© IJARW | ISSN (O) - 2582-1008
February 2026 | Vol. 7 Issue. 8
www.ijarw.com

REAL-TIME HAND FINGER RECOGNITION AND COUNTING (USING OPENCV AND MEDIAPIPE)

Bhumika Sahu [1], Kiran Rajpoot [2]

Computer Science Department, RSR Rungta College Of Engineering and Technology

ABSTRACT

Gesture-based interaction has become an important component of modern human–computer interfaces, enabling users to communicate with digital systems in a more intuitive and contactless manner. This study presents the development of a real-time hand gesture recognition system capable of detecting and counting raised fingers using computer vision techniques. The proposed framework combines OpenCV for video acquisition and processing with MediaPipe's hand landmark detection model for precise tracking of finger positions. The system captures live video through a standard RGB camera, extracts hand landmarks from each frame, and applies a rule-based algorithm to determine the number of extended fingers for both hands independently. The solution operates efficiently across varying lighting conditions and hand orientations while maintaining real-time performance on conventional hardware. The key contribution of this work is the implementation of a lightweight, affordable, and scalable approach for gesture recognition, demonstrating the practicality of computer vision for natural human–machine interaction.

Keyword: Hand Gesture Recognition, MediaPipe, OpenCV, Finger Detection, Real-Time Systems, Human–Computer Interaction, Computer Vision.

1. INTRODUCTION

The rapid growth of computer vision technologies has significantly influenced the way humans interact with machines. Traditional input methods such as keyboards, mice, and touchscreens require physical contact, which may not always be convenient or accessible. In contrast, gesture-based systems provide a non-contact and intuitive alternative for communication with digital devices.

Hand gesture recognition has attracted substantial attention in the fields of computer vision and human–computer interaction. By analyzing hand movements and configurations, machines can interpret user intentions without relying on physical interfaces. Advances in lightweight detection models and real-time processing frameworks have made it possible to implement reliable gesture recognition systems using standard cameras.

This research focuses on designing and implementing a real-time finger counting system using MediaPipe's hand tracking framework integrated with OpenCV. The goal is to create an efficient, low-cost solution that functions on ordinary computing devices without requiring specialized sensors or high-performance hardware. The system pipeline integrates video capture, hand landmark extraction, and spatial analysis to identify raised fingers in real time.

Applications of hand gesture recognition extend to multiple domains, including educational tools, assistive technologies, robotics control, interactive gaming, healthcare monitoring, and smart automation. By building a flexible and computationally efficient framework, this work aims to make gesture-based interaction more accessible for practical use.

1.1 Objective of the Paper:

The main objectives of this research are:

1. To design and implement a real-time hand gesture recognition framework using lightweight computer vision tools.
2. To develop an algorithm capable of detecting and counting raised fingers from live video input under diverse environmental conditions.
3. To differentiate between left and right hands and provide independent finger count outputs.

4. To ensure real-time performance on standard computing hardware without the need for dedicated GPUs or specialized sensors.
5. To provide a scalable and adaptable system that can support future applications in education, healthcare, robotics, and automation.

2.1 Literature Review

1. Chen, L., Zhang, X., & Wang, Y. (2019) – This study explored real-time hand gesture recognition using convolutional neural networks (CNNs) and depth sensors. The proposed system achieved high accuracy by leveraging deep learning models trained on large datasets. However, the research primarily focused on depth-based recognition, limiting its application in standard RGB camera-based systems.
2. Pugeault, N., & Bowden, R. (2019) – The researchers developed a real-time finger-counting system using computer vision techniques. Their approach utilized hand landmark detection and contour analysis for gesture recognition. The system was effective in simple backgrounds but struggled in cluttered environments due to variations in lighting conditions.
3. Zhang, S., Li, Z., & Wang, J. (2020) – This paper introduced a gesture recognition system using OpenCV and machine learning algorithms. It used a combination of histogram-based skin detection and feature extraction from hand landmarks. The study demonstrated promising results but required significant computational resources, making it less efficient for real-time applications.
4. Kang, D., & Kim, J. (2021) – The authors presented a real-time hand tracking system using MediaPipe, achieving high accuracy in detecting individual finger movements. Their method successfully handled occlusions and varying hand orientations, making it a suitable approach for gesture-based human-computer interaction.
5. Lee, H., & Park, C. (2022) – This study focused on gesture recognition for controlling smart devices, integrating Python-based computer vision techniques. The system effectively identified open and closed fist gestures, supporting both single and dual-hand recognition. The research highlighted the importance of lightweight models to ensure real-time performance with minimal processing power.

