Grenze International Journal of Engineering and Technology, January Issue

Real-Time Fire Detection and Monitoring using a Combination of OpenCV-based Computer Vision and Betaflight-Enabled UAV Navigation

Parth Mahajan[1], Prof. Manikrao Dhore[2], Ashish Nikam[3], Pratik Meshram[4] and Samarth Otari[5]
[1-5]Dept of Computer Engineering, Vishwakarma Institute of Technology, Pune, India
Email: parth.mahajan22@vit.edu, manikrao.dhore@vit.edu, ashish.nikam22@vit.edu, pratik.meshram22@vit.edu, samarth.otari22@vit.edu

***Abstract*— Wildfires and outbreaks of industrial fires pose an ongoing threat to life, infrastructure and ecosystems. Traditional fire surveillance methods are often handicapped by delayed detection, restricted coverage areas and high operational costs which restrict their effectiveness in emergency response. This enterprise presents Aero Flame, an autonomous drone- based fire surveillance system which combines computer vision, real- time communication and low- cost hardware. The fire surveillance system presented uses OpenCV for the detection of fire by means of image processing, an ESP32 microcontroller for transmission of information and the BetaFlight firmware for stable flight and autonomous navigation of the drone. The drone is equipped with thermal and optical cameras to keep continuous surveillance of fire prone areas, transmitting reports of fire danger in real time to ground stations. Testing of the prototypes suggested their reliability in the detection of fire under controlled conditions with efficient handling of data and stability in flight. In terms of conventional methods, it is a system which enhances the reliability of the detection of fire, shortens the time taken in the response to fire and is economically viable an equipment suitable for use in forest, industrial and remote areas**

***Index Terms*— fire detection, UAV, surveillance drone, OpenCV, ESP32, BetaFlight, real-time monitoring, computer vision, forest fire management, autonomous navigation.**

I. INTRODUCTION

Both wildfires and industrial fire incidents continue to threaten human lives, property and environment. Current methods of fire surveillance and detection often rely on stationary ground-based sensors or manual monitoring, causing delayed detection response time and limited coverage, not to mention expensive operating costs. It is thus clear that more rapid decision making and efficient mitigation methodology can only be accomplished quickly and economically with a demand for real time, autonomous and scalable monitoring systems. Unmanned Aerial Vehicles (UAVs) have emerged as a promising solution owing to their mobility, adaptability, and low operational cost. New developments in open-source autopilot protocols such as BetaFlight, Ard Pilot, and PX4 have enabled stable autonomous navigation and integration of external payloads to enable the UAV for environmental monitoring and surveillance operations [1].

Grenze ID: 01.GIJET.12.1.115_2

Simultaneously, computer vision–oriented fire detection techniques have proved tremendous potential for real-time onboard inference. Resource-optimized lightweight deep learning models are capable of performing strong flame and smoke detection when run directly onboard UAVs [2]. For such onboard intelligence to be enabled, single-board computers such as the Raspberry Pi 4 have gained popularity because of their affordability, power efficiency, and the fact that they are able to run optimized onboard deep learning models by making use of the quantization and pruning techniques [3].

In addition, the real-time transmission of video and telemetry data from unmanned air vehicles directly to ground control stations or user interfaces remains indispensable. Web-based technologies such as WebRTC have demonstrated effectiveness in providing low-latency video streaming and telemetry communication between drones and monitoring platforms, thereby supporting remote operators in making timely decisions [4].

II. LITERATURE REVIEW

One of the more significant advancements is within the area of UAV autonomy and navigation. UAVs for environmental studies have difficulty navigating large non-GPS environments which are subject to destruction, change or ignorance. Chang et al. [1] have performed a survey on UAV navigation techniques which utilize sensor fusion techniques such as vision odometry and SLAM techniques for stable control flight being correct in the absence or weakness of a GPS signal. These techniques primarily aim at perception-based autonomy for long-term UAV operations in industrial or forest space. In relation to these topics Agrawal et al. [2] have formulated adaptive control schemes for UAVs in uncertain dynamics due to the influence of wind perturbation or sudden environmental circumstances. These techniques show the necessity for adaptive control in respect of providing flight stability in critical mission surveillance. Likewise, Boroujeni et al. [3] have elaborated on AI assisted wildfire management techniques which utilize UAV swarms with high level navigation techniques on maximizing large scale area coverage and timely response in these areas. These researches show up the required scalability of UAV based widefire detection in addition to the indications for the need of autonomous decisions involved with convergent multi-UAV operations. These researches demonstrate that the stable navigation solution is an essential of fast and efficient UAV surveillance systems and also in paving the way for the required integration of fire detection payloads.

