MJI
Media Jurnal Informatika
Vol. 18, No. 1, June 2026
p-issn : 2088-2114, e-issn : 2477-2542

Real-Time Webcam-Based Hand Gesture Recognition with Face Authentication for 3D Drone Simulation in Godot Engine

Muhammad Rizqi Sholahuddin [a,1,*], Siti Dwi Setiarini [a,2], Ardhian Ekawijana [a,3] , Muhammad Samudera Bagja [a,4], Firas Atqiya [b,5]

[a] Department of Computer Engineering and Informatics, Politeknik Negeri Bandung, Kab. Bandung Barat 40559, Indonesia
[b] Department of Computer Science, Math and Science Faculty, Universitas Padjadjaran, Bandung 45363, Indonesia
[1] muhammad.rizqi@polban.ac.id*; [2] siti.dwi@polban.ac.id; [3] ardhian.ekawijana@polban.ac.id ;
[4] muhammad.samudera.tif423@polban.ac.id; [5] firas.atqiya@unpad.ac.id
* corresponding author

ARTICLE INFO

Article history
Received 2026-06-28
Revised 2026-06-29
Accepted 2026-06-29

Keywords
hand gesture recognition
Godot Engine
drone simulation
face authentication
MediaPipe

ABSTRACT

Most hand gesture control systems for drones depend on specialized hardware such as Leap Motion or Kinect, which raises the cost barrier for educational institutions in developing countries. Integrating face authentication within the same low-cost pipeline remains under-explored. This study develops a real-time, webcam-based system that combines Google MediaPipe hand tracking with facial authentication and a Godot Engine 4.3 3D drone simulation for authenticated, responsive gesture control. A finger-counting algorithm classifies eight gestures across two hands. The left hand drives horizontal motion (forward, backward, left, right) and the right hand drives altitude and yaw (up, down, rotate left, rotate right). Commands travel over UDP to Godot, where a receiver node translates each packet into a native input action. Face authentication uses dlib and the face_recognition library with a 60-frame login counter. All metrics were collected under a fixed condition (normal lighting 300–500 lux, 0.8 m, one subject). The system achieved 100% gesture accuracy across 160 trials, 35.6 FPS pipeline throughput, 0.33 ms one-way UDP latency with 0% packet loss, and 23.9 ms end-to-end gesture-to-drone latency. Face authentication scored 100% recognition with 0% FRR and 19.0% FAR against an unregistered face at the default 0.6 tolerance. A standard-webcam pipeline built entirely from open-source components can deliver responsive, authenticated gesture control for interactive drone simulation, though the single-subject evaluation is an upper bound requiring multi-subject validation. However, the 100% accuracy represents an upper bound as evaluation was limited to a single subject under controlled lighting (300–500 lux) and a fixed distance (0.8 m), requiring further validation across diverse users and environments

1. Introduction

Human-computer interaction has moved well past keyboards and mice. Hand gesture recognition is one of the more practical contactless modalities, and advances in computer vision and deep learning have made real-time performance achievable. Methods now span from classic machine learning classifiers to Vision Transformers reaching 99.44% accuracy on benchmark datasets [1], [2].

Drones have proliferated across agriculture, surveillance, delivery, and entertainment [3], and training operators on real hardware is costly and risky. Simulators absorb that risk by letting pilots learn in a virtual environment [4]. Pairing gesture control with a simulator recreates the feel of operating a real drone without purchasing one. In their survey of human-drone interaction, Tezza and Andujar [5] found that 86% of users gravitate toward gestures before speech or brain-computer interfaces.

Most existing gesture-based drone systems depend on dedicated hardware. Leap Motion achieves over 95% accuracy but requires a USB peripheral priced around $100 [6]. Microsoft Kinect adds depth sensing but has been discontinued for consumers and demands substantial computational resources. Both raise the cost barrier, particularly for educational institutions in developing countries [7]. Web-based and ROS-based demonstrations built on Leap Motion or depth cameras inherit the same hardware dependency [8], [9]. Natarajan et al. [10] released an open-source Haar-feature library that surpasses 90% accuracy, but is limited to five gestures and a 3-foot operating range.

Lightweight ML frameworks have opened a cheaper path. Google MediaPipe tracks 21 landmarks per hand from a plain RGB feed and runs on a CPU without a GPU [11]. Sánchez-Brizuela et al. [12] reached a mean IoU of 0.869 at 90 FPS with MediaPipe hand segmentation on a standard CPU. Singh et al. [13] combined pose, face, and hand landmarks with a Random Forest classifier to reach 96.78% on real-time action detection. The authors' earlier work [14] used MediaPipe for presentation control, which serves as the starting point for the drone application in this paper.

What remains under-explored is tight integration with game engines. Kassab et al. [15] built gesture-based human-UAV interaction with Darknet-19 and Tiny-YOLOv2, but for real flight rather than simulation. Chen et al. [16] reached 99.5% accuracy in real flight by fusing skeleton and RGB recognition, at the cost of multiple sensors and heavy computation. Akagi et al. [17] controlled a UAV fleet from a hand-worn IMU at 95% for 10 gestures, but again the hardware limits accessibility. Few systems integrate face authentication with gesture control in a single pipeline, even though shared training rooms call for user identification. Imran et al.[18] reached 90.91% with ArcFace embeddings and temporal voting in surveillance video. Sikarwar et al. [19] reached 96.76% with MTCNN and FaceNet. Fayaz et al. [20] validated the dlib HOG detector with 128-dimensional embeddings at 99.38% reliability on the LFW benchmark.

The gap is a complete, low-cost system that (a) takes input from a standard webcam, (b) supports enough gestures for six-degree-of-freedom drone control using both hands, (c) integrates with a modern open-source engine for 3D rendering, (d) authenticates the user without extra biometric hardware, and (e) maintains a clean input architecture where the drone controller is decoupled from any one input source. Unity and Unreal have been used for drone simulation [4], [21], [22] but open-source engines like Godot receive little attention in the HDI literature despite their no-royalty licensing model.

This study presents a real-time, webcam-based hand gesture system built on MediaPipe that communicates with Godot 4.3 over UDP for drone simulation. Eight gestures are distributed across two hands: the left hand handles forward, backward, left, right, and the right hand handles up, down, rotate left, rotate right. A finger-counting rule classifies each gesture from MediaPipe landmarks. A dlib and face_recognition module authenticates the user first. Godot receives each UDP packet in a dedicated receiver node and fires the matching input action, so the drone script remains input-source agnostic. All components are measured under a controlled protocol with instrumented timing.

