Multi-model Human-Computer Interaction System with Hand Gesture and Eye Gesture Control

M. jayalakshmi
Department of Computer Science and Engineering.
Kalasalingam Academy of Research and Education,
Anand Nagar, Krishnankoil,
Tamilnadu, India
m.jayalakshmi@klu.ac.in

T. Pardha Saradhi
Department of Computer Science Engineering.
Kalasalingam Academy of Research and Education,
Anand Nagar, Krishnankoil,
Tamilnadu, India
saradhi.tulasi.l23@gmail.com

Syed Mohammed Rahil Azam
Department of Computer Science and Engineering.
Kalasalingam Academy of Research and Education,
Anand Nagar, Krishnankoil,
Tamilnadu, India
syedmohammadrahil@gmail.com

Sk. Fazil
Department of Computer Science and Engineering.
Kalasalingam Academy of Research and Education,
Anand Nagar, Krishnankoil,
Tamilnadu, India
Skfazil221@gmail.com

S. Durga sai sriram
Department of Computer Science and Engineering.
Kalasalingam Academy of Research and Education,
Anand Nagar, Krishnankoil,
Tamilnadu, India
saisriramsuggu@gmail.com

Thottempudi Amrutha
Department of Computer Science and Engineering.
Kalasalingam Academy of Research and Education,
Anand Nagar, Krishnankoil,
Tamilnadu, India
amrutha.thottempudi@gmail.

Abstract— ***This research paper study, a new virtual mouse system that combines hand and eye gestures to produce a non-contact interface for smooth computer interaction is designed and put into practice. Accurate cursor movements on the computer screen are produced by the system through exact detection of hand and face position and orientation. The real-time camera video feed is processed and analyzed using OpenCV, a powerful open-source computer vision toolkit, to identify these movements as dynamic pointer operations. One of the system's key features is its click capabilities, which enables users to give orders with a combination of hand and eye movements. Interestingly, the device incorporates eye-tracking functionality, adding a new level of interactivity to improve user engagement. The device tracks the user's eye movements to control the cursor, providing a hands-free option for those with limited hand mobility. A volume control function that is triggered by hand gestures is introduced by the system to address audio control. Users can effortlessly and successfully control the audio output by dynamically varying the computer's audio volume based on the separation between fingers. The user experience is extensive and engaging due to the unique combination of audio control, eye tracking, and hand gestures.***

Key words – Multi model – Human Computer Interaction system – Hand Gesture Control – Eye Gesture Control – Gesture based volume control.

Introduction

A multimodal system for human-computer interaction that combines hand gestures and eye tracking is presented in this paper. It suggests an updated virtual mouse technology to overcome the drawbacks of conventional computer mice, such as their limited lifespan and sporadic anomalies. This system uses the open-source computer vision library OpenCV to improve user engagement by tracking hand movements in real time, recognizing gestures, and detecting eye movements. The goal of the virtual mouse technology is to give users—especially those with limited mobility—a more natural and user-friendly interface. The system can manipulate the mouse cursor and do computer tasks thanks to OpenCV's tools. By controlling the cursor location through eye movements, the eye-tracking feature provides a hands-free option. Also, users can effortlessly change the computer's audio volume with hand gestures. The interface can be made simpler by using hand gesture-based mouse control, which enables cursor movement without physical contact. The project includes a Tkinter-based Python UI application with four buttons for activities related to mouse control, GUI automation with OpenCV, Mediapipe, and PyAutoGUI, and computer vision.

I. Related Works

Venkat Mukthineni [1] This paper discusses alternative technology and implementation strategies for a gesture detection and recognition system. Additionally covered in the article is how to create a facial recognition module for system authentication using the Viola-Jones algorithm.

Laura Hay [2] This study set out to characterize the properties of touchless/in-air hand motions applied to interaction interfaces. The results of that systematic review are presented in this article. The review's objectives were to map the landscape of gesture-based user interfaces, look into usage trends, and pinpoint typical gesture combinations for various combinations of technology and applications.

Ramanath Nayak [3] In this research, this paper propose a hardware implementation of a virtual mouse that enhances the performance of the current "virtual marker" by making it extremely responsive in real-time. In order to give the existing Virtual Marker access to all mouse functionalities, it is modified to act as both a mouse pointer and a marker.

