A research platform for vision-based UAV autonomy: Architecture and implementation

Yurii Lukash[1,*,†], Pylyp Prystavka[1,†]

[1]State University "Kyiv Aviation Institute", Liubomyra Huzara Ave., 1, Kyiv, 03058, Ukraine

Abstract

This paper presents a practical research platform that combines UAV hardware with flexible client-server software, designed for testing and development of visual autonomy algorithms. The system allows researchers to quickly integrate their own video analysis procedures with minimal effort, using a simple interface. The architecture supports multiple types of onboard computers (such as Raspberry Pi 4 and Jetson Nano), and uses MAVLink via MAVSDK to communicate with flight controllers. A web interface provides real-time video streaming, manual control, and dynamic configuration of processing parameters. The platform includes telemetry logging synchronized with video frames, and supports both manual and automatic control based on video analysis results. Initial test runs with several different video processing methods, including object tracking, YOLO-based detection, and SIFT-based position holding, have confirmed the usability and flexibility of the proposed system for real UAV experiments.

Keywords

UAV, visual autonomy, computer vision, MAVSDK, onboard processing, Raspberry Pi, object tracking,

1. Introduction

Modern unmanned aerial vehicles (UAVs) are increasingly used in a wide variety of areas - from agriculture and environmental monitoring to security, reconnaissance and rescue operations, delivery and logistics. In some applications, such as optical navigation and vision-based positioning, the use of onboard visual sensors is critical [1, 2]. And in today's conditions, the military sphere has also caused rapid development, which, in addition to the variety of possible applications, adds many different limitations and requirements, including bandwidth constraints, sensor availability, and computing tradeoffs in UAV deployments [3, 4]. These include the lack of any frequencies for communication, unusually long distances of use, problems or the absence of standard sensors, or the impossibility of their use. One of the key areas of UAV development is increasing the level of autonomy, in particular through the integration of computer vision and intelligent video processing algorithms in real time.

However, the implementation and testing of such algorithms is associated with a number of challenges. First, the algorithms must be adapted to limited computing resources, typical of compact on-board computers. Secondly, a convenient platform is needed that allows researchers to quickly change or compare different approaches to image processing and decision-making without disrupting the overall stability of the system. Thirdly, it is critically important to ensure the ability to transmit commands from algorithms to the drone controller with minimal delay.

This work presents a comprehensive solution that combines the UAV hardware platform with client-server software. The system allows researchers to explore, compare and validate new algorithms and video processing procedures, and perform automatic drone control based on image analysis and telemetry data from the device. Thanks to integration with MAVSDK and the use of logic separation into client and server parts, a stable infrastructure for experiments with autonomous control in real conditions is implemented. Additionally, the user is provided with a UI with real-time video streaming

CH&CMiGIN'25: Fourth International Conference on Cyber Hygiene & Conflict Management in Global Information Networks, June 20–22, 2025, Kyiv, Ukraine

*Corresponding author.

†These authors contributed equally.

y.lukash@gmail.com (Y. Lukash); chindakor37@gmail.com (P. Prystavka)

0009-0003-1824-8936 (Y. Lukash); 0000-0002-0360-2459 (P. Prystavka)

and a corresponding set of control elements for sending commands directly to flight controller, as well as tools for flexibly adjusting processing parameters or triggering specific processing stages as needed.

The purpose of this article is to describe the architecture of the developed system, demonstrate its capabilities in scenarios of autonomous control and object tracking, as well as justify the selected technical solutions. Special attention is paid to the practical aspects of implementing the interaction between the video processing modules and the UAV controller, which enables the use of this platform as a universal tool for researching intelligent control of aircraft.

2. Definition of visual autonomy in UAV systems

Unmanned Aerial Vehicle (UAV) autonomy typically refers to the capability of performing flight-related tasks—ranging from navigation to interaction with the environment without real-time operator control. In the context of visual autonomy, the drone relies primarily on onboard video data as the sensory input. That is, the UAV analyzes image sequences in real time to make decisions, opening up a wide range of research directions such as visual navigation, object tracking, detection, motion planning, and more.

To enable such autonomy, the system architecture generally includes:

- Sensing subsystem: camera, GPS, compass, IMU;
- Onboard computing unit: a single-board computer capable of processing video streams (e.g., Raspberry Pi or Jetson);
- Image analysis and decision-making module: software that implements detection, tracking, mapping, or motion vector generation;
- Flight control system: an autopilot that receives control commands (e.g., via MAVLink);
- Feedback and/or logging subsystem: for evaluation, synchronization, and debugging.

