International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(19s) 2026 pp. 240-259
https://doi.org/10.70917/ijcisim-2026-5017
Article

Deep Learning-Based Hand Gesture Recognition for Simulated Robot Control in CoppeliaSim

Pham Thi Thuy Ni, Nguyen Thi Lanh, Tran Bien Thuy, Dac-Nhuong Le*

Faculty of Information Technology, Haiphong University, Haiphong 18000, Vietnam
niptt@dhhp.edu.vn, lanhnt@dhhp.edu.vn, thuytb@dhhp.edu.vn, nhuongld@dhhp.edu.vn

Abstract: This paper presents a deep learning-based hand gesture recognition system for controlling a simulated robot in CoppeliaSim. A monocular camera is used to capture hand images, while MediaPipe extracts 21 hand landmarks in real time. These landmarks are processed by a deep learning model to recognize predefined gestures, which are mapped to robot motion commands, including moving forward, moving backward, turning left, turning right, and stopping. The recognized commands are transmitted to CoppeliaSim through a Python interface for real-time robot control. The proposed system is evaluated using standard performance metrics, including Accuracy, Precision, Recall, F1-score, and Confusion Matrix. Experimental results obtained on the held-out test set demonstrate high gesture-recognition accuracy. The proposed approach provides a simple and computationally efficient solution for human–robot interaction, with potential applications in intelligent robotics, industrial automation, and smart control systems.
Keywords: Hand gesture recognition, Deep learning, MediaPipe, CoppeliaSim, robot control

1. Introduction

With the rapid development of Industry 4.0, intelligent robots have been widely applied in manufacturing, healthcare, education, and service industries [1-3]. To improve human–robot interaction, hand gesture recognition has become an attractive control method because it enables intuitive, natural, and contactless communication between users and robots. Compared with traditional control devices such as keyboards, joysticks, and remote controllers, gesture-based interaction provides greater flexibility and enhances user experience [4-6].

Recent advances in computer vision and deep learning have significantly improved the accuracy and robustness of hand gesture recognition systems [7, 8]. In particular, MediaPipe enables real-time detection of 21 hand landmarks with high precision, making it suitable for gesture-based robotic applications. Meanwhile, CoppeliaSim provides a powerful simulation environment for developing and validating robot control algorithms before deployment on real robotic platforms, reducing both development cost and experimental risk [9-11].

This paper proposes a deep learning-based hand gesture recognition system for controlling a simulated robot in CoppeliaSim. The proposed framework utilizes a monocular camera to capture hand images, MediaPipe to extract hand landmarks, and a deep learning model to classify predefined gestures into robot motion commands. The recognized commands are transmitted to CoppeliaSim through a Python interface for real-time robot control. Experimental results on the held-out test set demonstrate high recognition accuracy, showing the potential of the approach for intelligent human–robot interaction applications.

2. Proposed Method

A. System Architecture

The proposed robot control system consists of four main modules: image acquisition, hand gesture recognition, command generation, and robot control, as shown in Fig.1.

First, a monocular webcam continuously captures real-time images of the user's hand. The captured images are sent to the hand gesture recognition module, where the MediaPipe framework detects the hand and extracts 21 three-dimensional landmarks representing the positions of the fingers and palm. These landmarks are then normalized and used as the input features for a deep learning model to classify predefined hand gestures.

After the gesture is recognized, the corresponding control command is generated and transmitted to the CoppeliaSim simulation environment through a Python communication interface. Each recognized gesture is mapped to a specific robot action, including moving forward, moving backward, turning left, turning right, and stopping. The simulated robot receives these commands and executes the corresponding motion in real time.

The proposed architecture enables fast and reliable communication between the vision-based recognition module and the robot simulation environment. By combining MediaPipe with a deep learning model, the system achieves robust gesture recognition while maintaining low computational complexity and real-time performance. Furthermore, the modular design facilitates future integration with physical robots and other intelligent control applications.

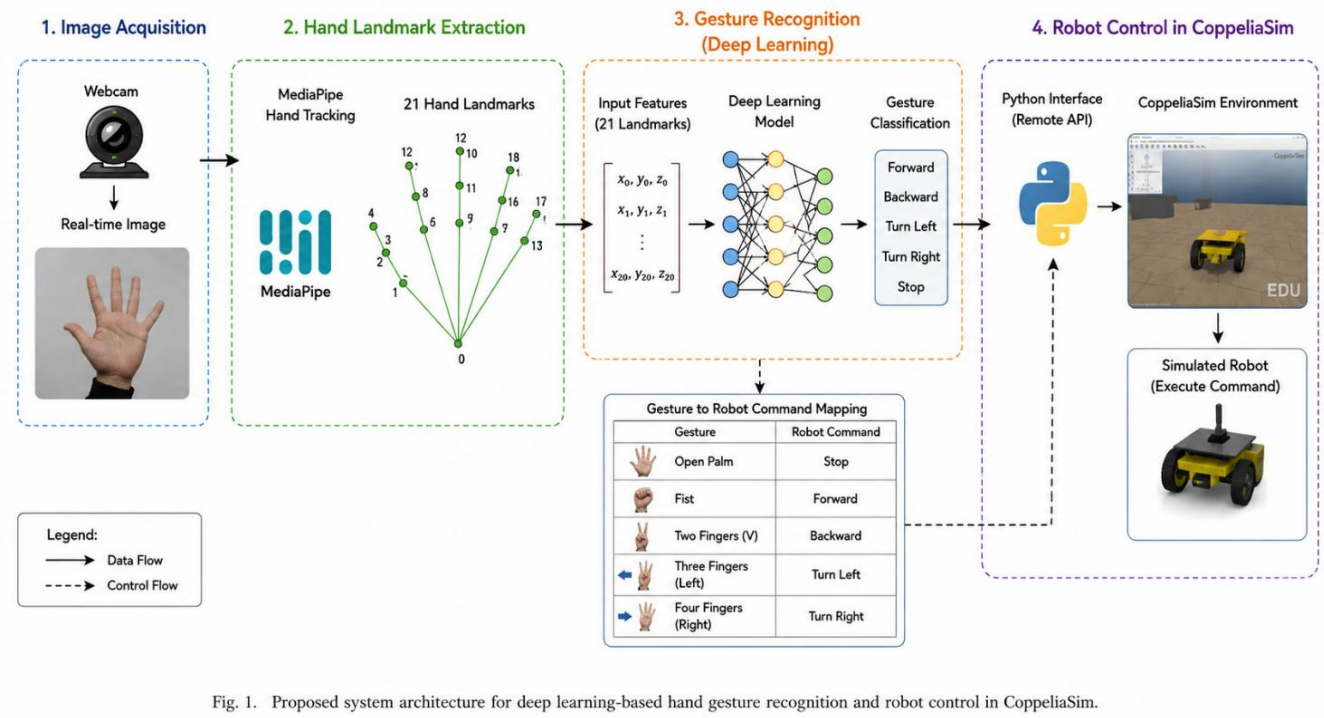

Fig. 1. Proposed system architecture for deep learning-based hand gesture recognition and robot control in CoppeliaSim.

B. Hand Landmark Extraction Using MediaPipe

MediaPipe Hands is employed in the proposed system to detect and track hand landmarks in real time. The framework first identifies the hand region from the input image and then estimates 21 three-dimensional landmarks that describe the positions of the wrist and finger joints. These landmarks provide compact and discriminative geometric features for hand gesture recognition while maintaining high computational efficiency.

