

Air Keys: Virtual Keyboard and Mouse Using Computer Vision

Vinaya Kulkarni¹, Nilesh Uke², Shital Karande¹,
Shreyasee Patil¹, Samruddhi Patil¹, Krutika Shende¹, Sakshi Zagade¹

¹ Department of Computer Engineering, Bharati Vidyapeeth's College of Engineering for
Women, Pune, Maharashtra, India

² Indira College of Engineering and Management, Pune, Maharashtra, India

Abstract

Traditional input devices such as the mouse and keyboard require human contact. As such, these input devices have limitations when it comes to malleability. The notion of the Virtual Mouse & Keyboard attempts to bypass the limitations inherent in traditional input devices by applying the use of OpenCV & MediaPipe. These technologies will enable a contactless method for human computer interaction through the use of hand gesture recognition. A webcam will be responsible for continuous video capture. In turn, the Hand Tracking module from the MediaPipe framework will generate 21 landmarks found on the hand. These landmarks will enable the detection of hand gesture interaction that may translate to cursor movement or clicking. These events will be processed using the creation of events by the Pynput & Autopy libraries. These events will be at the operating system level. The proposed model will have two operation modes. These will be the Virtual Mouse Mode & the Virtual Keyboard Mode. These will enable the operation of hand gesture interaction for the control of the computer mouse & the input of text.

Keywords: Gesture Recognition, Media Pipe, OpenCV, Contactless Interface, Hand Tracking.

I. INTRODUCTION

Human-Computer Interaction (HCI) has moved from the traditional mechanical way to a natural, vision-based approach that interprets gestures. The traditional mouse and keyboard method of interaction is a contact-based approach requiring a static working position. These limitations pose a problem to the user in different settings. The Virtual Mouse and Keyboard using OpenCV and Media Pipe approach seeks to offer a solution to the traditional mouse and keyboard methods by allowing the user to perform these tasks using hand gestures

from a webcam.

The system uses the integration of OpenCV for video processing and Media Pipe for hand landmark detection to determine the position and orientation of the fingers. In this setup, the index finger acts as a pointer to control the cursor on the computer screen, while a pinching motion performed using the index finger and middle finger simulates a mouse click. For keyboard actions, a keyboard layout needs to be projected on the screen using OpenCV, and a pointer motion accompanied by a click action will enable a user to key in characters. Libraries such as Pynput or Autopy can convert these actions to events recognized by the operating system. A unique gesture such as holding up three fingers can be employed for mode switching or quitting the application.

It does not need additional hardware, making it inexpensive and easy to apply to any computer that has a web camera attached. It is particularly beneficial in cases where touchless interaction is required, such as public terminals, healthcare facilities, and physically challenged individuals. The proposed approach highlights the way computer vision and gesture recognition technologies can be applied in order to facilitate accessibility, cleanliness, and convenience in using computing systems. This project applies open source solutions in order to advance the realm of intelligent human computer interaction systems with touchless features.

II. RELATED WORK

Recent A real-time virtual keyboard using CNN-LSTM is proposed in [1]. Hand key points detection is performed using MediaPipe. Temporal processing is addressed using sequence learning. The effect of spatial feature extraction, as well as temporal consistency, is demonstrated to enhance keyboard typing accuracy and reduce incidents of incorrect classifications

during continuous keyboard input. Yet, its performance is reduced for CPU-only computers, implying the requirement for optimizations and more efficient paths. This study validates the capability of deep sequence learning models as a solution to mid-air keyboard typing problems

A study on hand gesture classification for virtual reality using MediaPipe is presented in [2]. The system extracts 21 hand landmarks and evaluates different implementation methods such as Barracuda and TensorFlow Lite for performance comparison. The study demonstrates that gesture-based interaction provides an intuitive and controller-free experience in virtual environments. However, performance varies across platforms, where higher frame rates are achieved on PCs while mobile devices experience reduced efficiency, indicating limitations in real-time deployment on low-power systems.

A research on multimodal interaction is presented in [3]. In this research, hand gestures and voice commands are integrated to increase the reliability and vocabulary of the interaction process. The integration of the modes increases the system reliability by decreasing the rate of false activation, which occurs when the modes are used alone. Voice commands also have other drawbacks such as noise, privacy issues, and synchronization.

