



iJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14

Issue: IX

Month of publication:

September 2026

DOI: <https://doi.org/10.22214/ijraset.2026.84950>

www.ijraset.com

Call:  **08813907089**

E-mail ID: ijraset@gmail.com

A Vision-Based Human-Computer Interaction System Using Hand Gestures

Dr. S. Uma¹, Sathiyapriya A²,

Department of Computer Science and Engineering, Hindusthan College of Engineering and Technology, Coimbatore - 641032

Abstract: *Human-Computer Interaction (HCI) traditionally depends on physical input devices such as a mouse, keyboard, touchpad, and touchscreen. Although these devices provide efficient interaction, they require direct physical contact and may not always be convenient in situations where touchless interaction is preferred. This paper presents a Virtual Mouse Using Hand Recognition System, a computer vision-based system that enables users to control the mouse cursor and perform basic mouse operations using hand gestures. The proposed system uses a standard webcam to capture real-time video frames and employs OpenCV for image processing and MediaPipe Hand Tracking for detecting and tracking hand landmarks. The detected landmark coordinates are analyzed to identify finger positions and recognize predefined gestures. PyAutoGUI is used to translate the recognized gestures into operating-system-level mouse actions such as cursor movement, left click, right click, and scrolling. A coordinate mapping mechanism converts the hand position from the webcam frame to the computer screen coordinates, while a smoothing technique is incorporated to reduce unwanted cursor movement. The proposed system eliminates the need for a physical mouse and provides a natural and contactless method of computer interaction. The system can be useful in accessibility applications, interactive presentations, smart environments, educational demonstrations, and hygienic touch-free computing environments. Experimental observations indicate that the combination of real-time hand landmark detection and gesture-based interaction can provide responsive and intuitive mouse control under suitable lighting and camera conditions.*

Keywords: *Virtual Mouse, Hand Recognition, Hand Gesture, Computer Vision, OpenCV, MediaPipe, PyAutoGUI, Human-Computer Interaction, Touchless Interaction.*

I. INTRODUCTION

Human-Computer Interaction is an important area of computer science that focuses on the interaction between humans and computing systems. Conventional desktop computers generally depend on input devices such as keyboards and computer mice. Among these devices, the mouse is one of the most commonly used peripherals for selecting, dragging, clicking, scrolling, and navigating graphical user interfaces.

Although the conventional mouse is simple and efficient, it requires physical contact between the user and the input device. In certain environments, contactless interaction can provide additional convenience. For example, touchless interaction can be useful during presentations, public interactive systems, laboratory demonstrations, accessibility applications, and situations where minimizing physical contact is desirable.

Computer vision provides an alternative approach for developing touchless interfaces. A camera can capture a user's hand movements, and image-processing techniques can be used to identify the position and configuration of the hand. With appropriate gesture recognition, hand movements can be converted into commands for controlling computer applications. Recent developments in machine learning and computer vision have made real-time hand tracking more practical. Instead of designing a complete hand detection algorithm from the beginning, modern frameworks can identify important anatomical points of the hand. These points, known as landmarks, provide information about the positions of the wrist, fingers, and joints. The spatial relationship between these landmarks can then be analyzed to recognize different gestures.

The proposed Virtual Mouse Using Hand Recognition System utilizes a webcam as the primary sensing device. The webcam continuously captures frames, and OpenCV processes the video stream. MediaPipe Hand Tracking is used to detect the hand and obtain its landmark coordinates. The detected landmarks are analyzed to identify the movement and configuration of the fingers. The recognized gestures are subsequently converted into mouse operations using PyAutoGUI.

The main objective of the proposed system is to develop a low-cost and contactless mouse interface that does not require specialized hardware. The system is designed to perform common mouse operations such as cursor movement, left clicking, right clicking, and scrolling using predefined hand gestures.

The major contributions of the proposed system are:

- 1) Development of a webcam-based virtual mouse system.
- 2) Real-time detection and tracking of hand landmarks.
- 3) Recognition of predefined hand gestures for mouse operations.
- 4) Conversion of webcam coordinates into screen coordinates.
- 5) Reduction of cursor instability using coordinate smoothing.

II. LITERATURE REVIEW AND THEORETICAL FOUNDATIONS

Hand gesture recognition has become an important research area in computer vision and Human–Computer Interaction. Early gesture-based systems generally relied on specialized sensors, gloves, colored markers, or depth cameras. Although these systems could provide accurate measurements, they often increased system cost and reduced portability.