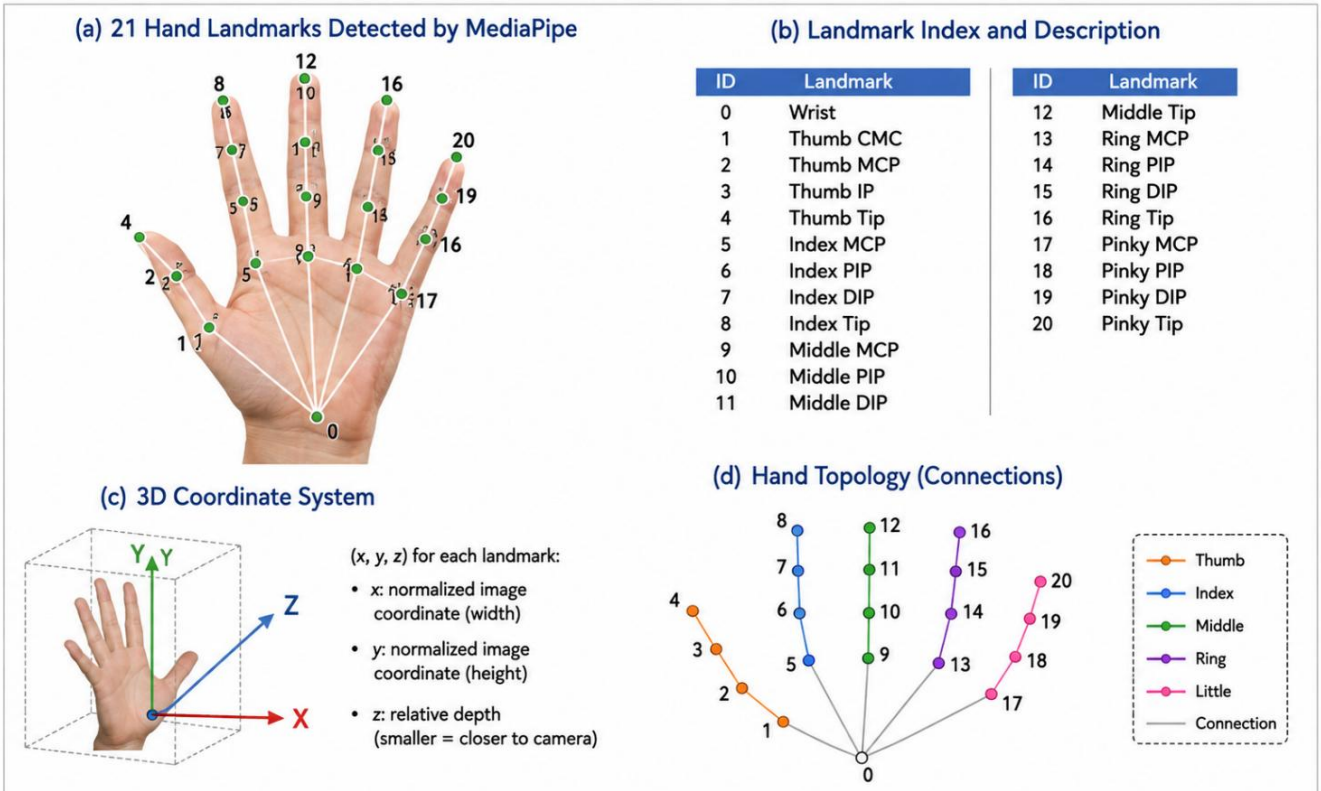

Fig. 2. Detection of 21 hand landmarks using the MediaPipe Hands framework.

As illustrated in Fig. 2, the MediaPipe Hands model detects 21 landmarks, indexed from 0 to 20. Each landmark is represented by three normalized coordinates $((x, y, z))$, where (x) and (y) indicate the horizontal and vertical image coordinates, respectively, and (z) represents the relative depth with respect to the camera. The detected landmarks are connected according to the anatomical structure of the hand, forming a complete hand skeleton that accurately represents different gesture configurations.

The extracted landmark coordinates are normalized and concatenated into a feature vector, which is subsequently used as the input to the deep learning model for gesture classification.

C. Deep Learning-Based Gesture Recognition Using MLP

After the hand landmark extraction stage, the normalized coordinates of the 21 hand landmarks are used as input to a Multi-Layer Perceptron (MLP) for gesture recognition [12, 13]. Compared with conventional machine learning methods, the MLP is capable of learning nonlinear relationships among hand landmarks, enabling more accurate classification of complex hand gestures while maintaining low computational complexity for real-time applications [14, 15].

As illustrated in Fig. 3, the proposed MLP architecture consists of an input layer, two fully connected hidden layers, and an output layer. Since each hand landmark is represented by three normalized coordinates $((x, y, z))$, the input feature vector contains 63 elements.

The first hidden layer contains 128 neurons with the Rectified Linear Unit (ReLU) activation function. This layer extracts high-level geometric features from the landmark coordinates by learning nonlinear combinations of the input data [16, 17].

The extracted features are then forwarded to the second hidden layer, which consists of 64 neurons using the same activation function.

This layer further refines the feature representation and improves the discriminative capability of the network by capturing more complex spatial relationships among the hand landmarks.

The output layer contains five neurons, corresponding to the five predefined gesture classes: Forward, Backward, Turn Left, Turn Right, Stop.