This structure provides a modular basis for experimental platforms where each component—from frame acquisition to command generation — can be studied and substituted independently, which aligns with contemporary modular approaches for implementing vision-based autonomy in UAV systems [5, 6].

In recent literature, such architectures are seen as promising foundations for UAV autonomy in environments with limited or unavailable GPS access [7, 8]. The significance of vision-based methods for tasks like landing [9], object interaction, and visual localization is also frequently highlighted [10]. Additionally, recent work has demonstrated the feasibility of integrating object detection, tracking, and obstacle avoidance in cost-constrained UAV platforms using AI-based visual pipelines [11].

3. Hardware design and description

To build a research platform that would allow for flexible configuration modification and various experiments, a number of basic requirements for the hardware layout were formulated.

The basis of the design was a quadcopter frame, which provides sufficient flight stability and ease of maintenance. An important design principle was the presence of backup communication channels - both the main and alternative connections to the drone were implemented to control the flight in the event of a failure of the main channel during experiments.

In order to be able to lift different types of single-board computers on board - such as OrangePi, Raspberry Pi, or NVIDIA Jetson Nano - and compare their computing capabilities, more powerful engines and an appropriate power system were used. The design provides for the possibility of connecting cameras of different types, which allows testing algorithms with different quality and video stream format. So far, launches have been carried out with 2 different CSI cameras, including those with infrared illumination and 3 different USB cameras.

The platform also includes a set of mandatory sensors for orientation in space and collection of telemetric data: GPS module, magnetometer (compass), IMU. This provides not only autonomous

Table 1
Comparison of Single-Board Computers for UAV Integration

Board	CPU	GPU / Accel.	RAM	Interfaces	Power	Price (USD)	Notes
Raspberry Pi 3B+	4× A53 @ 1.4GHz	Video Core IV	1 GB	GPIO, UART, CSI, USB	2.5–4W	$35–40	Minimal specs, good CSI support, low power use.
Raspberry Pi 4B	4× A72 @ 1.5GHz	Video Core VI	2–8 GB	GPIO, UART, CSI, USB 3	3–6W	$45–65	Well-supported, excellent CSI integration, broad community.
Raspberry Pi 5	4× A76 @ 2.4GHz	Video Core VII	4–8 GB	PCIe, UART, CSI, USB 3	7–10W	$80–95	Very powerful, needs active cooling, new interface standards.
Orange Pi 4	2× A72 + 4× A53	Mali-T860 MP4	4 GB	GPIO, UART, CSI, USB 3, PCIe	5–10W	$55–65	More compute power, modest support, PCIe availability.
Orange Pi 5	4× A76 + 4× A55	Mali-G610 + NPU	4–32 GB	GPIO, UART, MIPI, USB 3, PCIe	5–15W	$70–120	High performance, onboard NPU, complex integration.
Jetson Nano	4× A57	NVIDIA Maxwell GPU	4–8 GB	GPIO, UART, CSI, USB	5–10W	$100–130	CUDA-capable, suitable for CV/DL, higher cost.

navigation, but also allows you to integrate spatial information into the video analysis process or use it for data synchronization in further research.

The platform body is designed with sufficient space for placing power supplies, cooling systems and additional modules. This solution ensures long-term operation and flexibility in configuration at different stages of development.

One of the critical decisions in the design of the system was the selection of a single-board computer that would meet the requirements for performance, power consumption, and compatibility with other components. A number of alternatives available today were considered and the Raspberry Pi 4B was selected for the current stage of research, as it provides a balance between energy efficiency and computing capabilities, has CSI interfaces for the camera and GPIO for UART connection with the controller. This model also has a stable ecosystem, availability, and a sufficient number of modules of this series are available at the department. This choice aligns with recent developments in low-cost embedded vision systems for UAVs, which have demonstrated effective navigation and obstacle avoidance capabilities using platforms like Raspberry Pi and Pixhawk flight controllers [12].

Although experiments were conducted with the Raspberry Pi 5, which provides higher performance, it has significantly higher power consumption, which is critical for the duration of autonomous flight. In addition, newer models have new interfaces (PCIe, active cooling), which require more complex integration into the on-board system.

