Online-Journals.org

iJIM International Journal of Interactive Mobile Technologies

iJIM | eISSN: 1865-7923 | Vol. 19 No. 2 (2025) | OPEN ACCESS

https://doi.org/10.3991/ijim.v19i02.51879

PAPER

Hand Gesture Recognition for Virtual Mouse Control

Issam El Magrouni[1](✉), Abdelaziz Ettaoufik[2], Siham Aouad[3], Abderrahim Maizate[1]

[1]RITM-ESTC/CED-ENSEM, Hassan II University, Casablanca, Morocco

[2]LTIM, FS ben M'SICK, Hassan II University, Casablanca, Morocco

[3]SSL ENSIAS, Mohamed V University, Rabat, Morocco

issam.elmagrouni-etu@etu.univh2c.ma

ABSTRACT

Our work delved into the complexities of real-time hand motion interpretation and fingertip recognition to simulate the functionality of a traditional mouse. We developed a python-based technique that seamlessly translates hand movements into mouse commands by analyzing the angles between fingers and calculating the ratio of the hand's silhouette to its convex hull. Our methodology was refined to ensure an intuitive and accurate user experience. However, challenges remained in achieving robustness and accuracy in gesture recognition systems in various scenarios, including variations in lighting, hand orientation, and individual human characteristics. These factors had a significant impact on system performance and reliability. To address these challenges, our approach incorporated the algorithms and machine learning models designed to adapt to different conditions. Despite these advances, further research and development were essential to improve the reliability and comprehensiveness of gesture recognition technologies.

KEYWORDS

human-computer interaction (HCI), gesture recognition, virtual mouse, fingertip tracking, hand gesture monitoring, python, webcam interface

1 INTRODUCTION

This study presents a virtual mouse system that uses computer vision to perform mouse activities on a computer with hand motions and fingertip detection. The suggested system can be used to solve various challenges, such as limited space for using a physical mouse or situations where users are unable to operate a traditional mouse due to physical limitations.

By utilizing a built-in or external webcam to detect hand gestures and fingertip movements, this AI-powered virtual mouse eliminates the need for physical contact with devices, enhancing hygiene and reducing the risk of virus transmission. The system relies on computer vision to track fingertip gestures, enabling various mouse functions such as scrolling and cursor movement. Unlike wireless or Bluetooth mice, which require dongles and batteries, this solution operates through a webcam or built-in camera to recognize hand gestures.

El Magrouni, I., Ettaoufik, A., Aouad, S., Maizate, A. (2025). Hand Gesture Recognition for Virtual Mouse Control. *International Journal of Interactive Mobile Technologies (iJIM)*, 19(2), pp. 53–64. https://doi.org/10.3991/ijim.v19i02.51879

Article submitted 2024-08-30. Revision uploaded 2024-10-12. Final acceptance 2024-10-15.

The webcam captures images, processes them, and interprets different hand and fingertip gestures to execute specific mouse commands. The virtual mouse system is implemented using Python and the OpenCV (open-source computer vision library).

This paper has two primary goals: first, to create a virtual mouse that offers a seamless and intuitive user experience without physical touch, and second, to promote a cleaner, more hygienic interaction with computers, minimizing the need for physical device handling.

We begin by reviewing previous studies, comparing their methodologies to ours to demonstrate how our virtual mouse represents a significant advancement in enhancing computer usability (Section 2). Next, we detail the development of our virtual mouse, providing an in-depth explanation of the technical aspects involved (Section 3). Following this, we present the results of our testing, illustrating the effectiveness and reliability of our system (Section 4). Finally, we discuss the implications of our study and outline potential future directions, focusing on ongoing improvements to enhance human-computer interaction (Section 5).

2 RELATED WORKS

In the rapidly growing field of human-computer interaction (HCI), the idea of controlling virtual mouse systems through hand gestures has gained significant traction. Zhang et al. [3] pioneered the use of convolutional neural networks (CNNs) for gesture recognition, demonstrating the power of deep learning by achieving instantaneous and accurate gesture interpretation. Their work outperformed traditional methods, setting a high standard for gesture-based control. On the hardware side, Biswas and Basu [4] integrated depth-sensing capabilities with Microsoft's Kinect, merging skeletal tracking and gesture recognition to craft a more intuitive user interface, making gesture control feel natural.