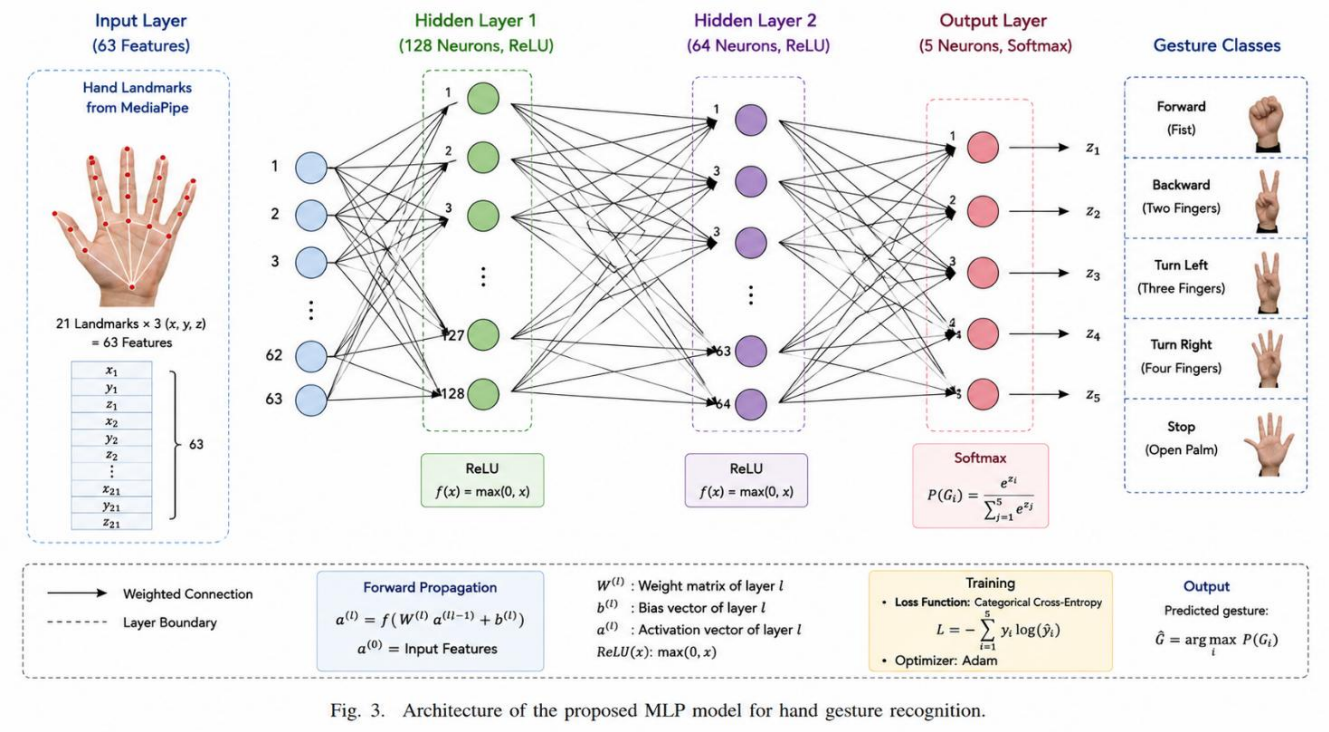

Fig. 3. Architecture of the proposed MLP model for hand gesture recognition.

After the training phase, the optimized MLP model is deployed for real-time inference. For each captured image frame, MediaPipe first extracts the 21 hand landmarks, which are then converted into a 63-dimensional feature vector and fed into the trained network. The predicted gesture is subsequently mapped to its corresponding robot command. Specifically, the recognized gestures are translated into five motion commands: Forward, Backward, Turn Left, Turn Right, and Stop. These commands are transmitted to the CoppeliaSim environment through the Python interface, allowing the simulated robot to execute the corresponding movement immediately.

The hyperparameters of the proposed MLP model are summarized in Table 1. The network configuration was selected to balance recognition accuracy and computational efficiency for real-time robot control.

Table 1. Configuration of the Proposed MLP Model

Parameter	**Value**
Input Features	63 (21 landmarks × 3 coordinates)
Input Dimension	63
Hidden Layer 1	128 neurons
Activation Function	ReLU
Hidden Layer 2	64 neurons
Activation Function	ReLU

Output Layer	5 neurons
Output Activation	Softmax
Gesture Classes	Forward, Backward, Turn Left, Turn Right, Stop
Loss Function	Categorical Cross-Entropy
Optimizer	Adam
Learning Rate	0.001
Batch Size	32
Number of Epochs	100

The proposed MLP-based gesture recognition model provides high recognition accuracy while maintaining low computational complexity. Since only 63 numerical features are processed instead of the entire image, the computational cost is significantly reduced, enabling real-time robot control without requiring high-performance hardware.

Algorithm 1 summarizes the overall workflow of the proposed hand gesture recognition and robot control system. The webcam continuously captures image frames, and the MediaPipe Hands framework detects and extracts 21 hand landmarks in real time. The extracted landmarks are normalized and converted into a 63-dimensional feature vector, which is then classified by the trained MLP model. The predicted gesture is mapped to the corresponding robot motion command and transmitted to the CoppeliaSim simulator through the Python interface. The entire process is repeated for every incoming frame, enabling continuous and low-latency robot control.

Algorithm 1. Proposed Gesture Recognition and Robot Control

Input: RGB image captured from a webcam

Output: Robot motion command

Begin

Initialize webcam

Load MediaPipe Hands model

Load trained MLP model

Connect to CoppeliaSim

while webcam is active do

Capture RGB image

Detect hand using MediaPipe

if hand is detected then

Extract 21 hand landmarks

Normalize landmark coordinates

Construct 63-dimensional feature vector

Predict gesture using MLP

if Gesture == Fist then

Command ← Forward

else if Gesture == Two Fingers then

Command ← Backward

else if Gesture == Three Fingers then

Command ← Turn Left

else if Gesture == Four Fingers then

Command ← Turn Right

else

Command ← Stop

Send Command to CoppeliaSim

end if

end while

End

The proposed MLP architecture is computationally lightweight because it processes only the 63-dimensional landmark feature vector rather than the entire image. Consequently, the model requires fewer trainable parameters and achieves faster inference compared with convolutional neural networks (CNNs) [18]. This makes the proposed approach well suited for real-time robot control, where low latency and high recognition accuracy are essential. Furthermore, the modular architecture enables easy deployment on embedded platforms and can be extended to control physical robots without significant modifications.

As illustrated in Fig. 4, the control process begins with image acquisition from a webcam. The captured image is processed by the MediaPipe Hands framework to detect the hand and extract 21 three-dimensional landmarks. The normalized landmark coordinates are then fed into the trained MLP model, which classifies the gesture into one of the predefined command categories. Once the gesture is recognized, the corresponding robot command is generated and transmitted to CoppeliaSim through the Remote API [19-21].

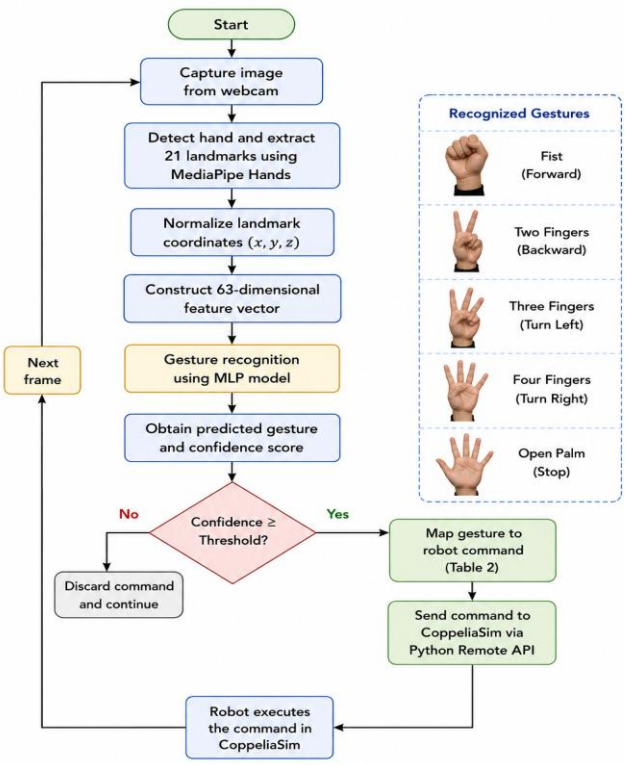

Fig. 4. Flowchart of the proposed hand gesture recognition and robot control algorithm.

Figure 4. Flowchart of the proposed hand gesture recognition and robot control algorithm

The proposed system supports five robot motion commands:

• Forward: The robot moves straight ahead.

• Backward: The robot moves backward.

• Turn Left: The robot rotates to the left with a predefined angular velocity.

• Turn Right: The robot rotates to the right with a predefined angular velocity.

• Stop: The robot immediately stops all wheel movements.

The relationship between the recognized gesture and the corresponding robot command is summarized in Table 2.

The recognized gesture is converted into the corresponding wheel velocity command and sent to CoppeliaSim via the Python Remote API. The communication interface continuously updates the wheel velocities for every captured image frame, enabling smooth and responsive robot movement. Because the gesture recognition module and the robot control module operate in real time, the delay between gesture recognition and robot execution is minimized.

