applied sciences

MDPI

Article

Enhancing Drone Navigation and Control: Gesture-Based Piloting, Obstacle Avoidance, and 3D Trajectory Mapping

Ben Taylor, Mathew Allen, Preston Henson, Xu Gao, Haroon Malik and Pingping Zhu *

Department of Computer Sciences and Electrical Engineering, Marshall University, Huntington, WV 25755, USA; taylor924@marshall.edu (B.T.); allen490@marshall.edu (M.A.); sellards11@marshall.edu (P.H.); gao32@marshall.edu (X.G.); malikh@marshall.edu (H.M.)
* Correspondence: zhup@marshall.edu

Abstract

Autonomous drone navigation presents challenges for users unfamiliar with manual flight controls, increasing the risk of collisions. This research addresses this issue by developing a multifunctional drone control system that integrates hand gesture recognition, obstacle avoidance, and 3D mapping to improve accessibility and safety. The system utilizes Google's MediaPipe Hands software library, which employs machine learning to track 21 key landmarks of the user's hand, enabling gesture-based control of the drone. Each recognized gesture is mapped to a flight command, eliminating the need for a traditional controller. The obstacle avoidance system, utilizing the Flow Deck V2 and Multi-Ranger Deck, detects objects within a safety threshold and autonomously moves the drone by a predefined avoidance distance away to prevent collisions. A mapping system continuously logs the drone's flight path and detects obstacles, enabling 3D visualization of drone's trajectory after the drone landing. Also, an AI-Deck streams live video, enabling navigation beyond the user's direct line of sight. Experimental validation with the Crazyflie drone demonstrates seamless integration of these systems, providing a beginner-friendly experience where users can fly drones safely without prior expertise. This research enhances human–drone interaction, making drone technology more accessible for education, training, and intuitive navigation.

Keywords: gesture-based drone control; autonomous obstacle avoidance; human–drone interaction; 3D mapping and navigation

Academic Editor: Rui Araújo

Received: 18 May 2025
Revised: 21 June 2025
Accepted: 23 June 2025
Published: 30 June 2025

Citation: Taylor, B.; Allen, M.; Henson, P.; Gao, X.; Malik, H.; Zhu, P. Enhancing Drone Navigation and Control: Gesture-Based Piloting, Obstacle Avoidance, and 3D Trajectory Mapping. *Appl. Sci.* **2025**, *15*, 7340. https://doi.org/10.3390/app15137340

1. Introduction

Drones have become essential for multiple fields, including disaster response, industrial inspection, and autonomous delivery systems [1–3]. Their versatility has led to widespread adoption in both commercial and research applications, but traditional drone operation requires precise manual control, often demanding significant training and experience [4]. This challenge is particularly evident in cluttered or dynamic environments where human pilots must react quickly to avoid obstacles [5]. To enhance accessibility and usability, researchers have explored alternative control methods such as gesture-based interfaces, brain–computer interaction, and voice-controlled navigation [6,7]. While these technologies have been studied independently, integrating them with other features into a seamless, user-friendly system remains an ongoing challenge.

Gesture-based control offers a natural alternative to conventional remote control systems. It leverages computer vision techniques to interpret human hand movements [8].

Specifically, Google's MediaPipe (Version 0.10.21) Hands software library can detect 21 landmarks of the human hand from static images and track these landmarks in videos, utilizing a machine learning model [9]. These landmark positions distinguish between different gestures [10]. Prior research has demonstrated the effectiveness of gesture-based interfaces in robotic control, virtual reality, assistive technology, and human–robot interaction [11–13]. However, their application in drone piloting—particularly when combined with autonomous safety mechanisms—remains underexplored. Some studies have attempted to integrate gesture recognition with drone control, but they often suffer from high latency, gesture misclassification with differing hand sizes, lighting issues, or limited adaptability in dynamic environments [14].

Autonomous obstacle avoidance, another key component of drone navigation, has been successfully implemented using infrared time-of-flight sensors, LiDAR, and deep-learning-based object detection [15,16]. These systems enable drones to detect and avoid obstacles, significantly reducing the risk of collisions in unknown environments [17]. However, many autonomous systems override user control entirely when avoiding obstacles, reducing pilot interaction and flexibility [18]. This raises important questions about how to balance human input with autonomous safety mechanisms to provide a smooth and responsive user experience.

This study proposes a multifunctional drone control system that integrates hand gesture recognition, autonomous obstacle avoidance, and 3D mapping using Bitcraze's Crazyflie drone to create an intuitive and beginner-friendly piloting experience [19]. The system ensures safe and interactive flight, allowing users to control drones with hand gestures. At the same time, an autonomous safety feature prevents collisions by detecting obstacles within a safety threshold ρ_{th} and retreating the drone by a fixed avoidance distance, ρ_{avoid}. A 3D mapping system visualizes the flight path and detected obstacles, and an AI-Deck streams live video, enabling navigation beyond direct line-of-sight. Unlike prior research, which often focuses on one aspect of drone control, this system bridges the gap between human interaction and autonomous safety, allowing for greater user engagement while maintaining obstacle avoidance [20,21].

By integrating gesture-based control with autonomous safety mechanisms and 3D mapping, this research advances human–drone interaction in a novel and practical way. Unlike many existing systems that treat gesture recognition and obstacle avoidance as separate functionalities, this work combines them to create a cohesive, intuitive user experience. This multifaceted integration directly addresses challenges in drone accessibility, offering a scalable and user-friendly solution for applications in education, training, and real-world navigation. The system contributes to the fields of robotic control, autonomous navigation, and interactive machine learning by lowering the barrier to entry for novice users while maintaining flight safety and environmental awareness.

Nonetheless, the system is not without limitations. Challenges such as occasional gesture misclassification, command latency, and sensitivity to lighting conditions remain areas for improvement. A recent overview of the UAV field emphasizes the growing need for multimodal and adaptive drone interfaces, underscoring the relevance and timeliness of this research [22]. Future efforts will aim to enhance gesture recognition reliability, reduce latency through software optimization, and increase system usability through better camera modules or dual-view configurations. Also, scaling the platform to support multi-drone coordination and adapting it for outdoor environments using GPS or SLAM-based localization would further expand its applicability and impact.

2. Materials and Methods

2.1. Integrated System Architecture and Workflow

The architecture of the proposed drone control system integrates multiple hardware and software components to enable intuitive, gesture-based drone navigation with obstacle avoidance and 3D mapping. The user initiates control by performing predefined hand gestures in front of a webcam. These gestures are captured and processed using Google's MediaPipe Hands framework. The landmark data are analyzed using custom Python classification functions that determine the type of gesture being shown. Using a command-mapping module, each recognized gesture is mapped to a specific flight command.

The mapped flight commands are wirelessly transmitted to the Bitcraze Crazyflie 2.1 drone via the Crazyflie Python library. Once airborne, the drone operates with three key expansion decks: the Multi-Ranger Deck for obstacle detection, the Flow Deck V2 for position estimation, and the AI-Deck for First-Person View (FPV) video streaming. The Multi-Ranger Deck uses infrared sensors to monitor proximity to objects in the front, back, left, and right directions.

Meanwhile, the Flow Deck V2 provides position data through a Kalman filter, which is logged every 0.1 s by a flight data logger. These coordinates generate a 3D map of the drone's path using Python's Plotly (Version 6.0.0) visualization library. Obstacle encounters are also marked on the map as red rectangular prisms. The AI-Deck streams live grayscale or color video to the user's screen using socket communication, enabling remote navigation beyond the user's direct line of sight. The FPV video feed and the interactive 3D map are displayed to the user, offering system feedback. This modular system functions in parallel, creating a responsive platform for safe, beginner-friendly drone piloting and for advanced mission-based piloting. This workflow and architecture can be visualized in Figure 1 below.

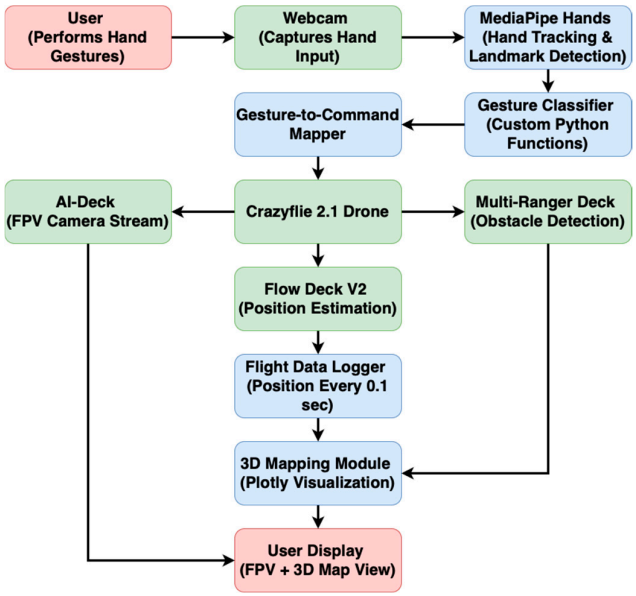


Figure 1. System architecture diagram illustrating the integrated components of the multifunctional drone control framework.