Benchmark comparisons indicate that while the Raspberry Pi 4's ARM Cortex-A72 CPU delivers strong single-thread performance, the Jetson Nano's 128-core Maxwell GPU excels in parallel processing tasks, making it more suitable for deep learning and computer vision applications [13]. However, GPU acceleration is not currently used and will be considered as a future enhancement. In terms of power consumption, the Raspberry Pi 4 operates between 3W and 7W, making it ideal for battery-powered applications, whereas the Jetson Nano consumes between 5W and 10W [14].

The platform architecture is designed in such a way that it is easy to replace the SBC. This allows you

Table 2
Comparison of Flight Controllers for UAV Applications

Controller	Processor	FMU Version	Firmware Support	Interfaces
Pixhawk 2.4.5	STM32F427(168 MHz)	FMUv2	PX4 (deprecated), ArduPilot (limited)	UART, microUSB, I2C, SPI
Pixhawk 4	STM32F765(216 MHz)	FMUv5	PX4, ArduPilot (fully supported)	Multiple UART, CAN, USB, I2C, SPI
Pixhawk 6C	STM32H743(480 MHz)	FMUv6C	PX4, ArduPilot (fully supported)	Multiple UART, CAN, USB-C, I2C, SPI
Matek H743	STM32H743(480 MHz)	Custom	ArduPilot only (well supported)	UART, CAN, USB, I2C, SPI

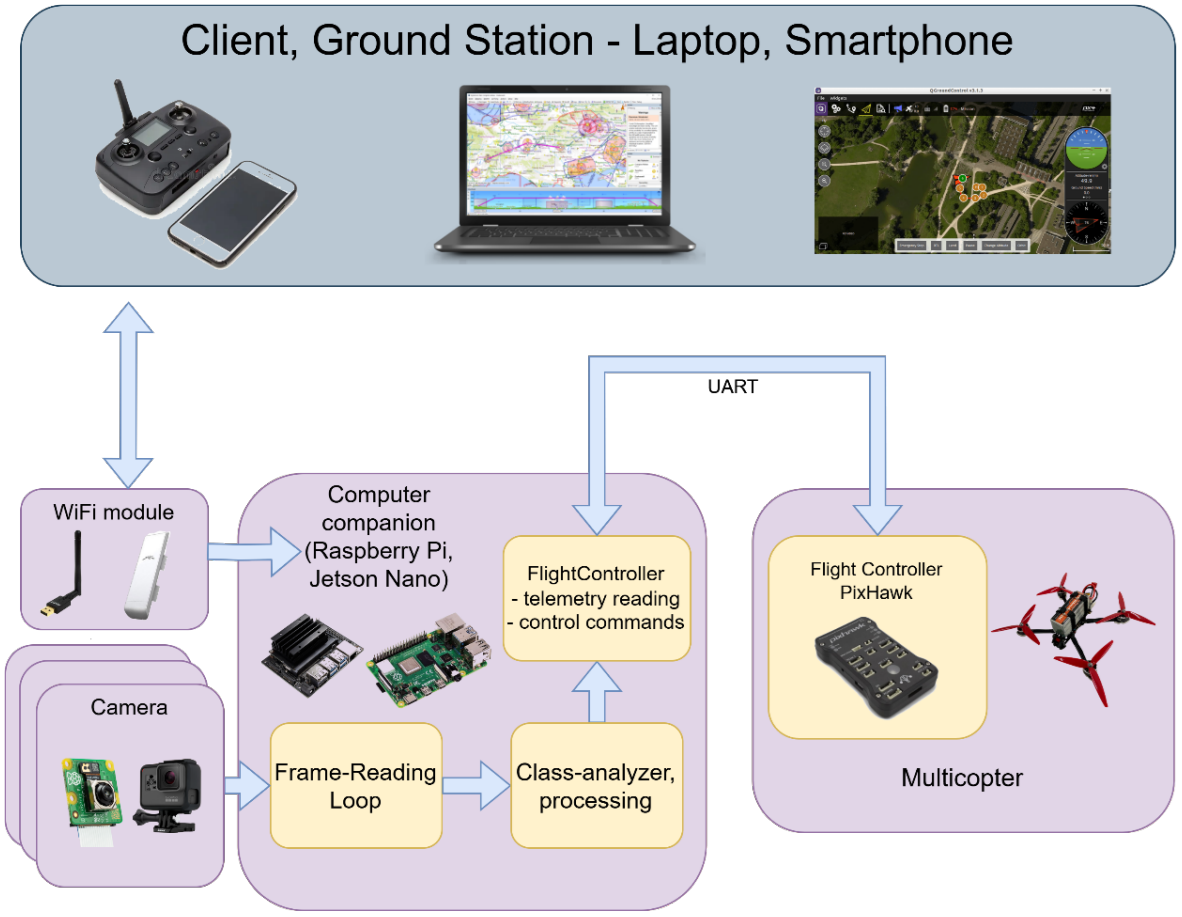


Figure 1: Research platform overview.