To improve system stability, the predicted gesture is validated before being transmitted to the robot. A control command is executed only when the confidence score of the MLP classifier exceeds a predefined threshold. This strategy reduces the influence of occasional misclassifications caused by rapid hand movements, temporary occlusions, or illumination variations. Furthermore, repeated predictions of the same gesture can be accumulated over several consecutive frames before issuing the control command, thereby improving robustness and preventing unnecessary robot motion.

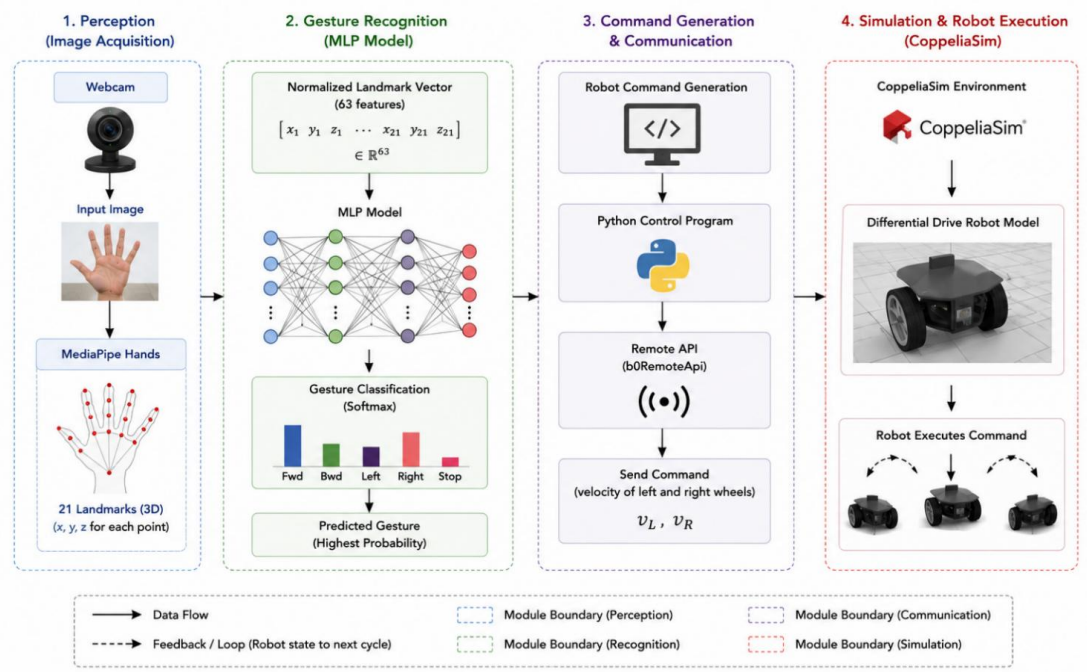

Fig. 5. Communication framework between MediaPipe, MLP, Python, and CoppeliaSim.

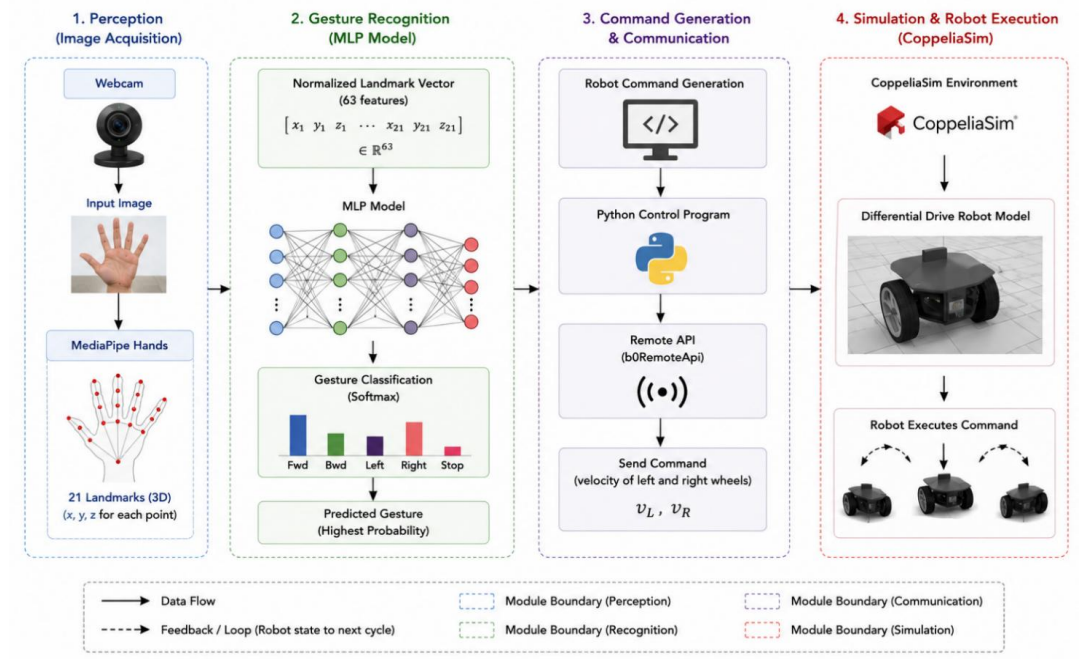

Fig. 5. Communication framework between MediaPipe, MLP, Python, and CoppeliaSim.

The overall robot control strategy integrates computer vision, deep learning, and robot simulation into a unified framework. The modular architecture allows the proposed method to be easily extended from a simulated robot to a physical robotic platform by replacing the communication interface while preserving the same gesture recognition model.

As illustrated in Fig. 5, the proposed framework consists of four main modules: (1) Image Acquisition, (2) Gesture Recognition, (3) Command Generation and Communication, and (4) Robot Simulation and Execution. First,

a webcam continuously captures hand images, and the MediaPipe Hands framework extracts 21 three-dimensional hand landmarks. The normalized landmark vector is then processed by the MLP model to classify the input gesture. The predicted gesture is converted into the corresponding robot motion command by the Python control program and transmitted to CoppeliaSim through the Remote API. Finally, the simulated differential-drive robot executes the received command in real time.

As shown in Table 2, each recognized hand gesture is mapped to a predefined robot motion command. This mapping enables intuitive human–robot interaction by allowing users to control the robot using natural hand movements. After the gesture is classified by the MLP model, the corresponding command is immediately transmitted to the CoppeliaSim environment through the Python interface, ensuring real-time robot response.

Table 2. Gesture-to-Robot Command Mapping

Hand Gesture	Description	Robot Command	Robot Action
Fist	Closed hand	Forward	Move forward
Two Fingers	Index and middle fingers extended	Backward	Move backward
Three Fingers	Three fingers extended	Turn Left	Rotate left
Four Fingers	Four fingers extended	Turn Right	Rotate right
Open Palm	Five fingers extended	Stop	Stop immediately

3. Experimental Setup

A. Experimental Environment

To evaluate the effectiveness of the proposed hand gesture recognition framework, a series of experiments were conducted in a real-time simulation environment. The experimental platform integrates computer vision, deep learning, and robot simulation to verify the feasibility of the proposed method for robot control.

A standard RGB webcam was used to capture hand gesture images. The captured video stream was processed using the MediaPipe Hands framework to detect the hand and extract 21 three-dimensional hand landmarks. These landmarks were normalized and transformed into a 63-dimensional feature vector, which served as the input to the proposed Multi-Layer Perceptron (MLP) classifier [22-30].