2. Method

This study adopts the Design Science Research (DSR) framework to develop and evaluate the proposed system as a functional IT artifact. The research is structured into iterative modules face recognition, hand gesture detection, UDP communication, and Godot integration each undergoing independent unit testing before full pipeline integration. The evaluation follows a rigorous quantitative approach: gesture accuracy is determined through controlled trials using majority-vote decisions against ground-truth labels; system performance (frame rate, latency, and end-to-end response) is measured via instrumented code with high-resolution timestamps; and security robustness is evaluated using False Acceptance Rate (FAR) and False Rejection Rate (FRR) at the operational tolerance.

2.1 Type and Approach of Research

This research adopts a Design Science Research (DSR) framework with a quantitative experimental approach. The DSR framework is selected as it focuses on the creation and evaluation of innovative IT artifacts in this case, a real-time gesture control and authentication pipeline to solve practical problems such as the high cost of specialized HCI hardware. The study develops a complete hardware-software pipeline and evaluates it under controlled laboratory conditions. The experimental approach is appropriate because the research questions concern measured system performance (accuracy, latency, throughput) rather than

subjective user experience, and because each independent variable (gesture type, network condition, authentication state) can be isolated during testing.

2.2 Object and Scope of Research

The object of study is a real-time hand gesture recognition and face authentication pipeline connected to a Godot Engine 4.3 3D drone simulation over UDP. Godot was selected as the simulation environment due to its open-source, royalty-free licensing model, which aligns with the goal of democratizing educational tools compared to proprietary engines like Unity or Unreal.

The scope is limited to finger-counting gestures, a single indoor lighting condition (300 to 500 lux), and a fixed camera distance (0.8 m). The system processes a single video stream from a standard USB webcam at 720p resolution. To ensure reproducibility, all performance tests were conducted on a system equipped with an AMD Ryzen 7 5800H, 32GB RAM, and running Windows 11.

The system utilizes UDP for communication between Python and Godot to minimize latency, which is critical for real-time interactivity, although it is acknowledged that UDP does not guarantee packet delivery. Evaluation data presented in this study, such as the 100% gesture accuracy, represents an upper bound as it was collected from a single test subject under optimized conditions

2.3 Data Collection Techniques

All performance data were collected through a dedicated Python evaluation suite (evaluate_paper.py) with five subcommands covering each measurement type. Gesture accuracy was measured through 160 controlled trials (8 gestures × 20 repetitions) with a 2-second hold window and majority-vote decision. Pipeline timing was recorded with Python's time.perf_counter() over 300 frames, breaking the pipeline into capture, detection, and classification stages. UDP latency was measured through a loopback test (500 packets at 30 pps) and an end-to-end echo test (200 packets) through Godot's eval_echo.gd node. Face recognition performance was measured through 100 authorized frames and 100 impostor frames at the default distance tolerance of 0.6. All raw data were written to JSON and CSV for reproducible analysis.

2.4 Tools and Materials Used

Table 1 lists the tools and materials used in the study. The software stack is built entirely on open-source libraries: Python 3.10+ with MediaPipe 0.10.21 for hand tracking, OpenCV 4.8.1.78 for video capture and display, and the face_recognition library with dlib for face authentication. The game engine is Godot 4.3 with the Forward Plus renderer, communicating with Python over raw UDP sockets with JSON payloads.

Table 1. Lists the tools and materials used in the study.

Category	Tool / Material	Version / Specification
Programming language	Python	3.10+
Hand tracking	MediaPipe	0.10.21
Computer vision	OpenCV (cv2)	4.8.1.78
Face recognition	face_recognition	≥ 1.3.0
Face landmark detection	dlib	≥ 19.22.0
Numerical computing	NumPy	1.24.3
Game engine	Godot Engine	4.3 (Forward Plus)
Communication	UDP socket (Python / PacketPeerUDP)	-
Hardware	Standard USB webcam	720p

2.5 Research Procedures or Stages

Development proceeded through seven stages. Stage 1 Requirements and gesture design: Eight gestures were chosen for finger-counting feasibility and user comfort. The left hand took planar motion (Forward, Backward, Left, Right) and the right hand took altitude and yaw (Up, Down, Rotate Left, Rotate Right). Stage 2 Face authentication: Registration captures 5 facial encodings per user at 1-second intervals. Login requires 60 consecutive per-frame matches at the default 0.6 distance tolerance. The match logic and login counter run entirely in Python; Godot only receives the final login result over a shared-secret token channel. Stage 3 Hand gesture recognition: MediaPipe Hands returns 21 landmarks per hand. A finger is classified as extended when its fingertip landmark is above its PIP joint. The gesture label is determined by a lookup of finger count combined with which specific fingers are raised. Stage 4 UDP communication: Two independent channels were implemented. The gesture channel (port 9999) sends JSON payloads with 100 ms

rate limiting. The video channel (port 5000) multiplexes JPEG frames (fragmented at 60 KB) and small JSON status messages for the login subsystem. Stage 5 Godot integration: A GestureReceiver node listens on UDP port 9999 and calls Input.action_press() and Input.action_release(). The drone controller reads Input.is_action_pressed() identically for any input source. Stage 6 Integration testing: The full pipeline was connected and tested end to end. Stage 7 Performance evaluation: All metrics were collected through a unified evaluation script with dedicated subcommands.

Figure 1 illustrates the system architecture, showing the dual-channel UDP design and the scene flow from Main_UI through Login to WorldEnv.

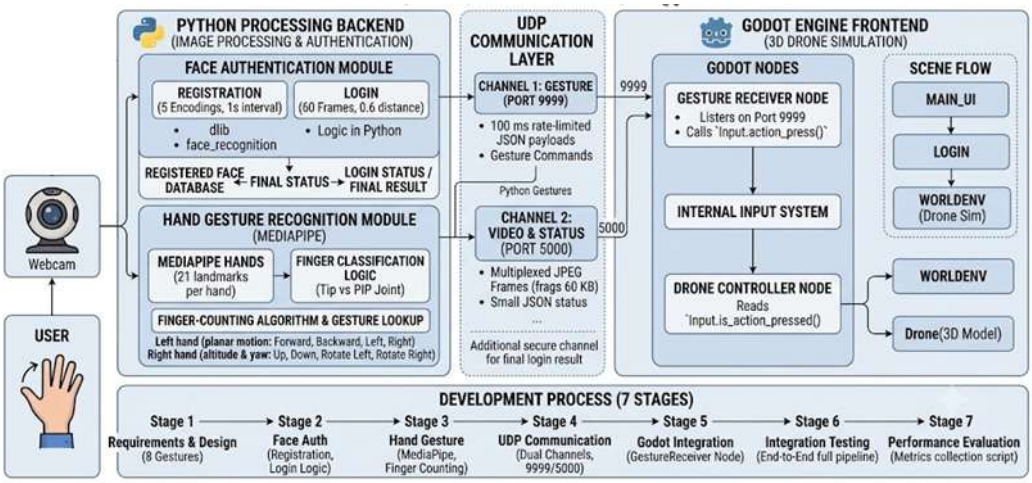

Fig. 1 System architecture, dual-channel UDP design, and scene flow.