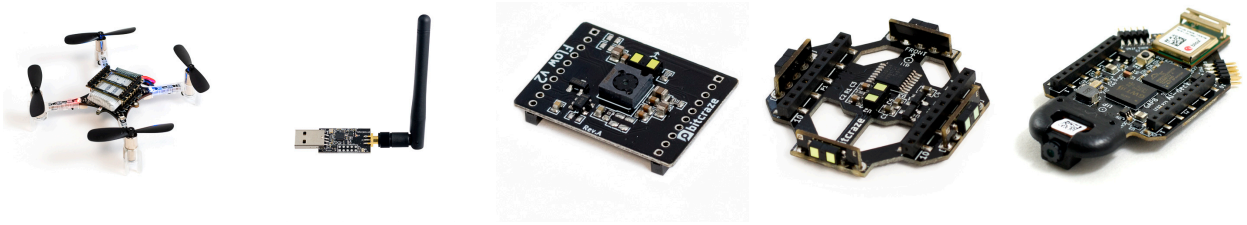
2.2. Hardware and Software Components

The Bitcraze Crazyflie 2.1 drone served as the experimental platform, selected for its compact size, modular design, and expandability. Several expansion decks and components were integrated to enhance its sensing and flight capabilities:

- **Crazyflie 2.1 Drone:** Figure 2a illustrates the main aerial platform for this research. It is a lightweight, open-source nano quadcopter developed by Bitcraze, equipped

with an STM32F405 microcontroller, built-in IMU, and expansion deck support via an easy-to-use pin header system [23]. Its modular design allows seamless integration of multiple decks for advanced sensing and control functionalities.

- **Crazyradio PA:** Figure 2b is a 2.4 GHz radio dongle used to establish wireless communication between the host computer and the Crazyflie 2.1 drone [24]. The PA (Power Amplifier) version enhances signal strength and communication reliability, particularly in environments with interference or extended range requirements.
- **Flow Deck V2:** Figure 2c provides optical flow-based position estimation and altitude measurement using a downward-facing time-of-flight (ToF) sensor and an optical flow camera [25]. This enables the drone to maintain stable flight without external positioning systems.
- **Multi-Ranger Deck:** Figure 2d utilizes infrared ToF distance sensors to detect obstacles in five directions—front, back, left, and right [26]. This facilitates obstacle avoidance by continuously monitoring the drone's surroundings.
- **AI-Deck 1.1:** Figure 2e features a camera module for video streaming, allowing users to navigate the drone beyond their direct line of sight [27]. The AI-Deck includes a GAP8 RISC-V multi-core processor, enabling onboard image processing and low-latency streaming to a ground station. A Linux system is required for setting up and flashing the AI-Deck, as the software tools provided by Bitcraze (Malmö, Sweden), such as the GAP8 SDK, are optimized for Linux-based environments.



(**a**) (**b**) (**c**) (**d**) (**e**)

Figure 2. (**a**) Crazyflie 2.1 Drone; (**b**) Crazyradio PA; (**c**) Flow Deck V2 Component; (**d**) Multi-Ranger Deck Component; (**e**) AI-Deck 1.1 Component.

Gesture-based control was implemented using Google's MediaPipe Hands, a computer vision framework capable of tracking 21 key landmarks on a user's hand. The gesture recognition software was implemented in Python (Version 3.11.0), processing webcam input to identify predefined gestures and mapping them to corresponding flight control commands for the drone. The system was executed in Visual Studio Code (Version 1.101) on a laptop running Windows 11 with an Intel Core i9 processor, 64 GB of RAM, and an integrated webcam, which provided sufficient performance for processing and drone communication. The system leveraged Bitcraze's Crazyflie Python library for wireless communication, ensuring seamless execution of gesture-based commands.

2.3. Code Implementation and System Logic

The full drone control system was implemented in Python, integrating hand gesture recognition, autonomous obstacle avoidance, 3D mapping, and AI-Deck video streaming. Upon startup, all necessary software libraries are imported, and the system establishes a wireless radio connection to the Crazyflie 2.1 drone. A socket connection is simultaneously opened to the AI-Deck for FPV streaming. Google's MediaPipe Hands framework is initialized to process webcam input and track 21 hand landmarks, which are later used for gesture classification. In parallel, the drone's position is logged using the Kalman filter output from the Flow Deck V2, and a blank 3D map is initialized using Plotly to visualize flight data and obstacle encounters.

Once the system is fully initialized, the main loop begins and handles multiple tasks concurrently. The AI-Deck continuously streams grayscale or color video, which is decoded and displayed in a live feed using OpenCV, enabling remote FPV-based navigation. Simultaneously, the Multi-Ranger Deck monitors the proximity of obstacles in the front, back, left, and right. If an object is detected within ρ_{th} in any direction, the drone overrides gesture commands and automatically moves ρ_{avoid} away from the object to avoid collision. The location of each obstacle encounter is recorded and visualized as a red rectangular prism on the 3D map.

Meanwhile, MediaPipe Hands detects the user's hand in the webcam frame and calculates geometric angles between finger joints and distances between landmark pairs. These features are passed into custom classification functions that recognize specific gestures such as "point", "thumb up", or "closed fist". Each gesture is mapped to a unique drone command. For example, a "point" gesture triggers forward motion, "hand open" results in vertical ascent, and "thumb up" moves the drone laterally to the left. The drone executes each command in small increments (typically 0.05 m) to ensure precision and responsiveness. When the landing gesture is detected, the drone initiates a landing sequence and stops logging data. At that point, the full 3D map is saved as an interactive HTML file, and flight coordinates are exported as a CSV dataset.

Throughout the flight, x, y, and z positions are recorded every 0.1 s using data from the Flow Deck. These positions are plotted and dynamically adjust the axis range of the 3D map. The AI-deck's FPV stream provides additional spatial awareness for navigating beyond the user's direct line of sight. These modules function in parallel, creating a responsive, beginner-friendly system that allows intuitive control while maintaining autonomous safety and environmental visualization. This system is visualized in Figure 3 below.

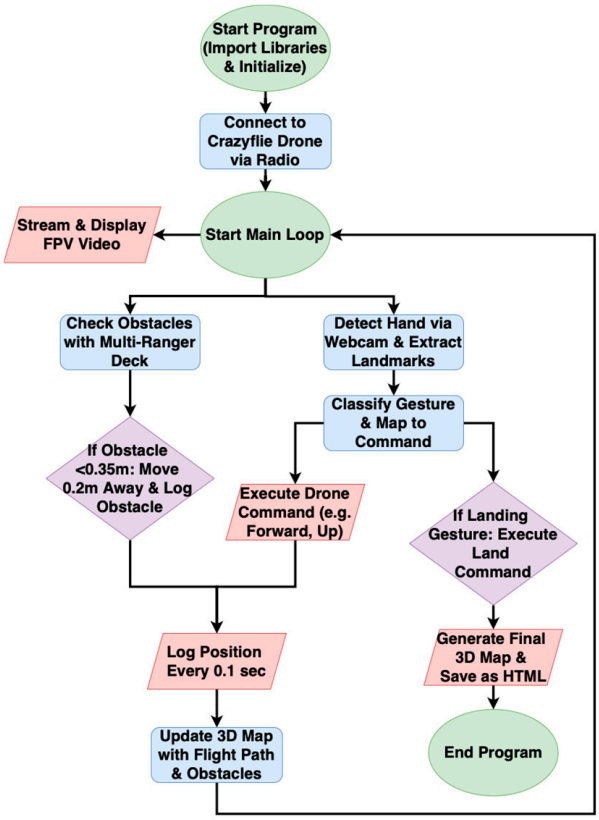


Figure 3. A block flowchart illustrating the full system logic of the multifunctional drone control framework.

2.4. Hand Gesture Recognition and Control

Gesture recognition in this study was implemented using Google's MediaPipe Hands, an open-source framework that leverages machine learning to detect and track 21 key hand landmarks using a single RGB webcam (Bitcraze, Malmö, Sweden), as illustrated in Figure 4. Each landmark corresponds to specific joints and tips of the fingers and wrist. Once detected, these landmark coordinates are used to classify predefined hand gestures by calculating joint angles and Euclidean distances between specific landmark pairs.

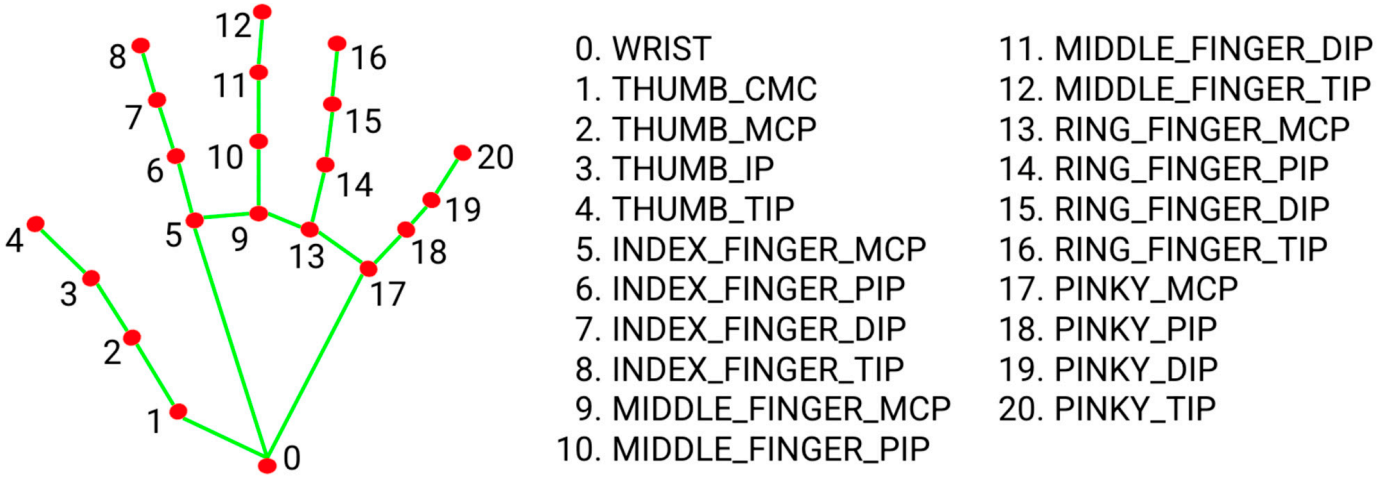


Figure 4. Hand landmark points used for gesture recognition [9].