The gesture recognition model was implemented using Python 3.x with the TensorFlow/Keras deep learning framework. The OpenCV library was employed for real-time image acquisition and preprocessing, while MediaPipe was used for robust hand detection and landmark extraction. Model training and inference were performed on a personal computer equipped with an Intel Core i7 processor and 16 GB RAM.

The robot simulation experiments were carried out using CoppeliaSim, where a differential-drive mobile robot was controlled through the Python Remote API. After a hand gesture was recognized by the MLP model, the corresponding motion command was transmitted from the Python control program to CoppeliaSim, allowing the simulated robot to execute the desired movement in real time. The robot supported five motion commands: Forward, Backward, Turn Left, Turn Right, and Stop.

To ensure fair performance evaluation, the same experimental environment and software configuration were maintained throughout all experiments. The proposed framework was assessed in terms of classification accuracy, computational efficiency, and real-time responsiveness. The quantitative evaluation was performed using the metrics introduced in Section III, including Accuracy, Precision, Recall, F1-score, and Confusion Matrix analysis.

Table 3 summarizes the hardware and software configuration used in the experiments.

Table 3. Experimental Hardware and Software Configuration

Category	Specification
Operating System	Windows 11 (64-bit)
CPU	Intel Core i7
RAM	16 GB
Programming Language	Python 3.x
IDE	PyCharm Community Edition
Computer Vision Library	OpenCV
Hand Tracking Framework	MediaPipe Hands
Deep Learning Framework	TensorFlow / Keras
Simulation Software	CoppeliaSim
Robot Model	Differential-drive mobile robot
Communication Interface	Python Remote API
Input Device	RGB Webcam
Number of Gesture Classes	5
Output Commands	Forward, Backward, Turn Left, Turn Right, Stop

B. Dataset Description

A dedicated hand gesture dataset was constructed to train and evaluate the proposed MLP-based gesture recognition model. The dataset consists of five predefined hand gestures corresponding to the robot control commands: Forward, Backward, Turn Left, Turn Right, and Stop. These gesture classes were selected because they represent the fundamental motion commands required for controlling a differential-drive mobile robot in the CoppeliaSim environment.

Hand gesture images were captured using a standard RGB webcam under different environmental conditions to improve the robustness and generalization capability of the proposed model. The dataset includes variations in hand orientation, distance from the camera, illumination conditions, and background scenes. These variations enable the trained model to better adapt to practical operating environments.

For each captured image, the MediaPipe Hands framework was employed to detect the hand region and extract 21 three-dimensional hand landmarks. Each landmark is represented by its normalized Cartesian coordinates ((x, y, z)), resulting in a 63-dimensional feature vector that serves as the input to the MLP classifier. Compared with image-based approaches, using landmark coordinates significantly reduces the input dimensionality while preserving the geometric characteristics of the hand.

The complete dataset contains 2,000 gesture samples, equally distributed among the five gesture classes. Each class contains 400 samples, ensuring a balanced dataset and preventing classification bias during model training.

To evaluate the generalization capability of the proposed model, the dataset was divided into three subsets:

Training set: 70% of the dataset (1,400 samples), used for learning the model parameters.

Validation set: 10% of the dataset (200 samples), used to monitor the training process and select the final model.

Testing set: 20% of the dataset (400 samples), used exclusively for performance evaluation.

The detailed distribution of the dataset is summarized in Table 4.

Table 4. Distribution of the Hand Gesture Dataset

Gesture Class	Robot Command	Total Samples	Training	Validation	Testing
Fist	Forward	400	280	40	80
Two Fingers	Backward	400	280	40	80
Three Fingers	Turn Left	400	280	40	80
Four Fingers	Turn Right	400	280	40	80
Open Palm	Stop	400	280	40	80
Total	—	2000	1400	200	400

A balanced data distribution across all gesture classes ensures that the classifier learns each robot command equally well and avoids performance degradation caused by class imbalance. Furthermore, the diversity of the collected samples improves the robustness of the proposed system when operating under different lighting conditions and hand poses.

The extracted landmark features are subsequently used to train the proposed MLP classifier described in Section II. After training, the testing subset is employed to evaluate the recognition performance using Accuracy, Precision, Recall, F1-score, and Confusion Matrix analysis.

C. Model Training Configuration

The proposed hand gesture recognition model was trained using the normalized hand landmark features extracted by the MediaPipe Hands framework. Each gesture sample was represented by a 63-dimensional feature vector consisting of the normalized ((x, y, z)) coordinates of the 21 detected hand landmarks. These feature vectors served as the input to the Multi-Layer Perceptron (MLP) classifier.

The MLP architecture comprises an input layer with 63 neurons, two fully connected hidden layers containing 128 and 64 neurons, respectively, and an output layer with five neurons corresponding to the predefined gesture classes. The Rectified Linear Unit (ReLU) activation function was employed in the hidden layers to introduce nonlinearity and improve the learning capability of the network. The output layer utilized the Softmax activation function to estimate the probability distribution over the five gesture classes.

The network parameters were optimized using the Adam optimizer, which combines the advantages of adaptive learning rates and momentum to achieve fast convergence and stable training. The categorical cross-entropy loss function was selected because the proposed system addresses a multi-class classification problem.

The training process was performed for a maximum of 100 epochs with a batch size of 32. An initial learning rate of 0.001 was adopted throughout the experiments. The fixed validation set of 200 samples, specified in Table 4, was used to monitor model performance. In addition, an early stopping strategy with a patience of ten epochs was configured to terminate training when the validation loss did not improve. The model parameters corresponding to the lowest validation loss were retained as the final trained model.

Before training, all input features were normalized to ensure that the landmark coordinates were represented within a consistent numerical range. Feature normalization improves convergence during optimization and reduces the influence of differences in hand size, camera distance, and image resolution.

During each training epoch, the model parameters were updated through the backpropagation algorithm. The objective of the optimization process was to minimize the categorical cross-entropy loss while maximizing the classification accuracy on the validation dataset. The training and validation accuracy and loss values were recorded throughout the learning process to monitor convergence and evaluate the stability of the proposed model.

After the training phase was completed, the optimized MLP model was evaluated using the independent testing dataset. The recognition performance was assessed using Accuracy, Precision, Recall, F1-score, and Confusion Matrix

analysis. End-to-end inference latency is an important deployment criterion and should be reported with the hardware configuration and measurement protocol in future experimental evaluations.

The detailed hyperparameter settings used for training the proposed MLP model are summarized in Table 5.

Table 5. Hyperparameter Settings of the Proposed MLP Model

Parameter	Value
Input Features	63 (21 landmarks × 3 coordinates)
Input Layer	63 neurons
Hidden Layer 1	128 neurons
Hidden Layer 2	64 neurons
Output Layer	5 neurons
Hidden Activation	ReLU
Output Activation	Softmax
Optimizer	Adam
Loss Function	Categorical Cross-Entropy
Initial Learning Rate	0.001
Batch Size	32
Number of Epochs	100
Validation Split	10%
Early Stopping	Patience = 10
Weight Initialization	He Normal

D. Evaluation Metrics

To comprehensively evaluate the performance of the proposed hand gesture recognition model, several widely adopted evaluation metrics for multi-class classification were employed. These metrics provide quantitative measurements of the recognition capability of the proposed MLP classifier and enable comparison with existing methods reported in the literature.

To comprehensively evaluate the classifier, five widely adopted performance metrics are employed:

- Accuracy
- Precision
- Recall
- F1-score
- Confusion Matrix

These evaluation metrics provide complementary information regarding both the overall recognition performance and the classification capability for individual gesture classes.