A lightweight and efficient virtual mouse solution using MediaPipe is proposed in [4]. The solution uses logic related to gestures and fingertip coordinates to simulate mouse movements as well as clicks. The solution is fully CPU based, and smooth cursor movement is accomplished using coordinate smoothing and threshold settings. However, it is still affected by changes in the intensity levels and needs user calibration.

A gesture-controlled virtual mouse system using artificial intelligence is proposed in [5]. In this system, OpenCV is used for image processing, MediaPipe for hand detection, and PyAutoGUI for controlling mouse movements. With such technology, one can move the cursor, make clicks, and scroll up and down using hand motions. Therefore, in order to control the computer, one will require no additional hardware. However, the downside is that this system requires an environmental setup.

A hands-free computing technique using a gesture-controlled virtual mouse implemented by an AI mouse is described in [6]. The hands-free computing method consists of computer vision technologies and real-time gesture recognition to simulate mouse functionality without the use of actual hardware. It enhances usability and accessibility, especially for disabled users but suffers from several limitations such as gesture recognition precision and delay, and dependency on stable environment conditions.

RingGesture [7], which focuses on solving the problem of mid-air typing, is a hybrid system of gesture and vision-based typing with an IMU ring attached to the user's finger. Along with the hybrid typing technique, it uses deep learning models that utilize word correction techniques for efficient text entry tasks. However, unlike other vision-only approaches, it requires additional hardware to be implemented.

A literature review paper focusing on the problems of visual hand gesture recognition is provided in [8]. The paper surveys existing solutions that utilize both traditional machine learning algorithms and more sophisticated approaches such as convolutional neural networks. The survey mentions some crucial issues in this area such as the necessity of real-time operation, occlusions processing, and robustness with respect to user's differences.

An easy-to-use VKM system with real-time computer vision is proposed in [9]. The VKM system makes use of MediaPipe and analyzes finger joints to determine keys pressing and mouse clicks. The proposed VKM system uses a projection-based method where a mini-projector creates a virtual keyboard on a surface and hands are captured by webcam in the video stream. The key idea behind this project is that this system uses geometric analysis of finger joints and Dynamic Time Warping (DTW) for detecting touch events efficiently even at various typing speeds. Experiments show a very good performance of the system in terms of keystrokes accuracy and typing speed, which reaches around 90% and 55 letters per minute respectively. Nevertheless, the limitation for practical usage of this system is the necessity of a projector and various parameters like the degree of confidence in the hand position detection etc. AR/VR-related hand tracking systems research is discussed in [10]. The paper considers effective and fast deep neural network approaches to hand tracking with the usage of techniques like pruning, quantization, and prediction. These approaches provide not only simplicity but also a good steady frame rate during hand tracking.

Paper [11] describes a vision-based virtual mouse that allows controlling computers with hand gestures detected via webcam. This approach utilizes MediaPipe, OpenCV, and PyAutoGUI to detect hand gestures in real time. Besides the usual cursor movements and clicks, this technique also allows controlling brightness and volume by means of finger distance detection. Such an approach provides a cost-efficient method that needs only one device, a camera, to facilitate user interaction with computer applications including VR and gaming applications and medical applications.

Research in paper [12] suggests a contactless virtual mouse that eliminates any need for mouse or touchpad operations and uses hand gestures instead. Hand detection techniques include

MediaPipe and OpenCV for hand tracking and image processing, respectively, whereas PyAutoGUI is used to perform mouse operations. In contrast to previous studies which were focused on static hand gestures, this research proposes the usage of static as well as dynamic hand gestures recognition techniques. The approach could be beneficial in medicine, public PCs, and gaming.

The paper [13] proposes an autonomous intelligent robot that responds to simple hand wave signals. The algorithm uses MediaPipe and OpenCV to detect and recognize hand movements from the camera in real time. While the Raspberry Pi processes the images, the Arduino is responsible for controlling the motor motion. When the system recognizes any waving hand gesture, the robot automatically approaches the user for further instructions. Experimental evaluations prove effective detection of the hand movements regardless of the light conditions and emphasize the usefulness of the solution for developing contactless services.

Proposed by [14], the virtual mouse application utilizes hand gestures along with Convolutional Neural Networks (CNNs) for hand gesture recognition. The algorithm analyzes live videos, recognizes hand movements, classifies them through CNN, and turns them into mouse actions. Emphasizing the benefits of using deep learning methods rather than image processing techniques, the researchers explain the necessity of implementing such applications to improve human-computer interaction and support virtual reality technologies, smart computing systems, and healthcare applications. Thus, substituting hardware components of a computer input device with software, the algorithm provides a more comfortable solution.

