WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

AirNav

Mr. Parwateeswar Gollapalli[1], Mk Revan[2], K. Niveditha Sai[3], G Bhanu Prakash[4], G Praveen Kumar[5]

[1] Assistant Professor, Dept. of CSE-Data Science,ACE Engineering College, India
[2345] B. Tech CSE-Data Science, ACE Engineering College, India
Email: parwateeswar.g@gmail.com[1], revanmk08@gmail.com[2], kondalaniveditha@gmail.com[3], bhanuprakash6705@gmail.com[4], gugulothp262@gmail.com[5].

A B S T R A C T

In response to the growing demand for intuitive and hygienic human-computer interaction, this paper presents AirNav, a vision-based virtual mouse and navigation system that utilizes real-time hand gesture recognition to interact with computers. The system enables users to control cursor movement, perform mouse clicks, scroll pages, and adjust system volume and brightness without physical contact. Built using Python, OpenCV, MediaPipe, and PyAutoGUI, AirNav offers a seamless and touchless experience suitable for public, medical, and assistive technology contexts. Real-time performance, minimal hardware requirements, and open-source architecture make it an effective and accessible alternative to traditional input devices.

Keywords: Gesture Recognition, Virtual Mouse, OpenCV, Mediapipe, Human-Computer Interaction, Touchless Control, Assistive Technology.

1. Introduction

The rapid evolution of human-computer interaction (HCI) has shifted focus toward more natural and intuitive control systems. Traditional input devices such as keyboards and mice have limitations in terms of hygiene, accessibility, and ergonomics. In certain scenarios—like medical laboratories, public kiosks, or for users with physical impairments—contactless interaction is essential.

AirNav was developed to bridge this gap by offering a fully touchless, gesture-based control system using a standard webcam. Users can perform actions like cursor movement, clicking, scrolling, volume adjustment, and brightness control with specific hand gestures. The system integrates several computer vision and automation tools to deliver a high-accuracy, real-time user experience.

2. Literature Review

2.1 Existing Systems

- Leap Motion Controller: Uses infrared sensors but requires expensive proprietary hardware.
- Microsoft Kinect: Suitable for full-body motion, lacks portability
.- Voice Assistants: Not suitable for visual interactions or noisy environments.
- Touchscreens: Not hygienic for public or sterile use.

2.2 Drawbacks of Existing Systems

- High cost and hardware dependency
- Learning complexity and calibration needs
- Low flexibility and closed-source limitations.

2.3 Related Research Studies- 2024

– AI Virtual Mouse (Deekshith et al.) combines voice and gesture- 2023

– Real-time MediaPipe fingertip control (Shriram et al.)

- 2020 – Colored fingertips with OpenCV (Teja et al.)

2.4 Proposed System

AirNav uses only a basic webcam and an open-source software stack. It improves upon the accuracy, cost-efficiency, and adaptability of previous systems, enabling dynamic hand gesture recognition under various environmental conditions.

2.5 Summary

AirNav offers a cost-effective and adaptable HCI alternative, especially in assistive and public interfaces.