The system evaluates angles at key finger joints by applying geometric calculations to three-point sequences. For example, to detect if a finger is extended or bent, the angles formed at the carpometacarpal (CMC), metacarpophalangeal (MCP), proximal interphalangeal (PIP), and distal interphalangeal (DIP) joints are measured. An extended finger typically shows an angle greater than 120°, whereas a bent finger produces a significantly smaller angle (usually <100°). In addition to angles, distances between landmark points (e.g., between the thumb tip and the base of the index finger) are used to determine the spread of the hand or proximity of fingers, further enhancing gesture classification. To control the drones using the different hand gestures, furthermore, we select eight common gestures in this paper, including "pointing", "peace", "thumbs up", "pinky", "open hand", "fist", "okay", and "love", to indicate the corresponding drone commands, as described below and demonstrated in Figure 5.

- A "pointing" gesture moves the drone forward.
- A "peace" gesture moves the drone backward.
- A "thumbs up" gesture moves the drone to the left.
- A "pinky" gesture moves the drone to the right.
- An "open hand" gesture triggers the drone to ascend (up).
- A "fist" gesture triggers the drone to descend (down).
- An "okay" gesture triggers the drone to land.
- A "love" gesture triggers the drone to rotate.

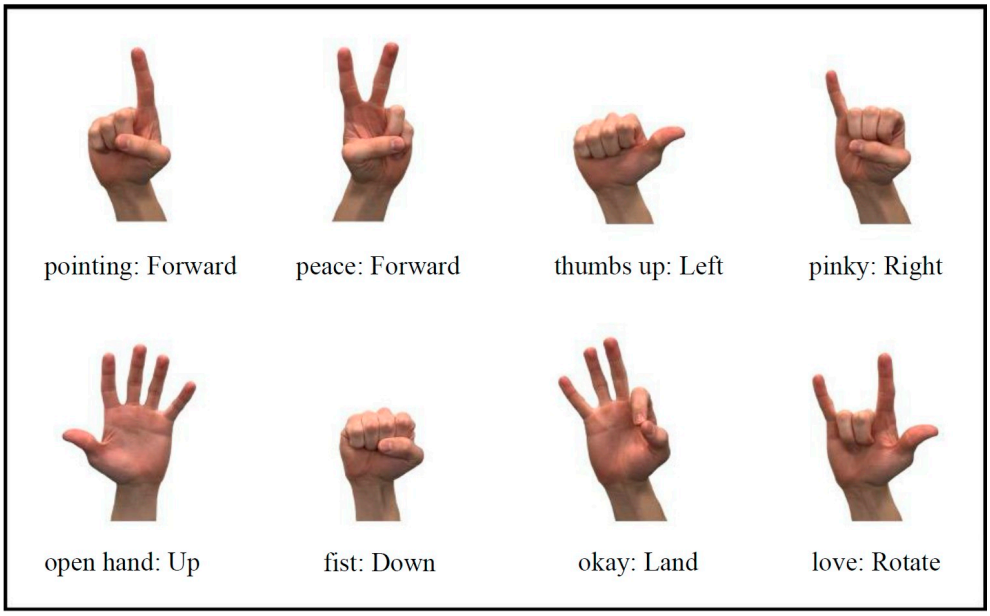


Figure 5. Visual representation of all hand gestures showing each mapped drone command and its corresponding hand pose for clarity.

First, the system detects $i = 0, 1, 2, \ldots, 20$ hand landmarks per frame and each landmark is represented by $v_i \in \mathbb{R}^2$ in 2D image coordinates:

$$\boldsymbol{v}_i = [x_i, y_i]^T \text{ for } i = 0, 1, \ldots, 20 \tag{1}$$

where "T" indicates the transpose. Then, a gesture in a frame can be represented by a 21×2 matrix $\boldsymbol{V} = \begin{bmatrix} \boldsymbol{v_0} & \ldots & \boldsymbol{v_{20}} \end{bmatrix}^{\boldsymbol{T}}$, which are the raw data for the gestures.

To classify hand gestures into the eight classes, such that $\mathcal{G} = \{G_{pointing}, G_{peace}, G_{thumbsUp}, G_{pinky}, G_{openHand}, G_{fist}, G_{okay}, G_{love}\}$ as shown in Figure 5, in the next step, the system extracts two types of features from the matrix $\boldsymbol{V}$, including the distance feature d_{ij} and the angle feature θ_{ijk} where $i, j, k \in \{0, 1, \ldots, 20\}$. Specifically, the distance features are defined by the Euclidean distances between landmark pairs i and j, such that

$$d_{ij} = |\boldsymbol{v}_i - \boldsymbol{v}_j| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{2}$$

In addition, the joint angles features are defined using the cosine rule for three consecutive landmarks, i,' j, and k, such that

$$\theta_{ijk} = arccos\left(\frac{(\boldsymbol{v}_i - \boldsymbol{v}_j)^T(\boldsymbol{v}_k - \boldsymbol{v}_j)}{d_{ij}d_{kj}}\right) \tag{3}$$

Finally, considering the system effectiveness and efficiency, we implemented this hand gesture classification only based a subset of ten features, such that $\boldsymbol{\Theta} = \{\theta_i\}_{i=0}^{8} \cup d_0$, where $\theta_0 = \theta_{0,5,6}$, $\theta_1 = \theta_{0,9,10}$, $\theta_2 = \theta_{0,13,14}$, $\theta_3 = \theta_{0,17,18}$, $\theta_4 = \theta_{5,6,7}$, $\theta_5 = \theta_{5,6,8}$, $\theta_6 = \theta_{9,10,11}$, $\theta_7 = \theta_{13,14,15}$, $\theta_8 = \theta_{17,18,19}$ $d_0 = d_{4,17}$. Specifically, the gesture classification can be determined by using the following Boolean combinations of these features:

$$G = \begin{cases} G_{pointing}, & if \quad \theta_1 > 120^\circ\ \theta_5 > 120^\circ\ \theta_6 > 100^\circ\ \theta_7 > 100^\circ\ \theta_8 > 120^\circ \\ G_{peace}, & if \quad \theta_0 > 120^\circ\ \theta_5 > 120^\circ\ \theta_1 > 120^\circ\ \theta_6 > 120^\circ\ \theta_7 > 100^\circ\ \theta_8 > 100^\circ \\ G_{thumbsUp}, & if \quad \theta_4 > 100^\circ\ \theta_6 > 100^\circ\ \theta_7 > 100^\circ\ \theta_8 > 100^\circ\ d_0 > 225 \\ G_{pinky}, & if \quad \theta_3 > 120^\circ\ \theta_5 > 130^\circ\ \theta_6 > 130^\circ\ \theta_7 > 130^\circ\ \theta_8 > 120^\circ \\ G_{openHand}, & if \quad \theta_0 > 120^\circ\ \theta_5 > 120^\circ\ \theta_6 > 120^\circ\ \theta_7 > 120^\circ\ \theta_8 > 120^\circ \\ G_{fist}, & if \quad \theta_4 > 100^\circ\ \theta_6 > 120^\circ\ \theta_7 > 100^\circ\ \theta_8 > 100^\circ\ d_0 < 225 \\ G_{okay}, & if \quad \theta_4 < 100^\circ\ \theta_1 > 120^\circ\ \theta_2 > 120^\circ\ \theta_3 > 120^\circ\ \theta_6 > 120^\circ\ \theta_7 > 120^\circ\ \theta_8 > 120^\circ \\ G_{love}, & if \quad \theta_0 > 120^\circ\ \theta_3 > 120^\circ\ \theta_5 > 120^\circ\ \theta_6 < 130^\circ\ \theta_7 < 130^\circ\ \theta_8 > 120^\circ \\ G_{other}, & otherwise \end{cases} \tag{4}$$

where the determined gesture class $G \in \mathcal{G} \cup G_{other}$ and G_{other} indicates any unspecified hand gestures. This classification can also be implemented using a decision tree, as demonstrated in Figure 6. It is noteworthy that in the decision-tree diagram, the unspecified hand gesture G_{other} is ignored.