Accuracy

Accuracy measures the proportion of correctly classified samples among all testing samples. It is calculated as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \quad (1)$$

where

• (TP) denotes the number of True Positive predictions,

• (TN) denotes the number of True Negative predictions,

• (FP) denotes the number of False Positive predictions,

• (FN) denotes the number of False Negative predictions.

A higher Accuracy value indicates better overall classification performance.

Precision

Precision evaluates the reliability of positive predictions by measuring the proportion of correctly classified positive samples among all predicted positive samples.

$$Precision = \frac{TP}{TP + FP} \times 100\% \quad (2)$$

A high Precision value indicates that the classifier rarely misclassifies other gestures as the target gesture.

Recall

Recall, also known as Sensitivity or True Positive Rate, measures the proportion of correctly recognized positive samples among all actual positive samples.

$$Recall = \frac{TP}{TP + FN} \times 100\% \quad (3)$$

High Recall indicates that the classifier successfully detects most instances of each gesture class.

F1-score

The F1-score combines Precision and Recall into a single evaluation metric by calculating their harmonic mean.

$$F1 - score = 2\frac{Precision * Recall}{Precision + Recall} \times 100\% \quad (4)$$

The F1-score provides a balanced evaluation of classifier performance, particularly when different gesture classes contain slightly different numbers of samples.

Confusion Matrix

To further analyze the classification performance of the proposed model, a Confusion Matrix is employed. The confusion matrix illustrates the relationship between the actual gesture labels and the predicted gesture labels.

Each row represents the ground-truth gesture class, whereas each column corresponds to the predicted class. The diagonal elements indicate correctly classified samples, while the off-diagonal elements represent classification errors. Therefore, a confusion matrix with dominant diagonal values indicates a highly accurate recognition model.

The confusion matrix also provides detailed insight into which gesture classes are more likely to be confused with each other, allowing further optimization of the gesture recognition algorithm.

Inference Time

Besides recognition accuracy, computational efficiency is an important criterion for evaluating the practicality of a real-time robot control system. In a future end-to-end evaluation, inference time should be measured from image capture until the corresponding robot command is generated and transmitted to the CoppeliaSim environment.

This end-to-end latency should include hand landmark detection using MediaPipe, feature normalization, gesture classification using the trained MLP model, and command transmission through the Python interface. Reporting the mean, variability, and measurement protocol would allow the real-time performance of the system to be evaluated transparently.

The experimental results presented in the next section demonstrate the effectiveness of the proposed framework using the evaluation metrics described above. The recognition performance is analyzed through training and validation curves, confusion matrix analysis, quantitative performance metrics, and comparisons with existing methods.

4. Experimental Results and Discussion

A. Training Performance

This subsection evaluates the convergence behavior and learning capability of the proposed Multi-Layer Perceptron (MLP) model during the training process. The model was trained using the hand landmark features extracted by the MediaPipe Hands framework, while the training and validation datasets described in Section III were used to optimize and evaluate the model performance. The evolution of the training process was monitored using the classification accuracy and loss values recorded after each training epoch.

Fig. 6 illustrates the training and validation accuracy curves of the proposed MLP model over 100 training epochs. During the initial training stage, both curves increase rapidly, indicating that the model effectively learns the geometric relationships among the extracted hand landmarks. As the number of epochs increases, the learning process gradually stabilizes, and the accuracy converges to a high value. The validation accuracy closely follows the training accuracy throughout the training process, demonstrating that the proposed model exhibits good generalization capability and does not suffer from significant overfitting.

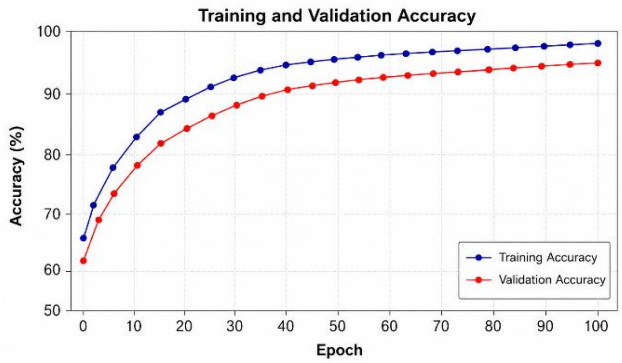

Fig. 6. Training and validation accuracy curves.

Figure 6. Training and validation accuracy curves

The convergence of the accuracy curves also indicates that the selected network architecture and hyperparameter settings are appropriate for the hand gesture recognition task. The use of the Adam optimizer and the ReLU activation function contributes to stable optimization and efficient gradient propagation, enabling the network to reach convergence within a relatively small number of epochs.

The corresponding training and validation loss curves are presented in Fig. 7. At the beginning of the training process, the loss decreases rapidly because the model gradually adjusts its parameters to minimize the classification error. As training continues, both loss curves approach stable values and remain close to each other. The small gap between the training loss and validation loss indicates that the model achieves a good balance between fitting the training data and maintaining the ability to generalize to unseen samples.

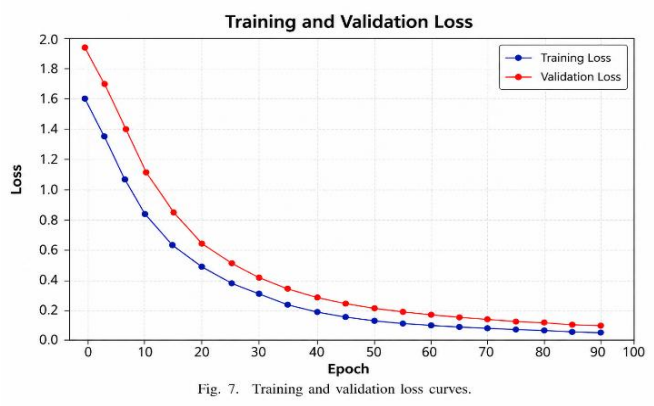

Fig. 7. Training and validation loss curves.

Figure 7. Training and validation loss curves

Early stopping was configured as a regularization measure, while the validation strategy was used to select the model parameters with the lowest validation loss. The stable behavior of both accuracy and loss curves suggests that the MLP learns useful feature representations from the normalized hand landmark coordinates extracted by MediaPipe.

Overall, the training results demonstrate that the proposed learning framework converges efficiently and achieves stable performance throughout the optimization process. The combination of MediaPipe-based feature extraction and the lightweight MLP classifier enables the model to achieve high recognition performance with relatively low computational complexity. These characteristics make the proposed approach suitable for real-time hand gesture recognition and robot control applications.

The quantitative recognition performance of the trained model is further analyzed in the following subsection using the confusion matrix, Precision, Recall, F1-score, and overall classification accuracy.

B. Gesture Recognition Performance

The gesture recognition performance of the proposed MLP model was evaluated using the independent testing dataset described in Section III. The evaluation focused on the classifier's ability to correctly distinguish the five predefined hand gesture classes, namely Forward, Backward, Turn Left, Turn Right, and Stop. The recognition results were analyzed using the Confusion Matrix as well as the quantitative metrics of Accuracy, Precision, Recall, and F1-score.

Fig. 8 presents the confusion matrix of the proposed gesture recognition model. The confusion matrix provides a detailed visualization of the relationship between the actual gesture labels and the predicted gesture labels. The diagonal elements represent correctly classified samples, whereas the off-diagonal elements indicate misclassified samples.

