Gesture-Based UAV Navigation Using Real-Time Vision and ArduPilot Simulation

Riya Bhalavat*, Rishi Kaushik †, Diya Uday Nayak‡
*†‡Department of Computer Science and Business Systems,
Mukesh Patel School of Technology, Management and Engineering, NMIMS University, Mumbai, India
Email: riyabhalavat01@gmail.com, rgravity15@gmail.com, diyan6151@gmail.com

***Abstract*—This paper presents a gesture-controlled drone navigation system that leverages real-time vision-based fist detection to guide UAV movement. Utilizing MediaPipe's hand tracking solution integrated with a live Iriun webcam feed, the system accurately detects the presence and position of a closed fist within the camera frame. Based on the detected location—left, center, or right—the drone responds with relative heading adjustments: turning right, left, or continuing forward accordingly. This design enables intuitive, contactless control without the need for external transmitters or handheld remotes. The drone operates in guided mode using DroneKit, maintaining continuous flight for at least one minute while dynamically responding to gesture inputs. The system avoids erratic behavior by preserving heading direction in the absence of a gesture, resulting in stable and predictable navigation. The proposed setup offers a low-cost, flexible framework for hands-free drone operation with potential applications in constrained or hands-busy environments such as disaster zones, warehouses, or inspection sites.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Unmanned Aerial Vehicles (UAVs), more commonly referred to as drones, have become increasingly versatile across various applications, ranging from aerial photography to emergency response and smart surveillance. The ability to control UAVs using intuitive human gestures can significantly enhance their accessibility and usability, especially in scenarios where conventional remote controls may not be practical or where hands-free navigation is essential. Recent advancements in real-time computer vision and embedded systems have paved the way for gesture-based drone control systems that are lightweight, affordable, and reliable.

This project explores a vision-based drone navigation system that utilizes real-time hand gesture recognition, specifically focusing on the detection of a closed fist using a live webcam feed. A fist gesture is selected due to its simplicity and reduced likelihood of false positives compared to open-hand gestures or dynamic motion cues. The system integrates Google's MediaPipe framework for robust hand landmark detection and gesture classification, combined with DroneKit Python API for commanding flight operations.

The drone is programmed to interpret the spatial position of the detected fist within the camera frame and accordingly adjust its trajectory: a fist detected in the center or right side of the frame triggers a left turn, whereas a fist on the left side initiates a right turn. Crucially, the system implements relative heading changes rather than absolute GPS-based jumps, ensuring smooth and intuitive maneuverability. In the absence of a detected gesture, the drone continues to move forward in the direction of its current heading.

The flight control logic was carefully designed to avoid unnecessary oscillations or direction reversals, enabling stable and continuous motion for a minimum duration of one minute. The integration with Iriun webcam allows for flexible deployment using a mobile phone as the video capture device, further reducing the hardware requirements and cost. Overall, this work presents a lightweight and efficient proof-of-concept for intuitive gesture-based UAV navigation, with potential applications in search-and-rescue operations, warehouse automation, and hands-free drone piloting.

II. Software Stack and Dependencies

The implementation of the gesture-based drone control system required a coordinated setup of multiple software tools and libraries, chosen specifically for their compatibility, performance, and ease of integration. The following subsections detail the key components and justify their selection.

A. Programming Language: Python

Python was selected as the primary programming language due to its extensive support for computer vision, robotics, and real-time communication with UAVs. Its simple syntax, combined with robust libraries, made it an ideal choice for rapid prototyping and modular development.

B. Drone Control: DroneKit-Python

DroneKit-Python is an open-source Python library designed to communicate with drones running ArduPilot firmware via MAVLink protocol. It enables direct scripting of flight commands such as arming, takeoff, landing, and waypoint navigation. This library was used to issue real-time flight control commands based on gesture recognition outputs. Its compatibility with simulated environments such as SITL (Software-In-The-Loop) also facilitated early testing.

C. Gesture Recognition: MediaPipe

MediaPipe, developed by Google, was employed for real-time hand and fist detection. Specifically, the Hand Tracking module was utilized, offering lightweight and accurate detection of hand landmarks without the need for GPU acceleration. MediaPipe was chosen for its proven efficiency, cross-platform support, and ease of integration with OpenCV-based video streams.

D. Webcam Integration: Iriun Webcam

To simulate a real-world mobile camera feed, Iriun Webcam was used to stream live video from a smartphone to the host system. This allowed greater flexibility in positioning the camera and closely mirrored practical use cases. Iriun's compatibility with standard virtual camera interfaces enabled seamless access via OpenCV without custom drivers.