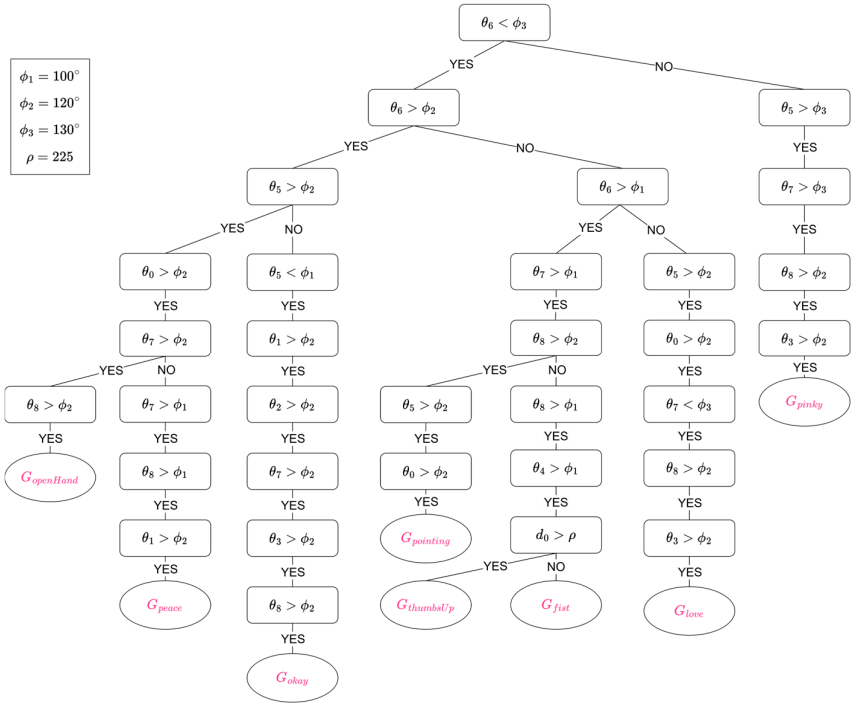


Figure 6. The decision-tree diagram of eight-class gesture classification.

Once a gesture was classified, it was mapped to a corresponding drone flight command through a gesture-command control function implemented in Python. Using the Crazyflie Python API and the MotionCommander interface, each recognized gesture triggered a predefined flight behavior. For instance, when the system detected an open hand gesture, $G = G_{openHand}$, the drone executed a vertical ascent using the *mc.up*() command [10]. This mapping system allowed for smooth, real-time gesture-to-command execution and was designed to be modular, enabling additional gestures and flight behaviors to be added with minimal modification to the underlying code.

These mappings enabled the drone to be controlled solely through hand gestures, offering an intuitive and accessible user experience. The system processed webcam input frame-by-frame, continuously checking for changes in gesture state and executing flight commands accordingly. This approach eliminated the need for physical controllers and reduced the learning curve for beginner users.

2.5. Obstacle Avoidance System

The obstacle avoidance system utilized sensor data from the Multi-Ranger Deck to detect obstacles in the front, back, left, and right directions. Unlike autonomous navigation systems that follow a predetermined path, this drone only moves in response to user hand gestures. While executing a commanded movement, the drone continuously scans its surroundings.

Let the drone be situated in a 3D space with its local coordinate system aligned such that the front direction corresponds to the positive x-axis and left to the positive y-axis. To assess the proximity of obstacles, the system defines directional obstacle distances by $\rho_d \in \{\rho_F, \rho_B, \rho_L, \rho_R\}$ indicating the measured distances to obstacles in the front, back, left, and right directions. These distances are obtained using infrared time-of-flight (ToF) sensors on the Multi-Ranger Deck, which actively measure the shortest line-of-sight distance between the drone and any nearby surface in the specified directions.

To ensure safe navigation, the obstacle avoidance algorithm uses a dynamic safety threshold, denoted as ρ_{th}. This value accounts for the physical limitations of the Crazyflie drone and its ability to decelerate upon detecting an obstacle. This safety threshold is computed using principles derived from Newton's Laws of Motion, such that

$$\rho_{th} = vt_{latency} + \frac{v^2}{2a} \tag{5}$$

where v represents the drone's velocity, $t_{latency}$ indicates the command execution latency, and a indicates the deceleration rate of the drone, respectively. This equation calculates how far ahead the drone should start reacting to an obstacle, based on its current velocity and ability to decelerate, including the first term *reaction distance* and the second term *braking distance*. The reaction distance, $vt_{latency}$, indicates the distance the drone travels during the delay in command execution, and the braking distance, $v^2/(2a)$, indicates the distance needed to decelerate to a stop. The reaction distance contributes minimally to the threshold due to its small magnitude, making the braking distance the dominant factor at higher velocities. The deceleration rate is inherent to the drone's hardware and was verified through controlled testing. This formulation enables the safety threshold to scale dynamically with velocity, providing a greater margin at higher speeds to prevent collisions, while at lower speeds, decreasing the margin for more precise navigation.

In this paper, specifically, we consider the maximum velocity of the Crazyflie, $v_{max} = 1$ m/s, according to Bitcraze Forums, the command execution latency, $t_{latency} = 5$ milliseconds (ms) (more details are provided in Section 3.4.3), and the deceleration rate of the drone $a = 1.5$ m/s^2. Here, the deceleration rate is obtained from experimental measurements, which vary depending on the different weights of the drones and the sensors equipped. Then, the safety threshold can be obtained as, $\rho_{th} = 0.3383 \approx 0.35$ m, by substituting these specifications into (5). Furthermore, a reasonable avoidance distance is set by $\rho_{avoid} = 0.2$ m.

When any detection distance ρ_d between the drone and the obstacles falls below this threshold, the system initiates an automatic avoidance maneuver, moving it by $\rho_{avoid} = 0.2$ m in the opposite direction to maintain a buffer from the obstacle.

$$\Delta x = \begin{cases} -\rho_{avoid}, & \text{if } \rho_F < \rho_{th} \\ +\rho_{avoid}, & \text{if } \rho_B < \rho_{th} \end{cases} \quad \text{and} \quad \Delta y = \begin{cases} -\rho_{avoid}, & \text{if } \rho_R < \rho_{th} \\ +\rho_{avoid}, & \text{if } \rho_L < \rho_{th} \end{cases} \tag{6}$$

These avoidance movements are issued using the Bitcraze Python API's Motion Commander interface, allowing precise, small-scale control of the Crazyflie drone's trajectory. It is noteworthy that, since the safety threshold, $\rho_{th} = 0.35$, is calculated based on the maxi-

mum velocity, $v_{max} = 1$ m/s, this approach results in safe navigation while maintaining user control as long as the velocity is lower than v_{max}. Furthermore, an adjustable safety threshold associated with the time-varying velocities can be proposed and implemented easily according to (5), although we did not apply it in our experiments for simplicity.

2.6. 3D Mapping and Flight Data Logging

A 3D mapping system was implemented to visualize the drone's flight path and detect obstacles. The system recorded the drone's x, y, and z coordinates at 0.1 s intervals, logging its movement throughout the flight. Obstacle detection data from the Multi-Ranger Deck were incorporated into the map, with obstacles represented as rectangular prisms positioned at a distance from the detection point based on the safety threshold.

The collected flight data were processed and visualized using Python's Plotly library, generating an interactive 3D representation of the environment. The resulting visualization provided a clear depiction of the drone's trajectory and mapped obstacles, aiding in post-flight analysis and system evaluation.

2.7. Experimental Setup and Testing Procedure

Experiments were conducted in a controlled indoor environment to evaluate the performance of the integrated system, focusing on gesture recognition, obstacle avoidance, and 3D mapping accuracy. The testing procedure consisted of three primary evaluations:

- **Hand Gesture Control Test:** Users performed predefined hand gestures in front of a webcam to verify accurate classification and execution of drone commands. Each gesture was tested multiple times to assess recognition accuracy and command responsiveness.
- **Obstacle Avoidance Test:** The drone was commanded to move forward while obstacles were placed in randomized positions along its path. The system's ability to detect and avoid obstacles within the safety threshold was analyzed, recording the reliability of avoidance maneuvers and any unintended collisions.
- **3D Mapping Validation:** To assess mapping accuracy, the recorded flight path and obstacle locations were compared to real-world measurements. Errors in obstacle placement and discrepancies in flight path representation were quantified.

Each test was repeated multiple times to ensure consistency.

2.8. Data Availability

The Python code used for gesture recognition, obstacle avoidance, 3D mapping, AI-Deck streaming, and supporting resources will be publicly available in a GitHub repository upon publication. The repository will also include images of the generated 3D maps and a YouTube video demonstrating the system's functionality.

3. Results

3.1. Obstacle Avoidance Performance

3.1.1. Obstacle Detection and Response

The obstacle avoidance system was designed to prevent collisions by integrating the Multi-Ranger Deck, which utilizes time-of-flight (ToF) sensors to detect objects in the drone's surroundings. Upon detecting an obstacle within $\rho_{th} = 0.35$ m, the drone autonomously retreats $\rho_{avoid} = 0.2$ m to ensure safe clearance. These values were selected to ensure the drone reacts promptly without excessive movement, striking a balance between responsiveness and stability.

The Multi-Ranger Deck can measure distance up to 4 m within a few millimeters, as provided by Bitcraze [26]. This level of accuracy allowed for consistent and reliable obstacle

detection throughout testing, as proved by Table 1. Also, the avoidance parameters were implemented as adjustable variables in the code, enabling customization based on the user's specific requirements or environmental constraints.