The research paper [15] analyses the ability of Google MediaPipe Hand (GMH) and the improved depth-based method GMH-D in the clinical analysis of hand movements. The performance of both approaches is tested with respect to gold standard motion capturing systems during hand opening, finger tapping, and dexterity tests. GMH exhibits remarkable performance in terms of high consistency and accuracy in tracking. Moreover, GMH-D ensures better spatial accuracy owing to the use of depth measurements. This research proves that MediaPipe based tracking technology can be used effectively as a non-invasive solution for monitoring motor disorder therapies.

Paper [16] discusses a real-time gesture recognition method that can recognize American Sign Language gestures using the features obtained from MediaPipe and a CNN model. It uses MediaPipe for extracting features from hands and classifies them as different ASL alphabets with an amazing accuracy rate

of 99.95%. The proposed method recognizes ASL alphabets in real time without any delays, which makes it appropriate for developing an assistive communication device.

As stated in this research paper by Google [17], MediaPipe Hands is an efficient tool developed for hand tracking in real-time using only RGB input from a camera. It utilizes a two-step pipeline that involves a palm detector and a hand-landmark model able to detect 21 hand landmarks in 2.5D space. MediaPipe Hands can work on edge devices and mobile phones without requiring any special equipment; its accuracy rate and real-time capabilities have contributed greatly to its popularity in applications involving gesture recognition, augmented and virtual reality programs, sign languages, and virtual mouse technology.

This research paper [18] describes a gesture-based system that acts as an alternative to a conventional mouse and keyboard. Instead, it uses a computer camera to replace a mouse with cursor control, clicking, scrolling, and keyboard commands. This tool is based on MediaPipe, OpenCV, and several other computer vision and machine learning software products. One of the main benefits of this technology lies in the possibility of touchless interaction which makes it valuable for physically disabled users and sanitary conditions.

This paper [19] describes a virtual mouse system that controls the position of a mouse using hand gestures captured by the camera. Hand detection, gesture recognition, and controlling the mouse position are accomplished using computer vision and machine learning technologies. The approach is suitable for AR, VR, and smart surroundings applications since it does not depend on any physical devices. Accessibility for people with limited motor skills can also be improved with this approach.

In this work [20], a hands-free computer control method using gesture control, facial expression recognition, and object detection is developed. The system is implemented with MediaPipe, OpenCV, and other Python libraries and allows moving the mouse cursor, scrolling, and controlling volume, and even blinking and opening one's mouth. The goal of the system described is the reduction of physical contact between humans and computers and accessibility improvement for disabled people. This paper focuses on the significance of such solutions for the period before and during the COVID-19 outbreak.

Table 1.1: Literature Survey

Author(s) & Year	Paper Name	Methodology Used	Key Findings	Future Scope
Mallik et al. (2024)	Virtual Keyboard: A Real-Time Hand Gesture Recognition-Based Character Input System Using LSTM and MediaPipe	MediaPipe Holistic, CNN-LSTM, sequence learning	Improved typing accuracy and reduced misclassification in mid-air typing.	Optimize for CPU-only systems and lightweight deployment.
Jo & Kim (2024)	A Study on the Classification of Hand Gesture for Mobile Virtual Reality with MediaPipe	MediaPipe (21 landmarks), TensorFlow Lite, Barracuda	Enabled intuitive controller-free interaction in VR environments.	Improve performance on low-power mobile devices.
Rahim et al. (2024)	Advanced Touchless HCI: A Gesture Recognition-Based Multilingual Virtual Keyboard	Gesture recognition + voice command integration	Reduced false activations and improved interaction reliability.	Address synchronization issues and noisy environments.
Wang & Xie (2024)	HandMouse: A Gesture-Based Virtual Mouse System	MediaPipe, fingertip tracking, coordinate smoothing	Achieved smooth cursor movement using a CPU-based solution.	Improve robustness against lighting changes and reduce calibration needs.
Yadav et al. (2024)	Gesture Controlled Virtual Mouse Using AI	OpenCV, MediaPipe, PyAutoGUI	Enabled cursor movement, clicking, and scrolling through gestures.	Increase robustness under varying lighting and background conditions.
Kavuri et al. (2025)	Hands-Free Computing: A Gesture-Controlled Virtual AI Mouse	Computer vision, real-time gesture recognition	Improved accessibility and hands-free computer interaction.	Reduce latency and enhance gesture precision.
Bisht et al. (2024)	VKM: A Virtual Keyboard and Mouse Solution Towards a Lightweight Computing System	MediaPipe, DTW, fingertip tracking, projection-based keyboard	Achieved >90% keystroke accuracy and 55 letters/min typing speed.	Eliminate dependence on projectors and simplify calibration.
Meng et al. (2024)	Real-Time Hand Gesture Monitoring Model Based on MediaPipe	Deep neural networks, MediaPipe optimization	Achieved efficient real-time hand tracking for AR/VR systems.	Further optimize lightweight models for edge devices.
Goel (2025)	A Vision-Based Virtual Mouse Using Hand Gesture Recognition	MediaPipe, OpenCV, PyAutoGUI	Added volume and brightness control along with mouse functions.	Improve complex gesture support and VR integration.