2.6 Data Analysis Techniques

The evaluation of the system artifact follows the quantitative metrics established in the DSR framework. System performance is analysed using the formalisms described in the following subsections.

2.6.1 Recognition Accuracy

Gesture recognition accuracy (Acc) is the proportion of correctly classified trials relative to the total number of trials (N):

$$Acc = \frac{\sum_{i=1}^{N} |f(g_i) - \hat{g}_i|}{N} \times 100\%$$

Here, $f(g_i)$ is the predicted gesture label obtained through majority voting and $\hat{g}_i$ is the ground-truth label

2.6.2 Security Metrics (FAR and FRR)

Face authentication robustness is evaluated through the False Acceptance Rate (FAR) and the False Rejection Rate (FRR). FAR is the probability that the system incorrectly authorises an unregistered user:

$$FAR = \frac{\text{Total False Acceptances}}{\text{Total Unauthorized Attempts}} \times 100\%$$

FRR is the probability that the system incorrectly rejects a registered user:

$$FRR = \frac{\text{Total False Rejections}}{\text{Total Authorized Attempts}} \times 100\%$$

2.6.3 Temporal Performance (Latency)

End-to-End Latency (L_{ea2e}) measures the total elapsed time from image capture (T_{CAP}) to the execution of the corresponding input action in Godot ($T_{exe}C$):

$$L_{e2e} = T_{exec} - T_{cap}$$

2.6.4 Network Reliability

Packet loss is the percentage of UDP packets that fail to reach the receiver over the course of a fixed trial:

$$\text{Packet Loss} = \frac{P_{sent} - P_{received}}{P_{sent}} \times 100\%$$

3. Results and Discussion

3.1 Presentation of Research Results

All measurements come from one fixed condition: one subject, normal indoor lighting (300 to 500 lux), approximately 0.8 m from the camera.

Table 2 presents the per-gesture recognition accuracy over 160 trials. Each gesture was evaluated in isolation with a hand and gesture filter active, a 2-second hold window, and a per-trial majority vote. Because each run only classifies one intended gesture, the table reports per-gesture accuracies rather than a combined confusion matrix. No misclassifications were recorded in any of the 160 trials, yielding 100% accuracy across all eight gestures.

Table 2. Gesture recognition accuracy per gesture

Gesture	Hand	Correct	Incorrect	Accuracy
FORWARD	Left	20	0	100.0%
BACKWARD	Left	20	0	100.0%
LEFT	Left	20	0	100.0%
RIGHT	Left	20	0	100.0%
UP	Right	20	0	100.0%
DOWN	Right	20	0	100.0%
ROTATE_LEFT	Right	20	0	100.0%
ROTATE_RIGHT	Right	20	0	100.0%
Total / Average		160	0	100.0%

Table 3 collects the system-level and latency metrics across all measurement categories. These include pipeline throughput in frames per second, one-way UDP latency and packet loss for both the loopback and Godot echo tests, end-to-end gesture-to-drone latency, and face recognition accuracy with its corresponding False Acceptance Rate (FAR) and False Rejection Rate (FRR) at the operational tolerance of 0.6. The table also lists the system parameters used during evaluation, including the gesture timeout and UDP rate-limiting interval.

Table 3. System performance metrics

Metric	Value
Average FPS (gesture detection pipeline)	35.6 fps
Average UDP latency (one-way, loopback)	0.33 ms
UDP packet loss (loopback, 500 packets)	0.00%
End-to-end latency (gesture to drone)	23.9 ms
UDP packet loss (echo test, Godot, 200 packets)	0.00%
Face recognition accuracy (registered user)	100.0%
Face FAR (cross-subject, tolerance 0.6)	19.0%
Face FRR (tolerance 0.6)	0.0%
Gesture timeout (release after no packet)	200 ms
UDP rate limiting interval	100 ms

Table 4 breaks the 23.9 ms end-to-end latency into its three constituent components: MediaPipe detection time, UDP one-way transit time, and Godot ingest time (parse plus input action injection).

Table 4. End-to-end latency breakdown

Component	Average (ms)	Share
MediaPipe detection	22.16	93%
UDP transmission (one-way)	1.71	7%
Godot ingest (parse + input action)	0.05	<1%
End-to-end (sum)	**23.9**	**100%**

Table 5 provides per-stage timing statistics for the gesture detection pipeline over 300 consecutive frames. The pipeline is broken into three stages: capture (webcam read), detection (MediaPipe Hands processing), and classification (finger-counting rule).

Table 5. Detailed pipeline stage timing (300 frames)

Stage	Average (ms)	Min	Max	Median	StdDev
Capture	5.65	1.37	507.40	2.25	29.52
Detection (MediaPipe)	22.16	20.80	27.38	21.83	1.08
Classification (gesture)	0.001	0.000	0.003	0.000	0.000
Pipeline total	28.07	22.55	533.14	24.62	29.71

Table 6 reports the UDP loopback latency distribution over 500 packets sent at approximately 30 packets per second on localhost (port 9990). The average one-way latency is 0.33 ms with a median of 0.30 ms, indicating a symmetric distribution. The 95th percentile is 0.57 ms and the 99th percentile is 0.80 ms, with a maximum of 1.50 ms and a standard deviation of 0.14 ms. The narrow distribution confirms that local UDP transport is highly consistent and effectively loss-free.

Table 6. UDP loopback latency distribution (500 packets)

Statistic	Value (ms)
Average	0.33
Median	0.30
Minimum	0.12
Maximum	1.50
P95	0.57
P99	0.80
StdDev	0.14

Table 7 places the proposed system alongside existing work in the hand gesture control literature. The comparison spans dedicated hardware (Leap Motion, Kinect), CNN-based approaches (Darknet-19, CNN glove), and webcam-based rule-based methods (Haar cascade, MediaPipe SLR). Among the webcam-based, rule-based group, the proposed system supports the largest gesture vocabulary (eight gestures) and the lowest measured end-to-end latency (23.9 ms). It is also the only entry that integrates face authentication and game-engine control into a single open-source pipeline.