Table 1. Obstacle avoidance success rates based on the direction of obstacle encounter relative to the drone.

Obstacle Location	Avoidance Success Rate (%)
Forward	100
Backward	100
Left	100
Right	100
Angled	100

To evaluate the obstacle avoidance system's reliability, structured tests were conducted where the drone was flown perpendicularly toward an obstacle placed at various orientations. For each direction—front, back, left, right, and angled (with the obstacle positioned diagonally)—the drone was commanded via gesture control to fly directly toward the object. Each scenario was tested 25 times, for a total of 125 trials. As shown in Table 1, the drone successfully avoided collisions in all cases, achieving a 100% avoidance rate across all directions.

During testing, when the drone's battery was fully charged and all systems were functioning optimally, there were no instances where the drone failed to react to an obstacle and flew directly into it. The system reliably detected and responded to obstacles, enhancing the drone's ability to navigate safely in constrained environments.

3.1.2. Obstacle Response During Perpendicular vs. Angled Approaches

During testing, the drone's reaction to obstacles differed depending on how it approached them—either perpendicularly as seen in Figure 7a or at an angle as seen in Figure 7b. When flying straight toward an obstacle, the Multi-Ranger Deck reliably detected the object within the 0.35 m threshold and immediately retreated 0.2 m to avoid a collision. These perpendicular encounters typically resulted in smooth, consistent avoidance maneuvers, as the object was well aligned with the front-facing sensor.

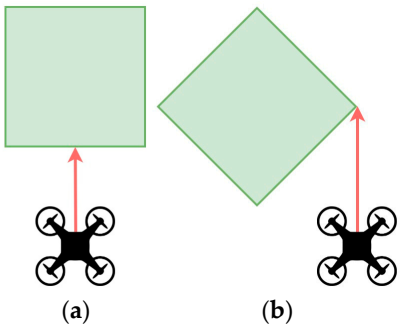


Figure 7. (**a**) Typical scenario of drone encountering obstacle perpendicularly. (**b**) Scenario of drone encountering an obstacle at an angle.

However, when the drone encountered an object at an angle, avoidance behavior became more nuanced. Angled obstacles—such as those turned or placed diagonally—were not always detected symmetrically by the front or side sensors. This asymmetry increased the possibility of the drone clipping a propeller on the edge of the object, particularly when the contact point fell between two sensor fields. Despite this, the drone successfully avoided collisions in these situations by responding based on whichever side

of the drone's boundary was first triggered. The system adjusted, initiating a horizontal or vertical avoidance maneuver depending on which sensor crossed the safety threshold first. A demonstration video of these two scenarios is available in Supplementary Materials (Video S1).

Furthermore, the drone could approach obstacles at awkward angles due to its ability to rotate and then fly forward. Although the control system does not include a specific gesture for diagonal flight, this functionality effectively allows the drone to fly at an angle relative to its original grid orientation. As a result, obstacles may be encountered off-axis, challenging the avoidance system's ability to interpret directionality accurately. Nonetheless, the obstacle detection logic proved adaptive enough to handle these situations, maintaining safe distances and adjusting flight paths as needed.

3.2. Hand Gesture Recognition Performance

3.2.1. Gesture Recognition Accuracy

To assess the accuracy of the hand gesture recognition system, a controlled test was performed using all eight predefined gestures: pointing, peace, thumbs up, pinky, open hand, fist, okay, and love. Each gesture was presented perpendicularly and approximately one foot away from the laptop's webcam—identified as the optimal distance and orientation for consistent detection. The gesture was raised and removed from the frame 25 times per test. The pointing, pinky, open hand, fist, okay, and love gestures were correctly identified in all 25 attempts, achieving a 100% recognition rate. However, the peace and thumbs up gestures occasionally exhibited inconsistencies in recognition. The peace gesture was sometimes initially misclassified as an open hand before correcting itself shortly after. Similarly, the thumbs up gesture was misidentified once as a fist, likely due to the visual similarity between the two gestures when the thumb is bent. Despite these minor issues, the system demonstrated high overall accuracy, as proven in Table 2, particularly when gestures were presented in the ideal position relative to the camera.

Table 2. Accuracy of gesture recognition for each predefined hand gesture used in the system.

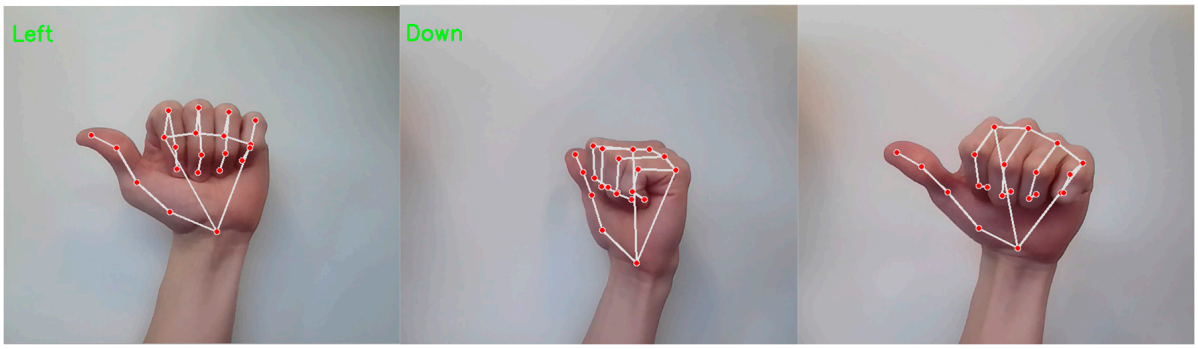
Hand Gesture	Accuracy
Pointing	100
Peace	88
Thumbs Up	96
Pinky	100
Open Hand	100
Fist	100
Okay	100
Love	100

While MediaPipe Hands provided tracking capabilities, its accuracy could be inconsistent due to the system's sensitivity to hand positioning. For optimal recognition, the user's hand needed to be perfectly perpendicular and close to the camera as seen in Figure 8a. If the hand was at an angle or too far away, the system sometimes failed to recognize gestures properly, leading to delays as users had to adjust their hand positioning multiple times.

Several factors impacted the system's reliability:

- **Gesture Misclassification:** The system occasionally misinterpreted one gesture as another (Figure 8b), especially when switching between gestures, leading to unintended drone movements. Also, if a gesture was presented at an unfavorable angle or partially obscured, it sometimes failed to register as any labeled gesture (Figure 8c), resulting in no response from the drone.

- **Lighting Conditions:** Bright, even lighting improved recognition, while dim or uneven lighting sometimes led to unstable tracking of landmarks.
- **External Hand Interference:** If another person's hand entered the camera's field of view, the system sometimes detected it as an input, disrupting the control process.



(**a**) (**b**) (**c**)

Figure 8. (**a**) Correct gesture representing "left"; (**b**) "Left" gesture misclassified as "down"; (**c**) Gesture failing to register as "left." The red points in the figures indicate the land marks of hands.

Despite these limitations, keeping the hand close to the camera and maintaining a stable positioning significantly improved detection reliability. Future improvements could include additional filtering techniques to reduce misclassification and enhance reliability in changing scenarios.

3.2.2. User Experience and Usability

The hand gesture control system provided an intuitive and engaging way to interact with the drone, especially for users with little to no prior experience in drone piloting. By eliminating the need for a traditional controller, the system allowed users to fly the drone using natural hand movements, making the experience more accessible.

Most users required some initial practice to become comfortable with the system. The main learning curve involved the following:

- Memorizing multiple gestures and associating them with their corresponding drone commands.
- Maintaining proper hand positioning in front of the camera to ensure accurate recognition.
- Balancing attention between their hand gestures and the drone's movement, particularly when not relying on the AI deck's streamed FPV footage.

Once users became familiar with the system, they found it to be a responsive and effective alternative to traditional control methods. The ability to adjust hand positioning and recognize gestures contributed to an overall smooth and immersive control experience.

3.3. 3D Mapping and Flight Path Analysis

The 3D mapping system recorded obstacles and the drone's flight path based on sensor data collected during flight. The accuracy of the mapping in Table 3 was evaluated by comparing the generated 3D representations with the physical obstacle and path locations.

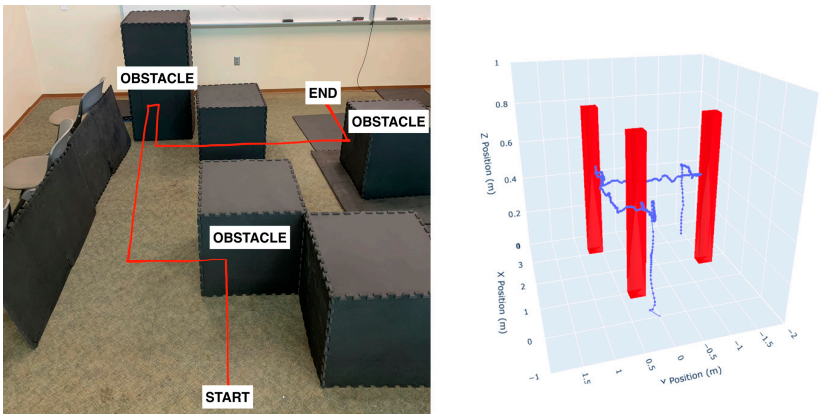
The drone's movements during this test, including minor wobbles and deviations, were consistently captured on the 3D map, resulting in zero deviation between the recorded flight path and the 3D map. For obstacle placement accuracy, the dimensions and layout of the physical maze were measured and compared with the mapped results. The first obstacle showed the most deviation, likely due to instability during the drone's initial takeoff. The obstacle placement is based on the safety threshold distance of $\rho_{th} = 0.35$ m,

meaning that any adjustments to the code could easily result in misplaced obstacles: a flaw within the customizable system.

