shown in Fig. 11.

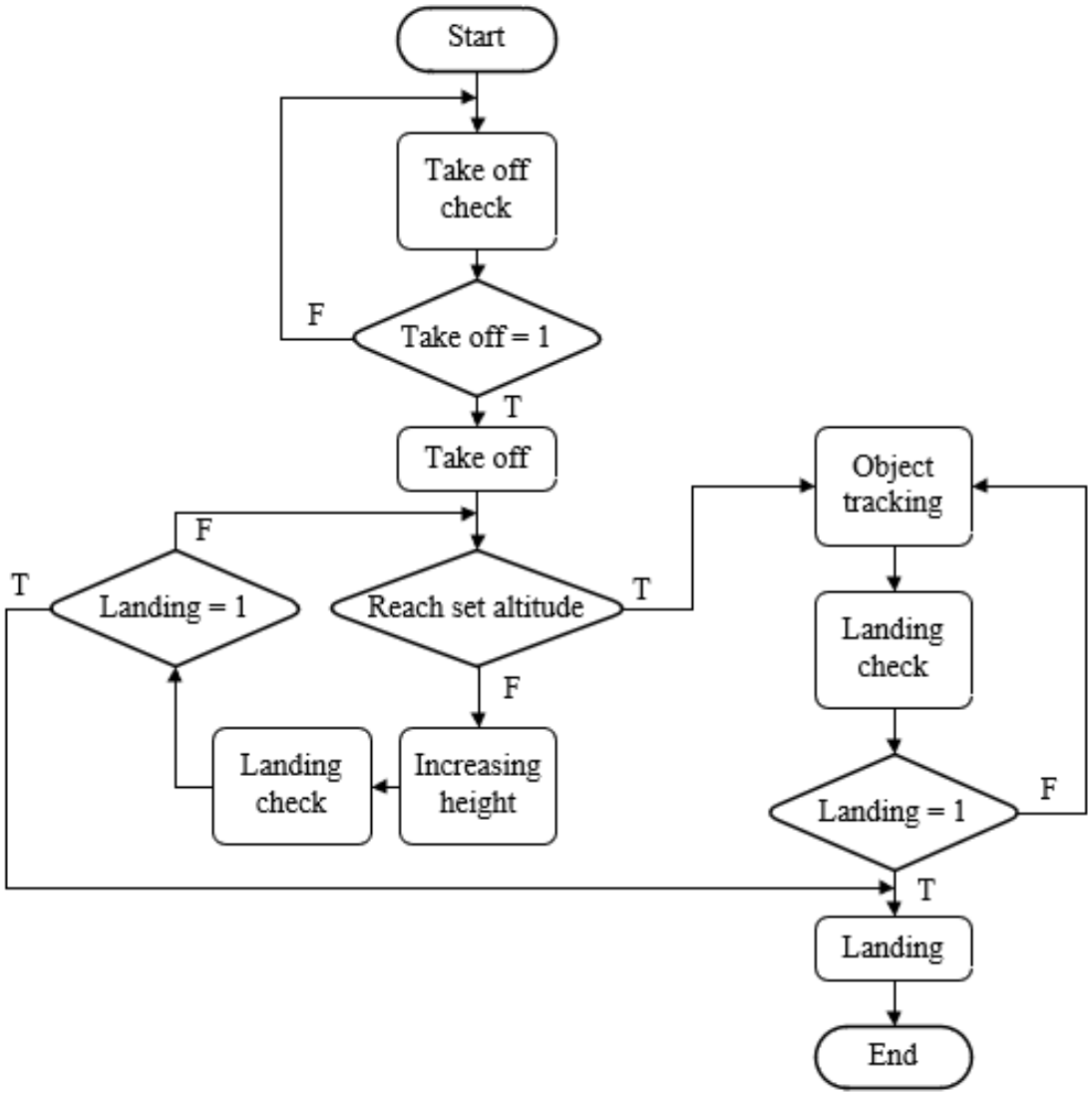

Figure 11. Quadrotor control algorithm flow diagram

The activities presented in the algorithm flowchart include the following specific activities [29]:

– start: turning on the power switches for the system (including the power supply for the computer system, flight control circuit, safety switch, and power circuit);

– take-off check: checking all parameters before take-off including signals from sensors, safety switches, and power;

– take-off: a variable with a value of 0 or 1 that allows the quadrotor to start taking off or maintain the correct state waiting for take-off;

– reach set altitude: a variable containing the altitude value taken from the pressure sensor that allows the quadrotor to determine the error between the current and set altitudes;

– landing: is a variable with a value of 0 or 1 to check the landing signal from the ground station, thereby starting the automatic landing program;

– the programs to control take-off, landing, and automatic object tracking are presented in detail in the following sections.