B. Nagaraj [4] The suggested AI virtual mouse system can overcome this restriction by using a webcam or a built-in camera for computer vision-based hand motion and hand tip identification. The technique relies on deep learning to find the hands. The suggested remedy will stop the spread of COVID-19 because it won't require any human involvement or extra equipment to operate the computer.

Mr. R. Raj Bharath [5] The suggested method offers an alternative to a physically handled virtual keyboard and mouse for those who are paralyzed or have physical limitations by leveraging their face expression through the web camera as the primary input system. The device operates using facial emotions including eye and tongue movement.

S Vickers [6] The article examines gaze-based interaction techniques and draws attention to the issues with gaze control in online virtual environments. The research then introduces a novel software tool called "Snap Clutch" that solves these issues and permits gaze control. An experiment using the tool demonstrates that, despite prolonged work times, effective gaze control is still achievable.

Khushboo Banjarey [7] This paper Human activity recognition (HAR) represents one of the most formidable challenges in computer vision. The primary objective of an intelligent video system is to ascertain the behaviors and pursuits of the person. Human-computer interaction, tracking, security, and health monitoring are just a few of the applications for this activity monitoring system. Even with continuous attempts, it is still difficult to detect activity in an uncircumscribed territory due to a variety of factors. According to the researchers, the vision-based approach has emerged as a popular HAR method.

Sang Min Yoon [8] This paper suggest employing triaxial accelerometer data obtained from users' smartphones to develop a one-dimensional (1D) Convolutional Neural Network (CNN)-based technique for identifying human activity. An accelerometer sensor on a smartphone is used to collect data on three types of human activity: walking, running, and standing motionless. The vector magnitude data obtained from the acceleration data for x, y, and z is utilized as the input to train the 1D CNN. Our 1D CNN-based technique outperformed the baseline random forest strategy with 92.71% accuracy in ternary activity identification.

II. METHODOLOGY

In order to enable hands-free and natural computer interaction through hand gestures and eye motions, this project intends to create a versatile user interface application that makes use of computer vision and GUI automation technologies. The program incorporates the Python libraries PyAutoGUI, Tkinter, OpenCV, and Mediapipe. In this project OpenCV plays a major role among all libraries. Here we created four buttons using Tkinter frame work. These four buttons makes interact with computer for the user. First button represents the volume controlling using hand gestures, second button represents clicking and controlling the mouse, third button represents controlling mouse using eye and enables the click feature when user blink an eye, The fourth and final button captures the image through the webcam and captured image will save as "captured_photo.jpg, saved image is opened and displayed by using a library called pillow. Here this paper explains the working of each and every button:

*A. **Button 1:** Hand Gesture – Based Volume Controlling*

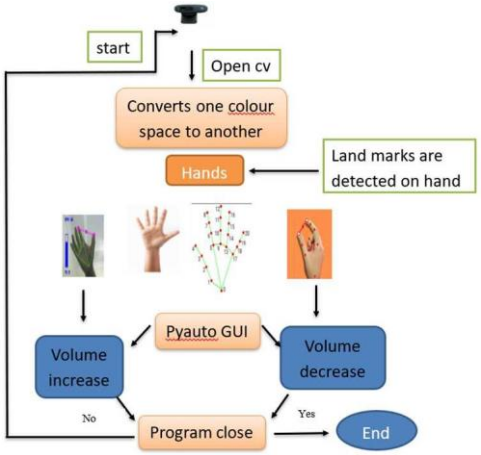


Fig. 1. Flow chart for Gesture-Based Volume controlling

This flow chart for volume control using hand gestures is displayed. The following steps must be done in order to detect hands and change the volume.

The step 1 is to captures instantaneous video through webcam by using open source computer vision (OpenCV). To track hand movements, the collected frames are continuously processed. Step 2 is to detect the hand land marks using the mediapipe library, hand landmarks such as tips of the fingers, base of the fingers and palm. While it detecting the hand it will convert one color space (RGB) to another color space (BGR).

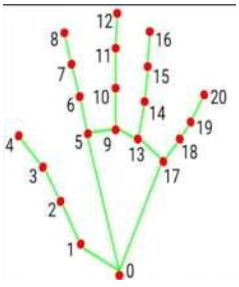


Fig. 2. Hand landmarks

From the above fig 2 We can observe the landmarks of the hand of every person. Step 3 is to calculates the distance between index finger tip (Landmark 8) and the thumb finger tip (Landmark 4). The distance between the two fingers is represented by the measurement in pixels. In final step when it detect the hand gesture for volume control based on calculated distance it increases and decreases volume. The

program continuously changes the system volume based on real-time monitoring of the hand motions.

