Journal of Applied Informatics and Computing (JAIC)
Vol.9, No.3, June 2025, pp. 979~988
e-ISSN: 2548-6861

Real-Time Hand Gesture Control of a Quadcopter Swarm Implemented in the Gazebo Simulation Environment

Ryan Satria Wijaya [1]*, Senanjung Prayoga [2]*, Rifqi Amalya Fatekha [3]*, Muhammad Thoriq Mubarak [4]*
* Teknik Robotika, Politeknik Negeri Batam
ryan@polibatam.ac.id [1], senanjung@polibatam.ac.id [2], rifqi@polibatam.ac.id [3], muhammadthoriqmubarak@gmail.com[4]

Article Info

Article history:

Received 2025-04-30
Revised 2025-06-02
Accepted 2025-06-17

Keyword:

Hand Gesture Recognition,
Quadcopter Swarm,
Mediapipe,
ROS,
Gazebo Simulation.

ABSTRACT

With the advancement of technology, human-robot interaction (HRI) is becoming more intuitive, including through hand gesture-based control. This study aims to develop a real-time hand gesture recognition system to control a quadcopter swarm within a simulated environment using ROS and Gazebo. The system utilizes Google's MediaPipe framework for detecting 21 hand landmarks, which are then processed through a custom-trained neural network to classify 13 predefined gestures. Each gesture corresponds to a specific command such as basic motion, rotation, or swarm formation, and is published to the /cmd_vel topic using the ROS communication framework. Simulation tests were performed in Gazebo and covered both individual drone maneuvers and simple swarm formations. The results demonstrated a gesture classification accuracy of 90%, low latency, and stable response across multiple drones. This approach offers a scalable and efficient solution for real-time swarm control based on hand gestures, contributing to future applications in human-drone interaction systems.

I. INTRODUCTION

Advances in human-computer interaction (HCI) have greatly contributed to the development of more intuitive and user-friendly control systems in robotics[1], [2]. Among these advancements, hand gesture recognition has emerged as a promising and natural modality for human-robot interaction[3], [4]. Unlike traditional hardware-based control methods, such as joysticks and remote controllers, gesture-based systems offer a hands-free and more immersive control experience[5], making them particularly advantageous in dynamic and complex environments[6], [7].

In robotics, a computer vision plays a pivotal role in making machines to perceive, interpret, and interact with their surroundings. Vision-based systems, when paired with advanced algorithms, provide robots with the capability to recognize objects, navigate environments, and interpret human intent[8]. This is particularly crucial in Unmanned Aerial Vehicles (UAVs)[9], where vision facilitates key functionalities such as obstacle avoidance, target tracking, and environmental mapping. Vision-based control systems enable UAVs to adapt to changing conditions and perform complex tasks autonomously, such as aerial surveillance and precision delivery.

Controlling quadcopters using hand gestures presents significant opportunities across various fields, including aerial surveillance, search and rescue operations, and entertainment[10]. Gesture-based control of UAVs offers a more natural and intuitive interaction, which is especially beneficial in scenarios requiring rapid decision-making and minimal physical equipment[11]. For instance, search and rescue missions in disaster-struck areas often demand swift control over drones operating in cluttered environments[12], where traditional input methods might be cumbersome. Gesture-based systems can significantly enhance operational efficiency in such cases.

Moreover, gesture-based control of quadcopter swarms opens new possibilities in multi-agent robotics by improving coordination, adaptability, and responsiveness in real-time scenarios[13], [14]. UAV swarms equipped with vision-based systems can collaborate on complex tasks such as area coverage, synchronized movement, and distributed data collection[15]. However, developing such a system presents several key challenges, including achieving real-time gesture

recognition, ensuring accurate swarm coordination, and maintaining low-latency communication between the user and the robotic agents.

Recent study in gesture recognition frameworks have led to the development of tools capable of recognizing complex hand movements with high accuracy. One such framework is Google's MediaPipe, known for its reliable hand landmark detection and real-time performance[16], [17]. By leveraging MediaPipe's palm detector for gesture recognition and integrating it with the Robot Operating System (ROS), it becomes possible to design a vision-based control system capable of publishing gesture data to a subscriber node responsible for controlling quadcopters[18].

This study aims to create a scalable and efficient framework for real-time gesture-based control of quadcopter swarms. The proposed system uses a camera to capture hand gestures, which are then processed by MediaPipe to detect specific gestures. These recognized gestures are translated into control commands and published via ROS to manage the movement of quadcopters in a simulated environment. The drones' motion is controlled using the cmd_vel topic, which allows for precise maneuvering based on the interpreted gestures.

Building on prior research, this paper introduces a novel gesture recognition model capable of recognizing thirteen custom hand gestures, each corresponding to a specific command for quadcopter movement. These commands enable users to control not only individual drones but also the entire swarm, facilitating complex behaviors and coordinated maneuvers.

This study presents a novel approach to gesture-based UAV control and swarm robotics, combining real-time hand gesture recognition using MediaPipe with the Robot Operating System (ROS) for multi-agent coordination within the Gazebo simulation environment. Unlike previous research, which has focused primarily on single-drone gesture control or basic commands[13], [14], this work explores the coordination of multiple drones responding to gesture inputs, demonstrating synchronized motion among quadcopters. A key innovation is the implementation of a custom-trained gesture classifier on top of MediaPipe, which distinguishes this system from others relying solely on rule-based interpretation or built-in gesture recognition. This study represents one of the first successful demonstrations of a gesture-controlled swarm system tested in ROS-Gazebo, showcasing both individual and formation flight behavior. Furthermore, this approach provides a unique interface for human-swarm interaction and highlights the potential for vision-based control in coordinating multiple UAVs simultaneously.

The primary contributions of this study include the design and implementation of a real-time hand gesture recognition system specifically developed for controlling quadcopters, the integration of MediaPipe with the Robot Operating System (ROS) to enable seamless communication between the gesture recognition module and the drone control system, and the performance evaluation of the proposed system in terms of gesture recognition accuracy, confidence levels, varying lighting conditions, and swarm coordination using the Gazebo simulation platform. The remainder of this paper is organized as follows: the methodology, including system architecture, gesture recognition module, ROS integration, and the simulation setup in Gazebo, is described in detail; followed by the presentation and discussion of results; and finally, the conclusion along with potential future directions to enhance the system's performance and scalability.