Liu et al. [5] took a more accessible approach, utilizing common webcams to offer an affordable hand-tracking solution. By blending background subtraction with skin tone detection, they created a practical virtual mouse system that emphasized both cost-effectiveness and accessibility. Similarly, Y. Zhang et al. [6] explored wearable technologies, implementing sensors to detect hand movements in detail. Their method, while highly accurate due to machine learning, required additional equipment, which posed challenges in terms of user convenience. In contrast, Y. Li et al. [7] revisited dynamic time warping (DTW), traditionally used in voice recognition, to interpret hand gestures with precision. This showed that classical algorithms still hold relevance in modern digital interaction. Jo et al. [8] took the next leap by blending gesture recognition with augmented reality (AR). Their work enabled hands-free virtual mouse control via AR eyewear, opening up new possibilities by merging technologies for more immersive and seamless user experiences.

Shibly et al. [9] proposed a practical solution, developing a virtual mouse system that relies on hand gestures captured through webcams and processed with OpenCV. While it showcased potential in replacing physical devices, the system's performance was sensitive to environmental lighting, indicating areas for improvement. Tran et al. [10] enhanced the experience by using RGB-D images and fingertip detection, proposing a more natural way for users to interact with computers through gestures, thanks to machine learning techniques. Finally, Haria et al. [11] explored gesture recognition in human-computer interaction by comparing multiple machine learning algorithms such as support vector machines (SVM), random forest, and neural networks.

Their comparative analysis highlighted the need for efficient feature extraction and adaptive recognition techniques, offering pathways for future innovation in gesture-based control (see Table 1).

Table 1. Comparative analysis of related work on virtual mouse control through hand gesture recognition

Related Work	Methods	Accuracy	Real Time
Zhang et al.	CNNs	High	Excellent
Biswas & Basu	Depth Cameras	Moderate-High	Excellent
Liu et al.	Webcams	Moderate	Good
Y. Zhang et al.	DTW	High	Excellent
Y. Li et al.	hand-type algo	High	Excellent
Jo et al.	AR Glasses	Moderate-High	Excellent
Shibly et al.	color segmentation	78–90%	Good
Tran et al.	RGB-D	Real-time	Excellent
Haria et al.	SVM, Random Forest, Neural Networks)	Improvement in accuracy	Excellent

When comparing the various methods for virtual mouse control through hand gesture recognition, we can extract three main criteria: accuracy and real-time response.

Our proposed approach synthesizes the strengths of the existing methods while considering their limitations. We aim to achieve high accuracy and real-time interaction, but without the need for extra hardware, aligning with the simplicity of webcam-based systems. By doing so, we strive to create a virtual mouse system that is both effective and accessible, reducing the barriers to entry for users and addressing the common challenges faced by previous works.

3 PROPOSED APPROACH

The proposed gesture recognition process is divided into five key stages, each of which is designed to perform specific tasks essential to the accurate recognition and interpretation of gestures (see Figure 1).

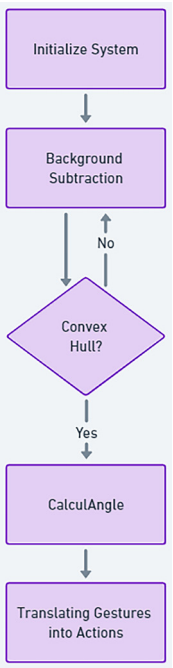

Fig. 1. Virtual mouse

3.1 The gesture recognition process

Initialize system: The first step in getting the gesture recognition system up and running is to set up the necessary hardware to capture the live video feed. This is done using a powerful computer vision tool called the OpenCV library. The main task is to start the video capture process, which involves taking pictures with the webcam.

Video capture setup: The webcam is connected to the system using the video capture function from the OpenCV package. This function creates a video capture object that acts as a bridge between the camera and the system, allowing real-time video data to be streamed.

Typically, the webcam is either built into a laptop or connected externally to a PC. This connection enables continuous video capture, which is essential for subsequent processing.

Initialization process: OpenCV is used to construct a video capture object pointing to the default camera device (usually device index 0). This is crucial as it allows the system to start video capture directly from the camera.

This initialization stage is fundamental to effective and reliable gesture recognition. It ensures the smooth acquisition of video frames and prepares the system for immediate analysis and response. This setup provides the essential data required for accurate gesture identification and understanding.

Background subtraction: Our retrieved frame images often contain more than one object. We use OpenCV's background subtraction to separate the foreground objects, which allows us to ignore hand shadows by adjusting the detect shadows parameter. After applying the background subtraction, the resulting output is a masked greyscale image that highlights the foreground objects, removes the background clutter, and focuses on the area of interest.