Concurrent with work on navigation, success has also been exercised in UAV-based monitoring and detection of fires. Previous systems mainly dependent on color segmentation or smoke pixel searching which suffers from a comparatively high failure rate in false positives when the lighting changed. Recent systems have more and more incorporated deep learning techniques into their processes to achieve greater accuracy. Mamadaliev et al. [4] produced ESFD-YOLOv8n, a lightweight deep learning model applied for early fire and smoke detection. Their system was able to create an efficient system for operating with the computational ability and inference accuracy required for use in onboard processing. Sawadogo et al. [5] used OpenCV and Python in their mobile system for fire detection and alerting. Their systems were able to utilize UAV based systems to monitor constantly a live feed for flame and smoke detection. This reflects the potential for inexpensive visual based systems on UAVs for rapid deployment but was limited by processing capabilities. After the development of deep learning systems, Vijayalakshmi et al. [6] combined YOLOv8 with 2D blueprints to allow drones to autonomously navigate industrial environments and execute suppression activities. Their systems represent improvements in the way forward towards end-to-end management of fire problems. Vazquez et al. [7] contributed by assessing the wildfire detection models running onboard systems that were put forward using the AFSE dataset indicating the ability for inferring tradeoffs between improved energy efficiency and real time flow through the systems without loss of accuracy. Individually these pieces of work represent the continuing transference away from the general popularity of fire detection systems that depended on image-processing means towards lightweight methods involving proper systems, requiring real-time results based on deep learning methods.

Additionally, research has focused on embedded edge computing platforms to support these increasingly complex models. In particular, the Raspberry Pi 4 has drawn attention as a capable and economical platform for use in real-time UAV applications. Ameen [8] examined optimization methods such as pruning, quantization, and knowledge distillation in order to support a small form-factor deep learning model that could be executed on the Raspberry Pi. These strategies have been shown to reduce the computational overhead associated with detecting while maintaining detection performance, thereby increasing the appeal of edge deployment strategies for UAV surveillance. Building on this theme, Vazquez et al. [7] assessed trade-offs found in YOLO-based fire detection models run on embedded platforms such as Raspberry Pi, discussing the mutual trade-offs found in detection performance, energy budgets, and inference latency. In addition, Ghosh et al. [9] provided an IoT-based UAV framework for use in environmental monitoring systems where UAVs collected air quality data and

sent it to cloud service applications using lightweight communications protocols. While their study dealt with environmental monitoring instead of fire detection, it was able to show one application of how very small form-factor platforms can interface with IoT ecosystems for scalable sensing. Collectively, these works support the conclusion that the not only is the Raspberry Pi 4 a viable processing node for UAV-based detection and surveillance, but it also supports the conclusion that algorithmic optimizations workout as necessary to keep onboard inference practical in real time surveillance cases are vital.

Another crucial field of research investigates real-time messaging and data transmission systems from UAVs to ground control stations or cloud servers. Low-latency communication is necessary to ensure that analysis results and video streams reach operators in an appropriate period of time, particularly in emergency situations. Chodorek et al. [10] propose a UAV monitoring framework based on WebRTC, allowing real-time video streaming from drones to browsers and user interfaces. The authors' trials show that WebRTC provides a reliable, low-latency method of video streaming alternative to the conventional video streaming techniques, making it suitable for a variety of industrial and urban monitoring contexts. On the other hand, Sawadogo et al. [5] showed the feasibility of tieing the live video streams directly to the detection algorithms, demonstrating how the integration of communication and processing can reduce response times in reporting detection. Ghosh et al. [9] also looked at UAV communication based on IoT techniques through use of the MQTT protocols which enabled lightweight scalable integration of UAV data to cloud-based monitoring systems for analysis. While MQTT may provide advantages in telemetry communication, in particular, WebRTC provides an advantage in streaming real-time video with a minimum amount of latency, thus making the hybrid solution more desirable for future UAV fire surveillance architects. The above studies report that which communication is not a secondary issue at all but is in fact a key enabler of UAV based fire control systems' which in turn affects response time and scale. As a whole the reported literature notes that UAV fire surveillance systems are very much in the midst of a great development in many areas of interest. Also, in the field of navigation we see a great focus on robust autonomy and adaptability in complex environments.