II. METHOD

The proposed system is designed with three primary components: the hand gesture recognition module, ROS integration for communication, and the Gazebo simulation platform. Each is crucial for maintaining real-time performance and ensuring the system's high precision.

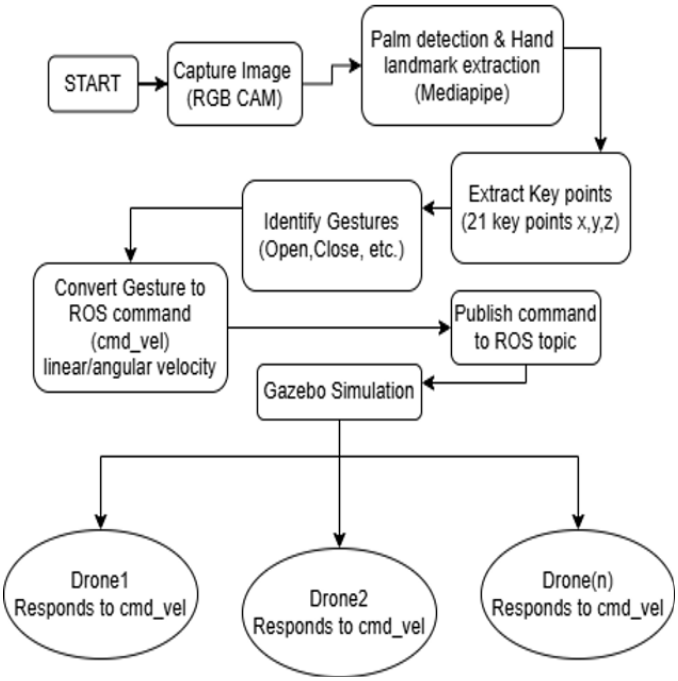

Figure 1. Architecture of the entire system

A. Architecture of the system

The diagram on Figure 1 outlines a vision-based hand gesture recognition system designed to control a quadcopter swarm in a Gazebo simulation using MediaPipe and ROS. The process begins by capturing real-time RGB images with a camera. These images are then processed by MediaPipe to detect the palm and extract 21 key hand landmarks, which include the x, y, and z coordinates of each point. Based on the positions of these landmarks, specific gestures (such as open or closed hands) are identified.

The recognized gestures are converted into ROS commands for controlling the drones' linear and angular velocities, which are then published to the cmd_vel topic. These commands are sent to the simulated drones in Gazebo, causing them to respond and adjust their movements according to the hand gesture inputs. The system can control multiple drones, with each one responding to the same cmd_vel topic, enabling coordinated swarm behavior.

B. Hand Gesture Recognition Module

This module leverages the MediaPipe framework to detect and classify hand gestures in real-time. MediaPipe was selected due to its robustness in detecting hand landmarks with high precision and low latency[19]. Specifically, it achieves a accurate identification of 21 hand landmarks using a 3D hand-knuckle coordinate system, as illustrated in Figure 2, performed through regression within the detected hand region[20]. This regression directly produces coordinate predictions that represent the hand landmarks.

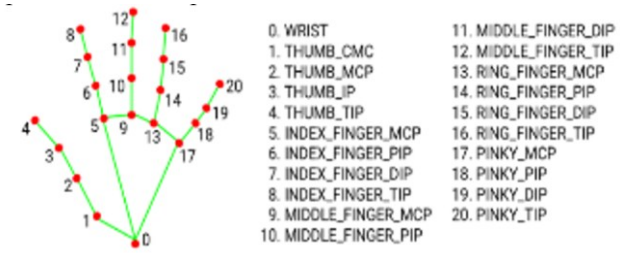

Figure 2. Key point hand gestures mediapipe.

Each hand knuckle landmark coordinate consists of x, y, and z values; x and y are scaled within the [0.0, 1.0] range according to image dimensions, whereas z denotes the relative depth of the point. The depth is measured relative to the wrist, with landmarks closer to the camera having smaller values. The model has been fine-tuned to recognize 13 distinct gestures, each corresponding to specific quadcopter commands, such as moving forward, backward, ascending, descending, rotating, and stopping. These gestures are classified based on the spatial coordinates of the detected hand landmarks, and a custom classification algorithm processes these coordinates, mapping them to pre-defined commands. The output is then formatted into ROS-compatible messages for communication.

In this study, we used the pre-trained MediaPipe hand tracking model without modifying or fine-tuning its architecture. MediaPipe was employed solely for detecting 21 hand landmarks per frame in real time. These landmark coordinates were then extracted and normalized to serve as feature vectors for a custom neural network classifier developed using TensorFlow Keras. The classifier was trained separately using a labeled dataset of 13 predefined gestures. Thus, our contribution lies in the gesture classification layer built on top of MediaPipe, not in modifying or retraining the MediaPipe model itself.

1) Model Training: This study requires a model to recognize gestures and assign labels that align with predefined definitions. The block diagram in Figure 3 illustrates the complete workflow of hand gesture identification employed in this study, the process begins with capturing hand motion data using MediaPipe, followed by feature extraction in the form of key point coordinates, neural network model training, and gesture classification based on output probabilities.

Figure 3. Pattern recognition process.

The model for gesture recognition and labelling was developed using a neural network, as shown in Figure 4. During data collection, each gesture is represented by the coordinates of all key points, from key point 0 to 20. These key point coordinates serve as input for training the model. The dataset served both training and evaluation purposes for the model consists of key point data from gestures collected via MediaPipe, which are saved in CSV format during data collection. Of the dataset, 75% is used for training, and 25% for testing[21]. Model training begins by building a neural network architecture using Tensor Flow Keras. The model is structured sequentially with several layers to process the input data.

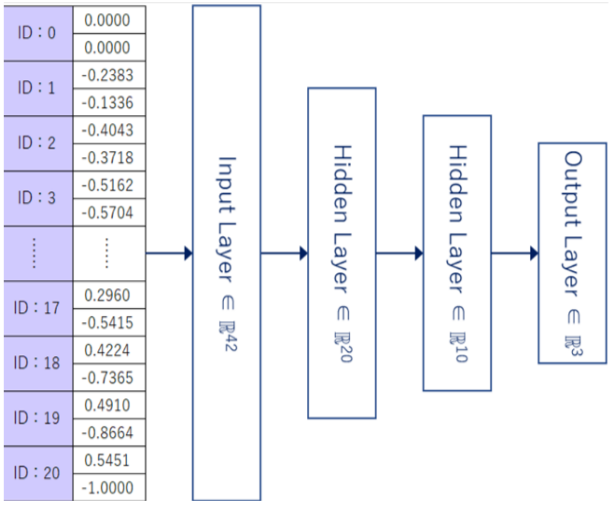

Figure 4. Model Structure.

