ElectroSphere: Electrical Electronics Engineering Bulletin

PRIME OPEN ACCESS

Volume 1, Issue 2

Research Article

Date of Submission: 25 June, 2025
Date of Acceptance: 11 August, 2025
Date of Publication: 19 August, 2025

A Vision-Based Virtual Mouse Using Hand Gesture Recognition

Aaditya Goel*

Undergraduate Student, Department of Electronics and Communication Engineering, Maharaja Agrasen Institute of Technology, India

***Corresponding Author:**
Aaditya Goel, Undergraduate Student, Department of Electronics and Communication Engineering, Maharaja Agrasen Institute of Technology, India.

Citation: Goel, A. (2025). A Vision-Based Virtual Mouse Using Hand Gesture Recognition. *Electro Sphere Electr Electronic Eng Bull, 1*(2), 01-08.

Abstract
Recent advancements in gesture detection and hand tracking have created both significant opportunities and challenges in the field of virtual reality (VR) and user interaction. With the increased focus on reducing physical interactions due to the COVID-19 pandemic, the need for alternative, non-contact methods of computer operation has grown [1]. These innovations hold the potential to reduce reliance on traditional input devices like keyboards and mice, fostering a more immersive and intuitive user experience.

One exciting direction for future development is the ability to perform complex tasks, such as clicking, dragging objects and controlling purely through hand gestures. This could open up new possibilities for VR environments, where users can interact naturally with digital objects. The key advantage of such an input method is that it requires only a camera, making it accessible and adaptable to various settings without the need for additional hardware.

In the future, gesture recognition may become an essential part of our interaction with technology, leading to seamless and efficient virtual experiences. The proposed project, leveraging Python and OpenCV for gesture recognition, seeks to explore these possibilities and inspire further research in this field.

Keywords: Gesture Control, Media Pipe, OpenCV, Pyauto-GUI

Introduction
This paper introduces a cost-effective visual AI system using computer vision to emulate mouse, keyboard, and stylus functions through hand gesture and fingertip tracking, all with a standard webcam. Developed in Python with OpenCV, MediaPipe, and libraries like PyAutoGUI, PyCaw, and screen brightness-control, it performs tasks such as clicking, scrolling, and adjusting volume or brightness, offering precise control even on a CPU [2].

This approach allows real-time gesture interpretation, enhancing Human-Computer Interaction (HCI) for applications like sign language recognition and motion control interfaces. Inspired by motion-based systems like the Nintendo Wii, which demonstrated the appeal of gesture-based interaction, this system is suitable for VR, gaming, and healthcare applications [3,4].

Hand gestures enable intuitive and immersive computer interaction, supporting complex actions for applications like sign language or therapy. The system's real-time tracking provides precise feedback, vital for simulations and educational tools [5,6]. As hand tracking advances, it will impact areas from communication to rehabilitation, making digital interactions more natural and accessible [7].

Major Contributions

- The visual AI mouse system marks a notable advancement in human-computer interaction by introducing intuitive hand gesture controls for volume and brightness adjustments. This system utilizes a combination of libraries, including Mediapipe, OpenCV, PyAutoGUI, Pycaw, and the set brightness library, to create a comprehensive, hands-free interface that goes beyond traditional mouse functionalities.
- The implementation begins with Mediapipe's robust hand tracking framework, which efficiently detects and tracks hand landmarks in real-time. This framework forms the basis for interpreting various hand gestures. OpenCV is then used to process image frames captured from the webcam, analyzing hand movements and mapping them to mouse-like actions. For instance, gestures such as moving the hand across the screen control the mouse cursor, while pinching or tapping gestures simulate click actions. These intuitive interactions provide an accessible and ergonomic way to interact with digital environments without the need for physical hardware.
- A significant enhancement of the system is the integration of volume and brightness control through simple hand gestures. The Pycaw library is employed to manage audio settings, interfacing with the Windows Core Audio API to enable smooth and responsive volume adjustments. By analyzing the distance between the thumb and index finger, the system dynamically increases or decreases the volume. Similarly, the set brightness library is used for adjusting screen brightness. Specific gestures mapped to brightness control allow users to effortlessly modify the display settings. PyAutoGUI ties everything together by ensuring seamless execution of these system level changes.
- This AI-driven system's ability to control both volume and brightness hands-free significantly enhances user experience and accessibility. By incorporating these features, the system meets the modern demand for more natural, intuitive, and immersive human-computer interactions. Its applications are extensive, ranging from providing accessibility for individuals with physical limitations to offering immersive experiences in gaming and virtual reality. The touchless interface is versatile and adaptable, paving the way for more advanced and ergonomic ways to engage with digital technology. Overall, this visual AI mouse system demonstrates the potential for more seamless, intuitive, and inclusive technology in everyday applications.