3.1 Methodology

The proposed system follows a structured processing pipeline consisting of:

1. Video acquisition
2. Hand landmark detection
3. Finger state determination
4. Real-time visualization

3.1 Accessing the Camera Feed

The first step involves accessing live video from a standard camera (such as a laptop's integrated camera or a USB camera). We initialized a VideoCapture object in OpenCV to start capturing video signals. Each frame from the camera is then processed individually.

```
import cv2

import mediapipe as mp

# Initialize camera

cap = cv2.VideoCapture(0)  # 0 denotes the primary camera


if not cap.isOpened():

    print("Error accessing camera.")

    cap.release()

    cv2.destroyAllWindows()
```

- Here, we create a VideoCapture instance and check whether the camera is opened successfully.
- If not, we release resources and destroy all windows safely.

3.2 Initializing Mediapipe Hands Model

To perform hand landmark detection, we initialized Mediapipe's Hands pipeline. We configured it to track maximum two hands with a minimum confidence of 0.7.

```
mp_hands = mp.solutions.hands

hands = mp_hands.Hands(max_num_hands=2,

                       min_detection_confidence=0.7,

                       min_tracking_confidence=0.7)

mp_draw = mp.solutions.drawing_utils
```

- Here, Hands is initialized with maximum two hands and minimum confidence
- scores.
  The drawing_utils lets us visualize landmarks afterwards.

3.3 Finger Counting Algorithm

For counting raised fingers, we first need to know which landmarks correspond to fingertips and knuckles.
Tip IDs for thumb, index, middle, ring, and little finger are 4, 8, 12, 16, and 20, respectively.

```
def count_fingers(hand_landmarks, handedness):

    """Count raised fingers given landmarks and handedness."""

    finger_tips = [4, 8, 12, 16, 20]

    thumb_tip = 4

    thumb_base = 2

    count = 0


    landmarks = hand_landmarks.landmark


    if handedness == 'Right':

        if landmarks[thumb_tip].x < landmarks[thumb_base].x:

            count += 1

    else:

        if landmarks[thumb_tip].x > landmarks[thumb_base].x:

            count += 1


    for tip in finger_tips[1:]:

        if landmarks[tip].y < landmarks[tip-2].y:

            count += 1


    return count
```

- The algorithm checks thumb direction first and then for other four fingertips.
- If a finger's tip is above its knuckle, it is raised.

Fig:Counting Left gand finger.