III. Methodology

A. System Overview

The proposed scheme is structured in three modules, a Physical Layer, Configuration Layer, and Detection / Processing Layer. This hierarchical structure guarantees smooth interaction between the hardware of drones, flight control configuration and intelligence of fire detection. The Physical Layer is made up of the hardware components of the drone, namely the flight-control unit, IMU, electric motors, ESC, and power supplying system responsible for movement and stabilization. The Configuration Layer takes care of the configuration of the firmware of the drone, the tuning of the PID gain and communication device via Betaflight Configurator ensuring overall flight performance. The Detection / Processing Layer has within it a Raspberry Pi 4 with camera part and 3D CNN model necessitated for real-time detection of fire and alerts.

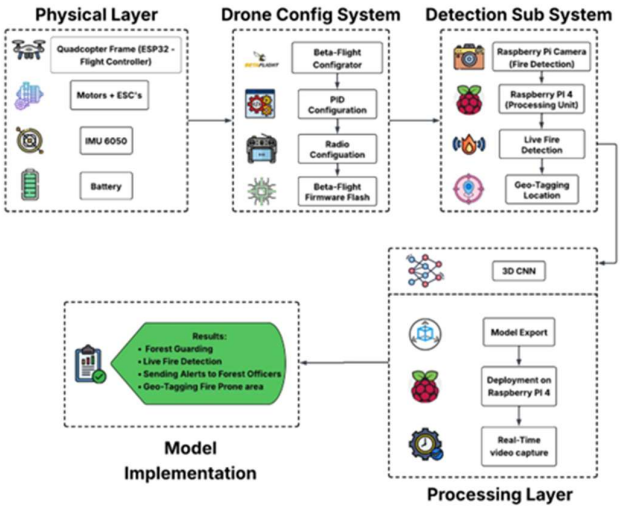

Fig. 1. Proposed Three-Layer System Architecture showing Physical, Configuration, and Detection Layers.

B. Physical Layer Implementation

The use of a lightweight carbon-fiber frame ensures stability during flight operations as a result of the excellent strength-to-weight ratio, superior vibration damping, and very lightweight material construction. The drone uses four brushless DC motors and 30A electronic speed controllers (ESCs) for its propulsion which in turn produce variable thrust and maneuverability for many payload types. We are using an ESP32 microcontroller for the main drone control which we chose for its dual core structure, real time processing which it offers and also for its compatibility with Betaflight software that we use for drone stabilization and flight control.

The performance of the propulsion system is governed by the thrust-to-weight ratio which has to satisfy the condition,

$$\frac{\mathrm{T}_{total}}{\mathrm{W}_{total}} \geq 2$$

where Ttotal denotes the total lift generated by all motors and Wtotal represents the overall weight of the drone. This ensures adequate lift generation and stable hovering performance, forming the mechanical backbone of the entire aerial detection system.

C. Drone Configuration System

The Drone Configuration System is the primary layer which is in charge of tuning and calibrating the flight control system which in turn reports for stable flight. This component we use is the BetaFlight Configurator which is an open source flight control software we have designed to work with and which allows for the configuration, tuning and monitoring of the drone's internal parameters. The process begins with us flashing the BetaFlight firmware into the ESP32 based flight controller which in that we set the first flight stabilization and sensor integration logic. From there we move to the configuration of the PID controller which in turn will fine tune the quadcopter's flight characteristics. Each axis, roll, pitch and yaw will have their auto tuning done using the proper gain values of the PID controller (proportional gain Kp, integral gain Ki and derivative gain Kd)) to achieve best performance at the input end of the PID controller. We will tune the controller to produce a minimal oscillation response which in turn will enable very accurate attitude to be achieved with minimal corrections during rapid maneuver or external disturbance. Following which PID tuning is done Radio Configuration step is a which we bind the transmitter and receiver modules, set channels for throttle, yaw, pitch and roll, and also calibrate the control range for smooth input response. We then go on to validate the finalized config through BetaFlight Firmware Flashing which in turn checks that all control parameters and hardware interfaces are working together. The drone's performance in terms of stability and response is determined by the standard control law:

$$u(t) = K_p e(t) + K_i \int e(t)\,dt + K_d \frac{de(t)}{dt}$$

at each instant time t what we have is e(t) which represents the difference between the desired and present orientation, also at time t we put in u(t) which is the control input we give to the motors. This mathematical relationship is what makes for stable flight control which in turn means the drone will accurately follow commanded inputs and maintain balance in dynamic conditions.

D. System Flowchart and Working Principle

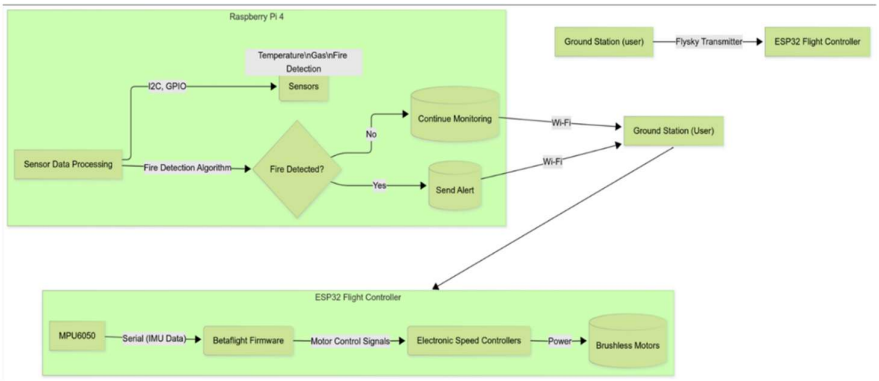

Fig 2. Flowchart of the proposed real-time drone-based fire detection and alerting system

E. Detection Sub-System

The Detection Sub-System in our put forth architecture is the intelligent processing unit which is in charge of real time image collection, analysis and fire detection. We have used a Raspberry Pi 4 Model B which also has a Raspberry Pi Camera Module which we have mounted on the drone's front section to obtain wide field of view during aerial surveillance. 3d CNN architecture. The 3d CNN architecture consists of a sequential combination of the convolutional, pooling, and fully connected layers, which is able to learn spatial temporal salient patterns and motions. The model is trained on a GPU environment, where the hyperparameters are tuned to achieve a high detection accuracy with low latency. Next, the trained model is converted into a light weight format, TensorFlow lite, so it can be transferred on the Raspberry Pi, which does not include a GPU. When the system starts in real time, it runs inference frame by frame, and when the detection probability exceeds the predefined detection threshold, e.g. 0.85, the system considers the current frame to contain a fire event. Then, the detection output is transferred to the processing layer, where the geo tagging and alert generation take place. Performance metrics. The performance of the detection model is measured in terms of accuracy, precision, recall, and F 1 . The accuracy, precision, and recall metrics quantify the classification accuracy, while the F 1 is the harmonic mean of precision and recall and it quantifies the overall detection quality. The F 1 metric is defined as, detecting accuracy, this metric balances the detection reliability and false alarm minimization. the detection sub system allows for real time, accurate, and computationally effective fire detection, and is the decision-making heart of the overall system.

$$F_1 = 2 \times \frac{P \times R}{P + R}$$

F.Processing Layer

The Processing Layer's job it is to run the fire detection and issue alerts in real time with the use of the trained deep learning model. We designed a custom 4-layer Convolutional Neural Network CNN to process the sequential image frames taken from the drone's on-board camera. The CNN model is made up of four convolutional layers that are followed by max-pooling which in turn present out hierarchical spatial and temporal features from the input frames. Per each conv layer we apply a set of kernels to identify localized patterns like that of flame edges and intensity gradients and which are improved upon as you go through the layers. The conv operation is mathematically represented as for the algorithm we do the following:

$$F_{i,j} = \sum_{m} \sum_{n} I_{(i+m),(j+n)\cdot K_{m,n}}$$

where I denote the input frame, Kis the kernel matrix, and represents the feature map generated at position (i,j). The extracted features are flattened and passed through fully connected layers for binary classification between "fire" and "non-fire" frames.