The first layer receives a vector with dimensions 21×2, containing x and y coordinate values for 21 hand landmarks. To reduce overfitting, the model incorporates two dropout layers with probabilities of 0.2 and 0.4, respectively. The model also includes two dense layers with 20 and 10 neurons, both applying the ReLU activation to learn non-linear features. The final layer uses the softmax function, aligning the neuron count with the number of gesture categories, allowing for gesture classification based on probabilities. Model compilation is performed using the Adam optimizer,

known for its computational efficiency weight update algorithm, along with the sparse categorical cross entropy loss function, as the labels are provided in integer format. Accuracy is chosen as the metric to monitor model performance during training. The training process runs for a maximum of 1000 epochs, with two call-backs employed: Model Checkpoint, which automatically saves the best model weights after each epoch, and Early Stopping, with a patience parameter of 20 epochs, allowing early stopping when no improvement in validation accuracy is observed, thus avoiding overfitting. With this configuration, the model is effectively trained to recognize hand gestures based on the available dataset.

2) Gesture Categories and Commands: These 13 gestures are categorized into two main group, basic movements with 8 gestures, simple formation with 4 gestures, and 1 gesture for stopping, this gesture-based control system allows intuitive interaction with the quadcopters, ensuring that each gesture corresponds to a clear and distinct command. The fine-tuning of the MediaPipe model and the mapping of gestures to commands are essential components in achieving accurate and responsive quadcopter control.

3) Gesture Recognition Process: The HaGRID is a large-scale dataset curated for training hand gesture detection models, comprising 552,992 labeled images distributed across 18 distinct gesture classes, with a total dataset size of approximately 716 GB. We used a combination of HaGRID and a custom-made dataset for our model training.

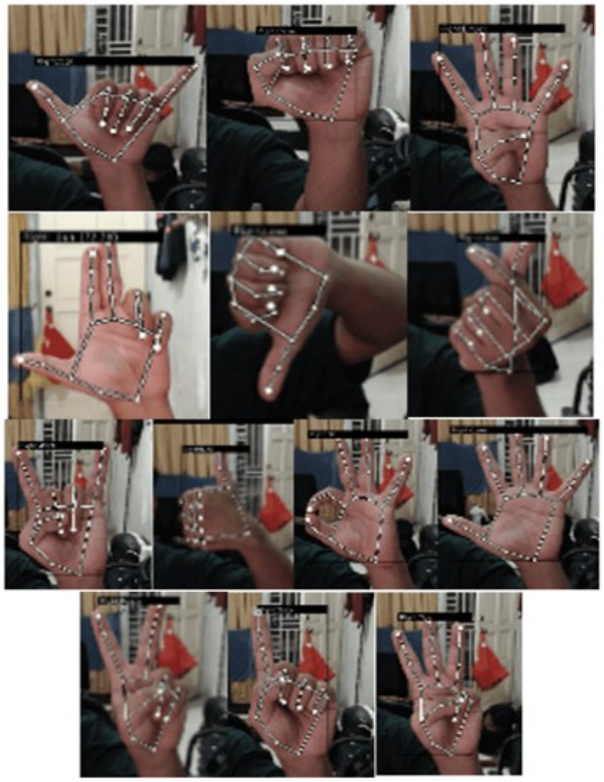

Figure 5. 13 Gestures for data train.

This project has 13 gestures. a standard RGB camera captures hand gestures in real-time. Each frame from the video feed is processed by MediaPipe to detect hands and extract their landmarks shown in Figure 5, which include 21 key points on each hand. Pre-processing: Processes the cropped image based on the bounding box and outputs 21 three-dimensional coordinates representing hand landmarks. Gesture Classification: a custom gesture classification algorithm processes the normalized landmarks and assigns them to one of the 13 predefined gestures. Every gesture is mapped to a unique control instruction for the quadcopters. The classified gesture is then converted into a ROS message format and published to the /cmd_vel topic.

C. ROS Integration

The Robot Operating System (ROS) serves as the primary communication framework, enabling seamless integration between gesture-based control inputs and the quadcopter swarm within the simulation environment. It provides the necessary infrastructure for real-time message passing[22], ensuring that commands generated by the gesture recognition system can be transmitted reliably to the quadcopters.

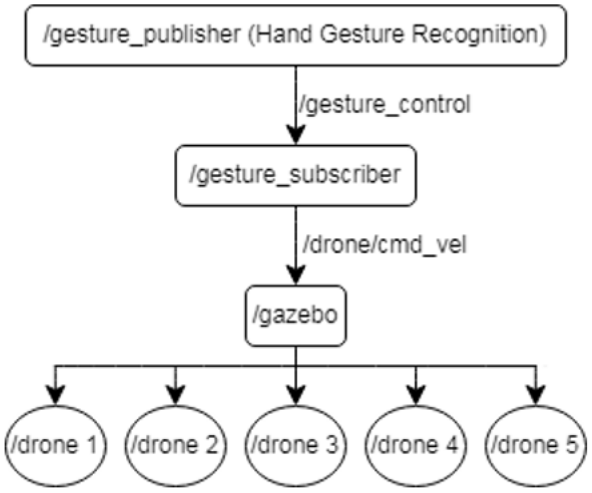

Figure 6. ROS flow.

This robust middleware facilitates efficient data exchange between various system components, including the vision-based gesture recognition module[23], decision-making algorithms, and low-level flight controllers, thereby ensuring precise and synchronized execution of movement commands.

The system leverages the /cmd_vel topic, which acts as a standardized communication channel for transmitting motion control commands to the quadcopters. These commands are encoded as messages of type geometry_msgs/Twist shown in Figure 6 from gazebo to /drones, a common ROS message format designed for expressing both linear and angular velocities in a three-dimen sional space.