to adapt the system to different research scenarios, as well as test compatibility with more powerful modules, such as Jetson Nano, Orange Pi 5. It is also possible to add hardware computing accelerators, such as Google Coral Edge TPU or Intel Movidius Neural Compute Stick, although they have not yet been used in this project. And they will be tested in further research.

Another important decision was the choice of a flight controller. To build a research platform, a flight controller must meet several key criteria. In particular, it must support the MAVLink protocol for interaction with client software via MAVSDK, have access to UART for connecting the SBC, and provide a flexible configuration environment using open firmware (ArduPilot or PX4, and it is better to support both). Support for micro-USB or USB-C is also desirable, which allows you to vary the

connection options for the SBC, GroundStation, and power during testing. Also, to simplify the change of components - sensors and communication are connected via connectors, do not require additional soldering/desoldering. In our work, two controllers were used: Pixhawk 2.4.5 (selected due to its availability in the department) and later Pixhawk 6C (selected for the final implementation due to its modern characteristics). The selection of these controllers is also consistent with recent comparative studies on Pixhawk-based architectures and open-source flight control platforms commonly adopted in research UAV systems [15, 16]. Both support both PX4 and ArduPilot, which allowed us to test the MAVSDK client-server system in different modes. The final integration was done with PX4.

Schematically, the main elements of the platform are shown in Figure 1. The UAV directly contains a companion computer with a program that provides video processing, telemetry reading, formation and transmission of commands to the controller. Connection to the controller via UART. Cameras can be CSI, USB. Not simultaneously, but the program provides several video-capturing classes for this - so changing the camera can be done quickly. Connection to the operator - implemented via Wi-Fi for video transmission, control elements. Connection and control from a Laptop, Smartphone, Tablet are possible. At the same time, as backup channels - control of the remote control remains, as well as an additional communication channel from the Laptop directly with QGroundControl. Software implementation will be discussed in more detail in the next section.

4. Software architecture and implementation

4.1. Overview

The software architecture of the platform is designed as a modular and extensible system that provides interaction between the video stream, frame analysis algorithms and telemetry, the flight controller and the client interface. The main attention is paid to the possibility of quick integration of new video analysis methods, client independence from settings, as well as automatic collection and synchronization of telemetry data and their recording for post-analysis. The system consists of several independent components that interact through clearly defined interfaces.

For flexibility and ease of expansion and replacement of processing algorithms - Python was chosen as the programming language. In addition to various standard modules - it should be noted the use of the numpy, opencv, imutils, vidgear, picamera2, ultralytics, mavsdk packages to support various video camera variants, processing of the video stream and frames directly, the use of CV and ML algorithms.

4.2. Video frame acquisition and preprocessing

Video frame capture is implemented with support for both USB cameras and Raspberry Pi cameras (rpicamera2). The capture component is encapsulated in a frame queue module, which allows buffering the input stream for further processing or skipping frames if processing takes longer. A separate class is responsible for pre-processing (resizing, normalization, format conversion), which allows standardizing the input for different algorithms.

4.3. Video analyzer class

The analysis module is implemented as a separate class, where the researcher is required to implement at least one method:

```
def process_frame(frame):
    # Analyze frame and apply any logic researcher wants
    result = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    return result
```

This approach allows you to easily connect new computer vision algorithms - for example, object detection, segmentation, tracking, etc. The main program automatically connects the analysis class, transmits frames in real time, and if necessary - the result is used to form commands.

Figure 2: User Interface for video streaming and controls.

Additionally, optional methods are defined for this class, by implementing them, the re-searcher will be able to transmit ROI from the client, change the parameters of his processing, and can implement phasing - by reacting to pre-defined control elements.