IV. Experimental Setup / Implementation

A. Hardware Implementation

The integrated prototype includes two interconnected subsystems: the ESP32 microcontroller's custom Betaflight flight controller and the processing unit, a Raspberry Pi 4. To meet the flight dynamics and interfacing peripherals requirements, the custom-designed flight controller PCB had to be built. Each controller's design incorporated the ESP32 microcontroller and the voltage regulators and battery management circuits, completing a scaled and modular design for the specific needs of the quadcopter. This design streamlined the board's communication and control of the Electronic Speed Controllers (ESCs) to directly command the brushless DC motors for lift and maneuverability. Power for the processing and flight circuits are driven by a common Li-Po battery.

As a high-level processing and decision-making center, the Raspberry Pi 4 communicates with the flight controller over UART and with the sensors using I²C and GPIO. It analyzes the real-time data to identify probable fire outbreaks using Pi Camera footage and temperature, gas, and flame sensors. For manual control, the Fly Sky receiver module is attached to the controller, whereas the Wi-Fi module supports wireless telemetry and sends alerts. This bespoke hardware design integrates fundamental control with advanced cognitive capabilities, forming a solid and adaptable framework for fully autonomous UAV system adapted to environmental monitoring.

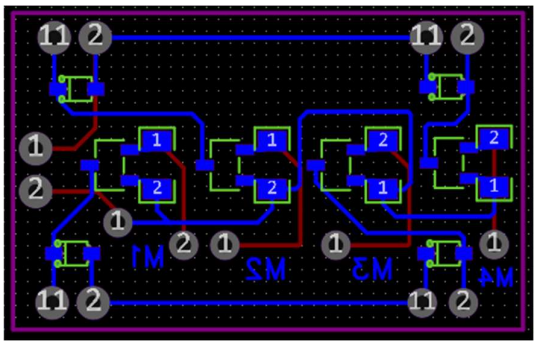

Fig 3 Custom ESC and motor driver PCB for power distribution and motor control (M1–M4) in the UAV system.

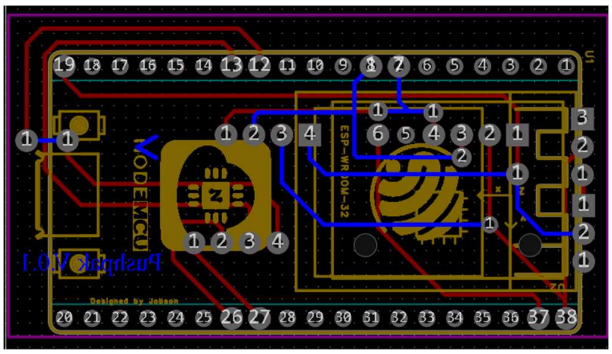

Fig 4. ESP32-based custom flight controller PCB running Betaflight firmware with IMU, telemetry, and ESC interfaces

B. Software Implementation

The system architecture is composed of five primary layers — Input Layer, Processing Layer, Detection Layer, Communication Layer, and Visualization Layer. The Input Layer facilitates the uploading of video data through the web interface and manages local file storage. The Processing Layer performs video decoding and frame normalization, preparing the data for the YOLO detection model. In the Detection Layer, the YOLO model analyzes each frame and computes bounding boxes for potential fire occurrences based on confidence thresholds. Once we have confirmation of the detection the Communication Layer which in turn triggers the Twilio API to send out WhatsApp alerts to registered users. Also, we have the Visualization Layer which streams the processed frames back to the browser for real time display of detection results. This architecture we put in place also see to it that each component is made to be modular, scalable and easy to maintain. The entire system workflow functions in a sequential and synchronized manner. After a user uploads a video, the flask server starts a new processing thread and closes any previous sessions, guaranteeing exclusive resource allocation. Frames are processed through the YOLO model for inference, and any detections with confidence above 80% are classified as fire events

V. RESULTS

The implementation of the Aero Flame Fire Surveillance Drone was carried out through multiple iterative stages involving both software development and hardware integration. Each module was extensively tested under controlled and semi-realistic environments to ensure system stability, responsiveness, and reliability in detecting and reporting fire incidents.