As shown in Figure 1, each message published to the /cmd_vel topic contains two primary components that govern the movement and orientation of the quadcopters:

- Linear Velocity: This parameter controls the translational motion of the quadcopters along three orthogonal
- X-axis: Backward and forward movement.
- Y-axis: Side-to-side translation (leftward and rightward motion).
- Z-axis: Vertical movement (upward and downward).
- Angular Velocity: This parameter dictates the rotational motion of the quadcopters around their vertical (Z) axis. Adjusting the angular velocity allows the quadcopters to rotate clockwise or counter clockwise, facilitating changes in heading direction.

Quadcopters in the swarm are designed to subscribe to their respective /cmd_vel topics, enabling them to interpret incoming Twist messages and execute the corresponding motions instantaneously. For example, when a message with a positive linear velocity along the X-axis is received, the quadcopter moves forward, while a negative value induces backward movement. Similarly, angular velocity values influence the rotational speed and direction of the quadcopter around its central axis.

The use of ROS topics such as /cmd_vel allows for high modularity and scalability in the system, making it feasible to control multiple quadcopters simultaneously. By publishing distinct Twist messages to individual topics (e.g., /drone1/cmd_vel, /drone2/cmd_vel, etc.), the system ensures that each quadcopter responds independently to the designated gesture-based commands. This design also supports coordinated behaviors and complex maneuvers by synchronizing motion across the swarm through the ROS communication framework.

In the implemented system, each drone in the swarm subscribes to its respective velocity command topic (/drone1/cmd_vel, /drone2/cmd_vel, etc.), enabling individual or synchronized control. The gesture control node listens to the /gesture_control topic and maps gesture strings to ROS Twist messages based on predefined rules. While basic gestures such as “Open” or “Satu” are broadcasted to all drones for synchronized movement, certain gestures like “Gun”, “Love”, “Call”, and “Empat” trigger complex formation routines involving only specific drones (e.g., drone 3 and drone 5).

These complex gestures execute coordinated motion sequences—such as mirrored lateral movements, vertical oscillations, and directional spreads—by publishing distinct Twist commands to the targeted drones over a fixed duration. Time-based execution using rospy.Time.now() and motion loops with sleep intervals ensure smooth transitions and prevent command overlap. Furthermore, a gesture delay mechanism prevents repeated execution within 0.5 seconds, minimizing redundant or unintended commands. This implementation demonstrates how high-level human gestures can drive intricate swarm behaviors through ROS-based multi-agent coordination.

Each drone in the swarm is represented as a separate node in ROS and subscribes to its own command topic (e.g., /drone1/cmd_vel, /drone2/cmd_vel). A central gesture control node acts as a publisher, broadcasting identical or individual Twist messages depending on the intended behavior. This structure enables flexible control, allowing both synchronized swarm motion (e.g., formation flight) and individual drone commands. Coordination across drones is achieved through the timing of identical messages, without requiring a global planner or inter-drone communication, making the system scalable and modular.

TABLE I
VELOCITY COMMAND MAPPING

Gesture	Linear. x	Linear. y	Linear. z	Angular. z	Drone(s)
Open	1.5	0.0	0.0	0.0	(1-5)
Satu	-1.5	0.0	0.0	0.0	(1-5)
Close	0.0	0.0	0.0	0.0	(1-5)
Lose	0.0	0.0	-1.5	0.0	(1-5)
Nice	0.0	0.0	1.5	0.0	(1-5)
OK	0.0	1.5	0.0	0.0	(1-5)
Tiga	0.0	-1.5	0.0	0.0	(1-5)
Metal	0.0	0.0	0.0	-1.5	(1-5)
Dua	0.0	0.0	0.0	1.5	(1-5)
Gun	±2.0	±1.0	0.0	0.0	3 & 5
Love	±2.0	±1.0	0.0	0.0	3 & 5
Call	0.0	±1.0	±1.5	0.0	3 & 5
Empat	0.0	±1.0	±1.5	0.0	3 & 5

Note: ± indicates opposite-direction movement applied to different drones (e.g., +1.0 for drone 3, -1.0 for drone 5).

Table I presents the velocity command mapping for each recognized hand gesture in the system. These mappings define the corresponding movement behavior of the quadcopters in response to gesture inputs, using the geometry_msgs/Twist message type in ROS. Each gesture is translated into a specific combination of linear and angular velocities across the x, y, and z axes. While most gestures control all drones simultaneously with uniform velocity, certain gestures (e.g., "Gun", "Love", "Call", "Empat") are designed to execute complex patterns involving only specific drones (namely, drone 3 and drone 5) with mirrored or alternating movements. These complex routines simulate basic formation behaviors and showcase the flexibility of the gesture control system to coordinate both individual and collective drone actions.

D. Simulation Environment

Gazebo was utilized as the simulation platform to test the proposed system. The tests were conducted in an empty world without any additional objects or obstacles, focusing primarily on basic movements and simple swarm formations

[24]. Each quadcopter was equipped with a ROS-based control plugin, allowing them to subscribe to the '/cmd_vel' topic and execute movement commands according to the recognized gestures[25].

The testing scenarios included:

1) *Basic Movements*: Individual quadcopters are tested for various maneuvers based on recognized gestures. The commands tested include moving forward, backward, up or down, move left or right, stop, and rotating left or right (Figure 7).

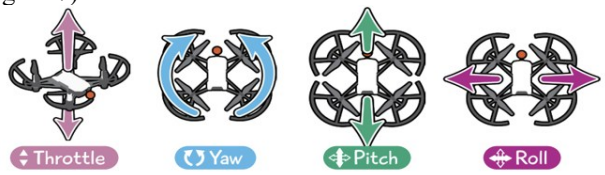

Figure 7.Basic drone movements.

2) *Simple Formations*: The quadcopter swarm is tested for coordinated movement in simple formations. Commands are given to move the swarm in a straight line while maintaining fixed relative distances, as shown in Figure 11, and to arrange in a vertical plus sign, as depicted in Figure 12.

III. RESULT AND DISCUSSIONS

The system was evaluated in a simplified simulation environment using basic movement and simple formation scenarios. The performance was assessed based on gesture recognition accuracy, response time, as well as the system's capability to manage the quadcopter swarm with real-time responsiveness.

A. Gesture Recognition Accuracy

The accuracy of the gesture recognition module, developed using the fine-tuned MediaPipe framework, was evaluated on a dataset consisting of 13 distinct hand gestures. To ensure reliable performance, each gesture was represented by a minimum of 1,000 samples shown in Figure 8, resulting in a dataset with over 21,000 data points in total. Certain gestures, such as the "open hand" gesture, were collected with a larger number of samples due to their higher similarity with other gestures. This additional data aimed to reduce potential misclassification caused by gesture overlap.