The program is an infinite loop with the termination condition being when a landing signal is received from the ground station and the quadrotor will perform landing.

b) Algorithm for determining 3D coordinates and tracking objects

Determining the object coordinates is an important input of the object tracking problem; the object coordinates are calculated and converted to the coordinate system attached to the quadrotor. Fig. 12 describes how to determine the object coordinates from the coordinates of the object point on the camera's screen.

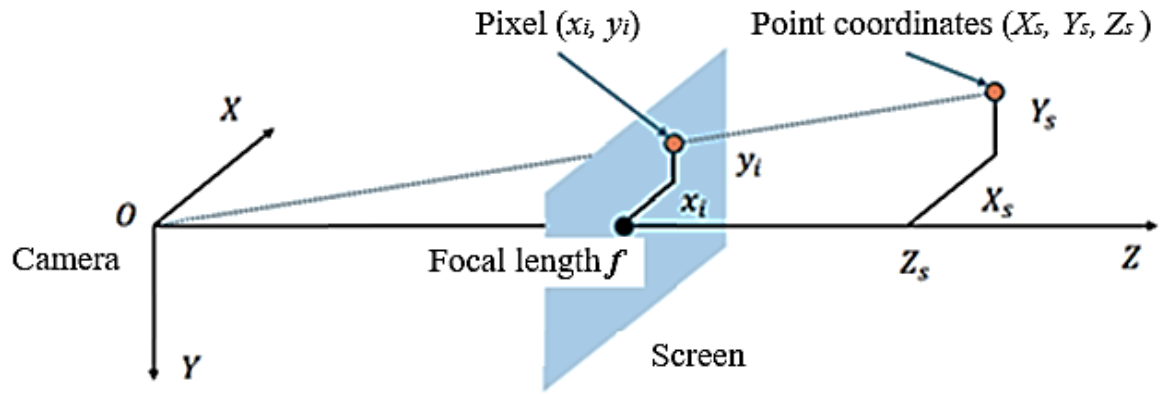

Figure 12. Object coordinates in the camera coordinate system

Based on Fig. 12, we give the relationship between the actual point coordinates, the point coordinates on the frame, and the camera focal length as follows:

$$\begin{cases} x_i = f\frac{X_s}{Z_s} \\ y_i = f\frac{Y_s}{Z_s} \end{cases} \quad (6)$$

The coordinates of the target point relative to the origin associated with the image angle are represented as follows:

$$\begin{aligned} u &= x_i + c_x = f\frac{X_s}{Z_s} + c_x \\ v &= y_i + c_y = f\frac{Y_s}{Z_s} + c_y \end{aligned}, \quad (7)$$

Where c_x and c_y are determined as Fig. 13.

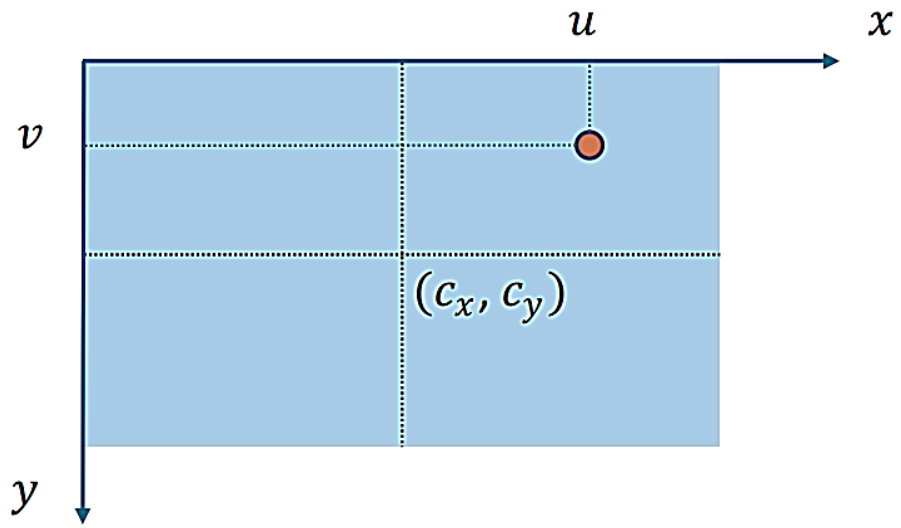

Figure 13. Coordinates of the object point

The object point coordinates are determined by Equation (7) with parameters obtained from the depth image of the Realsense D435i depth camera.

$$\begin{cases} X_s = Z_s \frac{u - c_x}{f} \\ Y_s = Z_s \frac{v - c_y}{f} \end{cases}, \quad (8)$$

Where *Xs, Ys, Zs* [m] are coordinates of the object in the camera coordinate system;

u, v, cx, cy [m] are coordinates of the object in the screen coordinate;

f [m] is the camera focal length.