Proposed System

The goal of this project is to design a finger-only virtual mouse system that tracks hand movements to perform mouse functions like left-clicking, right-clicking, and scrolling. This system is ideal for space-constrained environments and eliminates the need for additional hardware or sensors, making it an affordable, user-friendly alternative to traditional input devices [8,9].

The system uses the OpenCV library for image processing and the MediaPipe framework for precise hand gesture tracking [10]. Several libraries, such as PyAutoGUI, Pynput, and Autopy, enhance its functionality by simulating mouse clicks and keyboard inputs based on gestures. These integrations allow for smooth gesture-to-action mapping.

Additionally, the system incorporates PyCaw for volume control and the screen-brightness-control library to adjust screen brightness, offering a versatile hands-free user experience.

Combining these libraries, the system runs efficiently across desktop and mobile platforms, making it scalable for applications in accessibility, gaming, virtual reality, and productivity software, providing a comprehensive solution for hands-free interaction with digital devices.

Algorithm

- Start the program.
- Open the file and, using the file location, go to the command prompt (CMD).
- Use the required libraries (e.g., MediaPipe, OpenCV) to execute the program.
- Initialize the system and begin video capturing from the webcam.
- Capture frames continuously from the webcam. 6) Detect hands and hand tips using MediaPipe and OpenCV.
- Identify which fingers are raised (UP).
- Recognize the gesture based on the hand position.
- Perform mouse operations (e.g., left-click, right-click, scroll) according to the detected gesture.
- Detect which hand is raised for controlling additional functions.
- Perform volume or brightness control based on the hand gesture (e.g., pinch to adjust volume or brightness).
- Stop the program.