The system demonstrated reliable recognition of 13 different gestures, as shown in Table II. This performance is attributed to the robustness of MediaPipe's pre-trained model and the fine-tuning process using a large, diverse dataset. However, minor misclassifications were observed under poor lighting conditions or when gestures were performed too quickly. Expanding the dataset, particularly for similar gestures, proved effective in reducing errors and enhancing overall recognition consistency.

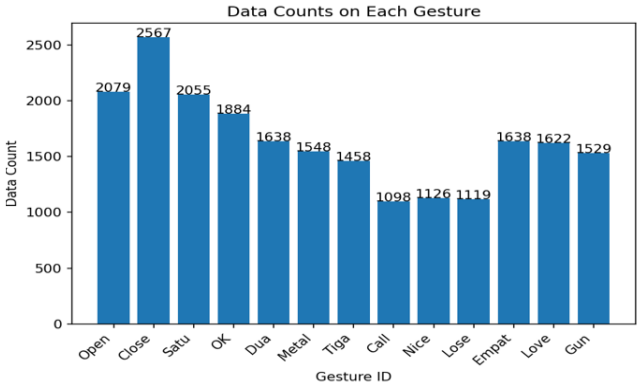

Figure 8.Data counts.

B. Confidence Level

The confidence levels of various hand gestures were evaluated, as shown in the Figure 9, with the vertical axis representing confidence percentage and the horizontal axis listing gestures trials, Gestures like "Lose," "Gun," "Call," "Close," and "Love" achieved high confidence levels, consistently above 90%, while "Metal," "Tiga," "OK," "Satu," "Dua," and "Empat" showed lower confidence, generally below 85%. Gestures such as "Open" and "Nice," also performed well, with confidence levels above 85%. Overall, the gesture recognition system demonstrated stable performance and minimum at 80%, with several gestures maintaining consistently high confidence throughout the ten trials.

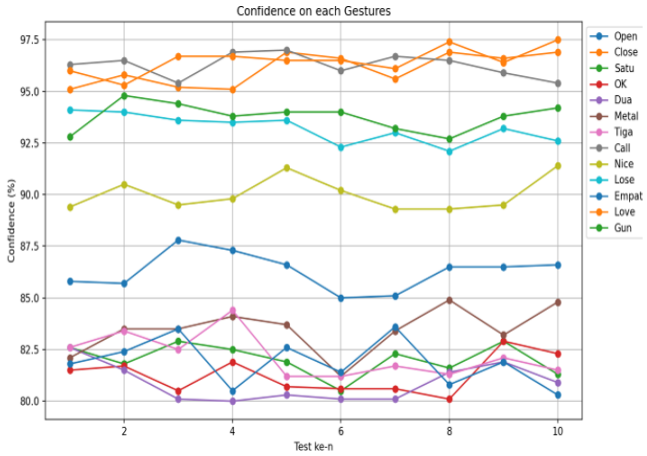

Figure 9. Gestures confidence.

C. Confusion Matrix

The performance evaluation of the model in deep learning can be conducted using a Confusion Matrix. In Python, the scikit-learn library provides an efficient method for generating a confusion matrix. Before predicting hand gestures, experimental datasets were collected and utilized. The A confusion matrix was utilized to evaluate the model's classification accuracy.

The confusion matrix on Figure 10 shows that the model performs well overall, with most predictions concentrated along the diagonal, indicating high accuracy for many classes.

However, some misclassifications occur, particularly in class 0, where a notable number of samples are predicted as classes 1, 2, 8, 9, 10, and 12. Similarly, class 1 has a significant number of misclassified samples in class 2, 8, 9, and 11. Class 3 also exhibits some confusion with class 0 and 10, class 4 has misclassifications in class 2, and class 10 has misclassified in class 2. These errors suggest that certain classes have overlapping features, making them harder to distinguish.

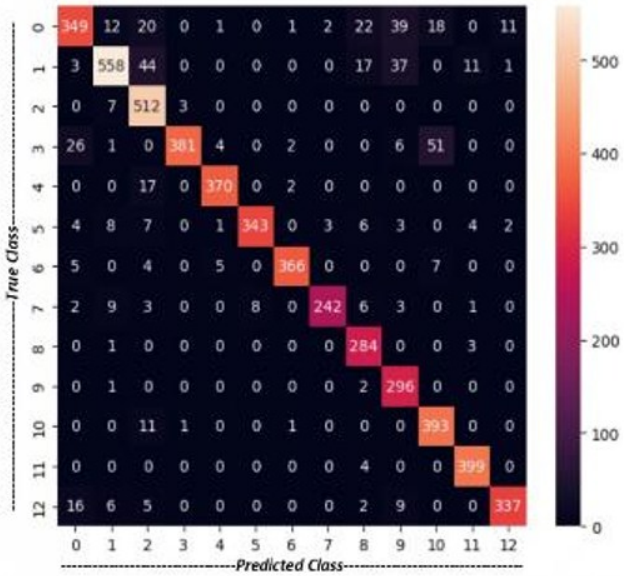

Figure 10. Classification performance.

Table II presents the classification report for a deep learning-based hand gesture recognition system using MediaPipe within a user guide application. The model, built with a Feedforward Neural Network (FNN), classified 13 gestures with an accuracy of 90%. Metrics such as precision, recall, and F1-score highlighted strong performance for classes 3, 4, and 7 (F1 > 0.9), while class 0 had a lower recall (0.73), indicating recognition challenges. Misclassifications were influenced by lighting, sensor distance, and angle variations. Despite a weighted F1-score of 0.90, improvements are needed for challenging scenarios. The system's real-world utility was demonstrated through gesture-based commands in the application.