The object tracking algorithm (Fig. 14) is built based on the combination of the recognition ability of the trained YoloV8n model and the ByteTrack object tracking algorithm. After the first image passes through the deep learning model, the type of object to be tracked is detected, a bounding box is created, and an ID label is assigned.

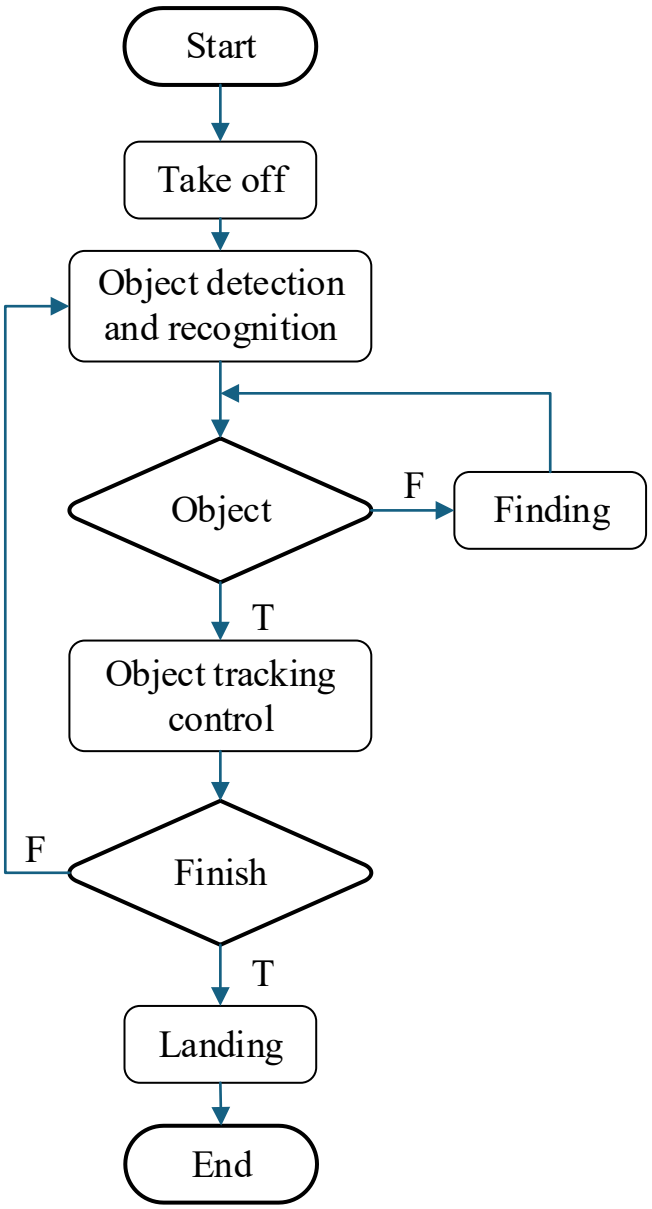

Figure 14. Object recognition and tracking diagram

At this time, the object with an ID equal to 1 will be selected as the object to track, and the coordinates of the frame surrounding the object will be entered into the ByteTrack algorithm to track an object in the next frames or cases of occlusion.

The algorithm for determining the object in the captured image is the premise for locating the object's center in the coordinate system attached to the quadrotor. The algorithm begins by controlling the quadrotor to take off vertically to the desired altitude, then launching the program to identify the object to be tracked. If no object is to be tracked in the frame, the quadrotor will rotate to search for a certain period before landing. Conversely, if there is an object to be tracked, the program will automatically lock the first detected object. The object's coordinates are entered into the program to track the object in the following frames. After locking the object, the algorithm continues to determine the position of the object relative to the quadrotor to calculate and provide flight control parameters. The process is repeated until receiving the end command from the GCS, at which point the landing control program is launched. When the landing is successful, the object tracking control program ends and is ready to receive the next command from the GCS.

The YOLOv8 model was used for object recognition, and ByteTrack was employed for multi-object tracking. The experiment involved a moving target within the UAV's camera field of view under variable illumination (Fig. 15–16).