Kotari et al. (2024)	Augmented Virtual Mouse System with Enhanced Gesture Recognition	MediaPipe, OpenCV, dynamic gesture recognition	Improved precision through static and dynamic gesture support.	Expand gesture vocabulary and improve real-time accuracy.
Muslimin et al. (2025)	Development of a Hand Gesture Detection-Based Robot System with MediaPipe	MediaPipe, OpenCV, Raspberry Pi, Arduino	Enabled robot navigation through hand-wave gestures.	Extend to more complex human-robot interaction tasks.
Sampreeth S. (2025)	Hand Gesture Recognition-Based Virtual Mouse Using CNN	CNN-based gesture classification, computer vision	Improved gesture recognition robustness in complex environments.	Develop adaptive deep-learning models for diverse users.
Amprimo et al. (2024)	Hand Tracking for Clinical Applications: Validation of GMH and GMH-D	MediaPipe Hand (GMH), RGB-D sensing, motion analysis	GMH-D achieved higher spatial accuracy for clinical hand tracking.	Apply framework to rehabilitation and healthcare systems.
Kumar et al. (2023)	Mediapipe and CNNs for Real-Time ASL Gesture Recognition	MediaPipe, CNN	Achieved 99.95% ASL recognition accuracy.	Extend to dynamic sign language recognition and multilingual datasets.
Zhang et al. (2020)	MediaPipe Hands: On-Device Real-Time Hand Tracking	Palm detection, hand landmark estimation, deep learning	Established a lightweight 21-landmark real-time hand tracking framework.	Improve 3D tracking accuracy and occlusion handling.
Jagnade et al. (2023)	Hand Gesture-Based Virtual Mouse Using OpenCV	MediaPipe, OpenCV, facial actions, object detection	Enabled cursor control, scrolling, volume adjustment, and eye-blink interactions.	Enhance accessibility and reduce environmental dependency.
Mallik et al. (2024)	Virtual Keyboard: A Real-Time Hand Gesture Recognition-Based Character Input System Using LSTM and MediaPipe	MediaPipe Holistic, CNN-LSTM, sequence learning	Improved typing accuracy and reduced misclassification in mid-air typing.	Optimize for CPU-only systems and lightweight deployment.
Jo & Kim (2024)	A Study on the Classification of Hand Gesture for Mobile Virtual Reality with MediaPipe	MediaPipe (21 landmarks), TensorFlow Lite, Barracuda	Enabled intuitive controller-free interaction in VR environments.	Improve performance on low-power mobile devices.
Rahim et al. (2024)	Advanced Touchless HCI: A Gesture Recognition-Based Multilingual Virtual Keyboard	Gesture recognition + voice command integration	Reduced false activations and improved interaction reliability.	Address synchronization issues and noisy environments.

Wang & Xie (2024)	HandMouse: A Gesture-Based Virtual Mouse System	MediaPipe, fingertip tracking, coordinate smoothing	Achieved smooth cursor movement using a CPU-based solution.	Improve robustness against lighting changes and reduce calibration needs.
-------------------	---	---	---	---

III. PROPOSED SYSTEM

The Air Keys system offers a contactless virtual mouse and keyboard solution that supports human-computer interface technology using hand gestures via a conventional webcam. The solution is implemented using a Python programming language and computer vision algorithms without relying on hardware components. The solution utilizes real-time gathering of hand keypoint locations supplied by MediaPipe and image processing functions offered by OpenCV to recognize natural human hand signals for mouse or keyboard operations.