4.4. Flight controller class

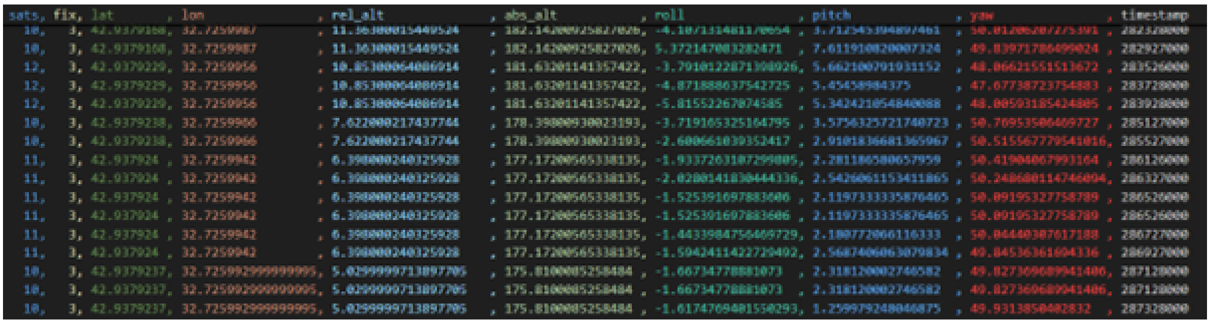
A separate class is responsible for working with the flight controller. It implements the connection to the controller, receives data from sensors and flight mode - about coordinates, orientation, speed and flight status from the controller. The FlightController class implements basic commands (takeoff, landing, movement in a given direction with a given speed and time, rotation), which are sent via the MAVLink protocol using MAVSDK. This integration follows the widely adopted MAVLink protocol, which has become the de facto standard in UAV communication, as highlighted in [17]. Both PX4 and ArduPilot are supported, which allows testing in different conditions. The connection to the controller is also flexibly parameterized - and can connect via GPIO, microUSB or set UDP for debugging with the simulator.

4.5. Integrated web interface for remote control

The program raises a built-in web server that broadcasts the video stream and provides an HTML interface with control buttons. Shown in Figure 2. The user can connect to the drone via Wi-Fi from any device (smartphone, tablet, laptop) without the need to install additional software. An interactive target selection function is available - clicking on the video determines the coordinates and parameterized size of the ROI, which can be used for tracking, landing or guidance or other types of use, as the researcher implements in his analyzer. There are direct control buttons - which send commands about Arm, Takeoff, Velocity etc directly to the drone. Additionally implemented buttons for parameterizing the analyzer algorithm - additional logic for their use can be used by the researcher in his processing class.

4.5.1. Logging, synchronization, and experimentation tools

During experiments, all telemetry data and processed frames are stored with timestamps, which allows for offline analysis, graphing, and evaluating algorithm behavior. Synchronization is provided through



sats	fix	lat	lon	rel_alt	abs_alt	roll	pitch	yaw	timestamp
10	3	42.9379108	32.7259987	11.36300015449524	182.14200925827026	-4.107131481170654	3.712545394897461	50.01206207275391	282328000
10	3	42.9379108	32.7259987	11.36300015449524	182.14200925827026	5.372147083282471	7.611910820007324	49.83971786499024	282927000
12	3	42.9379229	32.7259956	10.85300064086914	181.63201141357422	-3.7910122871398926	5.662100791931152	48.06621551513672	283526000
12	3	42.9379229	32.7259956	10.85300064086914	181.63201141357422	-4.871888637542725	5.45458984375	47.67738723754883	283728000
12	3	42.9379229	32.7259956	10.85300064086914	181.63201141357422	-5.81552267074585	5.342421054840088	48.00593185424805	283928000
10	3	42.9379238	32.7259966	7.622000217437744	178.39800930023193	-3.719165325164795	3.5756325721740723	50.70953506469727	285127000
10	3	42.9379238	32.7259966	7.622000217437744	178.39800930023193	-2.600661039352417	2.9101836681365967	50.515567779541016	285527000
11	3	42.937924	32.7259942	6.398000240325928	177.17200565338135	-1.9337263107299805	2.281186580657959	50.41904067993164	286126000
11	3	42.937924	32.7259942	6.398000240325928	177.17200565338135	-2.0280141830444336	2.5426061153411865	50.248680114746094	286327000
11	3	42.937924	32.7259942	6.398000240325928	177.17200565338135	-1.525391697883606	2.1197333335876465	50.09195327758789	286526000
11	3	42.937924	32.7259942	6.398000240325928	177.17200565338135	-1.525391697883606	2.1197333335876465	50.09195327758789	286526000
11	3	42.937924	32.7259942	6.398000240325928	177.17200565338135	-1.4433984756469729	2.180772066116333	50.04440307617188	286727000
11	3	42.937924	32.7259942	6.398000240325928	177.17200565338135	-1.5942411422729492	2.5687406063079834	49.84536361694336	286927000
10	3	42.9379237	32.725992999999995	5.029999713897705	175.8100085258484	-1.66734778881073	2.318120002746582	49.827369689941406	287128000
10	3	42.9379237	32.725992999999995	5.029999713897705	175.8100085258484	-1.66734778881073	2.318120002746582	49.827369689941406	287128000
10	3	42.9379237	32.725992999999995	5.029999713897705	175.8100085258484	-1.6174769401550293	1.259979248046875	49.9313850402832	287328000