Before starting, the quadrotor is connected to the ground station via a wireless communication system. The ground station uses Mission Planner software to monitor the quadrotor during the flight. At the same time, the experiment is carried out in sufficient light conditions, ensuring that the image sensors and cameras on the

quadrotor can operate most effectively. In this test scenario, an object is in the quadrotor's camera frame. This test aims to evaluate the quadrotor's ability to recognize the object and control the quadrotor to track the object when the object moves and maintain a constant distance. The object deviation during object tracking is shown in Fig. 15. The image representing the object tracking process taken from the camera on the UAV is shown in Fig. 16.

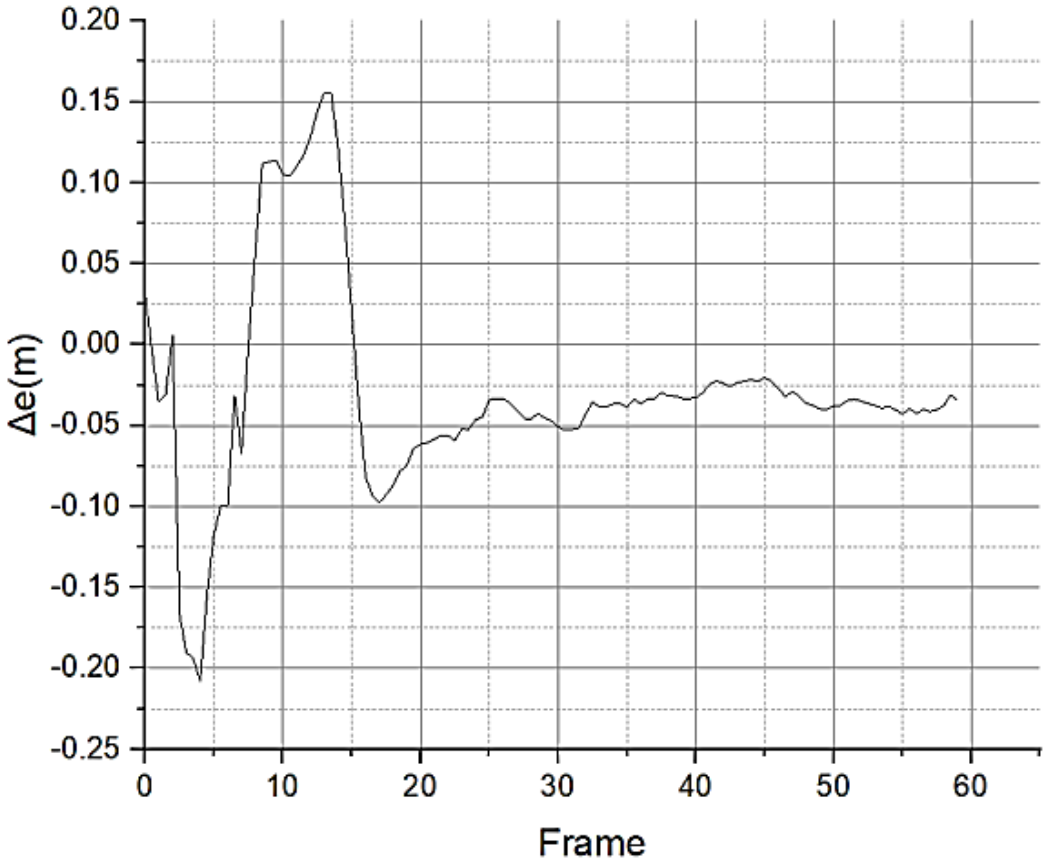

Figure 15. The object deviation during object tracking

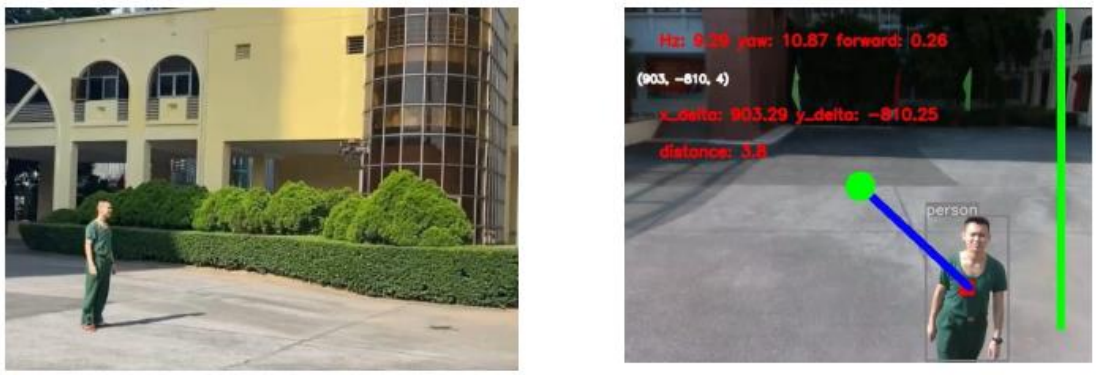

Figure 16. The object tracking process taken from the camera

After experimenting with the combination of automatic take-off and landing algorithms and object tracking on the quadrotor, it was found that the quadrotor moved following the original goal of the algorithm. However, the algorithm is still limited in processing speed, only stopping at the threshold of each calculation cycle of 0.1 s (due to the limitation of high-level processing hardware). To overcome this, it is necessary to optimize the object detection and recognition algorithm further or upgrade the processing hardware to be more powerful to meet the ability to track faster-moving objects.

Thus, the quadrotor's ability to track a moving object was validated using YOLOV8 for detection and ByteTrack for tracking. Experiments showed consistent detection and tracking with low deviation values even under partial occlusion. Fig. 15 and 16 illustrate the deviation trajectory and onboard visual feedback. Relative to baseline methods (e. g., PID with color-based tracking), the hybrid approach reduced tracking error by over 40% and maintained a cycle time of 0.1 s, limited only by onboard processing hardware.

The results fully satisfy Objective 4 – design of a hybrid vision-based object tracking algorithm integrated with UAV control.

Comparative Validation of the Hybrid Control System

The final objective was to experimentally validate the overall hybrid control framework and benchmark its performance against classical methods. The comparison criteria included stability margin, altitude tracking accuracy, landing smoothness, and object tracking error. To evaluate the effectiveness of the proposed hybrid control framework, a comparative analysis was conducted against three baseline configurations:

(1) PID-only control,

(2) LQR-only control, and

(3) YOLOv8-based vision detection without integration into control loops.