Convex hull: Finger identification is achieved by smoothing the images using the bilateral filter function, followed by background removal from the images. Next, each image is analyzed using the contour function to identify the shape of the palm, and the convex hull function is used to construct the convex hull. The contour function compares the color of the palm to a specified color range. This comparison determines the approximate boundary of the palm, which is then drawn to illustrate the shape and contour of the palm. The convex envelope is created by ensuring that every point of the contour is contained within the envelope, except for the outermost points.

By using the convex envelope and the palm contour areas, different hand movements can be detected. In addition, the number of peaks in the contour corresponds to the number of fingers extended, indicating how many fingers are freely extended. Figure 2a shows the HSV (hue, saturation, value) image derived from the original image. Figure 2b shows the image used for finger detection, where the convex hull technique produces red lines outlining the structure of the hand and green lines representing the contours along the palm.

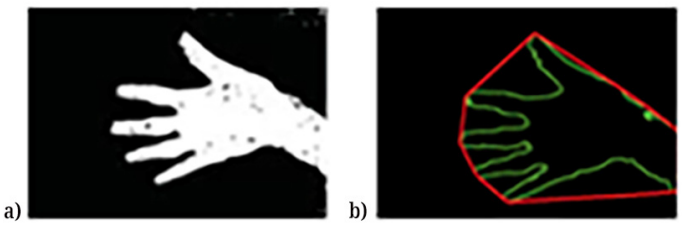

Fig. 2. a) Image of the skin mask; b) Contour and body formed

The area of the hand (palm) is represented by 'areacnt.' The area of the hull formed by the palm is represented by 'areahull.' The ratio of the areas is calculated as follows:

$$\text{arearatio} = ((\text{areahull} - \text{areacnt}) / \text{areacnt}) * 100$$

This ratio indicates the part of the convex envelope that is not occupied by the hand. Therefore, as shown in Table 2, gestures can be recognized by counting fingers and measuring the area ratio. The area ratio gives the name of the finger used.

Table 2. Gesture corresponding to area ratio

Area Ratio	Gesture
Hand area < 2000	No hand
a <= 12	Fist
12 < a <= 18	Thumb
a > 18	Index finger
a > 27	OK sign

Calculate angle: The calculate angle function is designed to measure the angle formed between any two fingers in a hand gesture. This angle provides valuable insight into the relative positions and movements of the fingers, allowing the system to discern gestures involving multiple fingers or hand positions. By analyzing these angles, the system can identify complex gestures and infer the user's intent with greater accuracy.

Let's denote two vectors representing the positions of the fingers involved in the gesture: $\overrightarrow{v_1}$ for the first finger and $\overrightarrow{v_2}$ for the second finger. The angle θ between these two vectors can be calculated using the dot product formula:

$$\theta = \arccos(v1.v2/|v1||v2|) \quad (1)$$

where:

v1.v2 represents the dot product of the two vectors.

|v1| and |v2| represent the magnitudes (lengths) of the vectors.

This formula yields the angle θ in radians between the two fingers.

To convert this angle to degrees, multiply by $180/\pi$.

Consider a scenario where a user brings their thumb and index finger close together to perform a 'pinch' gesture for zooming.

Let v1 represent the vector from the base of the thumb to the tip and let v2 represent the vector from the base of the index finger to the tip.

The "CalculateAngle" function calculates the angle θ between these two vectors. If the fingers are close together in a pinching motion, the resulting angle θ will be small, indicating an acute angle between the fingers.

Translating gestures into actions: Advanced decoding algorithms are essential in gesture recognition technology as they interpret data from the convex hull, which includes hand and finger positions. This technology is key to accurately identifying user gestures. The process begins by analyzing raw data, focusing on the convex hull—essentially the smallest convex shape that encompasses all hand movements. Using pattern recognition techniques, the algorithms are able to identify distinctive patterns within this data, which are then matched against an extensive library of

pre-defined gestures. Each gesture in the library is associated with specific commands, facilitating seamless user-computer interaction.

Here's how it translates certain hand gestures into mouse operations (see Figure 3).

Mouse control with fingertip and hand gesture recognition:

Cursor movement: Cursor movement is controlled when either the middle finger (Tip ID = 2) or index finger (Tip ID = 1) is raised, simulating the movement of a physical mouse.