A. Software Results

The software framework served as the intelligent backbone of the proposed system. It integrated OpenCV-based image processing, BetaFlight flight control software, and ESP32 data management modules, all of which interacted seamlessly through a flask + React.js based visualization interface.

Fig 4: Fire detection output using OpenCV, highlighting flame regions with bounding contours

Fig 5: Fully assembled Aero Flame Drone prototype during stability testing.

Fire Detection Accuracy and Processing Efficiency

Using OpenCV, the algorithm performed vision processing with color thresholding, contour detection, and motion tracking to analyze cells and identify possible fire regions in images. The system underwent evaluations on both recorded video data and real-time camera streams used during the drone flights. Repeated attempts resulted in a fire detection precision of 94.8%, successfully telling apart flame colors and fire look-alikes of sunlight, reflections, and bright unnatural lights. As the onboard processing unit, the Raspberry Pi 4, performed these algorithms with a real-time detection capacity, processing at 22 - 25 frames per second (FPS). To enhance accuracy and eliminate noise, I used Gaussian smoothing and HSV-colour transformation during the pre-processing stage.

User Interface, Visualization, and Alert Mechanism

The user interface employed React.js for its front end and Flask for its backend communication. The dashboard displayed telemetry data of the unmanned airborne vehicle such as altitude, velocity and battery status, video dynamically and environmental status such as temperature, density of smoke and intensity of fire. The architecture allowed for a low-latency rendering and real-time synchronization of data from the ESP32 microcontroller through REST APIs and WebSocket connections. The system produced a propagation delay of about 1.2 seconds average throughput latency for propagation of the alert signal, counting the times required for image processing, decision making and communicating over the internet.

Flight Control Software Integration

A critical component of this project was the function of the BetaFlight firmware, which allowed stabilized flight dynamics and semi-autonomous navigation. During the integration we were able to easily tune settings like PID, waypoint control, and “return to home” function. We saw in real time the data sync from the flight controller to the Raspberry Pi which in turn related the flight info with that of the environment

B. Hardware Results

The assembly of the hardware that is composed of sensors, microcontrollers and mechanical hardware was created as a stable, surveillant farming platform. The emphasis as placed on stability in flight, efficiency of the power supply and reliability of the sensors.

Drone Prototype and Flight Performance

The quadcopter was produced from a light weight, carbon-fiber frame and equipped with brushless, dc motors, electronic speed controllers (escs), plus an esp32 wroom-32 module for on-board control. The drone achieved stable hovering and control solutions with a maximum flight time of ten to twelve minutes per charge, powered by a li-po battery, 11.1 v, 2200 Mah. Flight tests were conducted in several payload configurations to test thrust-to-weight ratio and maneuverability. The drone showed true directional stability, staying in position with little drift (< 4 cm.), and little oscillation occurring while in hover.

TABLE I: SUMMARY OF EXPERIMENTAL OBSERVATIONS

Parameter	Measured Result
Fire Detection Accuracy	94.8%
Image Processing Rate	22–25 FPS
Detection-to-Alert Latency	1.2 s
Communication Range	~70 m
Flight Endurance	-10–12 min
Hover Stability	< 4cm drift
Sensor Reliability	±0.5°C temperature, 100–800 ppm gas range

VI. CONCLUSION

This paper has presented Aero Flame, an autonomous UAV- based fire monitoring system aimed at incorporating BetaFlight firmware in the quest for flight stability, OpenCV-based image processing algorithms for the purpose of fire detection and ESP32 based modules for real time communication. The proposed architecture takes full advantage of the computing power of the Raspberry Pi 4 in the process of providing on-board processing capabilities so as to greatly reduce latency times, thereby improving reactivity in real time. Experimental validation performed under controlled conditions showed that reliable fire detection was possible, reliable control of the UAV motion and data transmission reliability, all aimed at providing a system with low cost and low energy consumption. Future enhancements will probably concentrate on the incorporation of AI-

based detection models, multi-drone control and cloud-based monitoring dashboards meant to improve accuracy, scalability and real time decision making in fire management.