3. Methodology

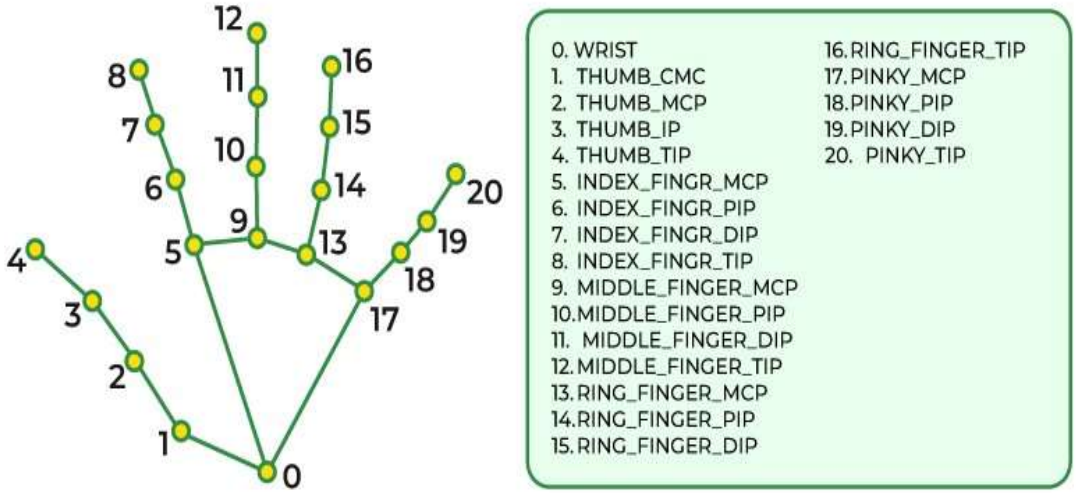

Fig. 1. AIRNAV System Methodology.

The development of the "Digital Navigator Using Deep Learning" follows a modular and layered methodology designed for scalability, real-time responsiveness, and adaptability. Built using Python and key libraries like OpenCV, MediaPipe, and PyAutoGUI, the system captures live video through a webcam and processes each frame for hand detection. MediaPipe identifies 21 key landmarks on the hand, while OpenCV converts and preprocesses the video stream to the RGB format required for accurate detection. Feature extraction relies on analyzing distances between fingertips and joints, with gesture patterns encoded into binary formats that are compared against known templates. To ensure reliability, gestures are confirmed only when consistently recognized across successive frames. Once identified, gestures are mapped to specific commands such as cursor movement, click actions, scrolling, or brightness/volume adjustments. These commands are executed through PyAutoGUI, while feedback is visualized using OpenCV overlays.

To optimize usability across diverse conditions, the system undergoes iterative testing in various lighting scenarios (e.g., low light or high contrast). Key performance indicators such as gesture accuracy, latency, CPU usage, and gesture misclassification rates are monitored and refined through feedback loops. Enhancements such as delay buffers and gesture stability checks are applied to maintain robustness. The system's design emphasizes real-time performance and accuracy, making it suitable for deployment in assistive technologies, smart environments, or touchless workspaces. This feedback-driven approach ensures the system is user-friendly, adaptive, and extensible for future upgrades.

To achieve clarity and maintainability, the system is divided into several modules. The **Hand Detection Module** extracts 21 landmarks using MediaPipe for each frame, delivering them as coordinate data. The **Cursor and Click Control Module** tracks gesture-based inputs to control the mouse cursor and perform click or drag actions using PyAutoGUI and NumPy-based distance calculations. The **Brightness Adjustment Module** alters screen brightness based on the distance between middle and ring fingers using the screen_brightness_control library. Similarly, the **Volume Control Module** adjusts system audio using gestures processed via the pycaw library. The **Main Driver Script** orchestrates the video stream, gesture recognition, and module execution using conditional logic. Meanwhile, the **Tkinter-based UI** allows the user to select modes, toggle gestures, and receive real-time visual feedback, thus offering an interactive and seamless user experience.