- **Left click:** Left click occurs when the thumb (Tip ID = 0) and index finger (Tip ID = 1) are both raised and within 30 pixels of each other, mimicking the left mouse button.
- **Right click:** The right mouse button is clicked when the index finger (Tip ID = 1) and middle finger (Tip ID = 2) are both raised and less than 40 pixels apart, similar to a standard mouse right click.
- **Scroll up:** Scroll Up is enabled when both the index finger (Tip ID = 1) and middle finger (Tip ID = 2) are raised with a separation of more than 40 pixels and moved upwards in a controlled manner, replicating the movement of a mouse scroll wheel.

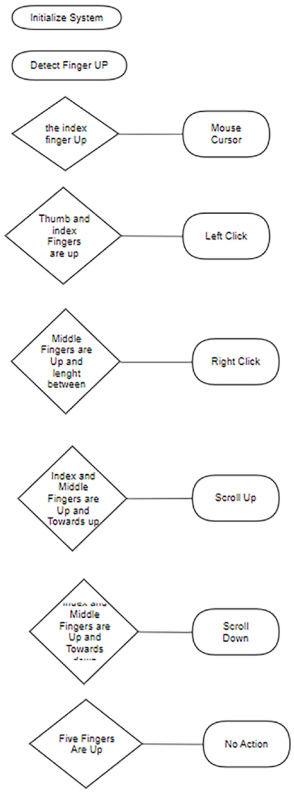

Fig. 3. Mouse virtual flowchart

3.2 Challenges in gesture recognition

- **Lighting conditions:** Lighting greatly influences the accuracy of background subtraction and hand detection. Bright or dim environments can create shadows or overexposed areas, causing the system to misidentify hand contours. To mitigate this, the system employs adaptive thresholding and shadow detection

to adjust for varying light levels. Background subtraction algorithms are enhanced with shadow removal techniques using OpenCV's detect shadows parameter.

- **Hand orientation:** Different hand orientations (rotations or tilts relative to the camera) can distort gesture recognition, leading to inaccuracies in finger detection and interpretation of gestures. The system applies **convex hull** algorithms to track contours in 3D space, combined with angle calculations to accurately detect hand movements across various orientations. Using the CalculateAngle function between finger vectors ensures consistent recognition even when the hand is slightly rotated.
- **Individual characteristics:** Hand size, finger length, and skin tone differences can cause variations in recognition accuracy. Larger hands may result in gesture misclassification, while skin tone may affect color-based detection methods. By using the **convex hull** to focus on shape-based recognition instead of color or size, the system reduces dependence on individual characteristics. The system adapts to user profiles, allowing personalized calibration based on hand size and shape. Additionally, the **area ratio** calculation helps detect gestures independent of skin tone.

4 RESULTS AND DISCUSSION

The model recognition gesture-based artificial intelligence (MoRGIA) project, developed using MediaPipe and OpenCV, aims to create and deploy a recognition system centered around virtual mouse interaction, providing a prime example of our methodology. MediaPipe is an open-source framework by Google Research that provides customizable pipelines for building machine learning models, particularly for multimedia tasks such as gesture recognition. OpenCV is another open-source library designed for computer vision and machine learning tasks, offering a wide range of tools and functions for image processing and analysis. Within the context of our virtual mouse application, this section presents a thorough case study encompassing two pivotal stages: "Translating Gestures into Actions" and "Finger Gesture Recognition." Testing has been carried out on hand gestures and fingertip detection under various lighting conditions and at different distances from the webcam to ensure precise monitoring of hand gestures and fingertip detection.

a) Case study

Step-by-step: Left-click gesture (see Figure 4) with angle calculation

- Gesture initiation: The system tracks the index finger and thumb, which must be extended, while the other fingers are curled inward.
- Vector representation: Two vectors are created: v1 from the base to the tip of the thumb and v2 from the base to the tip of the index finger.
- Angle calculation: Using the formula (1) of the function Angle Calculation
- The angle θ between the thumb and index finger is calculated.
- Angle and distance threshold: If the calculated angle θ is between 20° and 40° and the distance between the fingers is less than 40 pixels (or user-defined), the gesture is considered valid.
- Left-click trigger: Once the system verifies both conditions, the left-click action is triggered, enabling interaction with applications.
- Feedback: Visual and auditory cues confirm the successful execution of the left-click action.

The typical range for θ (between 20° and 40°) ensures reliable differentiation between gestures, while the pixel threshold ensures precise gesture recognition.