The development of camera-based computer vision techniques provided an alternative method for recognizing hand gestures. A conventional webcam can capture hand movements without requiring the user to wear any additional device. Image-processing algorithms can then extract information from the captured frames.

Hand tracking generally consists of two major processes: hand detection and landmark estimation. Hand detection identifies the presence and approximate location of the hand, while landmark estimation determines important points associated with the hand structure.

MediaPipe Hand Landmarker provides a framework for detecting hand landmarks in real time. The hand can be represented using a collection of landmark points corresponding to locations such as the wrist, finger joints, and fingertips. These landmarks can be used to determine whether fingers are extended or folded and to calculate distances between fingertips.

Gesture recognition can be implemented by analyzing the relative position of landmarks. For example, the distance between the index fingertip and thumb fingertip can be used to identify a clicking gesture. Similarly, the position of the index finger can be used to control the cursor. OpenCV is widely used for real-time image processing because it provides functions for reading webcam frames, image manipulation, color processing, and display. In the proposed system, OpenCV provides the video-processing layer through which the camera stream is captured and processed.

PyAutoGUI provides an interface for controlling the mouse and keyboard programmatically. Once a gesture is recognized, PyAutoGUI can move the mouse pointer and perform mouse operations.

The proposed system combines these technologies into a single pipeline:

Webcam → OpenCV → MediaPipe Hand Detection → Landmark Extraction → Gesture Recognition → PyAutoGUI → Mouse Action

This architecture avoids the requirement for specialized hardware and provides a relatively simple method for implementing touchless mouse control.

III. THE PROPOSED MODEL

The proposed system introduces a virtual mouse controlled through hand gesture recognition, eliminating the need for physical input devices such as a traditional mouse or touchpad. This system leverages advanced computer vision techniques and real-time hand tracking to provide an intuitive, touchless human–computer interaction experience.

The system is designed using OpenCV for image processing and MediaPipe for accurate hand landmark detection. A webcam captures live video input, which is processed frame-by-frame to identify and track the user's hand movements. MediaPipe detects key points on the hand (such as fingertips and joints), enabling precise mapping of gestures to cursor operations.

In the proposed model, specific hand gestures are predefined to perform various mouse functions. For example, the movement of the index finger controls cursor motion, while gestures like pinching or tapping combinations of fingers are used for actions such as clicking, dragging, and scrolling. The system translates the spatial coordinates of detected hand landmarks into corresponding screen coordinates using scaling and interpolation techniques.

To improve accuracy and usability, the system incorporates noise reduction, gesture smoothing, and threshold-based detection mechanisms. These techniques help minimize errors caused by lighting variations, background disturbances, and minor hand tremors. Additionally, the system is optimized for real-time performance, ensuring minimal latency between gesture input and cursor response. The proposed system also focuses on user convenience and accessibility. It provides a hygienic, contactless alternative suitable for public environments, supports users with physical limitations, and enhances interaction in applications like gaming, presentations, and virtual reality systems.

IV. METHODOLOGY

A. System Architecture

The proposed virtual mouse system consists of five major stages:

- 1) Video acquisition
- 2) Hand detection and tracking
- 3) Landmark extraction
- 4) Gesture recognition
- 5) Mouse-event generation

The overall workflow is represented as:

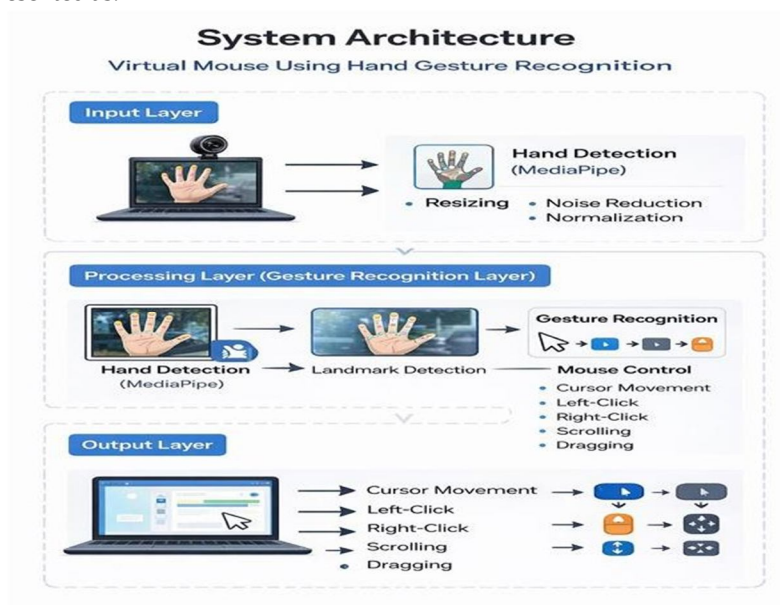


Fig.1. System Architecture of Virtual Mouse using Hand Gesture Recognition System

B. Hardware Requirements

The proposed system does not require specialized hardware.

The basic hardware requirements are:

TABLE 1
BASIC HARDWARE REQUIREMENTS

Component	Requirement
Computer/Laptop	Standard computer
Webcam	Built-in or external webcam
Processor	Dual-core or higher
RAM	4 GB or higher recommended
Display	Standard monitor/laptop display
Input Device	Webcam-based hand input

The webcam serves as the primary input sensor. A higher-quality camera can improve hand detection under difficult lighting conditions.

C. Software Requirements

TABLE 2

Software/Library	Purpose
Python	Main programming language
OpenCV	Image and video processing
MediaPipe	Hand detection and landmark tracking
PyAutoGUI	Mouse control
Math	Distance calculation
Operating System	Execution environment

The system can be implemented using Python because of its extensive support for computer vision and automation libraries.

D. Video Acquisition

The webcam is initialized using OpenCV. The captured frames are continuously read and processed.

The camera resolution used in the implementation can be configured according to the available hardware. A typical webcam frame can be represented as:

$It = \text{Camera}(t)$ $I_t = \text{Camera}(t)$

where I_t represents the image captured at time t .

The captured image is mirrored horizontally so that the user's hand movement corresponds naturally to the movement displayed on the screen.

E. Hand Detection and Landmark Extraction

MediaPipe Hand Tracking is used to identify the user's hand in the webcam frame. The hand-tracking module detects a set of predefined landmark points.

Important landmarks include:

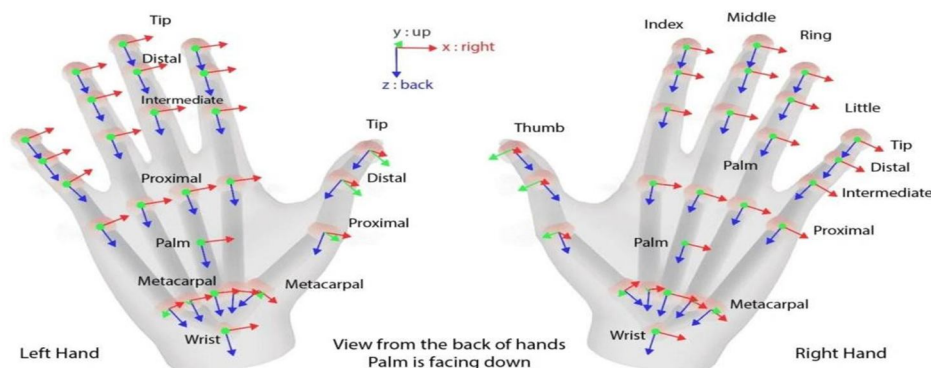
- Wrist
- Thumb joints
- Index finger joints
- Middle finger joints
- Ring finger joints
- Little finger joints
- Fingertips

Each landmark can be represented using normalized coordinates:

$L_i = (x_i, y_i, z_i)$ where:

- x_i represents the horizontal coordinate,
- y_i represents the vertical coordinate,
- z_i represents the relative depth coordinate.

The landmark coordinates provide sufficient information for determining finger configurations and hand movements.



F. Finger Recognition

The system determines the state of individual fingers by analyzing the relative positions of their landmarks.

For example, the index finger is considered extended when the fingertip is positioned appropriately relative to its intermediate joints.