The distance formula can be used to determine the distance between the hand landmarks returned from OpenCV.

Distance = ((x2-x1)**2+(y2-y1)**2)**(0.5)//4

Where

x1,x2 = frame width
y1,y2 = frame height

By using the above formula we can calculate the distance between the index finger and thumb finger. When the calculated distance is greater than 50 it triggers ‘volume up’ when the calculated distance is less than 50 it triggers ‘volume down’ this controlling will be done by the library called PyAutoGUI.

B. Button 2: *Hand controlled mouse*

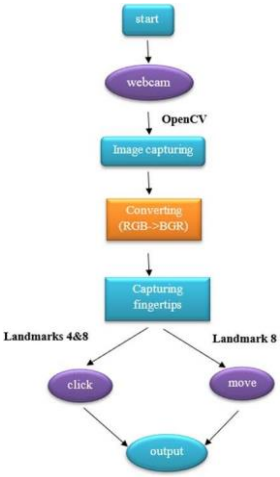


Fig. 3. Flowchart for hand-controlled mouse

The above fig 3 represents the flowchart for hand controlling mouse.

The system combines image processing for hand segmentation and OpenCV to record footage from a live camera. RGB to BGR video color conversion is carried out. The Mediapipe library is used to recognize landmarks, especially those of the index and thumb fingers (Landmarks 4 and 8). When there is less than 50 distance between these landmarks, a click mechanism is triggered. To move the cursor on the computer screen, use your index fingertip (Landmark 8). Clicking to pick items and pointing to move the pointer are examples of recognized gestures that the mouse reacts to. Feedback on the location of the hand and recognized motions is given via a graphical user interface (GUI). User interface design, machine learning, and image processing are all combined in the development of this computer vision-based hand-controlled mouse, with precise hand identification and motion recognition being given special attention.

The following equations must be used in Python programming to generate a virtual mouse:

index_x=screen_width/frame_width*x
index_y=screen_height/frame_height*y

thumb_x=screen_width/frame_width*x
thumb_y=screen_height/frame_height*y

We calculate the distance of both thumb finger and index finger when the distance value is absolutely less than 50 then clicking function will work easily. We can create any fingertips for clicking not only thumb and index. But we mainly make a focus on thumb and index so that every user can use easily when compared to all other fingertips. It will take the hand as a input and produce the output on the screen like virtual mouse. Gesture recognizing and detecting landmarks is a kind of image processing. This virtual mouse plays a major role when mouse was not carried in your bag. Hence, these all actions will be performed by using a library called PyAutoGUI.

C. Button 3: *Eye controlled mouse*

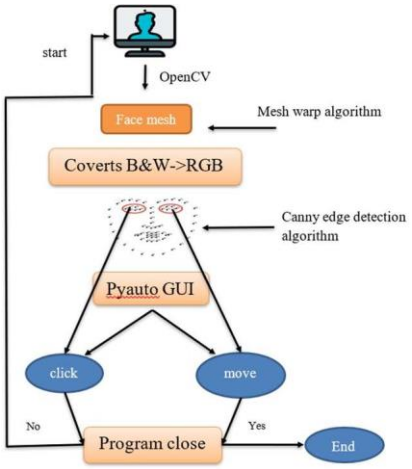


Fig. 4. Flowchart for eye-controlled mouse

The above fig 4 represents the eye tracking mouse using deep neural network.

There are multiple steps involved in developing an eye-tracking mouse control system. The camera takes uniformly preprocessed pictures and records eye movements together with the matching (x, y) screen coordinates. Eye regions are extracted by facial landmark identification, and landmarks are transformed into a fixed-size vector. The dataset is improved by data augmentation, and gaze point regression is achieved through the use of a DNN architecture. With optional capabilities like blink detection for mouse clicks, calibration adheres to hardware and software setups. Observing eye movements is encouraged with an interface that is easy to use. Tracking algorithms are able to continuously identify eye movements. Integration takes place after training with a program that converts the model's output into mouse operations. Users' eyes can be used for actions like scrolling and zooming. Data encryption is a component of security assurance. System efficiency is improved by design experimentation and performance-based fine-tuning, with optimization for several screen sizes and resolutions.