E. Computer Vision: OpenCV

OpenCV (Open Source Computer Vision Library) was used to handle image frame acquisition, resizing, and preprocessing tasks. It served as the bridge between the virtual camera input and the MediaPipe pipeline. OpenCV also enabled optional visualization and debugging features during development.

F. Development Environment

The development was performed on a Linux-based system running Python 3.10. Required packages were installed and managed using pip. Key dependencies included:

- `dronekit==2.9.2` – for drone control
- `opencv-python==4.8.0.76` – for image and video processing
- `mediapipe==0.10.0` – for hand and fist gesture recognition
- `pymavlink==2.4.30` – for MAVLink communication

G. Hardware Abstraction

The solution was tested using a simulated ArduCopter environment, with the potential for deployment on real UAVs. The software stack remains unchanged between simulation and live deployment, ensuring portability and scalability.

This combination of tools offered a robust, modular, and lightweight approach to developing an interactive drone system guided purely by human gestures, with minimal hardware requirements.

III. Methodology

The objective of this work is to enable intuitive drone navigation using hand gestures—specifically fist detection—to influence flight decisions in real time. The proposed methodology consists of sequential modules designed to capture, analyze, and respond to gesture-based input using computer vision and drone control APIs.

A. Video Frame Acquisition

The system initiates by capturing real-time video frames using a mobile device configured as a webcam through Iriun. This simulates practical deployment scenarios where a user utilizes a mobile camera for interaction, ensuring accessibility and portability.

B. Frame Preprocessing

Captured frames are resized and formatted to an appropriate color space for efficient processing. This preprocessing step, performed using OpenCV, optimizes detection speed and accuracy by reducing frame resolution and enhancing contrast for the vision model.

C. Fist Detection and Localization

MediaPipe's hand tracking model is employed to detect hand landmarks. A custom rule-based classifier is used to identify whether the detected hand forms a closed fist. Only confirmed fists are processed further to reduce the risk of misclassification due to other hand movements. The position of the fist within the frame is segmented into three zones—left, center, and right—to enable directional decision-making.

D. Flight Decision and Navigation

Based on the classified position of the fist:

- If the fist appears in the left one-third of the frame, the drone executes a relative right turn.
- If the fist appears in the center or right side of the frame, the drone executes a relative left turn.
- If no fist is detected, the drone continues moving in its current trajectory.

This logic enables dynamic decision-making based on spatial cues, allowing the drone to navigate autonomously without relying on fixed waypoints. The DroneKit library is used to update the vehicle's heading and send real-time commands via MAVLink.

E. Real-Time Loop and Timing Constraint

The control logic operates within a continuous loop bound by a 60-second time constraint. During this period, the drone repeatedly interprets hand gestures, updates its heading direction accordingly, and moves forward with a consistent speed. This window ensures stable, repeatable experimental results for analysis and testing.

F. Safe Termination and Landing

Upon completion of the predefined navigation interval, the system commands the drone to initiate a safe landing procedure. This ensures proper shutdown and prevents uncontrolled behavior post-mission. It also marks the end of a session, making the system ready for the next gesture-controlled operation

IV. Observations

The implemented system and simulation yielded the following key observations:

- **System Initialization:** The ArduPilot SITL environment initialized successfully with all subsystems including GPS, AHRS, EKF3, and pre-arm safety checks confirming proper functionality.
- **Gesture Recognition:** The MediaPipe-based hand gesture detection reliably recognized a closed fist in real time

and identified its position (left, center, right) for drone directional control.
- **Autonomous Flight Behavior:** The drone responded accurately to gesture-based commands while maintaining stable flight at a predefined altitude and speed.
- **Flight Mode Stability:** The drone retained safe operation in GUIDED mode, with MAVProxy confirming positional feedback and GPS health throughout the simulation.
- **Environmental Sensitivity:** The system performance was stable under controlled lighting, though it is susceptible to changes in illumination and background.

V. Learning

Several insights were gained during the development of the system:

- **Gesture-Based Navigation is Feasible:** Real-time hand tracking using a single webcam input can reliably guide drone movement using intuitive gestures.
- **Modular Open-Source Integration:** Combining open-source tools such as ArduPilot, MAVProxy, and MediaPipe enables rapid prototyping and development of intelligent UAV systems.
- **Simulated Testing Accelerates Development:** The SITL environment allowed for safe and repeatable testing of complex flight logic before real-world deployment.
- **Real-Time Constraints:** Gesture recognition systems must be optimized for latency and stability to be effective in real-time drone applications.