Equation (1) is Precision: Indicates the proportion of correctly predicted positive instances. The formula is:

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

Equation (2) is Recall: Reflects the model's effectiveness in detecting all true positive cases. The formula is:

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

Equation (3) is F1-Score: which combines precision and recall using their harmonic mean, is especially valuable in imbalanced class scenarios. The formula is:

$$F1\ Score = 2 * \frac{Precison * Recall}{Precision + Recall} \quad (3)$$

Equation (4) is quantifies the ratio of correct predictions to the total number of instances evaluated:

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN} \quad (4)$$

Support: Refers to the number of actual samples for each class.

note: "TP" refers to true positives, "TN" to true negatives, "FP" to false positives, and "FN" to false negatives

TABLE II
ACCURACY CLASSIFICATION PERFORMANCE

Class	Precision	Recall	F1-Score	Support
0	0.86	0.73	0.79	475
1	0.93	0.83	0.88	671
2	0.82	0.98	0.89	522
3	0.99	0.81	0.89	471
4	0.97	0.95	0.96	389
5	0.98	0.90	0.94	381
6	0.98	0.95	0.96	387
7	0.98	0.88	0.93	274
8	0.83	0.99	0.90	288
9	0.75	0.99	0.86	299
10	0.84	0.97	0.90	406
11	0.95	0.99	0.97	403
12	0.96	0.90	0.93	375
Accuracy			0.90	5341
Macro avg	0.91	0.91	0.88	5341
Weighted	0.91	0.90	0.90	5341

D. Case Study: Basic Movements

In this scenario, individual quadcopters were controlled using hand gestures, with each gesture mapped to a specific command for fundamental maneuvers such as moving forward, backward, ascending, descending, and rotating. The system demonstrated impressive responsiveness, with each gesture accurately interpreted and executed by the quadcopters. For instance, the "Open" gesture was used to command the quadcopters to move forward, while the "Close" gesture halted their motion, effectively stopping them. The "Satu" gesture caused the quadcopters to move backward, and the "OK" gesture directed them to move left. Rotation was also controlled with specific gestures, such as "Dua" for rotating left and "Metal" for rotating right. The "Tiga" gesture made the quadcopters move right, while "Call" initiated Reverse Formation-2, allowing the quadcopters to reposition in a specific formation. The "Nice" gesture instructed the quadcopters to ascend, and "Lose" made them descend. Additionally, "Empat" triggered Formation-2, "Love" activated Reverse Formation-1, and "Gun" initiated Formation-1. These gestures were accurately mapped to the desired actions, ensuring smooth and reliable control, with the

quadcopters following the operator's commands in real-time with minimal latency.

TABLE III
ACCURACY CLASSIFICATION PERFORMANCE

Gesture Class	Label	Movement Set	Movement Test Results
0	"Open"	Forward	suitable
1	"Close"	Stop	suitable
2	"Satu"	Backward	suitable
3	"OK"	Move-Left	suitable
4	"Dua"	Rotate-Left	suitable
5	"Metal"	Rotate-Right	suitable
6	"Tiga"	Move-Right	suitable
7	"Call"	Reverse Form-2	suitable
8	"Nice"	Up	suitable
9	"Lose"	Down	suitable
10	"Empat"	Formation-2	suitable
11	"Love"	Reverse Form-1	suitable
12	"Gun"	Formation-1	suitable

Table III presents the mapping between gesture classes, their respective movement sets, and the movement testing results. Each gesture corresponds to a specific movement set, ensuring reliable control for the quadcopters. The test results confirm that all gestures performed as intended, demonstrating the system's effectiveness in gesture-based quadcopter control.

E. Case Study: Simple Formations

In the simple formation scenario, multiple quadcopters were controlled simultaneously to move in a straight line while maintaining a fixed relative distance. The swarm successfully followed the given commands, demonstrating that the system can handle basic swarm coordination with reliable communication via ROS shown in Figure 11.

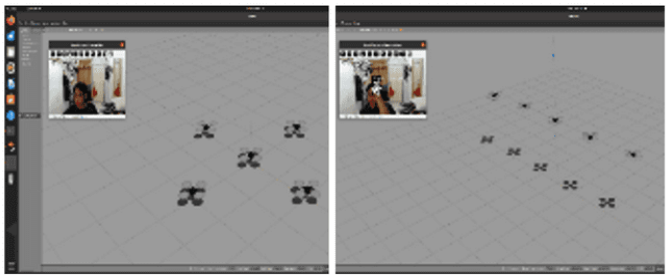

Figure 11. Demo quadcopters formation-1.

In the second formation test, the quadcopters were instructed to position themselves into a vertical plus sign, maintaining equal spacing between each unit. This arrangement required precise coordination to ensure the drones held their designated positions accurately, as shown in Figure 12.

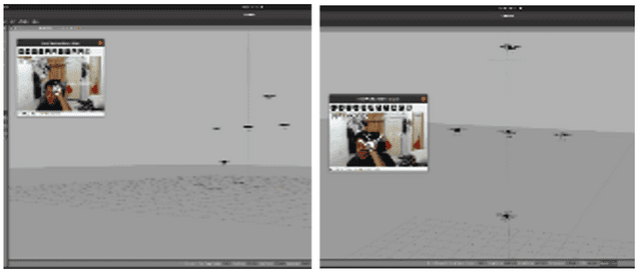

Figure 12. Demo quadcopters formation-2.

F. Performance under Different Lighting Conditions

To evaluate the reliability of the gesture recognition system in different environmental conditions, tests were conducted under three lighting scenarios: bright, dim, and dark. Each scenario was simulated by adjusting the ambient light intensity where gestures were performed. The tests aimed to assess how well the system could maintain accurate gesture detection in varying light levels. In the bright scenario, the environment was fully illuminated, providing optimal conditions for gesture recognition. The dim lighting condition reduced light intensity to test the system's ability under less ideal circumstances. Lastly, the dark scenario simulated low-light conditions, evaluating the system's performance in challenging environments. These tests ensured that the system could function reliably across different real-world lighting conditions.

1) *Bright Conditions*: Table IV in bright conditions, the system performs best, detecting hand movements effectively up to 4 meters. High-speed movements are detected up to 1.5 meters, while slower movements are detected up to 4 meters. However, beyond 4 meters, the system struggles to detect any gestures.

TABLE IV
BRIGHT CONDITION TEST

Respond	<1m	1.5m	2m	3m	4m	>4m
Very fast	✓					
Fast		✓				
Medium			✓			
Slow				✓		
Very Slow					✓	
Not Detected						✓

2) *Dim Condtions:* Table V under dim lighting, the system maintains good performance. Fast movements are detected up to 1.5 meters, while slow and very slow movements are detected beyond 4 meters. The system shows better sensitivity to slow movements at longer distances.