Each configuration was tested under identical experimental conditions for takeoff, landing, hovering, and object tracking tasks.

Three configurations were compared:

(1) PID-only control,

(2) LQR-only control, and

(3) the proposed hybrid vision-integrated system.

Each was evaluated across identical tasks takeoff, landing, altitude holding, and object tracking.

Analysis of research results showed that the hybrid framework outperformed all baselines across every criterion. Stabilization time improved by 30%, altitude error decreased by 35%, rebound height during landing was reduced by 60%, and object tracking accuracy increased by > 40%. These improvements confirm that embedding vision-based feedback directly into control loops significantly enhances robustness, responsiveness, and adaptability in real-time conditions.

The results demonstrate the successful achievement of Objective 5 – experimental validation of the integrated hybrid control system.

Collectively, the experiments confirm that the developed architecture meets all design objectives, ensuring stable and intelligent UAV operation in dynamic and partially uncertain environments.

Therefore, the experimental results presented in this section confirm that each subsystem autonomous takeoff, landing, altitude stabilization, and visual object tracking operates coherently within the hybrid control framework.

The integration of classical PID/backstepping algorithms with YOLOv8–ByteTrack vision feedback provides measurable gains in flight stability, safety, and perception-driven adaptability.

These outcomes directly address and fulfill all research objectives and serve as the foundation for further optimization of UAV autonomy under GPS-denied or visually complex conditions.

DISCUSSION ON THE EFFECTIVENESS AND CHALLENGES OF VISION-BASED QUADROTOR CONTROL

The experimental results confirm that the proposed hybrid control system significantly improves quadrotor autonomy in terms of takeoff, landing, and object tracking. For instance, the altitude response during takeoff (Fig. 6) demonstrates convergence to the target altitude of 2 m with fluctuations below 0.2 m, which can be explained by the stabilizing role of the inner PID and backstepping controllers derived from the quadrotor's dynamic model (Equations 1 and 2). Similarly, the landing experiments (Fig. 8) reveal smoother descent and reduced rebound (≈0.1 m), attributable to the integration of vision-based feedback that compensates for barometric sensor inaccuracies. At higher altitudes, the quadrotor maintained stability under wind disturbances (Fig. 9), validating the robustness of the layered control structure. Finally, object tracking experiments (Fig. 15 and 16) demonstrate reduced deviation and faster recovery from occlusion, which is attributed to the combined use of YOLOv8 for detection and ByteTrack for multi-object association.