3.1 System Architecture

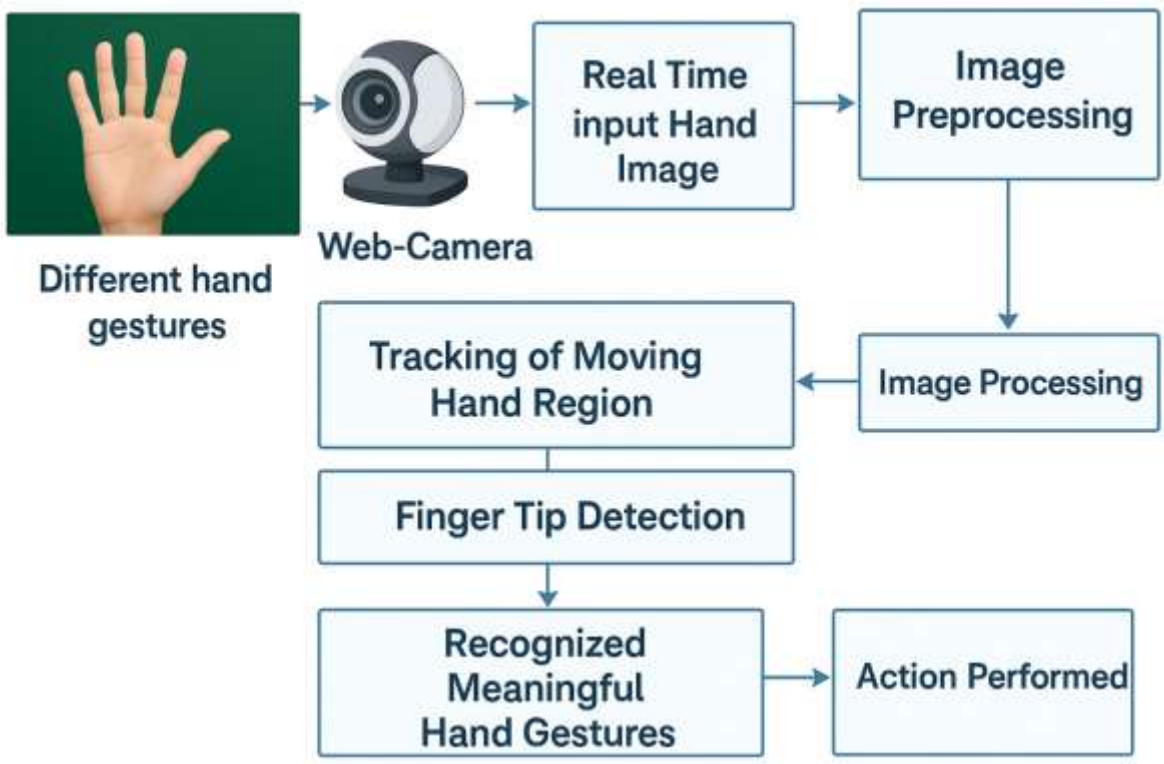

Fig. 2. AIRNAV System Architecture

1. Input: Different Hand Gestures
    - A user presents various hand gestures in front of a camera.
    - These gestures form the basic input to the system.

2. Capture Device: Web Camera
    - The webcam captures real-time images of the user's hand.
    - It continuously streams the hand gestures to the system.

3. Real-Time Input Hand Image
    - The image of the hand from the webcam is sent into the system as input data for processing.

4. Image Preprocessing
    - The raw image is first preprocessed to improve quality.
    - Operations may include resizing, filtering, background removal, and contrast adjustment.

5. Image Processing
    - Key features of the hand (like contour, edges, or skin tone) are extracted.
    - The processed image is then analyzed to isolate the hand region from the background.

6. Tracking of Moving Hand Region
    - The system tracks the motion of the hand in real-time.
    - Movement dynamics are monitored to differentiate between static and dynamic gestures.

7. Finger Tip Detection
    - The system detects fingertip positions from the tracked hand.

- Fingertip positions help define gesture types like pointing, clicking, or dragging.

8. Recognized Meaningful Hand Gestures
    - Based on fingertip data and hand movement, the system recognizes specific gestures.
    - Each gesture is mapped to a predefined command or function.
9. Action Performed
    - The system executes a specific action based on the recognized gesture.
    - For example: move cursor, click, scroll, zoom, etc.