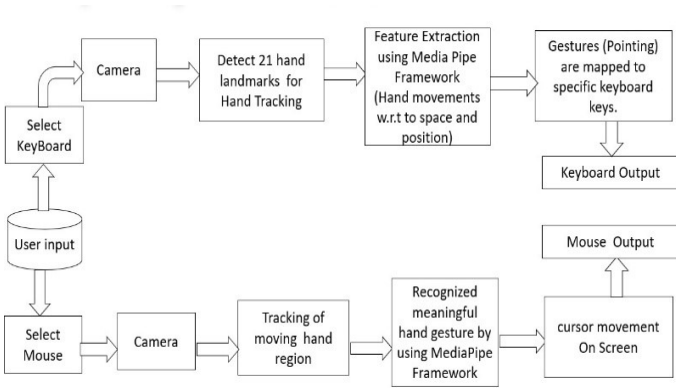


Fig. 1: System Architecture

The system architecture of this design, as shown in Fig. 1, has video capturing and processing, hand detection, recognition of gestures, function selecting, and execution of system-level input. This webcam keeps on processing live images frame by frame so that there is no hindrance to real-time processing. This design works on two different user functions, that is, Mouse Function and Keyboard Function. Users can interchange these operations through predefined gestures without halting their processing.

A. Image Acquisition and Preprocessing

The live video frames are captured using a conventional webcam with a preset resolution. Every video frame is then flipped horizontally for a mirror-like effect in the interaction process. The video frames are then converted from the BGR format to the RGB format for MediaPipe requirements. Simple image pre-processing is done to improve the stability of the landmarks while still allowing for interactive frame rates.

B. Hand Detection and Extraction of Landmarks

Hand detection is carried out through MediaPipe Hands, which uses a two-stage pipeline of palm detection followed by hand landmark regression. After a hand is detected, it locates 21 normalized points on the hand, which are finger joints and fingertips. These points are used to determine the precise location needed to detect gestures, as shown in Fig. 2.

The calculated coordinates are then normalized to be processed in terms of pixel values within a frame resolution, allowing precise mapping from hand to screen actions. This lightweight solution is able to operate on a system running only on a CPU.

C. Gesture Recognition

Gesture recognition is performed, as shown Fig 3, in a rule-based geometric approach rather than through data-driven classification. Finger states are recognized by comparing the coordinates of fingertips and their respective lower joints. The motion of the cursor is controlled by use of the index fingertip; while clicking and selection gestures are recognized from relative distance measurements among fingertips of selections.

The real-time virtual keyboard interface controlled through hand gestures. Hand landmarks are detected to enable precise key selection and highlighting. The system effectively translates gestures into text input, showcasing accurate gesture recognition, seamless touchless typing, and efficient human-computer interaction in a real-time operating environment.

Euclidean distance between the fingertip landmarks calculates authentication of pinch and click movements. Any gesture can be termed real only when certain threshold values concerning the distance are fulfilled. It really helps eradicate any spurious detection owing to sudden movement of fingers.

D. Mathematical Formulation and Gesture Computation

1. Hand Landmark Representation

The hand tracking module detects 21 landmarks on the human hand using a trained model. Each landmark is represented as a coordinate point in normalized space:

$$L_i = (x_i, y_i), i = 1, 2, \dots, 21$$

These normalized coordinates are mapped to pixel coordinates based on the frame resolution:

$$X_i = x_i \times W, Y_i = y_i \times H$$

where:

W = frame width
 H = frame height

2. Cursor Position Mapping

To control the cursor, the fingertip position (index finger) is mapped from camera space to screen space using linear interpolation:

$$X_{screen} = \frac{(X - X_{min})}{(X_{max} - X_{min})} \times ScreenWidth$$

$$Y_{screen} = \frac{(Y - Y_{min})}{(Y_{max} - Y_{min})} \times ScreenHeight$$

This ensures proportional mapping between hand movement and cursor movement.

3. Euclidean Distance for Gesture Detection

Gesture recognition is based on the distance between specific fingertip landmarks. The Euclidean distance between two points is calculated as:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

This distance is primarily used to detect pinch gestures (e.g., thumb and index finger).