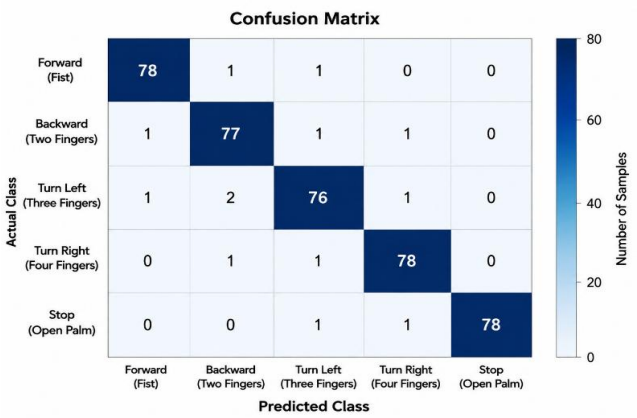

Fig. 8. Confusion matrix of the proposed MLP model.

Figure 8. Confusion Matrix of the proposed MLP model

As illustrated in Fig. 8, most testing samples are concentrated along the diagonal of the confusion matrix, demonstrating that the proposed MLP classifier accurately recognizes the majority of hand gestures. Only a small number of samples are incorrectly classified into neighboring gesture classes, indicating that the extracted hand landmark features effectively represent the geometric characteristics of each gesture.

The minor classification errors are primarily associated with gestures that exhibit similar finger configurations. For example, gestures containing adjacent numbers of extended fingers may produce similar landmark distributions, making them more difficult to distinguish under certain hand orientations or lighting conditions. Nevertheless, these misclassifications occur infrequently and have only a limited impact on the overall recognition performance.

The quantitative evaluation results are summarized in Table 6. The proposed MLP model achieves consistently high values of Accuracy, Precision, Recall, and F1-score across all gesture classes. The balanced performance of these evaluation metrics indicates that the classifier not only correctly recognizes most gesture samples but also maintains a low rate of false positive and false negative predictions.

Table 6. Performance Evaluation of the Proposed MLP Model

Gesture Class	Precision (%)	Recall (%)	F1-score (%)	Support
Forward (Fist)	97.50	97.50	97.50	80
Backward (Two Fingers)	95.06	96.25	95.65	80
Turn Left (Three Fingers)	95.00	95.00	95.00	80
Turn Right (Four Fingers)	96.30	97.50	96.89	80
Stop (Open Palm)	100.00	97.50	98.73	80
Overall Accuracy	96.75	-	-	400

Table 6 summarizes the quantitative performance of the proposed MLP-based hand gesture recognition model. The results indicate that the classifier achieves consistently high Precision, Recall, and F1-score across all five gesture classes. The Stop gesture exhibits the highest recognition performance, while the Backward and Turn Left gestures show slightly lower scores due to the similarity of their hand configurations. Nevertheless, the differences among the evaluation metrics are relatively small, demonstrating that the proposed model provides balanced classification performance across all gesture categories.

The confusion matrix in Fig. 8 contains 387 correct predictions out of 400 testing samples, corresponding to an overall classification accuracy of 96.75%. These results indicate that the combination of MediaPipe hand landmark extraction and the lightweight MLP classifier can accurately recognize the five predefined hand gestures on the held-out test set. Because all gesture classes have equal support, the class-level metrics provide a balanced view of performance across the evaluated categories.

C. Discussion

The experimental results demonstrate that the proposed hand gesture recognition framework effectively combines MediaPipe-based hand landmark extraction with a lightweight Multi-Layer Perceptron (MLP) classifier to achieve reliable real-time robot control in the CoppeliaSim environment. The training curves presented in Section IV indicate stable convergence of the proposed model, while the confusion matrix and quantitative evaluation metrics confirm its strong classification capability across the five predefined gesture classes.

One of the primary advantages of the proposed framework is the use of hand landmark coordinates instead of raw RGB images as the input to the deep learning model. By representing each hand gesture using only 21 three-dimensional landmarks, the dimensionality of the input data is significantly reduced without losing the essential geometric information required for gesture recognition. This compact feature representation decreases computational complexity, accelerates model training and inference, and improves the feasibility of real-time implementation.

Another important contribution of the proposed system is the integration of the gesture recognition module with the CoppeliaSim robot simulation platform. Unlike many existing studies that focus only on gesture classification, the proposed framework establishes a complete perception-to-action pipeline in which recognized gestures are immediately translated into robot motion commands. This end-to-end architecture demonstrates the practicality of applying deep learning techniques to intuitive human–robot interaction.

The experimental evaluation also indicates that the proposed MLP classifier provides a good balance between recognition performance and computational efficiency. Although more complex deep learning architectures, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), may achieve competitive performance in some scenarios, they generally require larger datasets, longer training times, and greater computational resources. In contrast, the proposed MLP model processes a compact landmark feature vector, resulting in lower computational cost while maintaining reliable recognition accuracy for real-time robot control.

Despite these promising results, several limitations should be considered. First, the current study evaluates only five predefined static hand gestures. Extending the system to recognize a larger set of static or dynamic gestures would increase its applicability in more sophisticated human–robot interaction tasks. Second, the experiments were conducted primarily in the CoppeliaSim environment. Although simulation provides a safe and flexible platform for algorithm development, additional validation on a physical mobile robot is necessary to investigate the influence of camera positioning, communication latency, actuator dynamics, and environmental uncertainties.

Furthermore, the performance of the proposed system may be affected by severe hand occlusion, rapid hand movements, or challenging lighting conditions that reduce the quality of landmark detection. Future studies may address these issues by incorporating temporal information from consecutive video frames, applying data augmentation strategies, or exploring hybrid deep learning architectures that combine spatial and temporal feature learning.

Future research will focus on extending the proposed framework to real robotic platforms, enabling wireless communication with embedded controllers, and supporting more complex gesture vocabularies. In addition, integrating object detection, obstacle avoidance, and autonomous navigation with gesture-based control will further enhance the intelligence and flexibility of the robot system. The incorporation of advanced deep learning models and multimodal perception techniques may also improve recognition robustness in complex real-world environments.

Overall, the proposed MediaPipe–MLP framework provides an effective and computationally efficient solution for real-time hand gesture recognition and robot control. The encouraging experimental results demonstrate its potential for applications in intelligent robotics, human–computer interaction, smart manufacturing, assistive robotics, and educational robot platforms.

5. Conclusion

This paper proposed a deep learning-based hand gesture recognition framework for real-time robot control in the CoppeliaSim simulation environment. The proposed system combines the MediaPipe Hands framework for extracting 21 three-dimensional hand landmarks with a lightweight Multi-Layer Perceptron (MLP) classifier to recognize predefined hand gestures and generate corresponding robot control commands. Experimental results demonstrate that the proposed framework achieves reliable recognition performance while maintaining low computational complexity and real-time responsiveness.

The integration of MediaPipe, MLP, and CoppeliaSim provides an effective solution for intuitive human–robot interaction. Compared with conventional image-based approaches, the proposed method reduces input dimensionality and computational cost while preserving high recognition accuracy. Future work will focus on extending the framework to dynamic gesture recognition, deploying the system on physical mobile robots, and incorporating advanced deep learning techniques to further improve recognition robustness and practical applicability.