Table 3. Deviation between the generated 3D map and the actual real-world flight path.

Components	Deviation (cm)
Flight Path	0
Obstacle 1	11
Obstacle 2	5
Obstacle 3	4

The mapping system only placed encountered obstacles on the map—meaning obstacles that directly affected the drone's path. If the drone flew near a wall or in front of an obstacle without attempting to move into it, that obstacle was not recorded in the 3D map. Instead, when the drone needed to determine a viable flight direction and an obstacle blocked that path, its location was noted and placed on the map. This explains why the 3D map (Figure 9b) contained fewer obstacles than the physical course (Figure 9a), and this discrepancy should not be interpreted as inaccuracy.



(**a**) (**b**)

Figure 9. (**a**) Physical maze setup for drone navigation testing; (**b**) 3D map reconstruction of the drone's flight path and encountered obstacles.

The flight path mapping was highly accurate, consistently starting at (0,0,0) and tracking movement relative to the takeoff position. Even when the drone wobbled or slightly deviated from a straight path, these deviations were accurately reflected in the 3D map. However, obstacle placement accuracy depended on sensor-based distance estimation. When an obstacle was detected, it was mapped at 0.35 m from the detection point. Any slight sensor inconsistencies in distance measurement could lead to minor errors in obstacle placement, but the overall flight path remained precise and reliable. This 3D mapping example along with gesture control, obstacle avoidance, and drone FPV streaming can be seen in the Supplementary Materials (Video S2).

3.4. Integrated System Performance: Gesture Control and Obstacle Avoidance

3.4.1. Interaction Between Gesture Control and Obstacle Avoidance

Integrating gesture control with obstacle avoidance introduced an interactive dynamic where user commands had to coexist with the drone's autonomous safety responses. When a user issued a command directing the drone toward an obstacle, the avoidance system overrode the gesture-based movement, causing the drone to back away from the detected object. This ensured collision prevention but occasionally led to unexpected movements from the user's perspective.

Despite this, users retained control over the drone's flight, as they could immediately issue a new gesture command after the avoidance maneuver. The 0.35 m safety threshold and 0.2 m retreat distance traded off safety and responsiveness, preventing crashes without making the drone feel unresponsive. Also, since these values were adjustable in the code, the system could be fine-tuned for different user preferences or environments.

During testing, users found that while the avoidance system sometimes interrupted their intended flight path, it did not completely disrupt their ability to navigate the drone. However, in confined spaces with multiple obstacles, the drone could enter a cycle of avoidance maneuvers, requiring the user to adjust their gestures to navigate effectively. Overall, the system provided a safe yet intuitive method for controlling the drone, even for users unfamiliar with manual drone operation.

3.4.2. AI-Deck Streaming and Remote Navigation

The AI-Deck's video streaming feature allowed for navigation beyond line of sight using both greyscale and color modes, enhancing user control and environmental awareness. While the color mode (Figure 10a) provided better scene interpretation, the greyscale mode (Figure 10b) offered better contrast, which could be beneficial in certain lighting conditions.

(**a**) (**b**)

Figure 10. (**a**) Image of AI-Deck streamed footage in color mode. (**b**) Image of AI-Deck streamed footage in greyscale mode.

The limited field of view (FOV) of the AI-Deck stream meant that users could only see directly in front of the drone, making it difficult to gauge surroundings beyond the camera's perspective. Combined with the low, 324 $\times$ 324 pixel resolution, this limitation made it challenging to discern fine details and detect distant obstacles during navigation. These constraints posed difficulties when maneuvering through complex environments. However, the obstacle avoidance system compensated for this limitation by detecting nearby obstacles outside the camera's view and preventing collisions. This combination of AI-Deck streaming and autonomous obstacle avoidance allowed for a more reliable navigation experience, even with the limitations of the video feed.

3.4.3. System Latency Analysis

The latency in gesture-based control primarily stems from the looping sequence of detecting a hand gesture and executing the corresponding drone movement. In each loop cycle, the system first detects the user's hand gesture, then issues a movement command to the drone, and only after completing the movement does it begin detecting the next gesture. To obtain prompt responsiveness, the movement distance per gesture is set to 0.05 m. This execution interval allows the drone's overall gesture-to-command latency to be approximately 209 milliseconds ($\pm$53 ms). This delay includes 200 milliseconds ($\pm$50 ms) for gesture recognition, 4 milliseconds ($\pm$1 ms) for command transmission via Crazyradio, and 5 milliseconds ($\pm$2 ms) for drone command execution. These values were

obtained through repeated experimental trials using custom Python commands designed to benchmark each phase of the control loop. A complete summary of these measured latencies is presented in Table 4.

Table 4. Latency measurements for key phases in the gesture-to-command control loop.

Action	Latency (ms)	Std Dev (ms)
AI-Deck to Host PC	100	±50
Gesture Detection	200	±50
Command Transmission	4	±1
Command Execution	5	±2

The latency associated with the AI-Deck's live video stream was measured at approximately 100 milliseconds, with a standard deviation of ±50 milliseconds. Although slightly noticeable, this delay did not significantly impair situational awareness or the user's ability to issue commands based on the visual feedback.

For the obstacle avoidance system, latency was even lower. Since obstacle detection and response are handled entirely within the internal logic loop of the Python program, without requiring gesture recognition or external input, the response time was limited to the drone's command execution delay. This resulted in a rapid, near-instantaneous obstacle avoidance reaction time of approximately 5 milliseconds (±2 ms), ensuring timely evasion of objects detected within the safety threshold.

3.5. Comparison with Existing Gesture-Controlled Drone Systems

To contextualize the novelty and performance of the proposed system, we compared it with two recent state-of-the-art gesture-based drone control systems from Yun et al. [28] and Lee and Yu [29]. Each system focused on hand gesture recognition for drone navigation but differed significantly in terms of autonomy, user feedback, and mapping capabilities.

The system proposed by Yun et al. [28] utilized an RGB camera to recognize multiple different single-handed gestures for drone control. While their system achieved a high accuracy of +90% using both motion- and posture-based gestures and introduces a lightweight architecture for mobile and edge devices, it lacks a built-in obstacle avoidance system. In contrast, our system not only offers gesture-based navigation with similarly high recognition accuracy but also integrates autonomous obstacle avoidance. This feature is valuable in gesture-controlled drone systems because it enables novice users to operate the drone intuitively while minimizing the risk of collision.

Lee and Yu's wearable drone controller [29] uses an inertial measurement unit (IMU) on the hand to classify gestures via machine learning and includes vibrotactile feedback to alert users of nearby obstacles. Although this provides a helpful sensory cue, the user must interpret the vibration and manually respond by steering the drone to avoid collisions. In contrast, our system eliminates this burden: the drone's obstacle avoidance is entirely autonomous, requiring no user input to prevent crashes. This makes our platform more accessible and safer, especially in obstacle-dense environments.

Also, neither of the two compared systems offers a mapping system. Our solution includes 3D mapping that visualizes the drone's flight trajectory and marks obstacles as red prisms in a virtual space. This capability provides insights into the environment and drone behavior over time and is especially useful for post-flight analysis.

In summary, unlike existing systems that rely heavily on manual control or external feedback, the proposed system is distinguished by its tight integration of gesture control, autonomous obstacle avoidance, real-time 3D mapping, and FPV video streaming, offering a unique and user-friendly platform for advanced drone interaction.

4. Discussion

4.1. Gesture Recognition and User Interaction

This study demonstrates the successful integration of gesture-based drone control, autonomous obstacle avoidance, and 3D mapping, providing a more accessible and intuitive method for drone navigation. The hand gesture control system, implemented using Google's MediaPipe Hands, accurately recognized predefined gestures and converted them into flight commands. However, lighting conditions, hand positioning, and background noise affected recognition accuracy, a limitation observed in prior research on vision-based gesture interfaces [30]. Despite this, the system provided hands-free control, reducing the need for traditional controllers and lowering the barrier to drone operation, aligning with previous studies that highlight the benefits of gesture-based interfaces for robotics and UAV applications [31].

A key aspect of this research was the integration of gesture control with autonomous navigation, allowing users to interact with the drone while ensuring safety through automatic collision avoidance. Studies on human–robot interaction suggest that balancing user input with autonomous decision-making is critical to achieving an effective shared control system [32]. Future work could focus on refining the balance between manual input and autonomous intervention, ensuring that the system adapts dynamically to user intentions while maintaining safety protocols. One possible approach is to use reinforcement learning or adaptive AI models to adjust the system's response based on feedback [33].