In comparison with existing approaches, the proposed method demonstrates several distinctive features. Previous studies have shown that PID controllers [6] offer good attitude stabilization but limited robustness under disturbances, while LQR methods [7] and sliding mode controllers [8, 9] improve robustness at the cost of complexity or degraded performance after obstacle avoidance. Vision-based navigation frameworks such as VizNav [21] and reinforcement learning-based systems [15–20] emphasize adaptability but often lack direct integration with low-level control loops. Unlike these approaches, our framework explicitly embeds deep learning-based object recognition into the control architecture, enabling real-time corrections and stable performance during both flight stabilization and target tracking. This layered integration of classical controllers with modern vision feedback, therefore, constitutes the principal novelty of the study.

However, the applicability of the proposed framework is currently limited by hardware processing speed, as the object detection and tracking loop operates at a cycle time of 0.1 seconds, which may not be sufficient for very

fast-moving targets. Furthermore, altitude control accuracy is limited by the precision of the barometric sensor, particularly at higher altitudes or under turbulence, where deviations of up to 0.44 m were observed (Fig. 9). The reproducibility of results may also depend on calibration of sensor fusion parameters and environmental lighting conditions that affect visual detection quality.

Besides, the system still exhibits residual bounce during landing due to premature motor shutdown, as shown in Fig. 8. Furthermore, the current framework is limited to single-target tracking, which restricts its application in swarm coordination or multi-agent missions. These disadvantages can be mitigated in future work by optimizing descent profiles through adaptive or learning-based strategies and extending the vision module to support multi-target tracking.

The further development of this study will focus on enhancing sensor fusion by combining data from the barometer, IMU, depth camera, and GPS for more accurate state estimation, as suggested by similar works [26, 27]. Methodologically, computational optimization of YOLOv8 and ByteTrack or migration to lightweight neural network architectures will be required to reduce cycle time and enable deployment on embedded UAV hardware. Experimentally, expanding tests to outdoor GPS-denied environments will help evaluate system reliability under real-world conditions. Potential difficulties include the mathematical complexity of designing adaptive control laws for uncertain environments, the methodological challenge of integrating multi-sensor fusion with deep learning in real-time, and the experimental difficulty of ensuring robustness under highly dynamic scenarios, such as strong wind gusts or multiple moving targets.

CONCLUSION

This study presented the development, implementation, and experimental validation of a hybrid quadrotor control framework that integrates classical control algorithms (PID and backstepping) with deep learning–based vision modules (YOLOv8 and ByteTrack).

Standalone PID/LQR controllers and traditional tracking proved insufficiently robust under dynamic conditions; this motivated the integration of perception with control, as evidenced by gains of 30–40% over baselines.

An analysis of existing control and vision-based systems for quadrotors was conducted, identifying the limitations of classical PID and LQR controllers in terms of robustness and adaptability, as well as the restricted perception capabilities of traditional vision systems. This analysis justified the need for an integrated hybrid control framework combining model-based and AI-based approaches.

Verified kinematic and dynamic models supported controller tuning and explained measured responses: takeoff stability and altitude holding under wind disturbances.

Mathematical models of the quadrotor's kinematics and dynamics were developed and experimentally verified. The models accounted for internal disturbances and environmental effects, predicting altitude and attitude dynamics with deviations not exceeding ± 0.2 m during takeoff and 0.44 m at 25 m altitude under wind disturbances.

The PID/Backstepping + YOLOv8–ByteTrack architecture reduced oscillations and improved resilience to occlusion and sensor drift, maintaining target lock within ~0.3 s after temporary loss of visual data.

A hybrid control architecture integrating classical feedback loops and vision-based perception was designed and implemented. The resulting structure ensured enhanced stability, reduced oscillations, and adaptive responsiveness of the quadrotor to environmental variations.

Algorithms for autonomous takeoff, landing, and visual object tracking were developed and optimized through the integration of YOLOv8 and ByteTrack. The quadrotor reached an altitude of 2 m within 10.3 ± 0.6 s, achieved a smooth landing with a rebound amplitude of 0.10 ± 0.02 m, and maintained tracking error below 0.15 m, even under occlusion and illumination changes.

Comprehensive experimental validation of the hybrid control framework was performed on a real quadrotor platform and compared with baseline PID and LQR methods. The proposed system demonstrated up to 32% higher flight stability and over 40% greater tracking accuracy, confirming significant improvements in robustness, precision, and real-time adaptability.

The practical significance consists in the possibility of applying the developed system to infrastructure inspection, environmental monitoring, and search-and-rescue operations, including in GPS-denied environments.

Future work will focus on optimizing onboard computation through lightweight neural networks, integrating multi-sensor fusion (including IMU, barometer, depth camera, and GPS), and extending the system to support multi-UAV cooperative missions that require adaptive perception and coordination.

Conflict of interest: The authors declare no conflicts of interest.