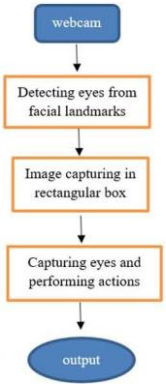


Fig. 5. Eye movement recognition

The above fig 5 represents the eye movement recognition for virtual mouse

To ensure precise eye movement tracking and real-time eye picture capture before using the eye-controlled mouse, first use an infrared-equipped eye tracker. Utilize techniques from computer vision, such as contour recognition, color segmentation, and image processing, to separate the eyes from their surrounds. Examine eye regions for gaze direction, dynamics, and blinking patterns utilizing optical flow and edge detection. Using a labeled dataset, apply machine learning techniques (ANN, k-NN, SVM) to identify and classify gaze directions and eye movements. Teach the system to associate virtual mouse movements with recognized gaze and eye gestures.For more efficient cursor control, integrate the eye tracking system with an API such as DirectInput or Windows API. When the eye difference is smaller than 0.008, click.

D. Button 4: *Capturing image*

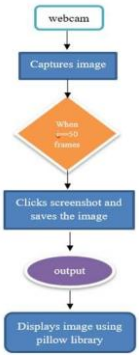


Fig. 6. Flowchart for capturing image

The above fig 6 represents flowchart for taking a screenshot of the user. Set the primary webcam's initial value to "0" by utilizing OpenCV's VideoCapture function. To improve responsiveness, add a 10-millisecond pause between frames. Using OpenCV's imshow(), loop over reading frames and displaying them. 50 frames later, break. Before leaving, make sure the frame read correctly. If so, use OpenCV's imwrite() function to store the frame as "captured_photo.jpg". Use release() to release the webcam, and destroyAllWindows() to end the OpenCV window. PIL's Image can be used to access and show the saved image.both show() and open() in the installed viewer.

III. RESULTS AND DISCUSSION

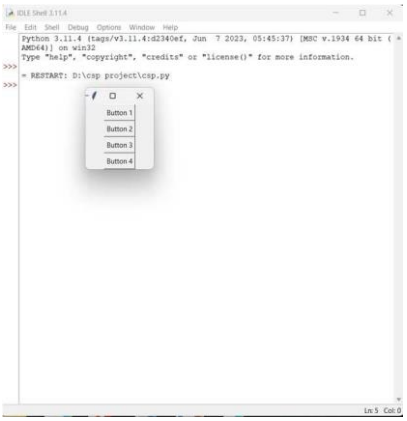


Fig. 7. Buttons

Fig 8 shows the initial set up of the human computer interaction **(HCI).** From the fig 8 we can say that there are 4 buttons

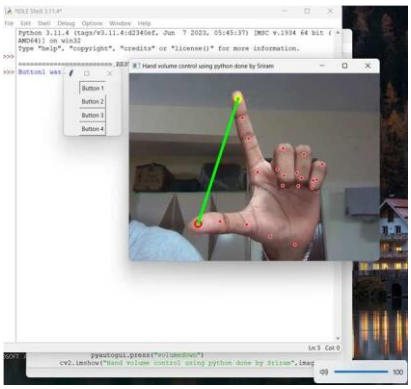


Fig. 8. Increases distance triggers 'volume up'

While the hand is in motion, the distance from the thumb to the index finger is measured.The space between the two fingers is wide. According to what we saw in Fig. 8, when the distance between the thumb and index finger is more than 50, "volume up" is activated.

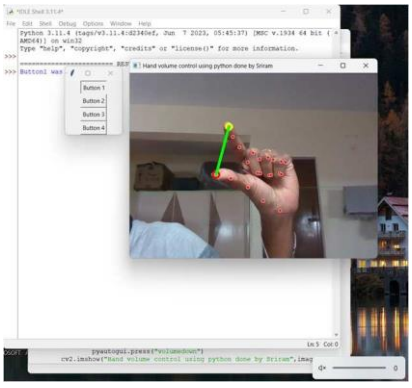


Fig. 9. Decreases distance triggers 'volume down'

From the fig 9 we observed that the distance between the thumb finger and index finger is less than 50 so it triggers 'volume down'.