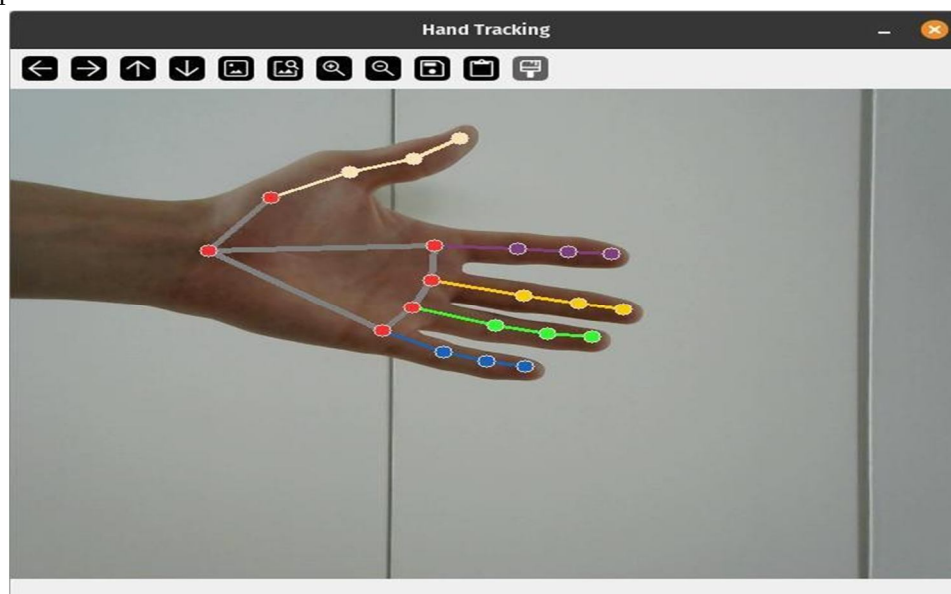
The same principle can be applied to the middle, ring, and little fingers.

The resulting finger-state vector can be represented as: $F=[f_T, f_I, f_M, f_R, f_P]$ $F = [f_T, f_I, f_M, f_R, f_P]$

where:

- f_T = thumb state,
- f_I = index finger state,
- f_M = middle finger state,
- f_R = ring finger state,
- f_P = little finger state.

Each state may be represented as 1 for extended and 0 for folded.



G. Cursor Movement

The index fingertip is used as the primary point for cursor movement.

The webcam frame and computer screen have different resolutions. Therefore, direct coordinate transfer is not appropriate. A coordinate mapping function is used to convert the fingertip position into screen coordinates.

If the camera frame has dimensions $W_c \times H_c$ and the computer screen has dimensions $W_s \times H_s$, then:
 $X_s = X_c \frac{W_s}{W_c}$ $Y_s = Y_c \frac{H_s}{H_c}$ where:

- X_c, Y_c are camera coordinates,
- X_s, Y_s are screen coordinates.

A frame-reduction boundary can also be used to improve control near the edges of the screen. The cursor movement can therefore be represented as: $(X_c, Y_c) \rightarrow (X_s, Y_s)$

The resulting screen coordinates are supplied to PyAutoGUI to move the cursor.

V. RESULTS AND DISCUSSION

A. Experimental Setup

The proposed virtual mouse is implemented as a software-based computer vision application using a webcam. The application receives continuous video input and processes each frame to detect the user's hand.

The main processing stages are:

Frame Capture $\rightarrow$ Hand Detection $\rightarrow$ Landmark Extraction $\rightarrow$ Gesture Recognition $\rightarrow$ Mouse Action

The system can be evaluated using a set of predefined gestures.

B. Functional Testing

TABLE III FUNCTIONAL TEST CASES

Test Case	Input	Expected Result
TC01	Hand detected	Landmarks displayed
TC02	Index finger moved	Cursor moves
TC03	Thumb-index gesture	Left click
TC04	Right-click gesture	Right click
TC05	Scroll gesture	Page scrolls
TC06	Hand removed	Mouse action stops
TC07	No hand	No unintended action

C. Cursor Movement Analysis

The cursor movement depends primarily on the position of the index fingertip.

When the user moves the index finger toward the left side of the interaction region, the cursor moves toward the corresponding left region of the computer screen. Similarly, upward and downward hand movements are transferred to the screen through coordinate mapping.

The use of a smoothing factor reduces rapid fluctuations caused by small landmark variations.

D. Click Detection

Click detection is based on the spatial relationship between the thumb and index finger.