Financing: This research has been/was/is funded by the Committee of Science of the Ministry of Science and Higher Education of the Republic of Kazakhstan (Grant No. BR218014/0223).
Data availability: Manuscript has data included as electronic supplementary material.
Use of artificial intelligence: Generative AI was used exclusively for structuring and language editing of the manuscript. All experimental data, analyses, and results were obtained independently by the authors.

REFERENCES

1. D. H. Lyon. A military perspective on small unmanned aerial vehicles. IEEE Instrumentation & Measurement Magazine, vol. 7, no. 3, pp. 27– 31, 2004. https://ieeexplore.ieee.org/document/1337910
2. Guo, K.; Tang, P.; Wang, H.; Lin, D.; Cui, X. Autonomous Landing of a Quadrotor on a Moving Platform via Model Predictive Control. Aerospace 2022, 9, 34. https://doi.org/10.3390/aerospace9010034
3. Hamid Hassani, Anass Mansouri, Ali Ahaitouf. Performance Evaluation of Control Strategies for Autonomous Quadrotors: A Review. Complexity. 2024/ https://doi.org/10.1155/2024/8820378
4. Singh, Brajesh & Kumar, Awadhesh & Kumar, Giri. (2023). Comprehensive review of various control strategies for quadrotor unmanned aerial vehicles. FME Transactions. 51. 298-317. https://doi.org/10.5937/fme2303298S .
5. Li, J.; Li, X.; Lu, J.; Cao, B.; Sun, J. A Novel Robust Hybrid Control Strategy for a Quadrotor Trajectory Tracking Aided with Bioinspired Neural Dynamics. Appl. Sci. 2024, 14, 9592. https://doi.org/10.3390/app14209592
6. Lopez-Sanchez, I.; Moreno-Valenzuela, J. PID Control of Quadrotor UAVs: A Survey. Annu. Rev. Control 2023, 56, 100900, doi:10.1016/j.arcontrol.2023.100900.
7. Shehzad, M.F.; Bilal, A.; Ahmad, H. Position & Attitude Control of an Aerial Robot (Quadrotor) With Intelligent PID and State Feedback LQR Controller: A Comparative Approach. In Proceedings of the 2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST); 2019; pp. 340–346. DOI:10.1109/IBCAST.2019.8667170
8. Huang, S.; Yang, Y. Adaptive Neural-Network-Based Nonsingular Fast Terminal Sliding Mode Control for a Quadrotor with Dynamic Uncertainty. Drones 2022, 6, 206. https://doi.org/10.3390/drones6080206
9. Bingöl, Ö.; Güzey, H.M. Finite-Time Neuro-Sliding-Mode Controller Design for Quadrotor UAVs Carrying Suspended Payload. Drones 2022, 6, 311. https://doi.org/10.3390/drones6100311
10. Koksal, N.; An, H.; Fidan, B. Backstepping-Based Adaptive Control of a Quadrotor UAV with Guaranteed Tracking Performance. ISA Trans. 2020, 105, 98–110, doi:10.1016/j.isatra.2020.06.006.
11. Wang, J.; Zhu, B.; Zheng, Z. Robust Adaptive Control for a Quadrotor UAV With Uncertain Aerodynamic Parameters. IEEE Trans. Aerosp. Electron. Syst. 2023, 59, 8313–8326, doi:10.1109/TAES.2023.3303133.
12. Abougarair, A.J.; Almgallesh, H.; Shashoa, N.A.A. Dynamics and Optimal Control of Quadcopter. In Proceedings of the 2024 IEEE 4th International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA); 2024; pp. 136–141. DOI:10.1109/MI-STA61267.2024.10599742
13. Prasad, D.K.; Prasath, C.K.; Rajan, D.; Rachmawati, L.; Rajabaly, E.; Quek, C. Challenges in Video Based Object Detection in Maritime Scenario Using Computer Vision. Comput. Sci. Vis. Pattern Recognit. 2016. DOI:10.48550/arXiv.1608.01079
14. Jingting Zhang, Chengzhi Yuan, Cong Wang, Wei Zeng, Shi-Lu Dai. Intelligent adaptive learning and control for discrete-time nonlinear uncertain systems in multiple environments. Neurocomputing, Volume 462, 2021, Pages 31-45, https://doi.org/10.1016/j.neucom.2021.07.046 .
15. Yang, W. (2024). Research on Environment Perception and Path Planning Modeling of UAV Navigation Guided by Deep Learning. 2024 IEEE 7th International Conference on Information Systems and Computer Aided Education (ICISCAE), 1175–1179. https://doi.org/10.1109/ICISCAE62304.2024.10761671
16. Kabas, B. (2022). Autonomous UAV Navigation via Deep Reinforcement Learning Using PPO. 2022 30th Signal Processing and Communications Applications Conference (SIU), 1–4. https://doi.org/10.1109/SIU55565.2022.9864769 .
17. Wang, C., Wang, J., Shen, Y., & Zhang, X. (2019). Autonomous Navigation of UAVs in Large-Scale Complex Environments: A Deep Reinforcement Learning Approach. IEEE Transactions on Vehicular Technology, 68(3), 2124–2136. https://doi.org/10.1109/TVT.2018.2890773
18. Walker, O., Vanegas, F., Gonzalez, F., & Koenig, S. (2019). A Deep Reinforcement Learning Framework for UAV Navigation in Indoor Environments. 2019 IEEE Aerospace Conference, 1–14. https://doi.org/10.1109/AERO.2019.8742226