4. Output Screens:

Fig 3.User Interface

Fig 4. Mouse Mode clicked

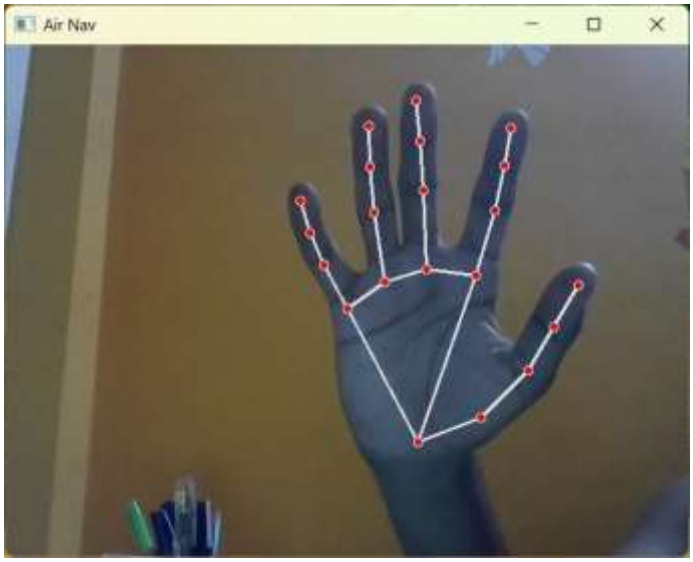

Fig 5. Tracking Mouse

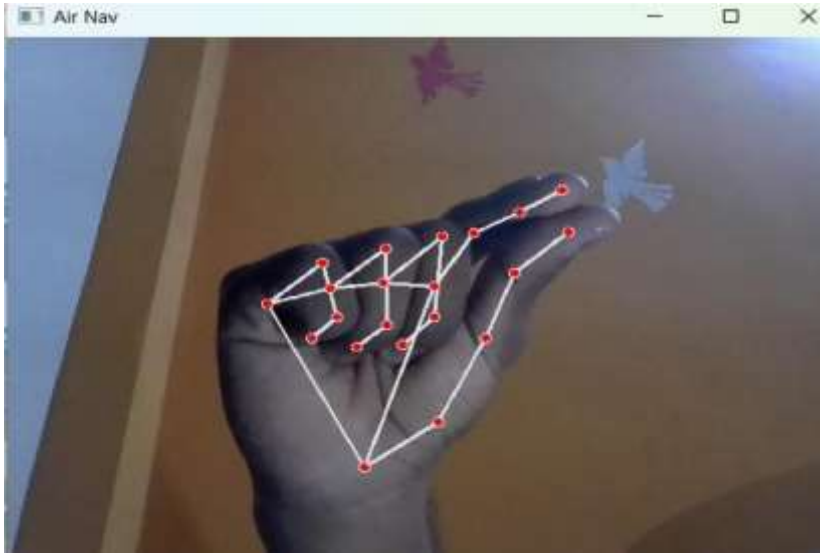

Fig 6. Left Click Performed

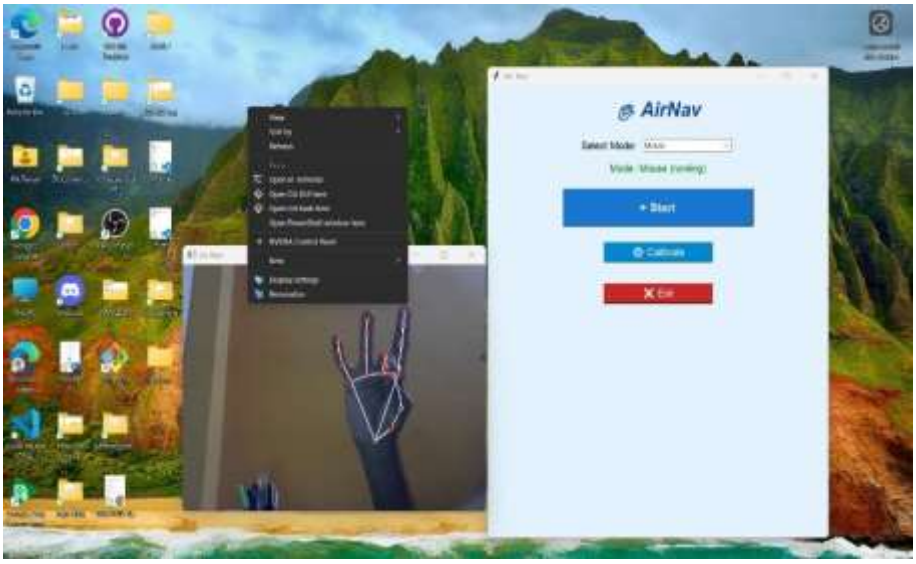

Fig 7.Right Click Performed

5. Work Flow:

1. Start & Mode Selection

- **User launches the GUI (Tkinter).**
- **Selects a mode: Mouse, Scroll, Volume, Brightness, or All.**
- **Webcam starts capturing live video using OpenCV.**