4. Gesture Classification Rule

A threshold-based decision mechanism is used to classify gestures:

If $d < T \Rightarrow \text{Gesture} = \text{Click}$

If $d \geq T \Rightarrow \text{Gesture} = \text{No Click}$

where:

T is a predefined threshold value determined experimentally.

5. Finger State Detection

Finger states (open/closed) are determined by comparing the vertical positions of fingertip and corresponding joint:

$$\text{Finger State} = \begin{cases} 1, & \text{if } y_{tip} < y_{joint} \\ 0, & \text{otherwise} \end{cases}$$

This helps in identifying gestures such as:

- Single finger $\rightarrow$ cursor movement
- Two fingers $\rightarrow$ click
- Three fingers $\rightarrow$ mode switch

6. Motion Smoothing (Exponential Filtering)

To reduce jitter in cursor movement, exponential smoothing is applied:

$$S_t = \alpha X_t + (1 - \alpha)S_{t-1}$$

where:

- S_t = smoothed position
- X_t = current position
- $\alpha \in (0,1)$ = smoothing factor

This improves stability and user experience.

7. Region-Based Mode Selection

The interaction space is divided into predefined regions:

$$\text{Mode} = \begin{cases} \text{Mouse}, & \text{if } X \in R_1 \\ \text{Keyboard}, & \text{if } X \in R_2 \end{cases}$$

where:

R_1, R_2 represent spatial regions on the screen.

8. Key Selection in Virtual Keyboard

Each key is represented as a rectangular region:

$$\text{Key} = \{k_i, \text{ if } (x, y) \in \text{Region}_i\}$$

A key press is triggered when:

- Cursor is inside region
- Click gesture is detected

E. Mode Selection and Control Logic

A region-based interaction is utilized for the mode-switching mechanism, as shown in Fig. 5. For operating the mouse cursor and keyboard, the display space is partitioned into predefined regions. When the index finger tip enters a region and a confirmation gesture is recognized, a mode of choice is triggered.

In the virtual mouse mode, cursor motion is controlled by a proportional coordinate mapping of the index finger tip. Jitter is reduced using exponential smoothing. Click and drag, as well as scroll, gestures are performed via disambiguated combination.

In Virtual Keyboard Mode, there is graphical representation of

keyboard layout. The selection is made by pointing to the region representing the key to be selected and performing a confirmation action. The character selected gets transmitted in the form of key input.

F. Output Execution and Feedback

Well-known gestures are processed to system-level mouse and keyboard events using input control libraries. Additionally, visual feedback in a highlighted landmark and interaction zone is made possible, as shown in Fig. 6, through processing of video feeds to the system. This system loops continuously unless a system exit gesture is recognized.

The real-time virtual keyboard controlled using hand gestures, where detected hand landmarks facilitate precise key selection and highlighting. The system accurately interprets gestures to generate text input, illustrating effective touchless typing, consistent gesture recognition, and smooth, real-time human-computer interaction.

IV. ALGORITHM

The designed system works on the principle of capturing the video feed, detecting the landmarks in the hand, recognizing gestures, and executing actions. In the first step, the video frames are continuously captured using a webcam, and each frame undergoes preprocessing to create a mirror view of the hand. Using the MediaPipe Hands framework, the landmarks in the hand are detected.

Once the landmarks have been successfully identified, the locations of the fingertips and the joints of the hand are extracted. Different gestures are recognized by calculating their geometrical relationship with respect to the landmarks as well as the distance between the landmarks. Specifically, the Euclidean distance between fingertip landmarks is calculated to recognize pinch gestures.

Step 1: Initialize the webcam and hand tracking algorithm.

Step 2: Take a video feed and transform it into an image that can be used for detecting landmarks.

Step 3: Identify the hand and detect the 21 landmarks with help from the MediaPipe Hands model.

Step 4: Show the AirKeys main menu and define the mode of operation depending on the user's gesture choice.