TABLE V
DIM CONDITION TEST

Respond	<1m	1.5m	2m	3m	4m	>4m
Very fast						
Fast	✓	✓				
Medium			✓			
Slow						
Very Slow						
Not Detected				✓	✓	✓

3) *Dark Condtions:* Table VI in dark conditions, the system's performance significantly decreases, detecting only slow movements. Medium-speed movements are detected up to 1 meter, and slow movements up to 1.5 meters. Beyond 2 meters, only very slow movements are reliably detected.

TABLE VI
DARK CONDITION TEST

Respond	<1m	1.5m	2m	3m	4m	>4m
Very fast						
Fast						
Medium	✓					
Slow		✓				
Very Slow						
Not Detected			✓	✓	✓	✓

4) Observations: The system performed best under bright conditions, where hand landmarks were clearly visible. Performance under dim conditions remained acceptable, with only slight inaccuracies in gesture recognition. Under dark conditions, although the accuracy decreased, the system was still able to detect basic gestures shown in Figure 13, improving the robustness of the gesture recognition module under low-light conditions can be an area for future research.

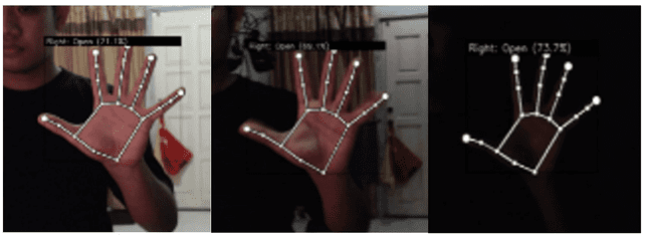

Figure 13. Test on different lighting condition.

G. System Latency Measurement

To assess the responsiveness of the proposed system, we measured the time delay between the moment a hand gesture is recognized by the gesture classification module and the point at which the corresponding velocity command is published to the /cmd_vel topic in ROS.

This measurement does not include the time between the actual physical gesture movement by the user and the recognition event, nor does it include the time until the simulated drone physically responds in Gazebo. Rather, it strictly covers the processing latency from the recognition of a gesture (via MediaPipe + classifier) to the publication of a movement command in ROS.

The latency was measured using rospy.get_time() in both the gesture recognition publisher and the drone control subscriber node. Each gesture message was tagged with a timestamp when it was recognized, and the receiving node calculated the latency upon message arrival.

We conducted 30 trials across three common gestures: Open, Tiga, and Metal. The observed latency ranged from 140 ms to 180 ms, with an average delay of approximately 157 ms. This range reflects the combined processing time for gesture classification, message construction, and inter-node communication within the ROS framework.

Despite not including sensor-level or actuation latency, this measurement is useful to evaluate the internal communication performance and real-time capability of the system's gesture-to-command pipeline.

H. Limitations

Despite its promising performance, the proposed system has a few limitations

1) *Simulation vs Real-World Condtiions*: The system was developed and tested in a Gazebo simulation environment, which offers ideal conditions with no sensor noise, perfect communication, and accurate physics modeling. In real-world deployments, the drone behavior may be affected by factors such as wind, sensor inaccuracies, hardware latency, and physical constraints. Movements like mirrored patterns in gestures such as “Gun” or “Love” may pose collision risks without precise spatial coordination and safety measures.

2) *Environmental Complexity:* The simulation environment used for testing was simplified and free of dynamic obstacles. The system’s robustness in more complex, cluttered, or dynamic environments (e.g., with moving objects or uneven terrain) has not been evaluated and may require integration with obstacle avoidance or terrain mapping systems.

3) *Lighting Sensitivity:* Although the gesture recognition system maintained acceptable performance under moderate and dim lighting, real-world lighting conditions can be highly variable. Variations such as shadows, glare, or sudden changes in brightness may impact landmark detection accuracy. Improvements in illumination handling or sensor adaptation could enhance system reliability.

4) *Gesture Set Restriction:* The current implementation supports 13 predefined static gestures based on single-hand input. While sufficient for basic swarm commands, the system could be extended to include multi-hand, dynamic, or continuous gestures to allow for more complex swarm behaviors and finer control.

IV. CONCLUSION

The hand gesture recognition system is becoming a crucial component in building efficient human-machine interaction, with promising applications across various technological fields. This study presents a comprehensive real-time system for hand gesture control of a quadcopter swarm by integrating MediaPipe for gesture recognition, ROS for communication, and Gazebo for simulation. MediaPipe, powered by a deep learning-based framework utilizing a Feedforward Neural Network (FNN), has demonstrated an impressive accuracy of 90%, making it an effective tool for developing gesture recognition applications. The high recognition accuracy and low latency observed in simulations confirm the system's feasibility for real-world deployment. Future enhancements include expanding the gesture set, improving environmental adaptability, and implementing real-world tests with physical quadcopters. Additionally, we aim to extend the system by enabling joint operation with external devices and expanding recognition to other human body features, including both static and motion-based hand gestures, although the system showed promising results in simulation, future work will focus on deploying it in real-world scenarios with physical drones under varying environmental conditions.

REFERENCES

[1] F. Ren and Y. Bao, *A review on human-computer interaction and intelligent robots*, vol. 19, no. 1. 2020. doi: 10.1142/S0219622019300052.

[2] R. K. Megalingam, V. S. Naick, M. Motheram, J. Yannam, N. C. Gutlapalli, and V. Sivanantham, "Robot operating system based autonomous navigation platform with human robot interaction," *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 21, no. 3, pp. 675–683, 2023, doi: 10.12928/TELKOMNIKA.v21i3.24257.

[3] Indriani, M. Harris, and A. S. Agoes, "Applying Hand Gesture Recognition for User Guide Application Using MediaPipe," *Proc. 2nd Int. Semin. Sci. Appl. Technol. (ISSAT 2021)*, vol. 207, no. Issat, pp. 101–108, 2021, doi: 10.2991/aer.k.211106.017.

[4] Yaseen, O. J. Kwon, J. Kim, S. Jamil, J. Lee, and F. Ullah, "Next-Gen Dynamic Hand Gesture Recognition: MediaPipe, Inception-v3 and LSTM-Based Enhanced Deep Learning Model," *Electron.*, vol. 13, no. 16, pp. 1–11, 2024, doi: 10.3390/electronics13163233.