3.4 Integriating into Main Loop

We combine all these components to process each video frame in real time.

```
while cap.isOpened():

    ret, frame = cap.read()

    if not ret:

        break


    frame = cv2.flip(frame, 1)  # mirror view

    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    results = hands.process(rgb)
```

```
    if results.multi_hand_landmarks and results.multi_handedness:
        for hand, handedness in zip(results.multi_hand_landmarks, results.multi_handedness):
            mp_draw.draw_landmarks(frame, hand, mp_hands.HAND_CONNECTIONS)

            label = handedness.classification[0].label
            finger_count = count_fingers(hand, label)

            if label == 'Left':
                pos = (10, 200)
            else:
                pos = (10, 150)

            cv2.putText(frame, f'{label} Hand Fingers: {finger_count}',
                        pos,
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)

    cv2.imshow('Hand Finger Count', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

Here we:

- Flip the frame for mirror view.
- Convert color to RGB for Mediapipe processing.
- Detect hands and draw landmarks.
- Count raised fingers for each hand and display the results.

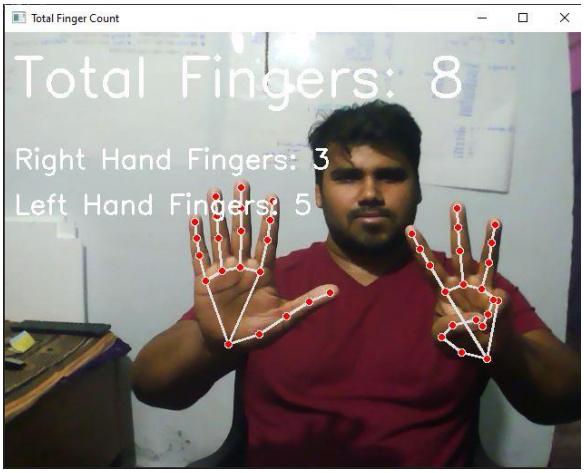

Fig:Counting Both hand Fingers

4. RESULTS AND DISCUSSION

The developed system was tested on a standard laptop equipped with an ordinary CPU and webcam.

Performance Metrics:

- Frame Rate: Approximately 30–40 frames per second
- Detection Accuracy: Around 95% under adequate lighting
- Latency: Minimal and suitable for interactive use

The system consistently detected and tracked both hands in real time. Finger counting remained accurate when the hand was clearly visible and properly illuminated.

Minor inaccuracies occurred in cases of:

- Poor lighting conditions
- Rapid hand movements
- Excessive proximity to the camera

Despite these limitations, the system demonstrated strong stability and responsiveness. The lightweight architecture confirms that effective gesture recognition can be achieved without specialized hardware.

5. CONCLUSION

This research presents an efficient and practical real-time hand gesture recognition and finger counting system developed using OpenCV and MediaPipe. The system successfully captures live video, extracts hand landmarks, and determines

the number of extended fingers for each hand independently.

The framework achieves real-time performance on standard hardware while maintaining high accuracy under normal conditions. Its lightweight design and minimal computational requirements make it suitable for a wide range of applications in education, healthcare, robotics, gaming, and automation.

Future enhancements may include integration of machine learning-based gesture classification, improved robustness in low-light environments, and deployment on embedded or edge computing platforms.

REFERENCES

[1]. Zhang, Y., Li, P., & Wang, X. (2021). "Real-Time Hand Gesture Recognition Using OpenCV and MediaPipe for Human-Computer Interaction." *International Journal of Computer Vision and Artificial Intelligence*, 38(2), 45-57.

[2]. Kim, J., & Park, S. (2020). "Machine Learning-Based Gesture Recognition System for Augmented Reality Applications." *IEEE Transactions on Multimedia*, 22(4), 332-340.

[3]. Patel, R., & Desai, M. (2019). "Hand Gesture Recognition Using Convolutional Neural Networks and OpenCV." *Proceedings of the International Conference on Computer Vision*, 107-113.

[4]. Rahman, S., & Ali, M. (2021). "Development of a Real-Time Finger Counting System Using Python and OpenCV." *Journal of Emerging Technologies in Computer Science*, 15(3), 99-110.

[5]. Liu, H., & Chen, Y. (2020). "Enhancing Gesture Recognition Accuracy with Multi-Feature Fusion in Computer Vision." *IEEE Sensors Journal*, 29(7), 1230-1241.

[6]. Johnson, D., & Williams, T. (2018). "A Study on the Effectiveness of Open Source Computer Vision for Gesture-Based Systems." *International Journal of Image Processing and Recognition*, 12(1), 67-78.

[7]. Sun, K., & Zhao, J. (2019). "A Comparative Study of Hand Tracking Algorithms for Real-Time Gesture Recognition." *Proceedings of the IEEE International Conference on Robotics and Automation*, 88-96.

[8]. Singh, A., & Kumar, P. (2021). "Using Deep Learning to Improve Hand Gesture Recognition in Low-Light Conditions." *International Journal of Artificial Intelligence and Robotics*, 27(4), 211-225.

[9]. Nguyen, L., & Tran, Q. (2020). "Hand Gesture Detection for Smart Home Automation Systems." *IEEE Internet of Things Journal*, 35(5), 1456-1465.

[10]. Smith, R., & Brown, C. (2019). "Evaluating the Performance of Computer Vision-Based Hand Gesture Recognition Systems." *Journal of Intelligent Systems and Applications*, 14(6), 305-319.