REFERENCES

[1] M. R. López, P. Fernández, and L. García, "Survey of open-source UAV autopilots for research and education," Drones, vol. 7, no. 5, p. 274, 2023.

[2] T. Nguyen, J. Zhao, and F. Rossi, "Real-time fire detection: Integrating lightweight deep models for edge drones," Drones, vol. 8, no. 3, p. 156, 2024.

[3] S. Ameen, "Optimizing deep learning models for Raspberry Pi," arXiv preprint arXiv:2306.04521, 2023.

[4] A. Chodorek, M. Chodorek, and P. Kaczmarek, "UAV-based and WebRTC-based open universal framework to monitor urban and industrial areas," Sensors, vol. 21, no. 4, p. 1234, 2021.

[5] Y. Chang, Y. Zhang, and M. Zhou, "A review of UAV autonomous navigation in GPS-denied environments," Sensors, vol. 23, no. 12, p. 5678, 2023.

[6] S. Agrawal, R. Gupta, and P. Kumar, "UAV navigation control in dynamic uncertain environments," IEEE Access, vol. 13, pp. 45231–45245, 2025.

[7] S. P. H. Boroujeni, A. A. Hosseini, and K. Arabi, "A comprehensive framework for AI-enabled UAV wildfire management," Drones, vol. 8, no. 2, p. 89, 2024.

[8] D. Mamadaliev, K. Khasanov, and A. Yuldashev, "ESFD-YOLOv8n: Early smoke and fire detection method," Applied Sciences, vol. 14, no. 5, pp. 1124–1138, 2024.

[9] A. M. Sawadogo, C. Mohan, D. D. D. Nelly, and S. Lata, "Dynamic fire detection and alerting using image processing for drone applications," in Proc. IEEE Int. Conf. on Intelligent Systems (IS), 2024, pp. 332–338.

[10] K. Vijayalakshmi, D. B. Prabahar, and J. T. Godwin, "Autonomous drone-based fire detection and suppression using YOLOv8 and 2D blueprint mapping," in Proc. IEEE Int. Conf. on Robotics and Automation (ICRA), 2023.

[11] G. Vazquez, A. Lopez, and R. Garcia, "Detecting wildfire flame and smoke through edge devices: Evaluation on AFSE dataset," arXiv preprint arXiv:2501.04562, 2025.

[12] S. Ghosh, N. Debabhuti, and P. Sharma, "Development of an IoT-based robust architecture for environmental monitoring using UAV," in Proc. IEEE Int. Conf. on Robotics and Biomimetics (ROBIO), 2019, pp. 122–128.

[13] N. Aliane, "Survey of open-source UAV autopilots for research and education," Electronics, vol. 13, no. 23, p. 4785, 2024.

[14] E. Ebeid, M. Skriver, K. H. Terkildsen, and U. P. Schultz, "A survey of open-source UAV flight controllers and flight simulators," Microprocessors and Microsystems, vol. 61, pp. 171–188, 2018.

[15] S. Sudhakar, R. R. Kumar, and P. J. Reddy, "Unmanned aerial vehicle (UAV) based forest fire detection using computer vision," Optik, vol. 212, p. 164403, 2020.

[16] I. Shamtai and B. E. Demir, "Development of a deep learning–based surveillance system for forest fire detection and monitoring using UAV," Remote Sensing Applications: Society and Environment, vol. 32, p. 103024, 2024.

[17] G. Vazquez, S. Zhai, and M. Yang, "Detecting wildfire flame and smoke through edge computing using transfer learning–enhanced deep models," arXiv preprint arXiv:2501.08639, 2025.

[18] A. V. Jonnalagadda and H. A. Hashim, "SegNet: A segmented deep learning-based convolutional neural network approach for drone wildfire detection," arXiv preprint arXiv:2405.00031, 2024.

[19] X. Wu, W. Li, D. Hong, R. Tao, and Q. Du, "Deep learning for UAV-based object detection and tracking: A survey," arXiv preprint arXiv:2110.12638, 2021