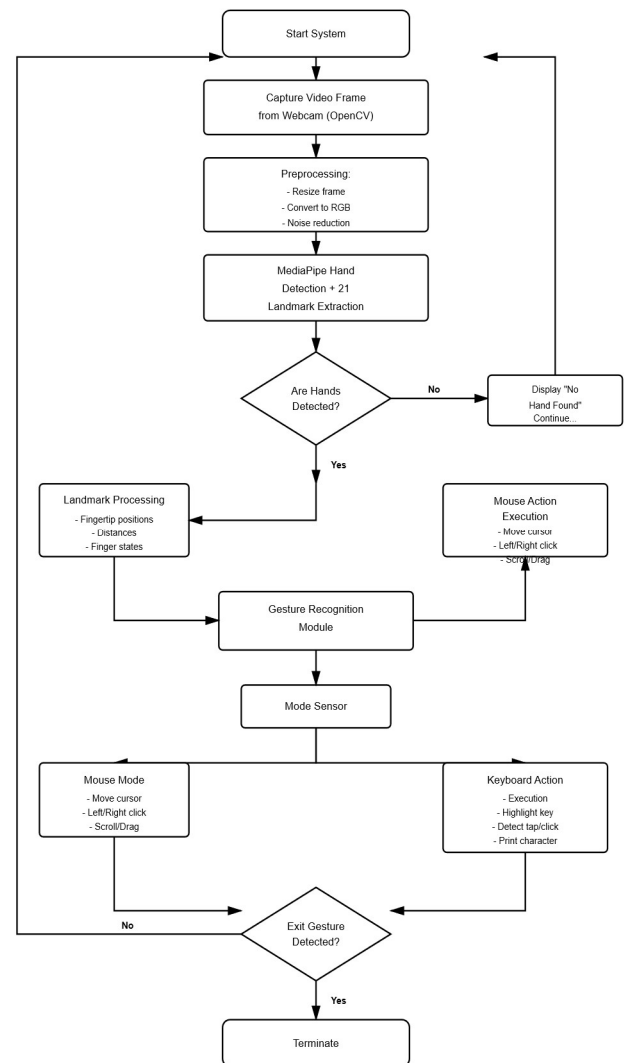


Fig. 8: Flow of System

Step 5: For Mouse Mode, execute the following commands:

- Move the mouse cursor depending on the location of the index finger tip.
- Perform left-click depending on the pinch detected between the index and middle fingers.
- Perform right-click depending on the pinch detected between the thumb and index fingers.
- Scroll up/down depending on whether four fingers are extended; if yes, the scrolling movement depends on the position of the hand on the vertical axis.
- Exit to the main menu when detecting a closed-fist gesture.

Step 6: For Keyboard Mode, show the keyboard interface and track the index fingertip.

Step 7: Determine which letter or symbol is under the hover of the fingertip.

Step 8: Validate the selection of keys by using the index-middle finger pinch gesture and create the required keyboard event.

Step 9: Perform the special functions such as Spacebar, Return, Backspace, Caps lock, and Exit via the gesture validation technique.

Step 10: Give a visual indication to show the active buttons, gestures, and commands.

Step 11: Proceed with processing the frames until the user gives an exit signal; subsequently, release the system resources.

The gesture recognition process primarily relies on Euclidean distance measurements between selected fingertip landmarks. The distance between two landmarks is computed as:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

where $((x_1, y_1))$ and $((x_2, y_2))$ represent the coordinates of the two landmarks. A gesture is considered valid when the computed distance falls below a predefined threshold value. This approach enables reliable real-time interaction while maintaining low computational complexity, making the proposed system suitable for touchless human-computer interaction applications.

V. CONCLUSION

In this paper, the Air Keys system was proposed. It is a virtual mouse and keyboard in real time to enable touchless human computer interaction using gestures of the hands and computer vision. The mouse and keyboard functionality was implemented using the hand landmarks offered by MediaPipe in a gesture recognition approach based on rules, using only a simple webcam without requiring any hardware. The approach was tested for 90% gesture recognition accuracy in real time on CPU alone hardware. The proposed approach is a low-cost clean and usable alternative for conventional input devices and thus applicable, for example, in the field of public computing, medical computing, or assistive computing. Future directions could include interaction using two-hand gestures, better low-light environments, and input fusion.