Table 7. Comparison with existing systems

System	Input device	Method	Gestures	Accuracy	Latency
Leap Motion [6]	Leap Motion	IR depth sensing	10+	95.5%	~30 ms
Kinect-based	Kinect	Depth + skeletal	6–8	93.2%	~80 ms
Darknet-19 HUI [15]	Webcam	Darknet-19	10	99.0%	28 fps
CNN glove	Color glove	CNN	6	97.1%	~60 ms
Haar cascade [10]	Webcam	Haar + AdaBoost	5	>90%	~50 ms
Adaptive ML [23]	Webcam	LSTM + adaptive	5–6	~90%	~50 ms
MediaPipe SLR	Webcam	MediaPipe rule	5	89.3%	~50 ms
MediaPipe pres. [14]	Webcam	Finger counting	4	~90%	~50 ms
Proposed system	Webcam	MediaPipe + finger counting	8	100%*	23.9 ms

Figure 2 shows the 21 MediaPipe hand landmarks used by the finger-counting classifier, along with a photographic reference of all eight gestures, arranges the gestures in a 2×4 grid organized by hand: the left hand controls planar motion (Backward,Forward, Right,Left) and the right hand controls altitude and yaw (Down, UP, Rotate Left, Rotate Right), with each cell annotated by hand, finger count, and action label.

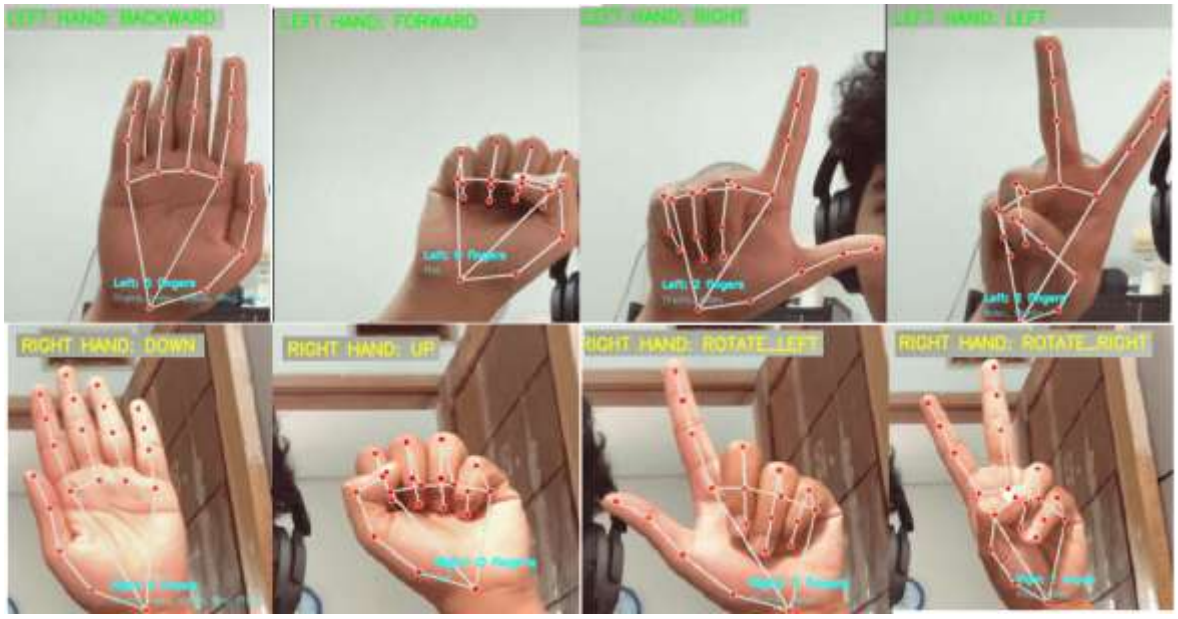

Fig. 2 Hand gesture vocabulary and corresponding drone action labels.

Figure 3 presents a screenshot of the Godot Engine simulation environment during development and testing. On the left, the Godot Engine renders a 3D DJI Tello drone navigating a corridor, actively responding to live input. Meanwhile, the right panel displays a custom Python application using Google MediaPipe to track the user's hand landmarks shown as red dots and white lines successfully interpreting a right-hand gesture as a 'ROTATE_LEFT' command, which is confirmed by the yellow overlay text. The console logs confirm real-time UDP communication, showing the transmission of 'UP' and 'ROTATE_LEFT' commands from the vision pipeline to the simulation environment via port 9999.

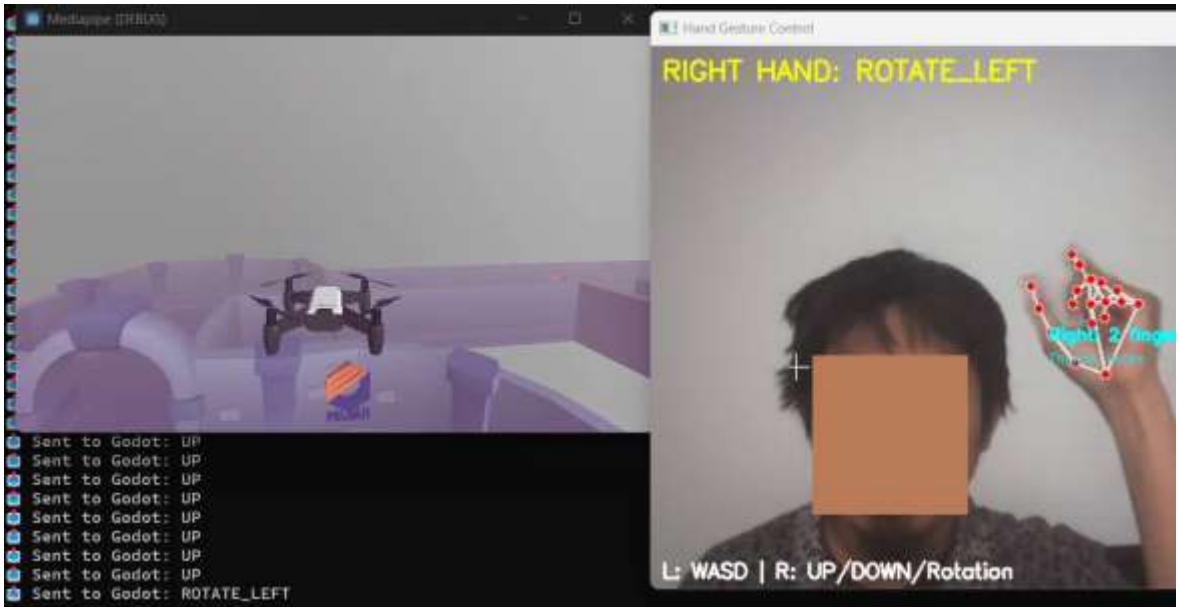

Fig. 3 Simulation interface and real-time gesture interpretation in Godot.

Figure 4 shows the WorldEnv scene running with the DJI Tello 3D drone model, where the GestureReceiver node is visible in the scene tree panel and the debug overlay displays incoming gesture packets in real time.