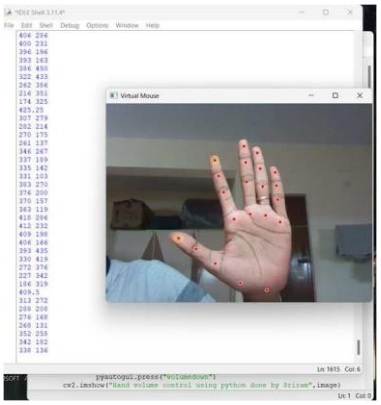


Fig. 10. Sensing the fingertips

By selecting this button 2, the webcam is set up and the video feed is processed to detect hand motions and familiar objects. The hand landmark detection is achieved by the mediapipe library. We can see from fig. 10 that the webcam can recognize the fingertips as landmarks.

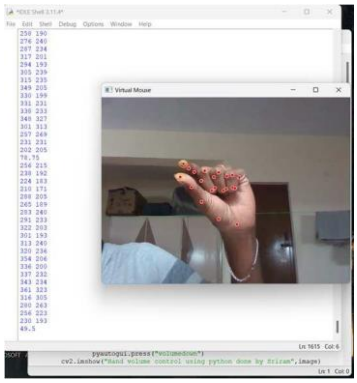


Fig. 11. Cursor performing clicking action

Here landmark 4 represents thumb fingertip and landmark 8 represents index finger. From fig 11 we observed that when thumb fingertip (Landmark 4) and index fingertip (Landmark 8) gets closer to each other (or) when the distance between those both fingertip is less than 50 then we can use click function.

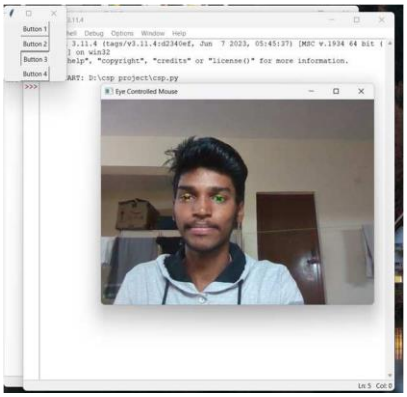


Fig. 12. Landmarks recognition of eyes

As we can see in fig. 12 we observed that when this button is pressed, the webcam tracks the user's eye motions and uses landmarks to direct the mouse pointer. The system moves the cursor based on the position of the eyes, and it uses blink detection to act out mouse clicks. The face's landmarks and the cursor position can be seen on the processed video feed.

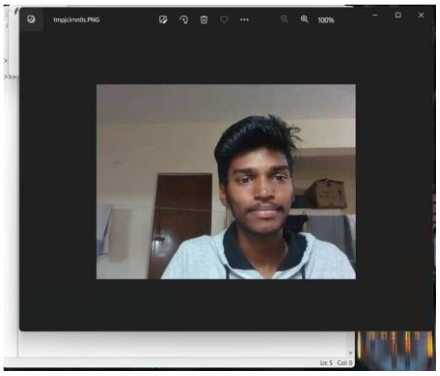


Fig. 13. Capturing image and display in our system

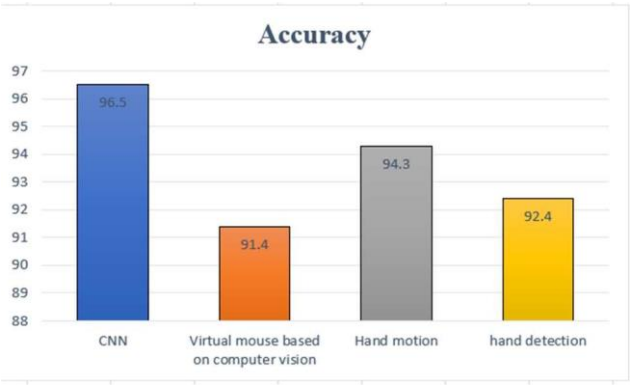
From the fig. 13 we observed that when this button is clicked, a webcam image is taken and saved as "captured_photo.jpg". The captured image is then opened using the PIL library. This feature offers a straightforward method for utilizing the application to take and view images.

Table 1 Testing results of hand-controlled mouse

Methods	Accuracy
CNN	96.5
Virtual mouse based on computer vision	91.18
Hand motion	94.5

The performance of the suggested method compared to the conventional method is shown in Table 1. The recommended approach found the position of the image for the item with an average value of 96.52%.