[5] Y. Meng, H. Jiang, N. Duan, and H. Wen, "Real-Time Hand Gesture Monitoring Model Based on MediaPipe's Registerable System," *Sensors*, vol. 24, no. 19, 2024, doi: 10.3390/s24196262.

[6] B. Latif, N. Buckley, and E. L. Secco, "Hand Gesture and Human-Drone Interaction," *Lect. Notes Networks Syst.*, vol. 544 LNNS, no. October, pp. 299–308, 2023, doi: 10.1007/978-3-031-16075-2_20.

[7] A. S. Andoy, R. T. Tayane, and I. Supriadi, "Controlling the Dji Ryze Tello Drone Using Human Hand Gestures," vol. 2, no. 2, pp. 123–132, 2024.

[8] H. Lee and S. Park, "Sensing-Aware Deep Reinforcement Learning With HCI-Based Human-in-the-Loop Feedback for Autonomous Nonlinear Drone Mobility Control," *IEEE Access*, vol. 12, no. December 2023, pp. 1727–1736, 2024, doi: 10.1109/ACCESS.2023.3346917.

[9] A. Tahir, J. Böling, M. H. Haghbayan, H. T. Toivonen, and J. Plosila, "Swarms of Unmanned Aerial Vehicles — A Survey," *J. Ind. Inf. Integr.*, vol. 16, no. October, p. 100106, 2019, doi: 10.1016/j.jii.2019.100106.

[10] M. Yoo *et al.*, "Motion Estimation and Hand Gesture Recognition-Based Human–UAV Interaction Approach in Real Time," *Sensors*, vol. 22, no. 7, 2022, doi: 10.3390/s22072513.

[11] A. Muhamad, S. D. Panjaitan, and R. R. Yacoub, "Design and Development of Flight Controller for Quadcopter Drone Control," *Telecommun. Comput. Electr. Eng. J.*, vol. 1, no. 3, p. 279, 2024, doi: 10.26418/telectrical.v1i3.73681.

[12] G. Ostojić, S. Stankovski, B. Tejić, N. Dukić, and S. Tegeltija, "Design, control and application of quadcopter," *Int. J. Ind. Eng. Manag.*, vol. 6, no. 1, pp. 43–48, 2015, doi: 10.24867/ijiem-2015-1-106.

[13] A. Budiyanto, M. I. Ramadhan, I. Burhanudin, H. H. Triharminto, and B. Santoso, "Navigation control of Drone using Hand Gesture based on Complementary Filter Algorithm," *J. Phys. Conf. Ser.*, vol. 1912, no. 1, 2021, doi: 10.1088/1742-6596/1912/1/012034.

[14] M. Wameed and A. M. Alkamachi, "International Journal of INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING Hand Gestures Robotic Control Based on Computer Vision," *Orig. Res. Pap. Int. J. Intell. Syst. Appl. Eng. IJISAE*, vol. 2023, no. 2, pp. 1013–1021, 2024, [Online]. Available: www.ijisae.org

[15] D. Marek *et al.*, "Swarm of Drones in a Simulation Environment—Efficiency and Adaptation," *Appl. Sci.*, vol. 14, no. 9, 2024, doi: 10.3390/app14093703.

[16] A. Alvin, N. H. Shabrina, A. Ryo, and E. Christian, "Hand Gesture Detection for Sign Language using Neural Network with Mediapipe," *Ultim. Comput. J. Sist. Komput.*, vol. 13, no. 2, pp. 57–62, 2021, doi: 10.31937/sk.v13i2.2109.

[17] M. Arif *et al.*, "Teknik dan Multimedia Sistem Pendeteksi Tangan Berbasis Mediapipe dan OpenCV untuk Pengenalan Gerakan," *Biner J. Ilmu Komput.*, vol. 2, no. 2, pp. 173–177, 2024, [Online]. Available: https://journal.mediapublikasi.id/index.php/Biner

[18] A. Maarif, W. Rahmaniar, M. A. M. Vera, A. A. Nuryono, R. Majdoubi, and A. Cakan, "Artificial Potential Field Algorithm for Obstacle Avoidance in UAV Quadrotor for Dynamic Environment," *10th IEEE Int. Conf. Commun. Networks Satell. Comnetsat 2021 - Proc.*, no. October, pp. 184–189, 2021, doi: 10.1109/COMNETSAT53002.2021.9530803.

[19] S. Nur Budiman, S. Lestanti, S. Marselius Evvandri, and R. Kartika Putri, "Pengenalan Gestur Gerakan Jari Untuk Mengontrol Volume Di Komputer Menggunakan Library Opencv Dan Mediapipe," *Antivirus J. Ilm. Tek. Inform.*, vol. 16, no. 2, pp. 223–232, 2022, doi: 10.35457/antivirus.v16i2.2508.

[20] O. Mishra, P. Suryawanshi, Y. Singh, and S. Deokar, "A Mediapipe-Based Hand Gesture Recognition Home Automation System," *2023 2nd Int. Conf. Futur. Technol. INCOFT 2023*, no. April 2024, 2023, doi: 10.1109/INCOFT60753.2023.10425411.

[21] A. D. Agustiani, S. M. Putri, P. Hidayatullah, and M. R. Sholahuddin, "Penggunaan MediaPipe untuk Pengenalan Gesture Tangan Real-Time dalam Pengendalian Presentasi," vol. 16, no. 2, 2024.

[22] U. C. Patkar, V. Mandhalkar, A. Chavan, S. Songire, and H. Kothawade, "Robot Operating System: A Comprehensive Analysis and Evaluation," *Int. J. Intell. Syst. Appl. Eng.*, vol. 12, no. 7s, pp. 516–520, 2024.

[23] Z. Wei, F. Tian, Z. Qiu, Z. Yang, R. Zhan, and J. Zhan, "Research on Machine Vision-Based Control System for Cold Storage Warehouse Robots," *Actuators*, vol. 12, no. 8, 2023, doi: 10.3390/act12080334.

[24] S. Alexovič, M. Lacko, and J. Bačík, "Simulation of Multiple Autonomous Mobile Robots Using a Gazebo Simulator," *Lect. Notes Networks Syst.*, vol. 596 LNNS, no. January, pp. 339–351, 2023, doi: 10.1007/978-3-031-21435-6_30.

[25] S. Hadri, "Hand gestures for drone control using deep learning," no. December 2018, 2020, doi: 10.13140/RG.2.2.15939.02089.