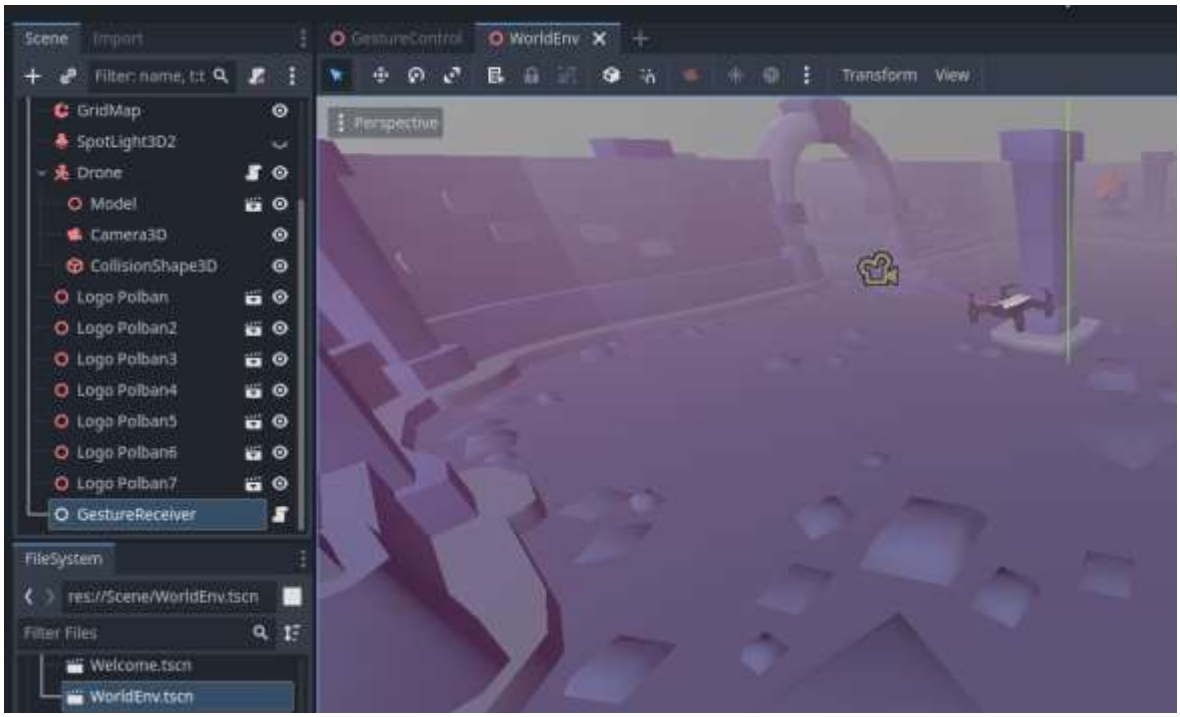

Fig. 4 Drone simulation scene tree and node hierarchy.

Figure 5 shows the Godot Engine screenshot during the UDP streaming demo for the face authentication login process. The Login scene displays the live webcam feed received over UDP port 5000, which multiplexes fragmented JPEG video frames alongside JSON login status messages prefixed with a shared-secret token ('FACELGN:'). The interface shows the user's face being captured and sent to the Python face recognition module, with the login progress indicator reflecting the 60-frame consecutive match counter. The LANJUT button remains disabled until a login_success status message is received, demonstrating the security boundary where the authentication logic executes entirely in Python and only the final result reaches the Godot client.

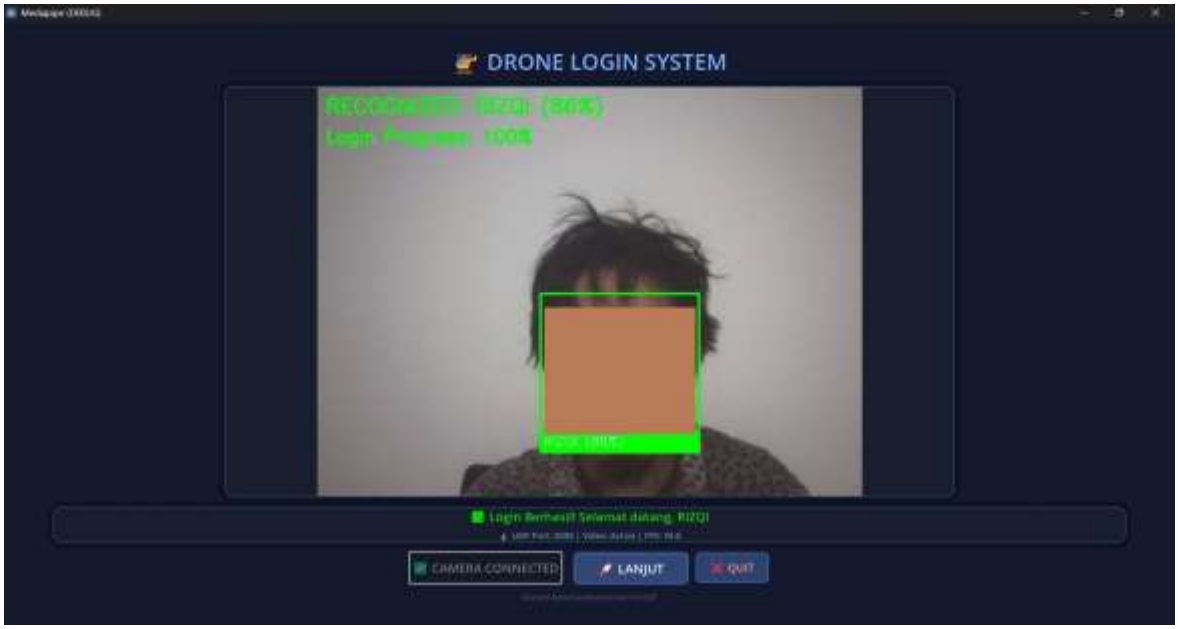

Fig. 5 Facial authentication interface and UDP video streaming demo.

3.2 Analysis of Findings

The system achieved 100% accuracy (160 of 160 trials) across all eight gestures under the fixed condition, including the two-finger combinations (Left, Right, Rotate Left, Rotate Right) that the literature flags as the most confusable. At the intended operating point normal lighting, close range, a rehearsed subject, and a per-trial majority vote over 2 seconds the rule-based finger-counting classifier is unambiguous for these eight shapes. This result sits above the 89% to 95% band reported for webcam-based MediaPipe with rule-based classification [10], [14] and matches the top of what dedicated hardware like Leap Motion achieves in a controlled setting[6]. It accomplishes this without training data or a GPU, on the same CPU path for which MediaPipe benchmarks report up to 90 FPS[12]. The caveat is that a perfect score from one subject under one condition is an upper bound, not a population estimate.