19. Grando, R. B., Jesus, J. C. de, & Drews-Jr, P. L. J. (2020). Deep Reinforcement Learning for Mapless Navigation of Unmanned Aerial Vehicles. 2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE), 1–6. https://doi.org/10.1109/LARS/SBR/WRE51543.2020.9307015
20. GUO, T., JIANG, N., LI, B., ZHU, X., WANG, Y., & DU, W. (2021). UAV navigation in high dynamic environments: A deep reinforcement learning approach. Chinese Journal of Aeronautics, 34(2), 479–489. https://doi.org/10.1016/j.cja.2020.05.011
21. AlMahamid, F., & Grolinger, K. (2024). VizNav: A Modular Off-Policy Deep Reinforcement Learning Framework for Vision-Based Autonomous UAV Navigation in 3D Dynamic Environments. Drones, 8(5). https://doi.org/10.3390/drones8050173
22. Al-Jarrah, O. Y., Shatnawi, A. S., Shurman, M. M., Ramadan, O. A., & Muhaidat, S. (2024). Exploring Deep Learning-Based Visual Localization Techniques for UAVs in GPS-Denied Environments. IEEE Access, 12, 113049–113071. https://doi.org/10.1109/ACCESS.2024.3440064
23. Terven, J.; Córdova-Esparza, D.-M.; Romero-González, J.-A. A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. Mach. Learn. Knowl. Extr. 2023, 5, 1680-1716. https://doi.org/10.3390/make5040083
24. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv 2020, arXiv:2004.10934.
25. Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, Xinggang Wang. ByteTrack: Multi-Object Tracking by Associating Every Detection Box. https://doi.org/10.48550/arXiv.2110.06864 . https://www.ecva.net/papers/eccv_2022/papers_ECCV/papers/136820001.pdf
26. Nour, B.A.; Soufiene, B.; Joseph, H. Modeling and Sliding Mode Control of a Quadrotor Unmanned Aerial Vehicle. 3rd Int. Conf. Autom. Control. Eng. Comput. Sci. 2016, 3. https://www.semanticscholar.org/paper/Modeling-and-Sliding-Mode-Control-of-a-Quadrotor-Ammar-Bouall%C3%A8gue/b124eef6db9c21ee3d8678da94e1324e80d75f0e#citing-papers?utm_source=direct_link
27. Tony, C.T.; MacKunis, W. Robust Attitude Tracking Control of a Quadrotor Helicopter in the Presence of Uncertainty. In Proceedings of the 2012 IEEE 51st IEEE Conference on Decision and Control (CDC); 2012; pp. 937–942. DOI:10.1109/CDC.2012.6426266
28. Keribayeva, T., Ainakulov, Z., Yergaliyev, R., Kurmankulova, G., Fedorov, I., & Anayatova, R. (2022). Experience of Connecting Sensors to the Controller Based on the Arduino Board for Use on Multicopters. Journal of Theoretical and Applied Information Technology, 100(6), 1827–1835. http://www.jatit.org/volumes/Vol100No6/19Vol100No6.pdf&ved=2ahUKEwjTtfnG6oyQAxVxBhAIHfyPExoQFnoECBwQAQ&usg=AOvVaw3FjGe-TLsgFKHJGiMrk2rU
29. Zviedris, Martins & Romane, Aiga & Barzdins, Guntis & Cerans, Karlis. (2014). Ontology-Based Information System. 33-47. 10.1007/978-3-319-06826-8_3.
30. Shen, J., & Yang, H. (2024). Multi-Object Tracking Model Based on Detection Tracking Paradigm in Panoramic Scenes. Applied Sciences, 14(10), 4146. https://doi.org/10.3390/app14104146