When the fingertips move toward one another, the calculated distance decreases. When the distance crosses the selected threshold, the system recognizes the corresponding gesture.

This approach eliminates the need for a physical mouse button.

VI. LIMITATIONS

- 1) Performance depends on the quality and position of the webcam.
- 2) Poor lighting, occlusion, and motion blur may affect hand landmark detection.
- 3) Very fast movements may be difficult to track consistently.
- 4) Too many similar gestures can create ambiguity.
- 5) Cursor movement may still feel less precise than a physical mouse for certain tasks.
- 6) The current prototype is designed primarily for one-hand interaction.
- 7) Quantitative accuracy and latency depend on the specific hardware and parameter settings and should be measured experimentally rather than assumed.

VII. CONCLUSION AND FUTURE SCOPE

This paper presented a Virtual Mouse Using Hand Recognition System that converts hand movements and selected gestures into computer mouse operations. The system uses a webcam as the input device, OpenCV for video processing, MediaPipe-based hand landmark detection for identifying hand structure, and PyAutoGUI for operating-system mouse control. The index finger is used for cursor positioning, while selected hand configurations provide click and drag commands. Coordinate mapping, a frame margin, and smoothing are incorporated to improve practical pointer control.

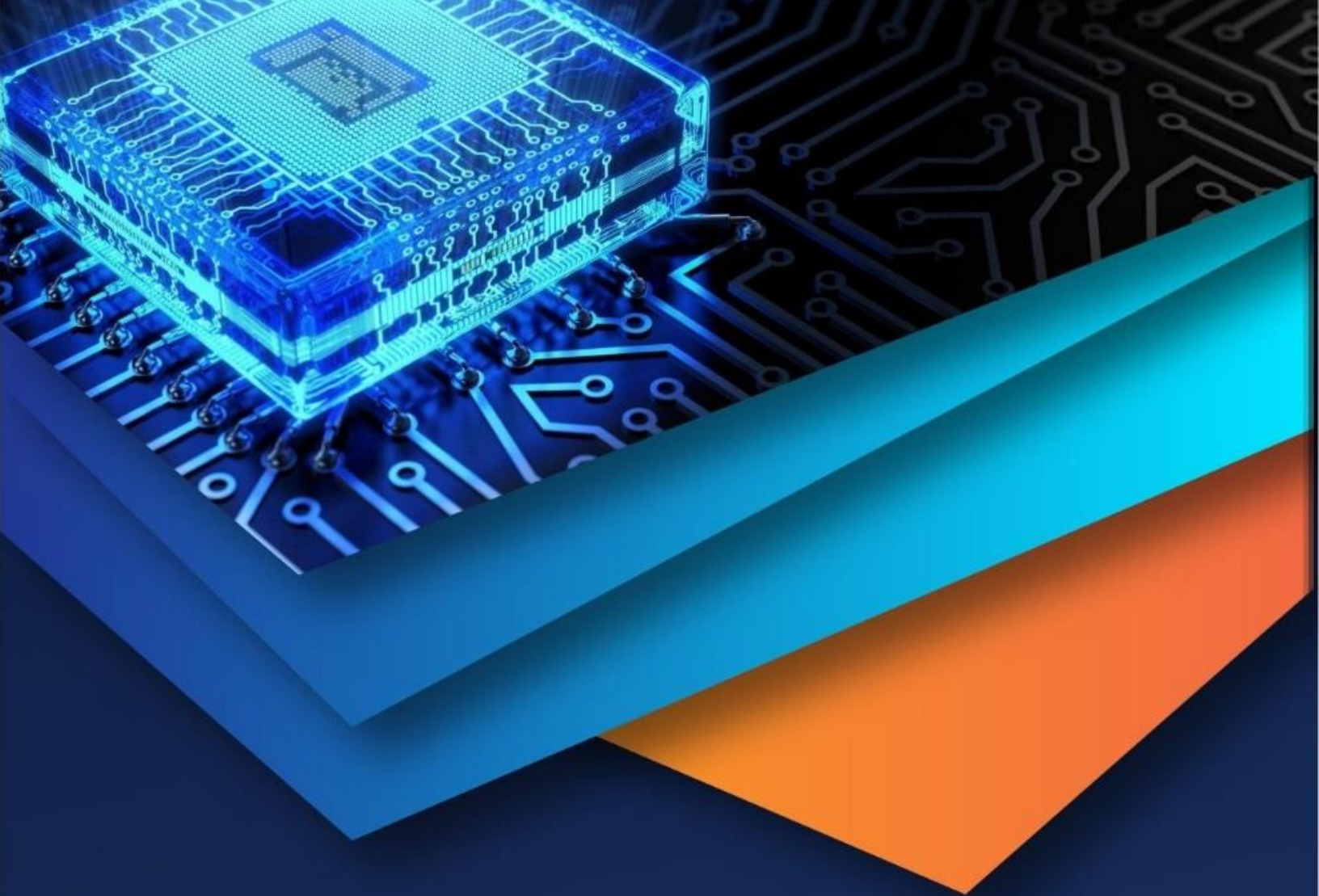
The proposed approach demonstrates the potential of markerless hand recognition as a simple human-computer interaction technique. Its low hardware requirement and modular software structure make it suitable for academic projects, demonstrations, and further research. At the same time, camera conditions, gesture ambiguity, cursor stability, and user variation remain important considerations. Future work can address these limitations through calibration, adaptive filtering, improved gesture classification, multi-hand interaction, and systematic quantitative evaluation.