2. Frame Capture & Hand Detection

- **Each frame is converted to RGB.**
- **MediaPipe detects 21 hand landmarks (fingertips, joints).**

3. Gesture Recognition

- **Calculates distances/angles between landmarks.**
- **Matches gesture to predefined templates.**
- **Confirms only stable gestures (avoids errors due to motion).**

4. Gesture-to-Command Mapping

Mode	Gesture → Action
Mouse	Finger movements → Move, Click, Drag
Scroll	Index up/down → Scroll screen
Volume	Thumb ↔ Ring finger distance → Adjust volume
Brightness	Middle ↔ Ring finger distance → Adjust brightness
All	Combines all controls

5. Action Execution

- **Uses PyAutoGUI to simulate:**
  - **Mouse actions**
  - **Scrolling**
  - **Volume/Brightness control**
- **Real-time visual feedback shown with OpenCV overlays.**

6. Loop or Exit

- **System keeps running until the user presses Exit or ESC.**
- **Webcam is released and windows are closed.**

6. Conclusion and Future scope:

AirNav demonstrates the feasibility of a webcam-based virtual mouse using hand gestures. It performs well under real-time constraints and is suitable for hygienic, accessible, and assistive technology use cases. The system is modular, extensible, and compatible with most modern PCs

Future Scope:

- Custom gesture training using neural networks
- Multimodal interaction (voice + gesture)
- Full keyboard emulation via gestures

- Integration with AR/VR platforms
- Packaging into a cross-platform executable (EXE/DMG)

7.Acknowledgement:

We express our sincere gratitude to all who supported us throughout the development of this project.We are especially thankful to Prof. Y.

V. Gopala Krishna Murthy, General Secretary, and Mrs. M. Padmavathi, Joint Secretary, for providing us the opportunity and environment to carry out this work.Our heartfelt thanks to Dr. P. Chiranjeevi, Head of the Department, for his guidance and encouragement.We are deeply grateful to our internal guide Mr. G. Parwateeswar, and project coordinator Mrs. B. Saritha, for their consistent support, valuable feedback, and motivation throughout the project.

Lastly, we thank all the faculty members, staff for their constant encouragement and support.

8.References:

[1] U Sairam, Dharani kumar Reddy Gowra, Sai Charan Kopparapu, "Virtual mouse using Machine Learning and GUI Automation", 07 June 2022.[https://ieeexplore.ieee.org/abstract/document/9784972]

[2] Roshnee Matlani, Roshan Dadlani, Sharv Dumbre, Shruti Mishra, Abha Tewari, "Virtual mouse using hand gestures", 14 January 2022.[https://ieeexplore.ieee.org/abstract/document/9673251]

[3] Manav Ranawat, Madhur Rajadhyaksha, Neha Lakhani, Radha Shankarmani, "Hand Gesture Recognition Based Virtual Mouse Events", 22 June 2021. [https://ieeexplore.ieee.org/abstract/document/9456388]

[4] Vantukala VishnuTeja Reddy, Thumma Dhyanchand, Galla Vamsi Krishna, Satish Maheshwaram, "Virtual Mouse Control Using Colored Finger Tips and Hand Gesture Recognition", 03 November 2020.[https://ieeexplore.ieee.org/abstract/document/9242677]

[5] Sugnik Roy Chowdhury, Sumit Pathak, M.D. Anto Praveena, "Gesture Recognition Based Virtual Mouse and Keyboard", 17 July 2020. [https://ieeexplore.ieee.org/abstract/document/9143016]

[6] Monali Shetty, Christina A.Daniel, Manthan K. Bhatkar, Ofrin P.Lopes, "Virtual Mouse Using Object Tracking", 10 July 2020. [https://ieeexplore.ieee.org/abstract/document/9137854