Figure 3: Sample of database that produces platform after flight.

a single time base. The system stores both input frames and processed frames with annotations or texts, which allows for visual comparison of algorithm efficiency. This functionality is implemented separately, allowing the researcher to not worry about its implementation and focus only on his processing class. This is what the telemetry saved after the experimental flight looks like.

4.5.2. Autonomous command generation (optional extension)

To implement fully autonomous drone control, the researcher needs to implement another method in his class:

```
def generate_vector_command(self):
    #any compution for vector and time of move
    return vx, vt, t
```

This allows you to generate commands based on video analysis without the operator's participation, for example, to follow an object, fly to the center of the frame, etc. Thus, the system provides a full cycle: from image to control.

4.5.3. Example of data for post-analysis

After the experiment, the researcher receives a telemetry database as shown in Figure 3. It contains the telemetry available from the Flight Controller. Also, all these records are synchronized with video frames, which are also stored and available for post-processing. Figure 4,5 shows examples of visualizations that can be easily generated, for example, with the matplotlib package, or for any other analysis, since the data is stored in the common .csv format.

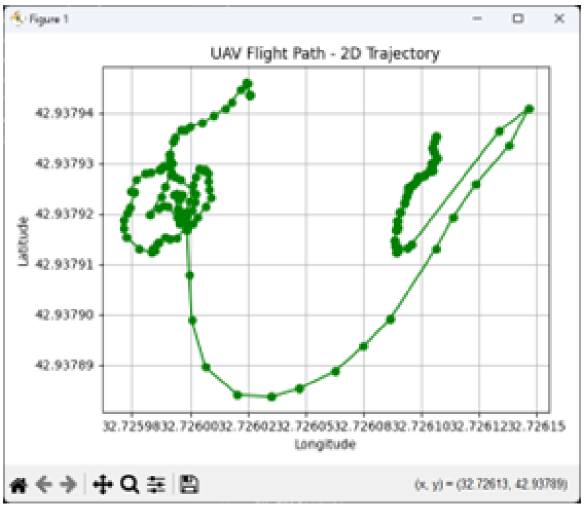


Figure 4: Flight visualization 2D

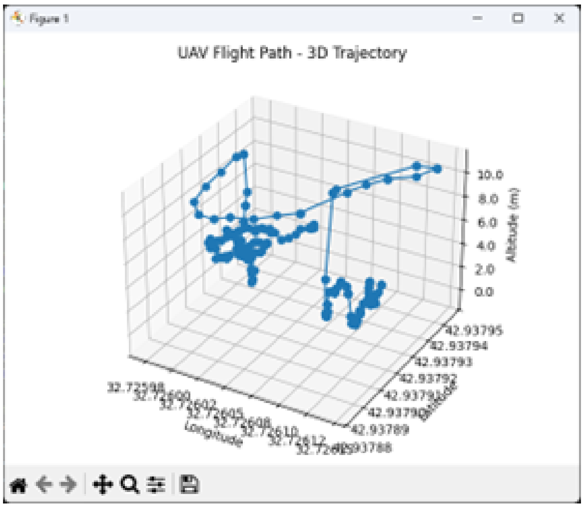


Figure 5: Flight visualization 3D

5. Preliminary experiments and use cases

For the initial assessment of the platform's functionality, several trial runs were carried out with the participation of several operator researchers, who implemented their algorithms exclusively in one class with the required 2-3 functions, as indicated in section 4.

From our side, we provided a unified class interface with clear documentation, specifying only the minimal required methods. The internal structure of the class, any helper functions, or additional private methods were left entirely at the discretion of each researcher.