VI. REFERENCE

- [1] Mallik, B., Rahim, M.A., Miah, A.S.M., Yun, K.S., Shin, J.: Virtual keyboard: A real-time hand gesture recognition-based character input system using LSTM and MediaPipe Holistic. *Computer Systems Science and Engineering* 48(2), 555–570 (2024).
- [2] Jo, B.J., Kim, S.K.: A study on the classification of hand gesture for mobile virtual reality with MediaPipe. *International Journal of Intelligent Systems and Applications in Engineering* 12(21s), 3555–3559 (2024).
- [3] Rahim, M.A., Shin, J., Hashem, M.A., Akash, H.S., Hossain, M.I., Miah, A.S.M.: Advanced touchless HCI: A gesture recognition-based multilingual virtual keyboard for character input. *Journal of Tianjin University Science and Technology* 57(11), 1–20 (2024).
- [4] Wang, Q., Xie, Z.: HandMouse: A gesture-based virtual mouse system. *International Journal of Advanced Computer Science and Applications (IJACSA)* 15(11), 657–668 (2024).
- [5] Yadav, V.J., Reddy, Y.K., Kumar, P.D., Ramya, C., Kumar, M.S.: Gesture controlled virtual mouse using AI. In: *IEEE Conference Proceedings*, pp. 1–5 (2024).
- [6] Kavuri, M., Madhumitha, T.S., Nagesh, Pratheek, B.K., Trisha, G.: Hands-free computing: A gesture-controlled virtual AI mouse. *Journal of Emerging Technologies and Innovative Research* 12(6), 545–550 (2025).
- [7] Shen, J., Boldu, R., Kalla, A., Glueck, M., Surale, H.B., Karlson, A.: RingGesture: A ring-based mid-air gesture typing system powered by a deep-learning word prediction framework. *arXiv preprint arXiv:2410.18100* (2024).
- [8] Linardakis, M., Varlamis, I., Papadopoulos, G.Th.: Survey on hand gesture recognition from visual input. *IEEE Access* 13, 135373–135377 (2025).
- [9] Bisht, D., Pal, A., Banerjee, S.: VKM: A virtual keyboard and mouse solution towards a lightweight computing system. In: *2024 20th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, pp. 254–258. IEEE (2024).
- [10] Meng, Y., Jiang, H., Duan, N., Wen, H.: Real-time hand gesture monitoring model based on MediaPipe's registerable system. *Sensors* 24(19), 6262 (2024).
- [11] A. Goel, "A Vision-Based Virtual Mouse Using Hand Gesture Recognition," *TechRxiv*, Jun. 2025. doi: 10.36227/techrxiv.174914054.46512037/v1.
- [12] G. Kotari, K. Jangam, K. S. Sai, and M. Aparna, "Augmented Virtual Mouse System with Enhanced Gesture Recognition," *International Journal of Engineering Research & Technology (IJERT)*, vol. 12, no. 3, 2024.

- [13] S. Muslimin, E. Prihatini, and T. Martin, "Development of a Hand Gesture Detection-Based Robot System with MediaPipe," *Indonesian Journal of Artificial Intelligence and Data Mining (IJAIDM)*, vol. 8, no. 3, pp. 737–745, Nov. 2025. doi: 10.24014/ijaidm.v8i3.37678.
- [14] S. S., "Hand Gesture Recognition-Based Virtual Mouse Using CNN," *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, vol. 13, no. 4, Apr. 2025. doi: 10.22214/ijraset.2025.68259.
- [15] G. Amprimo, G. Masi, G. Pettiti, G. Olmo, L. Priano, and C. Ferraris, "Hand Tracking for Clinical Applications: Validation of the Google MediaPipe Hand (GMH) and the Depth-Enhanced GMH-D Frameworks," *Elsevier Preprint*, 2024.
- [16] R. Kumar, S. K. Singh, A. Bajpai, and A. Sinha, "Mediapipe and CNNs for Real-Time ASL Gesture Recognition," *International Conference/Research Publication*, 2023.
- [17] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, "MediaPipe Hands: On-Device Real-Time Hand Tracking," *arXiv preprint arXiv:2006.10214*, Jun. 2020.
- [18] J. Nanajkar, A. Pisal, A. Patil, V. Sathe, and S. Dighe, "Virtual Mouse & Keyboard Using Hand Gesture," *International Journal of Creative Research Thoughts (IJCRT)*, vol. 13, no. 3, Mar. 2025.
- [19] G. M. Trupti, C. Kumar, D. P., V. Kumar, and P. K. Shivaraddi, "Virtual Mouse Using Hand Gestures," *International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE)*, vol. 13, no. 4, Apr. 2024. doi: 10.17148/IJARCCE.2024.134203.
- [20] G. Jagnade, M. Ikar, N. Chaudhari, and M. Chaware, "Hand Gesture-Based Virtual Mouse Using OpenCV," in *Proceedings of the International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)*, IEEE, 2023, pp. 820–825.