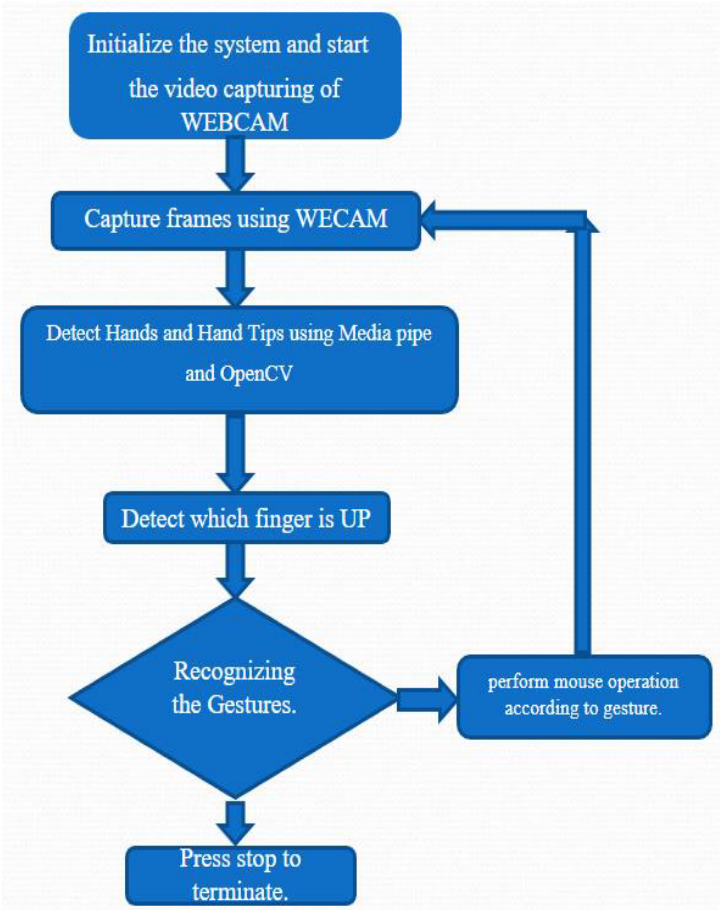

Figure 1: Flowchart of Project

Methodology

OpenCV

OpenCV is a widely-used computer vision library that provides a variety of tools and techniques for image processing and object detection. It enables the development of real-time computer vision applications in Python. By utilizing OpenCV, developers can analyze visual data from images and videos, including tasks such as face and object detection. OpenCV is an open-source, free-to-use library that focuses on computer vision and machine learning tasks. It provides a robust framework for building computer vision applications, which in turn accelerates the integration of artificial intelligence into products. Thanks to its Apache 2.0 license, OpenCV allows businesses to easily customize and adapt the code for their specific needs.

Media Pipe

MediaPipe, an open-source framework by Google, enables cross-platform machine learning pipelines using time series data. It supports various audio and video formats and uses graph-based models for system design and analysis. Key components include performance evaluation, sensor data access, and calculators, which are customizable building blocks linked by data streams. MediaPipe's flexible setup allows for seamless use on both desktop and mobile platforms [11].

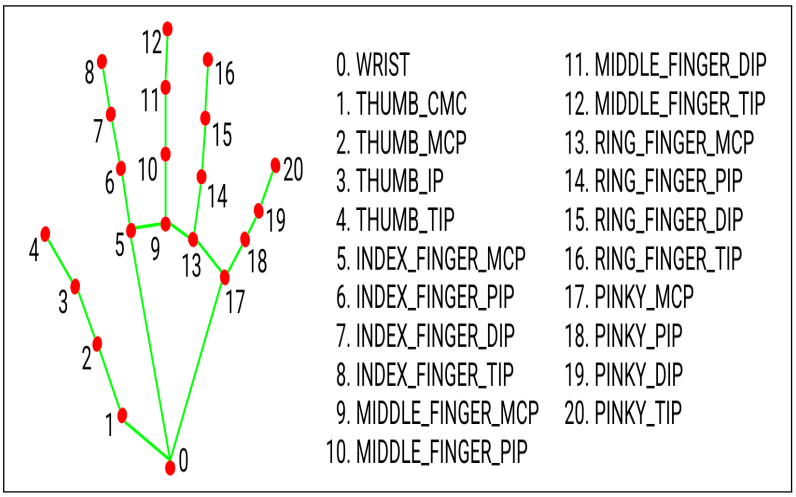

Figure 2: Media Pipe Hand Landmarks

MediaPipe Hands identifies 21 key points on the human hand. Each landmark has a specific index and position, enabling a comprehensive representation of the hand's pose and orientation. Here's a breakdown of these landmarks:

- **Wrist:** The first landmark (ID 0) represents the wrist and serves as the anchor point for the rest of the hand landmarks.
- **Thumb:** There are four key landmarks on the thumb:
  Thumb CMC (ID 1): The base joint connecting the thumb to the hand.
  Thumb MCP (ID 2): The metacarpophalangeal joint, near the knuckle of the thumb.
  Thumb IP (ID 3): The interphalangeal joint, located midway along the thumb.
  Thumb Tip (ID 4): The tip of the thumb, representing the outermost point.
- **Index Finger:** This finger also has four significant landmarks:
  Index MCP (ID 5): The metacarpophalangeal joint at the base of the index finger.
  Index PIP (ID 6): The proximal interphalangeal joint, near the middle of the finger.
  Index DIP (ID 7): The distal interphalangeal joint, closer to the fingertip.
  Index Tip (ID 8): The tip of the index finger.
- **Middle Finger:** Similarly, the middle finger includes:
  Middle MCP (ID 9): The base joint of the middle finger.
  Middle PIP (ID 10): The middle joint of the finger. 3) Middle DIP (ID 11): The joint closer to the fingertip. 4) Middle Tip (ID 12): The tip of the middle finger.
- **Ring Finger:** It features:
  Ring MCP (ID 13): The base joint of the ring finger.
  Ring PIP (ID 14): The middle joint.
  Ring DIP (ID 15): The joint near the fingertip.
  Ring Tip (ID 16): The tip of the ring finger.
- **Pinky Finger:** The smallest finger also has four landmarks:
  Pinky MCP (ID 17): The base joint of the pinky finger.
  Pinky PIP (ID 18): The middle joint.
  Pinky DIP (ID 19): The joint close to the tip.
  Pinky Tip (ID 20): The tip of the pinky finger.