VI. Future Scope

The project opens avenues for further development:

- **Multi-Gesture Control:** Expanding the gesture set to include commands such as hover, ascend, or return-to-home can enhance interaction.
- **Hardware Deployment:** Transferring the system to edge devices like Jetson Nano can make it field-deployable without tethering to a PC.
- **Obstacle Detection:** Integration with stereo cameras or LiDAR can allow dynamic obstacle avoidance and terrain-following capabilities.
- **Swarm Control:** Future systems could interpret gestures to manage coordinated movements among multiple UAVs.
- **AI-Based Adaptation:** Logging flight and gesture data to train adaptive models could result in predictive and autonomous learning systems.

VII. Applications

The system has potential use in various domains:

- **Search and Rescue:** Hands-free control allows responders to direct drones while navigating hazardous environments.
- **Military Missions:** Silent gesture-based navigation is valuable in stealth operations without electronic controllers.
- **Assistive Technologies:** Gesture interfaces provide accessibility to users with limited motor skills.
- **Educational Tools:** The system serves as an effective tool for learning robotics, AI, and embedded systems.
- **Agriculture and Warehousing:** Workers can direct UAVs without pausing tasks, enhancing productivity in large-scale environments.

VIII. Limitations

Despite its success, the system has some constraints:

- **Limited Gesture Set:** Only a closed fist is recognized, restricting the range of possible commands.
- **Environmental Sensitivity:** Varying lighting conditions can impact gesture detection accuracy.
- **Simulation-Based Validation:** Real-world flight tests are necessary to verify robustness against GPS errors, wind, and hardware constraints.
- **Short Detection Range:** The webcam restricts effective gesture recognition to a few feet in front of the drone.
- **No Obstacle Avoidance:** The system currently lacks real-time detection of physical barriers, which may limit safe deployment.
- **Fixed Altitude Navigation:** The drone maintains a constant height, not adapting to terrain or vertical gesture inputs.

IX. Conclusion

This research presents a vision-based, gesture-controlled drone navigation system that effectively interprets a closed-fist gesture to direct UAV movement. The system was built using open-source tools including ArduPilot SITL, MAVProxy, and MediaPipe, and demonstrated reliable performance in simulated environments. It provides an intuitive interface for UAV control that is hands-free, low-cost, and easily extendable. While the system currently operates with a limited gesture set and under ideal conditions, it lays the groundwork for more complex, real-world applications in assistive robotics, tactical operations, and autonomous drone missions. With future enhancements, such systems have the potential to significantly increase the usability and accessibility of UAVs in challenging environments.

A. Figures and Tables

The Heartbeat Detection Flowchart illustrates how ROS2 communicates with MAVROS and MAVLink to consistently monitor the heartbeat of the UAV. This ensures that the ground station remains aware of the UAV's active status and can trigger safety procedures when heartbeat loss is detected.

The Overall System Architecture Flowchart presents the entire gesture-controlled UAV system pipeline. It shows how user gestures are processed via computer vision modules, interpreted into commands, and then transmitted through ROS2 and MAVROS layers to control UAV behavior.

Fig. 1. Heartbeat Detection Flowchart

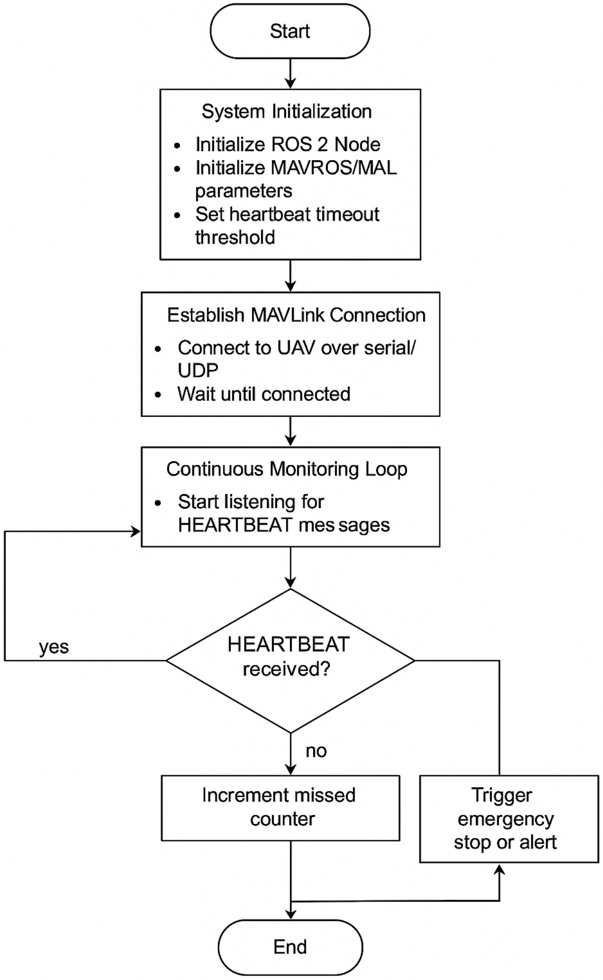
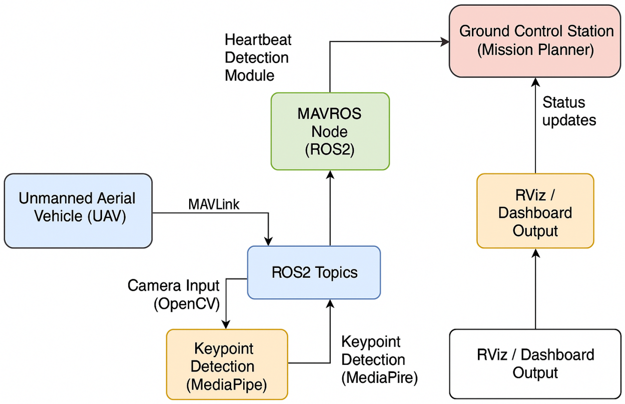