5.1. Object tracking with CSRT algorithm

One of the initial experiments focused on the use of a classical object tracking algorithm — CSRT [18]. The operator selected a region of interest (ROI) on the video stream via the web interface, and the tracker maintained this object in the field of view. The platform automatically generated control commands to make the UAV follow the object by adjusting its position relative to the moving target.

This experiment demonstrated the capability of integrating standard OpenCV-based tracking algorithms into the platform with minimal code — only two methods in the class had to be implemented. One for frame processing and another one - for generating vectors to move. The real-time responsiveness of the tracking loop and smooth command execution confirmed that the system could support simple visual servoing scenarios.

5.2. Object detection with YOLO

Another set of experiments involved integrating the YOLO (You Only Look Once) family of neural networks, particularly using the Ultralytics implementation [19, 20]. The video analyzer class was extended to include inference using a pre-trained model, with detected objects and bounding boxes drawn in real time and optionally logged.

Although no autonomous commands were issued in this case, the experiment validated that the platform was capable of real-time deep learning inference on the onboard computer, and that visual output could be transmitted to the operator. This opens up potential use cases for semantic detection, object classification, and more complex logic based on scene understanding.

5.3. Position holding using visual features

Third experiment involved implementing a position holding function based purely on image features. At the beginning of the flight, the drone captured a reference ROI and extracted SIFT (Scale-Invariant Feature Transform) keypoints. During flight, each new frame was compared to this reference, and a homography was computed to estimate the relative displacement of the UAV.

Based on the displacement vector, a movement command was generated to compensate and return the drone to its original visual location. This closed-loop control based on SIFT matches demonstrated the platform's ability to support advanced visual localization techniques and generate dynamic control signals in real time.

5.4. Summary of experimental validation

These initial experiments demonstrated the flexibility of the proposed architecture and confirmed that it effectively supports various modes of video-based UAV control — from traditional object tracking to deep learning and visual localization. Although a full quantitative evaluation is reserved for future studies, the platform demonstrated practical feasibility, ease of algorithm integration, and stable operation in real-world UAV flights.

6. Conclusions

The paper presents the development of a hardware-software platform focused on research and experiments on video stream analysis and formation of movement commands for a drone based on such analysis. The main hardware components are described, and their choice is justified. The constructed system demonstrates flexibility at all levels — from the choice of single-board computers and flight controllers to software components and video processing algorithms. A convenient interface for integrating own algorithms has been implemented, which has a defined interface and allows researchers not to think about all program classes, but to focus on their one analyzer class. Mechanisms for logging telemetry and video frames, synchronization and autonomous execution of movement commands for UAVs formed by the analyzer class implemented separately. Preliminary tests have confirmed the functionality and ease of use of the platform in research conditions. The paper presents what set of controls is available to the user, what kind of data the researcher receives after the flight, and how typically this data can be visualized for analysis.

In the future, it is planned to expand the system by improving the class interfaces, testing in more complex scenarios. Preparing for flights other single-board computers and graphics accelerators, so that it is also possible to assess their impact on the speed of the procedures being studied. Conducting more experiments and studies in flight conditions of those procedures that have already been prepared and described in non-flight simulations to confirm or correct the results already obtained.

It is also considered to implement a similar platform with reuse of the software part based on another UAV. The presence and availability of such a functioning platform opens up new opportunities for experiments.

Beyond the specific hardware and implementation details, the broader contribution of this work lies in offering a reproducible and modular baseline for vision-based UAV autonomy experiments. The clear separation of roles between flight logic, video processing, and control interface ensures ease of adaptation for various research directions — from academic investigations of perception pipelines to applied prototyping of autonomous systems. By minimizing the barrier to entry, the platform encourages more researchers to experiment, evaluate, and deploy intelligent UAV behaviors in real-world scenarios.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] P. Prystavka, O. Cholyshkina, Estimation of the aircraft's position based on optical channel data, CEUR Workshop Proceedings 3925 (2024).

[2] N. Ruzhentsev, et al., Radio-heat contrasts of UAVs and their weather variability at 12 GHz, 20 GHz, 34 GHz, and 94 GHz frequencies, ECTI Transactions on Electrical Engineering, Electronics, and Communications 20 (2022) 163–173. doi:10.37936/ecti-eec.2022202.246878.

[3] S. Liu, X. Li, H. Lu, Y. He, Multi-object tracking meets moving uav, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 8866–8875. doi:10.1109/CVPR52688.2022.00867.