Accuracy Graph of a propsed system



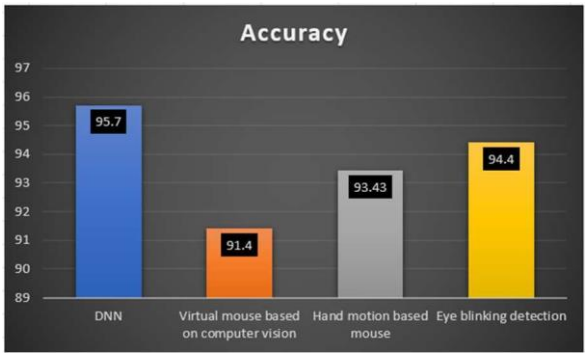
Bar graph. 1 for hand controlled mouse

Table 2 Testing results of eye-controlled mouse

Methods	Accuracy
DNN	95.7
Virtual mouse based on computer vision	91.18
Eye blinking detection	93.2

From the Table 2 we observed that it compares the effectiveness of the suggested approach versus the conventional approach.The suggested method identified the image's placement for the item with an average value of 95.72%.

Accuracy Graph of a propsed system



Bar garph. 2 for eye controlled mouse

IV. CONCLUSION

In conclusion, It presents the concepts of picture capture, hand-controlled, eye-controlled, and gesture volume control. An important development in human-computer interaction is the computer vision-based eye tracking mouse control system, which precisely tracks users' eye movements enabling intuitive, hands-free interaction. The system improves human-computer interaction by integrating hand tracking, eye tracking, and computer vision technologies to create a flexible virtual interface. However, for robustness and dependability, modifications must be made frequently due to factors like illumination and camera quality that might affect responsiveness and accuracy. Using computer vision to create a handheld mouse provides a user-friendly interface that is especially helpful for individuals with limitations or limited movement.

REFERENCES

1. Mukthineni, V., Mukthineni, R., Sharma, O., & Narayanan, S. J. (2020). Face authenticated hand gesture based human computer interaction for desktops. Cybernetics and Information Technologies, 20(4), 74-89

2. Vuletic, T., Duffy, A., Hay, L., McTeague, C., Campbell, G., & Grealy, M. (2019). Systematic literature review of hand gestures used in human computer interaction interfaces. International Journal of Human-Computer Studies, 129, 74-94.

3. Mhetar, A., Sriroop, B. K., Kavya, A. G. S., Nayak, R., Javali, R., & Suma, K. V. (2014, November). Virtual mouse. In International Conference on Circuits, Communication, Control and Computing (pp. 69-72). IEEE.

4. Shriram, S., Nagaraj, B., Jaya, J., Shankar, S., & Ajay, P. (2021). Deep learning-based real-time AI virtual mouse system using computer vision to avoid COVID-19 spread. Journal of healthcare engineering, 2021.

5. Shibly, K. H., Dey, S. K., Islam, M. A., & Showrav, S. I. (2019, May). Design and development of hand gesture based virtual mouse. In 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT) (pp. 1-5). IEEE.

6. Bharath, M. R. R. (2022). Controlling Mouse and Virtual Keyboard using Eye-Tracking by Computer Vision. Journal of Algebraic Statistics, 13(3), 3354-3368.

7. Vickers, S., Istance, H., Hyrskykari, A., Ali, N., & Bates, R. (2008). Keeping an eye on the game: Eye gaze interaction with massively multiplayer online games and virtual communities for motor impaired users.

8 F S. N. Yewale and S. S. Sambare, "Hand gesture recognition based mouse control using OpenCV," in International Journal of Computer Science and Mobile Computing, vol. 7, no. 2, pp. 123-128, February 2018.

9 Song-Mi Lee, Sang Min Yoon and Heeryon Cho, "Human activity recognition from accelerometer data using Convolutional Neural Network," 2017 IEEE International Conference on Big Data and Smart Computing (BigComp), Jeju, Korea (South), 2017, pp. 131-134, doi: 10.1109/BIGCOMP.2017.7881728.

10 Fu, Y., & Huang, T. S. (2007, February). hMouse: Head tracking driven virtual computer mouse. In 2007 IEEE Workshop on Applications of Computer Vision (WACV'07) (pp. 30-30). IEEE.

11 A. R. Vilela and J. L. A. Souza, "Real-time hand gesture recognition for controlling a computer mouse," 2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE), Halifax, NS, Canada, 2015, pp. 1477-1480, doi: 10.1109/CCECE.2015.7129458.