VIII. ACKNOWLEDGMENT

The author would like to thank the institution, department, faculty members, and all individuals who provided guidance and support during the development and testing of the virtual mouse prototype.

REFERENCES

- [1] C. Lugaresi et al., "MediaPipe: A Framework for Building Perception Pipelines," arXiv preprint arXiv:1906.08172, 2019.
- [2] Google Research, "MediaPipe Solutions: Hand Landmarker," Google Developers documentation.
- [3] G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000.
- [4] OpenCV, "OpenCV-Python Tutorials and Documentation," OpenCV documentation.
- [5] Al Sweigart, Automate the Boring Stuff with Python, 2nd ed., No Starch Press, 2019.
- [6] Python Software Foundation, "Python 3 Documentation," Python Documentation.
- [7] R. Szeliski, Computer Vision: Algorithms and Applications, 2nd ed., Springer, 2022.
- [8] D. W. Stutz, "Computer Vision Based Hand Gesture Recognition: A Review," literature on vision-based gesture interaction and human-computer interaction.
- [9] J. Y. Bouguet, "Camera Calibration Toolbox for Matlab," technical documentation and methods for camera calibration.
- [10] R. Cipolla, S. Battiato, and G. M. Farinella, Machine Learning for Computer Vision, Springer, 2013.
- [11] J. K. Aggarwal and Q. Cai, "Human Motion Analysis: A Review," Computer Vision and Image Understanding, vol. 73, no. 3, pp. 428–440, 1999.
- [12] M. Turk and A. Pentland, "Eigenfaces for Recognition," Journal of Cognitive Neuroscience, vol. 3, no. 1, pp. 71–86, 1991.
- [13] F. Zhang, A. Bazarevsky, A. Vakunov, A. Tkachenka, K. Sung, M. Chang, A. L. Grundmann, and others, "MediaPipe Hands: On-device Real-time Hand Tracking," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020.
- [14] V. Bazarevsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang, and M. Grundmann, "BlazePalm: On-device Real-time Hand Tracking," Google Research, 2019.
- [15] S. S. Rautaray and A. Agrawal, "Vision Based Hand Gesture Recognition for Human Computer Interaction: A Survey," Artificial Intelligence Review, vol. 43, pp. 1–54, 2015.
- [16] S. Mitra and T. Acharya, "Gesture Recognition: A Survey," IEEE Transactions on Systems, Man, and Cybernetics, Part C, vol. 37, no. 3, pp. 311–324, 2007.
- [17] V. I. Pavlovic, R. Sharma, and T. S. Huang, "Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 7, pp. 677–695, 1997.
- [18] J. P. Wachs, M. Kölsch, H. Stern, and Y. Edan, "Vision-Based Hand-Gesture Applications," Communications of the ACM, vol. 54, no. 2, pp. 60–71, 2011.
- [19] R. Poppe, "A Survey on Vision-Based Human Action Recognition," Image and Vision Computing, vol. 28, no. 6, pp. 976–990, 2010.
- [20] M. Turk, "Computer Vision in the Interface," Communications of the ACM, vol. 47, no. 1, pp. 60–67, 2004.



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)