4.2. Obstacle Avoidance and Power Limitations

The obstacle avoidance system, utilizing infrared time-of-flight sensors, effectively detected and prevented collisions by identifying obstacles within 0.35 m and retreating 0.2 m. This mechanism ensured continuous flight in cluttered environments, where multiple directions were sometimes obstructed simultaneously. While the system performed reliably, occasional issues occurred due to the drone's battery size, which led to rapid power depletion after only a few flights. This constraint not only restricted the duration of each test session but also introduced risks during extended flight operations, such as mid-flight shutdowns or degraded sensor performance. Power limitations affecting drone performance have been discussed in previous research, suggesting that optimizing power management strategies or using energy-efficient algorithms could enhance flight duration [34]. Furthermore, improving obstacle detection through sensor fusion with LiDAR or ultrasonic sensors has been proposed as a way to increase reliability in within different environments [35]. While the obstacle avoidance mechanism performed well in the given test setup, the experimental conditions were limited to short-range, single-obstacle scenarios. This system could be explored further to enhance its operation in more complex environments by incorporating real-time path planning and adaptive behavior to maintain safe navigation amid multiple obstacles or irregular layouts.

4.3. 3D Mapping and Visualization Accuracy

The 3D mapping system provided valuable insight into the drone's flight path and surrounding environment. By logging x, y, and z coordinates every 0.1 s, the system generated a spatial representation of obstacles as rectangular prisms positioned 0.35 m from the detection point. While the maps closely matched the lab's layouts, minor positioning inaccuracies were sometimes observed, likely due to improper coding adjustments. Such errors, though small, can compound over time and undermine the system's reliability, particularly in more complex environments where precise localization in essential. Prior research on SLAM (Simultaneous Localization and Mapping) systems suggests that incorporating adaptive filtering techniques, such as Kalman filtering, could improve mapping

precision [36]. Also, incorporating higher-resolution spatial data processing could enhance the system's ability to visualize complex environments [37].

The addition of an AI-Deck for live video streaming introduced a new capability: beyond-line-of-sight control. This feature allows users to pilot the drone remotely using live footage, which has promising applications in search and rescue, reconnaissance, and surveillance. However, the AI-Deck camera was limited by its low resolution and fixed focus, resulting in poor image quality. Prior research has explored alternative small-scale camera modules that improve image quality while maintaining low latency [38]. Exploring better hardware options or implementing image-processing techniques such as super-resolution reconstruction could enhance the user experience [39].

The integrated system functioned effectively as a cohesive unit, enabling intuitive control while maintaining a safety layer through obstacle avoidance. However, improvements are needed to enhance usability, responsiveness, and overall performance. The system's responsiveness was limited by occasional processing delays, likely caused by the computational overhead of gesture recognition and obstacle detection running simultaneously. Optimizing the software pipeline by utilizing multi-threading or GPU acceleration could improve processing efficiency [40]. Furthermore, reducing latency in the gesture recognition system could improve user experience, as research has shown that delays exceeding 100 milliseconds negatively impact interactive control systems [41].

4.4. Validity Considerations and Future Work

Although this system presents a significant advancement in human–drone interaction, several limitations remain. The gesture recognition system remains susceptible to improper hand positioning, lighting variations, and occlusion. Additionally, the obstacle avoidance system currently lacks adaptive decision-making capabilities and does not incorporate learning from prior interactions. To better evaluate and interpret the system's performance, we acknowledge the following potential threats to validity:

- Internal Validity—The gesture recognition and drone control were tested under controlled indoor conditions with stable lighting and minimal distractions. Performance may decline in less controlled, real-world environments where lighting, background activity, or camera positioning vary significantly.
- Construct Validity—The predefined gestures selected for this study may not be universally intuitive for all users, potentially influencing usability outcomes. Furthermore, users' hand size, motion speed, and articulation could impact gesture recognition reliability.
- External Validity—The findings are based on a specific hardware setup (Crazyflie 2.1, MediaPipe Hands, and AI-Deck). Results may not generalize to other drone models or hardware configurations without significant reengineering.
- Reliability Threats—System performance may degrade over prolonged use due to sensor drift, thermal effects, or battery limitations. Additionally, the gesture classification system may face reduced robustness in multi-user settings or when exposed to unintended hand movements.

The paper aims to mitigate these threats in future work through more diverse testing environments, user studies across broader demographics, and hardware-agnostic system design to ensure greater generalizability and robustness. Specifically, future research could explore deep learning models for more robust gesture recognition, sensor fusion techniques for improved obstacle detection, and adaptive control strategies to enhance user autonomy while maintaining safety [42,43]. Also, expanding the 3D mapping system with SLAM integration could enable more detailed environmental awareness, improving applications in multi-drone coordination and autonomous exploration [44]. By addressing these challenges,

future modifications of this system could improve accuracy, usability, and adaptability, making drone technology more accessible to a broader range of users and applications. Furthermore, another promising research direction is the application of hand gesture control to robotic swarm systems and very large-scale robotic systems [45]. Finally, it is worth noting that to improve the gesture classification rate, we have implemented gesture classification using the support vector machine (SVM) method, rather than the method proposed in Section 2.4. The results and findings will be reported in our future paper.

5. Conclusions

This research demonstrated a gesture-controlled drone system integrating obstacle avoidance, 3D mapping, and drone FPV streaming via the AI-Deck, offering an intuitive and accessible approach to drone navigation. The combination of hand gesture recognition and autonomous safety mechanisms reduces reliance on traditional controllers, making drone operation more approachable, particularly for beginners. While limitations such as gesture recognition sensitivity, video streaming quality, and processing delays were observed, the system successfully provided a safer and more interactive user experience. Future improvements could further refine usability, responsiveness, and reliability, including two camera views for better gesture recognition, a new camera module for higher-quality streaming, and further code optimization. Also, expanding the obstacle avoidance system to detect and respond to moving objects would enhance the drone's adaptability, opening pathways for use in unpredictable spaces.

Looking ahead, the system could be scaled to support multi-drone coordination, enabling collaborative missions in larger or more complex spaces. Moreover, generalizing the platform for outdoor use with improved localization, such as integrating GPS or SLAM, could extend its utility to critical applications like search and rescue, environmental monitoring, or infrastructure inspection. Altogether, this multifunctional system represents a meaningful step toward safer, more intuitive human–drone interaction and paves the way for broader adoption of drone technology in education, training, and mission applications.

Supplementary Materials: The following supporting materials are available: Video S1: Obstacle avoidance demonstration https://youtu.be/K5psfzL2qEo?si=YkmpECxuSVcb_bBJ (accessed on 22 June 2025); Video S2: Integrated system demonstration https://youtu.be/L6OvuF5S8QI?si=JP9HjhYPmkpfdgq6 (accessed on 22 June 2025).

Author Contributions: Conceptualization, B.T., M.A., P.H. and P.Z.; methodology, B.T., M.A., P.H. and X.G.; software, B.T., M.A., P.H. and X.G.; validation, B.T., M.A. and P.H.; formal analysis, B.T. and P.Z.; investigation, B.T. and M.A.; resources, P.Z.; data curation, B.T., M.A. and P.H.; writing—original draft preparation, B.T.; writing—review and editing, B.T., X.G., P.Z. and H.M.; visualization, B.T., M.A. and P.H.; supervision, P.Z.; project administration, P.Z. and H.M.; funding acquisition, H.M. and P.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Defense Advanced Research Projects Agency (DARPA)-Grant #000825, the NASA West Virginia Space Grant Consortium, NASA Agreement #80NSCC20M0055, and NASA Established Program to Stimulate Competitive Research, Grant #80NSSC22M0027.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original data presented in this study are included in the article and openly available in Gesture-based-Drone-Navigation at https://github.com/PING-ResearchLab/Gesture-based-Drone-Navigation.git (accessed on 22 June 2025).

Acknowledgments: The authors thank PiNG Lab and Marshall University for their support.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CMC	Carpometacarpal
DIP	Distal Interphalangeal
FOV	Field of View
FPV	First-Person View
IMU	Inertial Measurement Unit
MCP	Metacarpophalangeal
PA	Power Amplifier
PIP	Proximal Interphalangeal
SLAM	Simultaneous Localization and Mapping
ToF	Time-of-Flight
UAV	Unmanned Aerial Vehicle