Among the webcam-based, rule-based group in Table 7, the proposed system supports the largest gesture vocabulary (8) and the lowest measured end-to-end latency (23.9 ms). The rule-based approach buys interpretability, zero training data, and an easily editable gesture set, though it cannot match the cross-condition robustness of trained deep models such as Darknet-19 (99.0% in [15]) or Vision Transformers (99.44% in [2]). Crucially, it is the only entry that integrates face authentication and game-engine control into a single open-source pipeline.

The 23.9 ms end-to-end latency is well inside the 100 ms threshold typically cited for real-time interactive systems [1]. Table 4 reveals that MediaPipe detection consumes the dominant share at 22.16 ms (93%), UDP transit contributes 1.71 ms (7%), and Godot ingest is negligible at 0.05 ms (<1%). The transport layer is not where latency hides; if the latency budget ever needs to shrink, the detection stage is the only component with meaningful headroom. For comparison, Zhao et al. [8] reported average control latencies of 57.82 ms indoors and 58.84 ms outdoors for a web-based Leap Motion drone over WiFi, so the localhost UDP design is clearly faster for single-machine operation. The isolated loopback test (Table 6) recorded 0.33 ms average one-way latency with 0% loss over 500 packets, and the Godot echo test confirmed 0% loss over 200 packets, establishing that local UDP transport is effectively loss-free.

The pipeline runs at 35.6 FPS (Table 5). Detection is the dominant cost and remains tightly distributed (mean 22.16 ms, standard deviation 1.08 ms), while classification is essentially free (mean 0.001 ms). The capture stage shows a long tail (mean 5.65 ms but maximum 507.40 ms, standard deviation 29.52 ms) due to occasional webcam buffer stalls. These outliers push the pipeline mean (28.07 ms) above the detection mean and constitute the primary source of frame-to-frame jitter rather than the recognition logic itself.

Face recognition scored 100% on the registered user with 0% FRR, confirming that the HOG detector and 128-dimensional dlib embeddings reliably match the enrolled identity at the operating distance. However, the cross-subject impostor test produced 19.0% FAR. This is substantially higher than the 2.1% reported for dlib-based pipelines on standardized benchmarks [20], and higher than MTCNN plus FaceNet (96.76% accuracy[19]) and ArcFace with temporal voting (90.91% [18]). Two factors likely contribute. First, the 0.6 tolerance is the library default and is known to favor convenience; tightening it to 0.5 or 0.4 would reduce FAR at the cost of increased FRR. Second, per-frame real-time matching lacks the offline batch optimization that benchmarks employ, so the HOG detector can occasionally lock onto a partial impostor face whose embedding happens to fall within the registered user's tolerance radius. Despite the elevated FAR, the result validates a critical architectural property: both face recognition and the 60-frame login decision execute entirely in Python, so Godot only ever receives a login_success string gated by a shared-secret token. The client cannot be tricked into granting access by manipulating the video feed.

3.3 Implications of the Results

A low-cost alternative to dedicated hardware. The system requires only a standard USB webcam (under $20) and a consumer PC capable of running Python and Godot. Compared to Leap Motion ($89–100) or depth cameras, the cost reduction is substantial, which directly benefits polytechnics and vocational schools operating with limited budgets, and aligns with the democratization objective articulated in HDI research[5].

Educational value. The pipeline spans computer vision, socket programming, and game engine development, enabling students to gain exposure to multiple domains within a single project. Every layer is open source (MediaPipe, dlib, Godot, Python), removing the licensing friction that can block adoption of Unity or Unreal in educational settings, and builds on the learning gains associated with game-engine-based environments[24].

Drone training. Eight gestures cover the primary axes of quadcopter flight: translation in all four horizontal directions, altitude control, and yaw rotation. The mapping is symmetric and intuitive a fist maps to forward propulsion (left hand) or ascent (right hand), while an open hand maps to the opposite direction.

Mairaj et al. [4]identify intuitive control as a prerequisite for an effective simulator, and the proposed vocabulary satisfies this criterion.

Reusable input architecture. The input-action pattern keeps the drone controller cleanly separated from the gesture layer. The controller reads the same Input.is_action_pressed() calls that a keyboard or gamepad would trigger, so swapping in an alternative input source requires no modification. The measured 0.05 ms Godot ingest cost confirms that this abstraction imposes no meaningful runtime penalty.

Security with a single camera. Face authentication reuses the same webcam that the gesture layer depends on, eliminating the need for a separate biometric device. The 19.0% cross-subject FAR indicates that the default tolerance favors convenience over strict security, but the tolerance parameter provides a direct and adjustable lever for tightening that tradeoff.

3.4 Limitations of the Study

1. One subject, one condition. Every data point comes from a single individual under a single setting (normal lighting, 0.8 m). The 100% gesture accuracy is an upper bound for a rehearsed user at the intended operating point, not a population estimate across hand sizes, skin tones, or room configurations. Multi-subject and multi-condition trials are the most urgent next step.

2. Isolated per-gesture testing. Each gesture was evaluated in its own filtered session, so the protocol never exercises cross-gesture confusion directly. A mixed protocol in which any gesture can appear at any time would produce a true confusion matrix and would almost certainly register lower accuracy.

3. No depth information. A monocular camera provides no true depth measurement. MediaPipe estimates pseudo-3D landmarks, but distance-dependent gestures are inherently unreliable with a single RGB sensor. Fagundes-Junior et al. [25] identify simulation-to-real transfer of depth-dependent policies as an open research problem.

4. Vocabulary ceiling. Finger counting imposes a hard ceiling on how many gestures remain separable. Five binary fingers give a theoretical 32 states per hand, but many combinations are anatomically awkward or visually ambiguous. Expanding beyond eight gestures requires additional discriminators such as wrist orientation or motion trajectory, as explored in the 3D-CNN work of Channayanamath et al.[26].

5. Lighting and distance not characterized. Only the intended operating condition was measured, so the study provides no quantitative data on performance under low light (<100 lux) or at extended distances. The literature [10], [19] clearly establishes that vision-based accuracy degrades in poor lighting, and this system should follow the same trend, but the specific degradation curve remains to be measured.

6. UDP reliability. UDP provides no delivery guarantee. While both the loopback test and the Godot echo test recorded 0.00% loss on localhost, the study does not cover networked deployment. Moreover, the login_success packet is transmitted once without acknowledgment; a single drop can stall the client indefinitely.

7. Rule-based classification. The classifier is deterministic and rule-based rather than trained. Jin et al. [27] achieved 96.4% with a lightweight 2D CNN-LSTM for 21 gestures, and Mohammed et al. [28] reached 97.25% with an end-to-end deep learning pipeline, but both approaches require training data and GPU resources that the target classroom environment may not have. The rule-based approach is the pragmatic choice for the current deployment context.