Convolutional Neural Network (CNN)

MediaPipe uses a Convolutional Neural Network (CNN) architecture as shown in figure 3 for various tasks, including hand tracking and gesture recognition. In the context of hand tracking, MediaPipe employs a specialized model to detect and track hand landmarks in real-time, leveraging CNNs to process images and videos.

- **Hand Landmark Detection:** MediaPipe's hand tracking model is based on a CNN to detect and estimate the 3D positions of hand landmarks in images or videos. The network is trained to recognize key points of the hand, such as the wrist, knuckles, fingertips, and joints, in different poses and orientations.
- **Real-Time Processing:** The CNN architecture in MediaPipe is optimized for real-time performance. It uses lightweight layers and efficient design strategies, which help in processing input data quickly without significant latency. This makes MediaPipe suitable for interactive applications, such as virtual mouse control and gesture recognition.
- **Multi-Pose and Multi-View Support:** The CNN is designed to handle different hand poses and multiple views. It can accurately track the hand in various orientations, whether the hand is rotated or partially occluded, and even when multiple hands are present in the frame.
- **Feature Extraction:** CNNs are highly effective at extracting spatial features from images, such as edges, textures, and shapes. In the case of hand tracking, the network is trained to identify the key features of a hand, which allows it to determine the location of each hand landmark. The convolutional layers learn to recognize patterns specific to the hand, while the pooling layers help reduce the dimensionality and computational complexity.
- **Output:** The output of the CNN consists of a set of keypoints corresponding to different parts of the hand (21 in total) in 3D space, represented as coordinates (x, y, z). These keypoints can then be used to infer gestures, track movements, or perform actions such as controlling a mouse or adjusting volume and brightness [12].

PyAutoGUI

PyAutoGUI is a Python package designed for crossplatform GUI automation, compatible with Windows, macOS, and Linux. It allows users to simulate keyboard input, mouse movements, clicks, and other actions. By providing a simple API, PyAutoGUI abstracts the complexities of controlling the mouse and keyboard, making it easier to automate tasks. With PyAutoGUI, you can precisely click, drag, scroll, and move the cursor, among other functions. This automation tool simplifies what would otherwise be complex and technical processes across different operating systems, offering a unified solution for GUI control [13].