[4] S. A. H. Mohsan, N. Q. H. Othman, Y.-S. Chen, M. H. Alsharif, M. A. Khan, Unmanned aerial vehicles (uavs): practical aspects, applications, open challenges, security issues, and future trends, Intelligent Service Robotics 16 (2023) 109–137. doi:10.1007/s11370-022-00452-4.

[5] T. Hong, H. Liang, Q. Yang, L. Fang, M. Kadoch, M. Cheriet, A real-time tracking algorithm for multi-target uav based on deep learning, Remote Sensing 15 (2023) 2. doi:10.3390/rs15010002.

[6] O. Sushchenko, et al., Airborne sensor for measuring components of terrestrial magnetic field, in: 2022 IEEE International Conference on Electronics and Nanotechnology (ELNANO), 2022, pp. 687–691. doi:10.1109/ELNANO54667.2022.9926760.

[7] M. Y. Arafat, M. M. Alam, S. Moh, Vision-based navigation techniques for unmanned aerial vehicles: Review and challenges, Drones 7 (2023) 89. doi:10.3390/drones7020089.

[8] L. Xin, T. Zhi, G. Wen, L. Hua, Vision-based autonomous landing for the uav: A review, Aerospace 9 (2022) 634. doi:10.3390/aerospace9110634.

[9] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, M. Shah, Visual tracking: An experimental survey, IEEE Transactions on Pattern Analysis and Machine Intelligence 36 (2014) 1442–1468. doi:10.1109/TPAMI.2013.230.

[10] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, N. Roy, Visual odometry and mapping for autonomous flight using an rgb-d camera, in: Experimental Robotics, Springer, 2013, pp. 235–244. doi:10.1007/978-3-319-29363-9_14.

[11] V. Varatharasan, A. Rao, E. Toutounji, J.-H. Hong, H.-S. Shin, Target detection, tracking and avoidance system for low-cost uavs using ai-based approaches, in: Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS), 2019, pp. 142–147. doi:10.1109/REDUAS47371.2019.8999683.

[12] L. D. Ortega, M. A. Olivares-Mendez, P. Campoy, Low-cost computer-vision-based embedded systems for uavs, Robotics 12 (2023) 145. doi:10.3390/robotics12060145.

[13] P. M. Sánchez, J. M. J. Valero, A. H. Celdrán, G. Bovet, M. G. Pérez, G. M. Pérez, Lwhbench: A low-level hardware component benchmark and dataset for single board computers, Internet of Things 22 (2023) 100764. doi:10.1016/j.iot.2023.100764.

[14] I. Rakhmatulin, Increasing fps for single board computers and embedded computers in 2021 (jetson nano and yolov4-tiny), arXiv preprint (2021). doi:10.48550/arXiv.2107.12148. arXiv:arXiv:2107.12148.

[15] D. Singh, H. O. Verma, V. Kumar, R. K. Saluja, S. Singh, K. Radhakrishnan, A comparative study of pid and nonlinear backstepping flight control laws for pixhawk-based multicopter, in: Recent Advances in Aerospace Engineering, Lecture Notes in Mechanical Engineering, Springer, Singapore, 2024, pp. 253–267. doi:10.1007/978-981-97-1306-6_21.

[16] E. S. M. Ebeid, M. Skriver, K. H. Terkildsen, K. Jensen, U. P. Schultz, A survey of open-source uav flight controllers and flight simulators, Microprocessors and Microsystems 61 (2018) 11–20. doi:10.1016/j.micpro.2018.05.002.

[17] A. Koubaa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith, M. Khalgui, Micro air vehicle link (mavlink) in a nutshell: A survey, IEEE Access 7 (2019) 87658–87680. doi:10.1109/ACCESS.2019.2924410.

[18] I. Zhukov, Y. Lukash, Using an object tracking algorithm by video image for the implementation of an autonomous target tracking function for uav, Problems of Informatization and Management 2 (2024) 14–17. doi:10.18372/2073-4751.78.18956.

[19] Y. Zhang, G. Gao, Y. Chen, Z. Yang, Odd-yolov8: An algorithm for small object detection in uav imagery, The Journal of Supercomputing 81 (2025) article 202. doi:10.1007/s11227-024-06703-8.

[20] C.-H. Lin, K.-H. Chuang, J.-J. J. Lien, A comparative study of object detection algorithms on uavs: Yolov5, yolov6, and yolov8, Drones 7 (2023) 377. doi:10.3390/drones7060377.