8. Face authentication constraints. The 19.0% cross-subject FAR at tolerance 0.6 indicates that the default setting is too permissive for strict access control. The module also requires a near-frontal face and fails beyond approximately 30 degrees of yaw. The FAR-to-FRR tradeoff was evaluated at a single tolerance value rather than swept across a range (e.g., 0.4 to 0.8).

9. Single-user design. The system is designed for one operator at a time. Multi-operator configurations, such as the ROS-based framework in Yu et al. [9], represent a possible extension but are outside the current scope.

4. Conclusion

This study presented the design, implementation, and experimental evaluation of a low-cost, real-time, webcam-based hand gesture recognition system integrated with face authentication and a Godot Engine 4.3 3D drone simulation. By adopting a Design Science Research (DSR) framework, this research successfully developed a functional IT artifact that replaces expensive dedicated hardware with a standard USB webcam while maintaining the responsiveness required for interactive educational tools.

The principal findings are as follows. First, the finger-counting classifier achieved 100% accuracy (160 of 160 trials) across eight gestures under the fixed condition. This is an upper bound that validates both the gesture vocabulary design and the rule-based classifier at the intended operating point; generalization across subjects and conditions remains open. Second, UDP one-way loopback latency measured 0.33 ms with 0% packet loss. End-to-end gesture-to-drone latency was 23.9 ms, with MediaPipe detection contributing 22.16 ms (93%), UDP transit contributing 1.71 ms (7%), and Godot ingest contributing 0.05 ms (<1%). This places the pipeline well within the real-time interaction threshold and makes it faster than comparable web-based drone control systems. Third, the pipeline sustained 35.6 FPS on CPU. Classification was effectively cost-free (0.001 ms per frame), and frame-to-frame jitter originated primarily from occasional webcam capture buffer stalls rather than from the recognition pipeline. Fourth, face authentication achieved 100% recognition for the registered user. However, the measured False Acceptance Rate (FAR) of 19.0% at the default 0.6 tolerance indicates a bias toward user convenience. For high-security applications, this tolerance must be tightened (e.g., to 0.4 or 0.5) to reduce unauthorized access risks. Fifth, the Godot input-action pattern decouples gesture detection from drone control logic at a measured cost of 0.05 ms per packet, enabling the controller to accept keyboard, gamepad, or future input modalities with no code changes.

The primary contribution is a working, fully open-source pipeline that integrates MediaPipe hand tracking, Python-based face authentication, dual-channel UDP communication, and Godot Engine simulation components that prior work has largely treated in isolation or implemented on proprietary stacks. The system demonstrates that authenticated, responsive gesture control for interactive drone simulation is achievable with consumer-grade hardware and freely available software.

Future work should pursue several directions to address the identified limitations: (a) conducting multi-subject evaluation across varied hand sizes, skin tones, lighting conditions, and camera distances to produce a representative confusion matrix and population-level accuracy estimates; (b) exploration of CNN or LSTM classifiers for improved robustness on ambiguous gestures and unseen users, following Jin et al. [27]; (c) performing a systematic sweep of face recognition tolerance from 0.4 to 0.8, combined with temporal voting [18], to characterize and reduce the cross-subject FAR; (d) integration of a depth camera for distance-based gestures and improved lighting robustness; (e) porting the gesture recognition module to mobile platforms; (f) expanding the gesture vocabulary through 3D-CNN dynamic recognition[26]; (g) adding bidirectional feedback with acknowledgment and retransmission for the single-shot login_success packet to improve network reliability; and (h) conducting formal usability studies with multiple participants using established HCI instruments to evaluate user comfort, hand fatigue, and cognitive load.

Acknowledgment

The authors would like to express their sincere gratitude to the Department of Computer Engineering and Informatics, Politeknik Negeri Bandung, for providing the laboratory facilities and technical support that made this research possible. We also extend our profound appreciation to P3M Politeknik Negeri Bandung for their invaluable administrative and institutional support in facilitating this study.

Declarations

Author contribution. *Muhammad Rizqi Sholahuddin*: Conceptualization, Methodology, Software (Gesture Module), Writing Original Draft. *Muhammad Samudera Bagja*: Software (Godot Integration), Networking (UDP Communication). *Ardhian Ekawijana*: Visualization, Validation (System Integration). *Siti Dwi Setiarini*: Software (Face Authentication), Data Curation. Formal Analysis, *Firas Atqiya*: Supervision, Project Administration, Writing Review & Editing, all authors have approved the final manuscript.
Funding statement. POLBAN Research Grant for Laboratory Capacity Enhancement with reference No. B/1.13/PL1.R7/PG.00.03/2024
Conflict of interest. The authors declare no conflict of interest.
Additional information. No additional information is available for this paper.

Data and Software Availability Statements

The source code and project files are publicly available at: https://github.com/kyeiki/godotmp2026

References

[1] M. Oudah, A. Al-Naji, and J. Chahl, "Hand Gesture Recognition Based on Computer Vision: A Review of Techniques," *Journal of Imaging*, vol. 6, no. 8, p. 73, 2020, doi: 10.3390/jimaging6080073.
[2] Y. Zhang, J. Wang, X. Wang, H. Jing, Z. Sun, and Y. Cai, "Static Hand Gesture Recognition Method Based on the Vision Transformer," *Multimedia Tools and Applications*, vol. 82, no. 20, pp. 31309–31328, 2023, doi: 10.1007/s11042-023-14732-3.