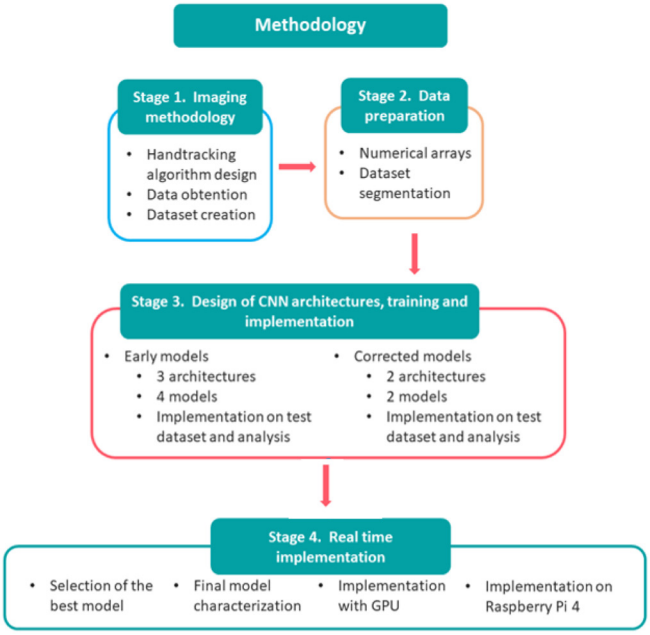

Figure 3: CNN Used by Media Pipe

Math

Mathematics plays a crucial role in various aspects of computer science and automation. In the context of automation with tools like PyAutoGUI, mathematical principles are applied to ensure accurate and efficient execution of tasks. Here's how:

- **Angles:** Calculating angles is essential for determining the direction of movements. For example, in automated mouse gestures or drawing shapes, angles help define the path between two points, enabling smooth transitions in a desired direction.

$$\theta = |\text{atan2}(y_C - y_B, x_C - x_B) - \text{atan2}(y_A - y_B, x_A - x_B)| \times \frac{180}{\pi} \qquad (1)$$

- **Hypotenuse Calculation:** The Pythagorean theorem is frequently used to compute the shortest distance between two points in a 2D space. This allows for efficient movement of the cursor, ensuring it travels the minimal distance to reach the target location. [14] The Euclidean distance L between the two points (x_1, y_1) and (x_2, y_2) is given by:

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

These mathematical concepts, when applied correctly, contribute to the precision and smoothness of automated actions, making tools like PyAutoGUI effective for GUI automation across various platforms.

Pycaw

PyCaw (Python Core Audio Windows Library) is a Python library used to interact with the audio system on Windows. It provides an easy-to-use interface for controlling various audio settings, such as volume, mute, and balance. With PyCaw, you can programmatically manipulate audio outputs on a Windows machine, including adjusting the volume levels, muting or unmuting sound, and even controlling specific application volumes. PyCaw allows you to adjust the system volume or the volume of specific applications.

Set Brightness Control

Set Brightness Control refers to the ability to programmatically adjust the screen brightness on a device. This can be achieved through various libraries and APIs depending on the operating system. This Python package provides a simple interface to control the brightness of the screen on both Windows and Linux systems. It allows you to set the brightness to a specific percentage or adjust it by a given amount.

Result

Camera Used in Visual AI System

The proposed AI mouse system is based on frames captured by a web camera on a portable computer or PC. Using the OpenCV Python computer library, a video recorder is created and the webcam will begin capturing video, as shown in Figure 4. The web camera captures and transmits frames to the visual AI system [8].

Figure 4: Capturing Video using the Webcam

Capturing the Video and Processing

The visual AI mouse system uses a webcam where each frame is scanned until the program is completed. Video frames are processed from BGR to RGB color space to get hands on the video frame by frame as shown in the following code:

Mouse Control

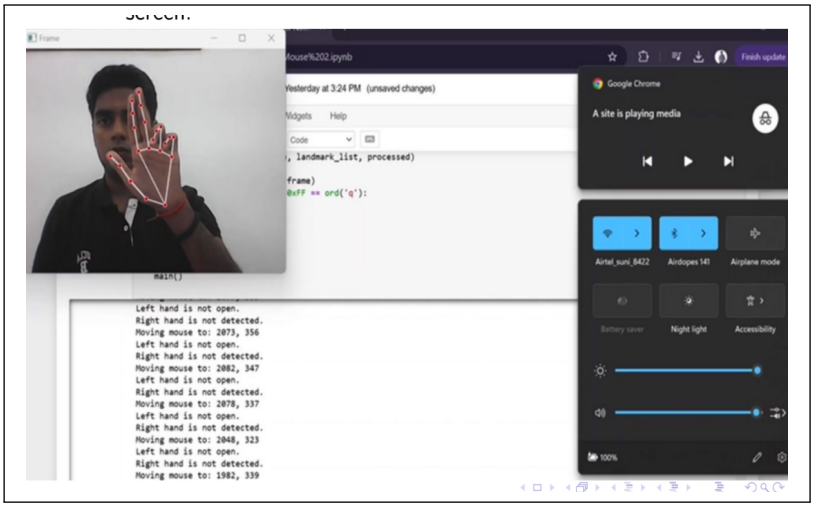

Figure 5: Mouse Control Using Hand Gesture

Key Features of Mouse Control

- **Mouse Movement:** The ability to move the mouse pointer to a specific position on the screen (often represented by x and y coordinates). This is useful for automating GUI interactions where specific areas on the screen need to be clicked or hovered over. The position of the index finger tip and middle finger landmarks can be used to move the mouse cursor as shown in figure 5. By mapping the coordinates of the fingertip to the ensure smooth cursor movement, you can average out or interpolate the positions of the fingertip over multiple frames.
  imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) self.results = self.hands.process(imgRGB) [9]
- **Mouse Click:** Simulating mouse button clicks (left, right, middle) at specific screen locations. This is essential for automating tasks that require user interaction, like clicking buttons, links, or menus. Index finger is used for 'left-click', middle finger is used for 'right-click' and both fingers are used for 'double-click'.

Gesture - Based Custom Controls

Figure 6: Brightness Control Using Hand Gesture

- **Brightness Control:** The ability to control brightness can be done by python libraries such as screen-brightnesscontrol which can be used to control brightness. The screen-brightness-control library is a Python package that allows you to interact with and control the screen brightness on various operating systems, such as Windows, Linux, and MacOS. It provides a simple and easy-to-use interface to get the current screen brightness and adjust it programmatically.
- The system will detect if the both hands are raised. Once the both hands are identified as active, the gesture control logic will be activated. By measuring the distance between the index finger and thumb of the right hand, you can interpret this distance as a way to adjust the brightness. The closer the fingers are, the lower the brightness, and vice versa.

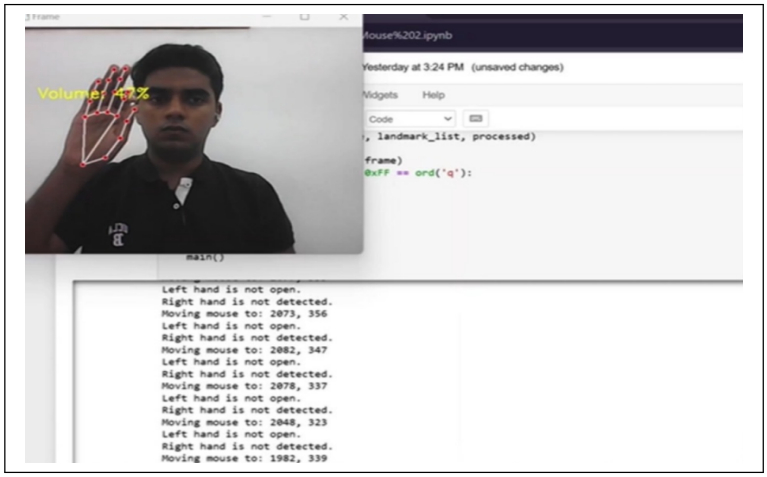

Figure 7: Volume Control Using Hand Gesture

- **Volume Control:** The ability to control volume can be done by python libraries such as Pycaw which can be used to control volume. Pycaw is a Python library that provides an interface for controlling the audio properties of a Windows system. It utilizes the Windows Core Audio API to give developers access to various features, such as adjusting volume levels, retrieving device information, and managing audio sessions.
- The system will detect if the left hand is raised. Once the left hand is identified as active, the gesture control logic will be activated. By measuring the distance between the index finger and thumb of the left hand, you can interpret this distance as a way to adjust the volume. The closer the fingers are, the lower the volume, and vice versa.