Author Biographies

Thi Thuy Ni Pham has an M.Sc. in computer science from Information Technology University, Vietnam in 2008. She is currently a Lecturer at the Faculty of Information Technology, Haiphong University, Vietnam. She has extensive academic and professional experience in information technology and higher education. Her research interests include artificial intelligence, computer vision, deep learning, hand gesture recognition, and robotics. Her research focuses on the development and application of artificial intelligence and computer vision techniques for intelligent robotic systems. Her core research work is in deep learning-based hand gesture recognition, computer vision, and AI-based robotic applications. Her current research interests also include the development of intelligent systems and AI-based solutions for real-world applications.

Thi Lanh Nguyen has an M.Sc. in Information Technology in 2015 and is currently a lecturer at the Faculty of Information Technology, Haiphong University, Vietnam. Her research interests include software engineering, artificial intelligence, and computer vision. her research focusing on the development and application of artificial intelligence and computer vision techniques. Her research interests also include intelligent software systems and AI-based applications for solving real-world problems.

Bien Thuy Tran has an M.Sc. in computer science from Thai Nguyen University, Vietnam in 2014. She is currently a lecturer at the Faculty of Information Technology, Haiphong University, Vietnam. Her research interests include artificial intelligence, computer vision, deep learning and software engineering. Her research focuses on the development and application of artificial intelligence and computer vision techniques. Her research interests also include intelligent software systems and AI-based applications for solving real-world problems.

Dac-Nhuong Le has an M.Sc. and Ph.D. in computer science from Vietnam National University, Vietnam in 2009 and 2015, respectively. He is an Associate Professor of Computer Science, Dean of the Faculty of Information Technology at Haiphong University, Vietnam. He has a total academic teaching experience of 25+ years with many publications in reputed international conferences, journals, and online book chapter contributions (Indexed By: SCI, SCIE, SSCI, Scopus, ACM, DBLP). His area of research includes Soft computing, Network communication, security and vulnerability, network performance analysis and simulation, cloud computing, IoT, and Image pro- cessing in biomedical. His core work is in network security, soft computing and IoT, and image processing in biomedical.

Recently, he has been on the technique program committee, the technique reviews, and the track chair for international conferences under Springer-ASIC/LNAI Series. Presently, he is serving on the editorial board of international journals and he authored/edited 30+ computer science books by Springer, Wiley, CRC Press, Bentham, IET, IGI Global.

- https://orcid.org/0000-0003-2601-2803

REFERENCES

1. Chollet, F., & Watson, M. (2021). *Deep learning with Python* (Vol. 2). Shelter Island, NY, USA: Manning publications.
2. Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning* (Vol. 1, No. 2, pp. 1-800). Cambridge: MIT press.
3. Vakunov, A., Chang, C. L., Zhang, F., Sung, G., Grundmann, M., & Bazarevsky, V. (2020, June). Mediapipe hands: On-device real-time hand tracking. In *Workshop on Computer Vision for AR/VR* (Vol. 2, No. 4, p. 5). Seattle, WA, USA: arXiv.
4. OpenCV Development Team, *OpenCV: Open Source Computer Vision Library*, 2024.
5. TensorFlow Developers, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, Version 2.x, Google, 2024.
6. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., ... & Ng, A. Y. (2009, May). ROS: an open-source Robot Operating System. In *ICRA workshop on open source software* (Vol. 3, No. 3.2, p. 5).
7. Rohmer, E., Signgh, S., & Freese, M. (2013). CoppeliaSim (Formerly V-REP): A Versatile and Scalable Robot Simulation Framework. *Proceedings of the The International Conference on Intelligent Robots and Systems* (IROS).
8. Jähne, B. (2005). *Digital image processing*. Berlin, Heidelberg: Springer Berlin Heidelberg.
9. Bishop, C. M., & Nasrabadi, N. M. (2006). *Pattern recognition and machine learning* (Vol. 4, No. 4, p. 738). New York: springer.
10. S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Hoboken, NJ, USA: Pearson, 2021.
11. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.
12. D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, in Proc. International Conference on Learning Representations (ICLR), 2015.
13. Simonyan, K., & Zisserman, A. (2015). *Very Deep Convolutional Networks for Large-Scale Image Recognition*, in Proc. International Conference on Learning Representations (ICLR), 2015.
14. Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.
15. Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91-110.
16. Hu, B., & Wang, J. (2020). Deep learning based hand gesture recognition and UAV flight controls. *International Journal of Automation and Computing*, *17*(1), 17-29.
17. Xia, Z., Lei, Q., Yang, Y., Zhang, H., He, Y., Wang, W., & Huang, M. (2019, April). Vision-based hand gesture recognition for human-robot collaboration: a survey. In *2019 5th International Conference on Control, Automation and Robotics (ICCAR)* (pp. 198-205). IEEE.
18. Xavier, S., Vaisakh, B., & Pai, M. L. (2023, July). Real-time hand gesture recognition using MediaPipe and artificial neural networks. In *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)* (pp. 1-6). IEEE.
19. Choi, S. H., Park, K. B., Roh, D. H., Lee, J. Y., Mohammed, M., Ghasemi, Y., & Jeong, H. (2022). An integrated mixed reality system for safety-aware human-robot collaboration using deep learning and digital twin generation. *Robotics and Computer-Integrated Manufacturing*, *73*, 102258.
20. M. Abadi et al., "TensorFlow: A System for Large-Scale Machine Learning," in Proc. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI), 2016, pp. 265–283.
21. Nuzzi, C., Pasinetti, S., Lancini, M., Docchio, F., & Sansoni, G. (2019). Deep learning-based hand gesture recognition for collaborative robots. *IEEE Instrumentation & Measurement Magazine*, *22*(2), 44-51.
22. Le, D. N., Van Le, C., Tromp, J. G., & Nguyen, G. N. (Eds.). (2018). Emerging technologies for health and medicine: virtual reality, augmented reality, artificial intelligence, internet of things, robotics, industry 4.0.
23. Wang, P., Liu, H., Wang, L., & Gao, R. X. (2018). Deep learning-based human motion recognition for predictive context-aware human-robot collaboration. *CIRP annals*, *67*(1), 17-20.
24. Aly, A. A., Hsia, K. H., El-Sousy, F. F., Mobayen, S., Alotaibi, A., Mousa, G., & Le, D. N. (2022). Adaptive neural backstepping control approach for tracker design of wheelchair upper-limb exoskeleton robot system. *Mathematics*, *10*(22), 4198.
25. Aly, A. A., Vu, M. T., El-Sousy, F. F., Hsia, K. H., Alotaibi, A., Mousa, G., ... & Mobayen, S. (2022). Adaptive neural network-based fixed-time tracking controller for disabilities exoskeleton wheelchair robotic system. *Mathematics*, *10*(20), 3853.
26. Aly, A. A., The Vu, M., El-Sousy, F. F., Alotaibi, A., Mousa, G., Le, D. N., & Mobayen, S. (2022). Fuzzy-based fixed-time nonsingular tracker of exoskeleton robots for disabilities using sliding mode state observer. *Mathematics*, *10*(17), 3147.

27. Lee, D., & Han, K. (2024). Vision-based construction robot for real-time automated welding with human-robot interaction. *Automation in Construction*, *168*, 105782.
28. Gao, Q., Liu, J., & Ju, Z. (2020). Robust real-time hand detection and localization for space human–robot interaction based on deep learning. *Neurocomputing*, *390*, 198-206.
29. Zhang, Y., Gao, Y., Yin, M., Jia, Y., Yang, Y., Zhang, C. H., ... & Hua, C. (2026). A novel human-robot interaction segmentation: Gesture-guided object segmentation based on deep learning. *IEEE Robotics and Automation Letters*.