[3] H. Shakhatreh *et al.*, "Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges," *IEEE Access*, vol. 7, pp. 48572–48634, 2019, doi: 10.1109/access.2019.2909530.
[4] A. Mairaj, A. I. Baba, and A. Y. Javaid, "Application Specific Drone Simulators: Recent Advances and Challenges," *Simulation Modelling Practice and Theory*, vol. 94, pp. 100–117, 2019, doi: 10.1016/j.simpat.2019.01.004.
[5] D. Tezza and M. Andujar, "The State-of-the-Art of Human-Drone Interaction: A Survey," *IEEE Access*, vol. 7, pp. 167438–167454, 2019, doi: 10.1109/access.2019.2953900.
[6] N. M. Bhiri, S. Ameur, I. Alouani, M. A. Mahjoub, and A. B. Khalifa, "Hand Gesture Recognition with Focus on Leap Motion: An Overview, Real World Challenges and Future Directions," *Expert Systems with Applications*, vol. 226, p. 120125, 2023, doi: 10.1016/j.eswa.2023.120125.
[7] S. S. Rautaray and A. Agrawal, "Vision Based Hand Gesture Recognition for Human Computer Interaction: A Survey," *Artificial Intelligence Review*, vol. 43, no. 1, pp. 1–54, 2015, doi: 10.1007/s10462-012-9356-9.
[8] Z. Zhao, H. Luo, G.-H. Song, Z. Chen, Z.-M. Lu, and X. Wu, "Web-Based Interactive Drone Control Using Hand Gesture," *Review of Scientific Instruments*, vol. 89, no. 1, 2018, doi: 10.1063/1.5004004.
[9] Y. Yu, X. Wang, Z. Zhong, and Y. Zhang, "ROS-Based UAV Control Using Hand Gesture Recognition," in *2017 29th Chinese Control And Decision Conference (CCDC)*, 2017, pp. 6795–6799. doi: 10.1109/ccdc.2017.7978402.
[10] K. Natarajan, T.-H. D. Nguyen, and M. Mete, "Hand Gesture Controlled Drones: An Open Source Library," in *2018 1st International Conference on Data Intelligence and Security (ICDIS)*, 2018, pp. 168–175. doi: 10.1109/icdis.2018.00035.
[11] C. Lugaresi *et al.*, "MediaPipe: A Framework for Building Perception Pipelines," *arXiv preprint arXiv:1906.08172*, 2019.
[12] G. Sánchez-Brizuela, A. Cisnal, E. de la Fuente-López, J.-C. Fraile, and J. Pérez-Turiel, "Lightweight Real-Time Hand Segmentation Leveraging MediaPipe Landmark Detection," *Virtual Reality*, vol. 27, no. 4, pp. 3125–3132, 2023, doi: 10.1007/s10055-023-00858-0.
[13] A. K. Singh, V. A. Kumbhare, and K. Arthi, "Real-Time Human Pose Detection and Recognition Using MediaPipe," in *Advances in Intelligent Systems and Computing*, Springer, 2022, pp. 145–154. doi: 10.1007/978-981-16-7088-6_12.
[14] A. D. Agustiani, M. R. Sholahuddin, S. M. Putri, and P. Hidayatullah, "Penggunaan MediaPipe untuk Pengenalan Gesture Tangan Real-Time dalam Pengendalian Presentasi," *Media Jurnal Informatika*, vol. 16, no. 2, pp. 147–153, 2024, doi: 10.35194/mji.v16i2.4788.
[15] M. A. Kassab, M. Ahmed, A. Maher, and B. Zhang, "Real-Time Human-UAV Interaction: New Dataset and Two Novel Gesture-Based Interacting Systems," *IEEE Access*, vol. 8, pp. 195030–195045, 2020, doi: 10.1109/access.2020.3033157.
[16] B. Chen, C. Hua, D. Li, Y. He, and J. Han, "Intelligent Human–UAV Interaction System with Joint Cross-Validation over Action–Gesture Recognition and Scene Understanding," *Applied Sciences*, vol. 9, no. 16, p. 3277, 2019, doi: 10.3390/app9163277.
[17] J. Akagi, T. D. Morris, B. Moon, X. Chen, and C. K. Peterson, "Gesture Commands for Controlling High-Level UAV Behavior," *SN Applied Sciences*, vol. 3, no. 6, 2021, doi: 10.1007/s42452-021-04583-8.
[18] A. Imran, R. Ahmed, M. M. Hasan, M. H. U. Ahmed, A. K. M. Azad, and S. A. Alyami, "FaceEngine: A Tracking-Based Framework for Real-Time Face Recognition in Video Surveillance System," *SN Computer Science*, vol. 5, no. 5, 2024, doi: 10.1007/s42979-024-02922-1.
[19] A. Sikarwar, H. Chandra, and I. Ram, "Real-Time Biometric Verification and Management System Using Face Embeddings," in *2020 IEEE 17th India Council International Conference (INDICON)*, 2020, pp. 1–4. doi: 10.1109/indicon49873.2020.9342551.
[20] F. A. Fayaz, S. Mohi-Ud-Din, I. Batool, S. Kaur, and M. Rashid, "Novel Face Recognition Based Examinee Authentication System Using Python D-Lib," in *2019 Fifth International Conference on Image Information Processing (ICIIP)*, 2019, pp. 480–485. doi: 10.1109/iciip47207.2019.8985983.
[21] C.-C. Tsai, C.-C. Kuo, and Y.-L. Chen, "3D Hand Gesture Recognition for Drone Control in Unity," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, 2020, pp. 985–988. doi: 10.1109/case48305.2020.9216807.
[22] A. H. Hoppe, D. Klooz, F. van de Camp, and R. Stiefelhagen, "Mouse-Based Hand Gesture Interaction in Virtual Reality," in *Communications in Computer and Information Science*, Springer, 2023, pp. 192–198. doi: 10.1007/978-3-031-36004-6_26.
[23] K. R. Dixit, T. Verma, U. S. Subramanya, and V. Umadevi, "Hand Gesture Based Quadcopter Control Using Image Processing and Adaptive Machine Learning," in *2018 3rd IEEE International Conference*

on Recent Trends in Electronics, Information & Communication Technology (RTEICT), 2018, pp. 1214–1218. doi: 10.1109/rteict42901.2018.9012139.

[24] J. Levy and D. Liu, "Extended Reality (XR) Environments for Flood Risk Management with 3D GIS and Open Source 3D Graphics Cross-Platform Game Engines," in *Lecture Notes in Networks and Systems*, Springer, 2023, pp. 271–285. doi: 10.1007/978-981-99-1912-3_25.

[25] L. A. Fagundes-Junior, K. B. de Carvalho, R. S. Ferreira, and A. S. Brandão, "Machine Learning for Unmanned Aerial Vehicles Navigation: An Overview," *SN Computer Science*, vol. 5, no. 2, 2024, doi: 10.1007/s42979-023-02592-5.

[26] M. Channayanamath, A. Math, S. Kamath, K. Chachadi, F. Sabeeh, A. Attar, and V. Peddigari, "Dynamic Hand Gesture Recognition Using 3D-Convolutional Neural Network," in *Lecture Notes in Networks and Systems*, Springer, 2020, pp. 145–153. doi: 10.1007/978-981-15-5397-4_16.

[27] H. Jin, Z. Jin, Y.-G. Kim, and C. Fan, "Integration of a Lightweight Customized 2D CNN Model to an Edge Computing System for Real-Time Multiple Gesture Recognition," *Journal of Grid Computing*, vol. 21, no. 4, 2023, doi: 10.1007/s10723-023-09715-5.

[28] A. A. Q. Mohammed, J. Lv, and M. S. Islam, "A Deep Learning-Based End-to-End Composite System for Hand Detection and Gesture Recognition," *Sensors*, vol. 19, no. 23, p. 5282, 2019, doi: 10.3390/s19235282.