References

1. Amicone, D.; Cannas, A.; Marci, A.; Tortora, G. A smart capsule equipped with artificial intelligence for autonomous delivery of medical material through drones. *Appl. Sci.* **2021**, *11*, 7976. [CrossRef]
2. Hu, D.; Li, S.; Du, J.; Cai, J. Automating building damage reconnaissance to optimize drone mission planning for disaster response. *J. Comput. Civ. Eng.* **2023**, *37*, 04023006. [CrossRef]
3. Nooralishahi, P.; López, F.; Maldague, X.P. Drone-enabled multimodal platform for inspection of industrial components. *IEEE Access* **2022**, *10*, 41429–41443. [CrossRef]
4. Nwaogu, J.M.; Yang, Y.; Chan, A.P.; Wang, X. Enhancing drone operator competency within the construction industry: Assessing training needs and roadmap for skill development. *Buildings* **2024**, *14*, 1153. [CrossRef]
5. Tezza, D.; Laesker, D.; Andujar, M. The learning experience of becoming a FPV drone pilot. In Proceedings of the 2021 ACM/IEEE International Conference on Human-Robot Interaction, Boulder, CO, USA, 8–11 March 2021.
6. Lawrence, I.D.; Pavitra, A.R.R. Voice-controlled drones for smart city applications. In *Sustainable Innovation for Industry 6.0*; IGI Global: Hershey, PA, USA, 2024; pp. 162–177.
7. Shin, S.-Y.; Kang, Y.-W.; Kim, Y.-G. Hand gesture-based wearable human-drone interface for intuitive movement control. In Proceedings of the 2019 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 11–13 January 2019.
8. Naseer, F.; Ullah, G.; Siddiqui, M.A.; Khan, M.J.; Hong, K.-S.; Naseer, N. Deep learning-based unmanned aerial vehicle control with hand gesture and computer vision. In Proceedings of the 2022 13th Asian Control Conference (ASCC), Jeju Island, Republic of Korea, 4–7 May 2022.
9. Google. Hand Landmarks Detection Guide. Available online: https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker (accessed on 4 May 2025).
10. Hayat, A.; Li, C.H.; Prakoso, N.; Zheng, R.; Wyawahare, A.; Wu, J. Gesture and Body Position Control for Lightweight Drones Using Remote Machine Learning Framework. In *Innovations in Electrical and Electronics Engineering, Proceedings of the ICEEE 2024, Marmaris, Turkey, 22–24 April 2024*; Springer: Singapore, 2025.
11. Bae, S.; Park, H.-S. Development of immersive virtual reality-based hand rehabilitation system using a gesture-controlled rhythm game with vibrotactile feedback: An fNIRS pilot study. *IEEE Trans. Neural Syst. Rehabil. Eng.* **2023**, *31*, 3732–3743. [CrossRef]
12. Manikanavar, A.R.; Shirol, S.B. Gesture controlled assistive device for deaf, dumb and blind people using Raspberry-Pi. In Proceedings of the 2022 International Conference on Smart Technologies and Systems for Next Generation Computing (ICSTSN), Virtual, 25–26 March 2022.
13. Paterson, J.; Aldabbagh, A. Gesture-controlled robotic arm utilizing opencv. In Proceedings of the 2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 11–13 June 2021.
14. Seidu, I.; Lawal, J.O. Personalized Drone Interaction: Adaptive Hand Gesture Control with Facial Authentication. *Int. J. Sci. Res. Sci. Eng. Technol.* **2024**, *11*, 43–60. [CrossRef]
15. Alanezi, M.A.; Haruna, Z.; Sha'aban, Y.A.; Bouchekara, H.R.; Nahas, M.; Shahriar, M.S. Obstacle avoidance-based autonomous navigation of a quadrotor system. *Drones* **2022**, *6*, 288. [CrossRef]
16. Xue, Z.; Gonsalves, T. Vision based drone obstacle avoidance by deep reinforcement learning. *AI* **2021**, *2*, 366–380. [CrossRef]
17. Ostovar, I. Nano-Drones: Enabling Indoor Collision Avoidance with a Miniaturized Multi-Zone Time of Flight Sensor. Master's Thesis, Politecnico di Torino, Torino, Italy, 2022.

18. Courtois, H.; Aouf, N.; Ahiska, K.; Cecotti, M. OAST: Obstacle avoidance system for teleoperation of UAVs. *IEEE Trans. Hum.-Mach. Syst.* **2022**, *52*, 157–168. [CrossRef]
19. Bouwmeester, R.J.; Paredes-Vallés, F.; De Croon, G.C. Nanoflownet: Real-time dense optical flow on a nano quadcopter. In Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA), London, UK, 29 May–2 June 2023.
20. Khoza, N.; Owolawi, P.; Malele, V. Drone Gesture Control using OpenCV and Tello. In Proceedings of the 2024 Conference on Information Communications Technology and Society (ICTAS), Durban, South Africa, 7–8 March 2024.
21. Zhang, N.; Nex, F.; Vosselman, G.; Kerle, N. End-to-end nano-drone obstacle avoidance for indoor exploration. *Drones* **2024**, *8*, 33. [CrossRef]
22. Telli, K.; Kraa, O.; Himeur, Y.; Ouamane, A.; Boumehraz, M.; Atalla, S.; Mansoor, W. A comprehensive review of recent research trends on unmanned aerial vehicles (UAVs). *Systems* **2023**, *11*, 400. [CrossRef]
23. Bitcraze. Crazyflie 2.1. Available online: https://www.bitcraze.io/products/old-products/crazyflie-2-1/ (accessed on 18 April 2025).
24. Bitcraze. Crazyradio PA. Available online: https://www.bitcraze.io/products/crazyradio-pa/ (accessed on 18 April 2025).
25. Bitcraze. Flow Deck v2. Available online: https://store.bitcraze.io/products/flow-deck-v2 (accessed on 14 April 2025).
26. Bitcraze. Multi-Ranger Deck. Available online: https://store.bitcraze.io/products/multi-ranger-deck (accessed on 14 April 2025).
27. Bitcraze. AI-Deck 1.1. Available online: https://store.bitcraze.io/products/ai-deck-1-1 (accessed on 14 April 2025).
28. Yun, G.; Kwak, H.; Kim, D.H. Single-Handed Gesture Recognition with RGB Camera for Drone Motion Control. *Appl. Sci.* **2024**, *14*, 10230. [CrossRef]
29. Lee, J.-W.; Yu, K.-H. Wearable drone controller: Machine learning-based hand gesture recognition and vibrotactile feedback. *Sensors* **2023**, *23*, 2666. [CrossRef] [PubMed]
30. Natarajan, K.; Nguyen, T.-H.D.; Mete, M. Hand gesture controlled drones: An open source library. In Proceedings of the 2018 1st International Conference on Data Intelligence and Security (ICDIS), South Padre Island, TX, USA, 8–10 April 2018.
31. Begum, T.; Haque, I.; Keselj, V. Deep learning models for gesture-controlled drone operation. In Proceedings of the 2020 16th International Conference on Network and Service Management (CNSM), Virtual, 2–6 November 2020.
32. Backman, K.; Kulić, D.; Chung, H. Reinforcement learning for shared autonomy drone landings. *Auton. Robot.* **2023**, *47*, 1419–1438. [CrossRef]
33. Schwalb, J.; Menon, V.; Tenhundfeld, N.; Weger, K.; Mesmer, B.; Gholston, S. A study of drone-based AI for enhanced human-AI trust and informed decision making in human-AI interactive virtual environments. In Proceedings of the 2022 IEEE 3rd International Conference on Human-Machine Systems (ICHMS), Orlando, FL, USA, 7–19 November 2022.
34. Khaksar, S.; Checker, L.; Borazjan, B.; Murray, I. Design and Evaluation of an Alternative Control for a Quad-Rotor Drone Using Hand-Gesture Recognition. *Sensors* **2023**, *23*, 5462. [CrossRef]
35. Moffatt, A.; Platt, E.; Mondragon, B.; Kwok, A.; Uryeu, D.; Bhandari, S. Obstacle detection and avoidance system for small UAVs using a LiDAR. In Proceedings of the 2020 International Conference on Unmanned Aircraft Systems (ICUAS), Athens, Greece, 1–4 September 2020.
36. Karam, S.; Nex, F.; Chidura, B.T.; Kerle, N. Microdrone-based indoor mapping with graph slam. *Drones* **2022**, *6*, 352. [CrossRef]
37. Krul, S.; Pantos, C.; Frangulea, M.; Valente, J. Visual SLAM for indoor livestock and farming using a small drone with a monocular camera: A feasibility study. *Drones* **2021**, *5*, 41. [CrossRef]
38. Sacoto-Martins, R.; Madeira, J.; Matos-Carvalho, J.P.; Azevedo, F.; Campos, L.M. Multi-purpose low latency streaming using unmanned aerial vehicles. In Proceedings of the 2020 12th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP), Porto, Portugal, 20–22 July 2020.
39. Truong, N.Q.; Nguyen, P.H.; Nam, S.H.; Park, K.R. Deep learning-based super-resolution reconstruction and marker detection for drone landing. *IEEE Access* **2019**, *7*, 61639–61655. [CrossRef]
40. Lee, S.H. Real-time edge computing on multi-processes and multi-threading architectures for deep learning applications. *Microprocess. Microsyst.* **2022**, *92*, 104554. [CrossRef]
41. Elbamby, M.S.; Perfecto, C.; Bennis, M.; Doppler, K. Toward low-latency and ultra-reliable virtual reality. *IEEE Netw.* **2018**, *32*, 78–84. [CrossRef]
42. Hu, B.; Wang, J. Deep learning based hand gesture recognition and UAV flight controls. *Int. J. Autom. Comput.* **2020**, *17*, 17–29. [CrossRef]
43. Lyu, H. Detect and avoid system based on multi sensor fusion for UAV. In Proceedings of the 2018 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Republic of Korea, 17–19 October 2018.

44. Habib, Y. Monocular SLAM Densification for 3D Mapping and Autonomous Drone Navigation. Ph.D. Thesis, Ecole Nationale Supérieure Mines-Télécom Atlantique, Nantes, France, 2024.
45. Gao, J.; Lee, J.; Zhou, Y.; Hu, Y.; Liu, C.; Zhu, P. SwarmCVT: Centroidal Voronoi Tessellation-Based Path Planning for Very-Large-Scale Robotics. *arXiv* **2024**, arXiv:2410.02510.