Start
System Initialization
Initialize ROS 2 Node
Initialize MAVROS/MAL parameters
Set heartbeat timeout threshold
Establish MAVLink Connection
Connect to UAV over serial/ UDP
Wait until connected
Continuous Monitoring Loop
Start listening for HEARTBEAT mes sages
HEARTBEAT received?
yes
no
Increment missed counter
Trigger emergency stop or alert
End

Fig. 2. Overall System Architecture Flowchart


Heartbeat Detection Module
Ground Control Station (Mission Planner)
MAVROS Node (ROS2)
Status updates
Unmanned Aerial Vehicle (UAV)
MAVLink
RViz / Dashboard Output
ROS2 Topics
Camera Input (OpenCV)
Keypoint Detection (MediaPipe)
Keypoint Detection (MediaPire)
RViz / Dashboard Output

Acknowledgment

This project was carried out by Riya Bhalavat, Rishi Kaushik, and Diya Nayak under the guidance of Prof. Sunny Nanade at Mukesh Patel School of Technology Management and Engineering. We express our sincere gratitude to Prof. Sunny Nanade for his constant support, insightful feedback, and encouragement throughout the course of this project. We also thank our institution for providing the necessary infrastructure and resources that enabled the successful completion of this work.

References

[1] Y. Zhang, J. Wu, H. Du, Y. Guo, Z. Shao, and C. Wang, "Trustworthy AI via scalable interpretable rule sets," *arXiv preprint*, arXiv:2406.00447, Jun. 2024.

[2] T. Garg, Y. Ren, M. S. Bernstein, and P. Mackenzie, "Crowd auditing fairness in hiring algorithms," in *Proc. 2023 CHI Conf. Human Factors in Computing Systems (CHI '23)*, Hamburg, Germany, Apr. 2023, pp. 1–16.

[3] P. Varshney, T. Garg, R. Ghosh, A. J. Biega, and M. S. Bernstein, "Designing participatory fairness audits: How domain experts and workers shape algorithmic fairness," in *Proc. 2023 ACM Conf. Fairness, Accountability, and Transparency (FAccT '23)*, Chicago, IL, USA, Jun. 2023, pp. 185–198.

[4] L. Chen, M. Lv, Q. Ye, G. Chen, and J. Woodward, "A personal route prediction system based on trajectory data mining," *Inf. Sci.*, vol. 181, no. 7, pp. 1264–1284, Apr. 2011.

[5] J. Selten, "Increasing multi-robot situation awareness of an intersection using prior knowledge and traffic rules to constrain driving behaviours," M.S. thesis, Dept. Mech. and Elect. Eng., Eindhoven Univ. Technol., Eindhoven, Netherlands, May 2023.

[6] M. Browne and S. S. Ghidary, "Convolutional neural networks for image processing: An application in robot vision," in *Proc. Aust. Joint Conf. Artif. Intell. (AI 2003)*, Perth, Australia, Dec. 2003, pp. 641–652.

[7] J. Roth, "Predicting route targets based on optimality considerations," in *Proc. 17th Int. Conf. Intell. Transp. Syst. Telecommunications (ITST 2014)*, Oct. 2014, pp. 1–6.

[8] C. Demetrescu and G. F. Italiano, "Engineering shortest path algorithms," in *Proc. 3rd Int. Workshop Exp. Algorithms (WEA 2004)*, Angra dos Reis, Brazil, May 2004, pp. 260–271.

[9] A. M. Raivi, S. M. A. Huda, M. M. Alam, and S. Moh, "Drone routing for drone-based delivery systems: A review of trajectory planning, charging, and security," *Sensors*, vol. 23, no. 3, p. 1463, Jan. 2023.