Conclusion

- The visual AI mouse system revolutionizes user interaction by using a webcam to translate hand gestures into mouse functions like cursor movement, clicking, scrolling, and adjusting settings, eliminating traditional hardware and providing a more ergonomic and hygienic experience.
- This AI system shows high accuracy compared to existing models, making it ideal for minimizing physical contact and assisting those with mobility impairments, while also addressing hygiene concerns in post-pandemic environments.
- However, challenges remain, such as inconsistent rightclick functionality and less precise actions for tasks like text selection or dragging, which can impact user efficiency and satisfaction.
- The system's potential lies in refining fingertip detection and improving accuracy for precision tasks, making it well-suited for environments prioritizing reduced contact.
- With further research, this touchless computing innovation could significantly enhance human-computer interaction,

offering a seamless, accessible, and intuitive user experience.

References

1. Raffel, C., and Shinn, C. (2017). Human-Computer Interaction through Gestures. IEEE Access, 5, 4253-4263.
2. Rao, A.K., Gordon, A.M., (2001). Contribution of tactile information to accuracy in pointing movements. Exp. Brain Res. 138, 438–445.
3. Lira, M., Egito, J. H., Dall'Agnol, P. A., Amodio, D. M., Gonçalves, Ó. F., & Boggio, P. S. (2017). The influence of skin colour on the experience of ownership in the rubber hand illusion. Scientific Reports, 7(1), 1-13.
4. Meta. (2021). Inside Facebook Reality Labs: Wrist-based interaction for the next computing platform.
5. Carlton, B. (2021). HaptX Launches True-Contact Haptic Gloves For VR And Robotics. VRScout.
6. Brenton, H., Gillies, M., Ballin, D., & Chatting, D. (2005, September). The uncanny valley: does it exist. In Proceedings of conference of human computer interaction, workshop on human animated character interaction. Pennsylvania: Citeseer.
7. Katona, J. (2021). A review of human–computer interaction and virtual reality research fields in cognitive InfoCommunications. Applied Sciences, 11(6), 2646.
8. Matlani, R., Dadlani, R., Dumbre, S., Mishra, S., & Tewari, A. (2021, November). Virtual mouse using hand gestures. In 2021 international conference on technological advancements and innovations (ICTAI) (pp. 340-345). IEEE.
9. Kumar, A., Chaudhary, A., Singhal, A., Dev, A., & Jaiswal, A. (2024, March). Gesture and Voice Controlled Virtual Mouse: A Review. In 2024 2nd International Conference on Disruptive Technologies (ICDT) (pp. 876-879). IEEE.
10. Mary, M. S. I., Anand, M. S., Manikandan, A. S., & Senthamilmuthu, M. (2024). Hand Gesture Recognition using Mediapipe and OpenCV. International Journal of Creative Research Thoughts (IJCRT), 12(2).
11. Kumar, R., Bajpai, A., & Sinha, A. (2023). Mediapipe and cnns for real-time asl gesture recognition. arXiv preprint arXiv:2305.05296.
12. Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C. L., & Grundmann, M. (2020). Mediapipe hands: On-device real-time hand tracking. arXiv preprint arXiv:2006.10214.
13. T. Tanner, (2019). "Python GUI Automation Using PyAutoGUI," Journal of Automation Scripting
14. P. Williams, (2018). "Mathematical Techniques for GUI Automation," Journal of Computer Science and Mathematics,.
15. Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., ... & Grundmann, M. (2019). Mediapipe: A framework for building perception pipelines. arXiv preprint arXiv:1906.08172.
16. Figure (1) - E. Sankar, B. N. Bharadwaj, and A. V. Vignesh, "Virtual Mouse Using Hand Gesture," International Journal of Scientific Research in Engineering and Management (IJSREM), vol. 7, no. 5, p. 1, May 2023.
17. Figure (2) - "Hands, MediaPipe Solutions: Hand Landmarker", Google Developers.
18. Figure (3) - Sánchez-Vicinaiz, T. J., Camacho-Pérez, E., Castillo-Atoche, A. A., Cruz-Fernandez, M., García-Martínez, J. R., & Rodríguez-Reséndiz, J. (2024). MediaPipe frame and convolutional neural networks-based fingerspelling detection in Mexican sign language. Technologies, 12(8), 124."