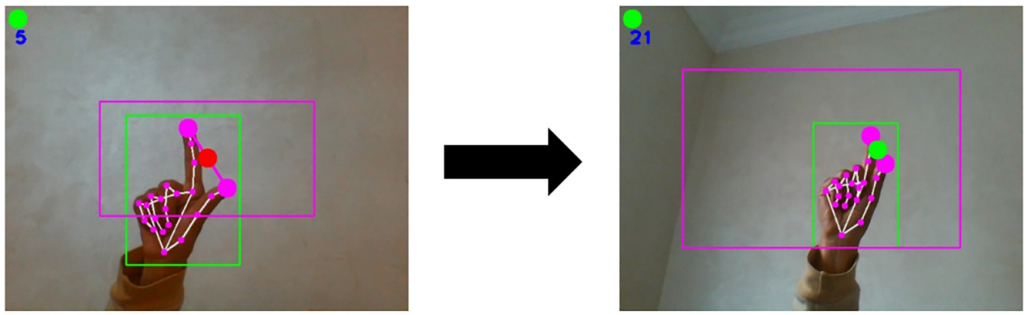

Fig. 4. Gesture for left click action

Master toggle gesture: Figure 5 illustrates the "Master Toggle" gesture, identified when all fingers are fully extended. This intuitive control gesture is pivotal for user interaction, acting as a switch to enable or disable the system's gesture recognition capability. When initially detected, the system activates and begins to process subsequent gestures. A subsequent recognition of the "Master Toggle" gesture pauses the detection process, maintaining this state until the gesture is performed once more.

To augment the user experience, the system offers instantaneous feedback upon recognizing a gesture. This is achieved through a visual indicator displayed on the screen, accompanied by a unique sound or vibration. This feedback mechanism ensures that users are consistently informed about the system's operational status—whether active or inactive—and empowers them to manage the gesture recognition functionality with confidence.

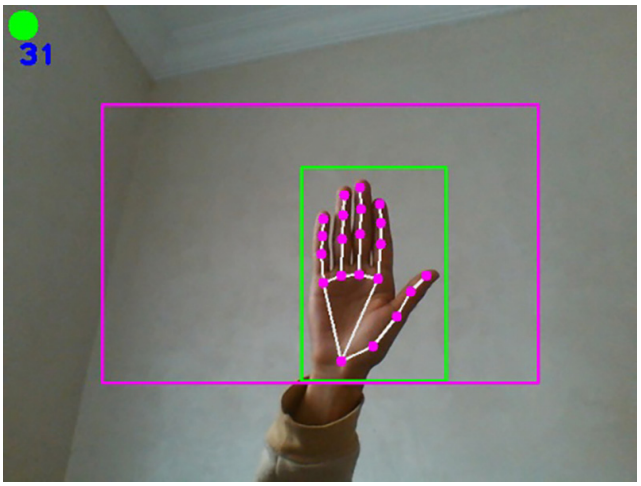

Fig. 5. Gesture for starting or stopping the program

Cursor movement control: As depicted in Figure 6, the cursor control gesture is initiated when the user raises their index finger while the other fingers are folded into the palm. Upon this gesture's recognition, the program precisely maps the hand's position to the cursor location, enabling the cursor to glide fluidly across the screen in harmony with the user's hand motion. The system employs advanced tracking algorithms to ensure responsive and accurate cursor movements, minimizing latency and providing an intuitive experience akin to a physical mouse. To accommodate different user preferences, our interface includes settings for adjusting cursor

sensitivity and movement dynamics, as well as a feedback mechanism that confirms the activation of cursor control mode.

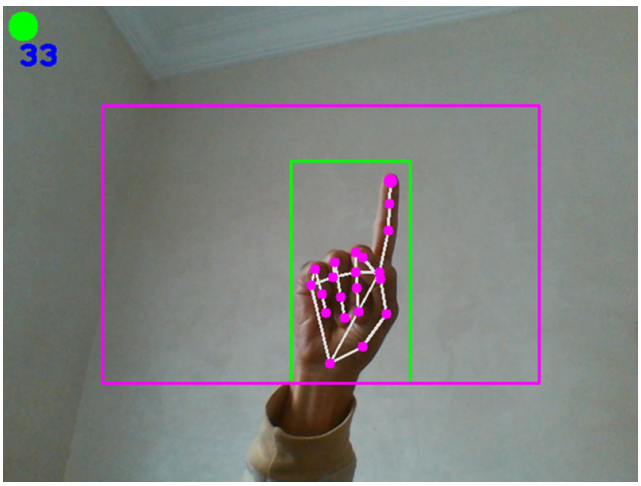

Fig. 6. Gesture for cursor movement

b) Evaluation

A comprehensive evaluation was conducted to assess the performance of the proposed virtual mouse system, focusing on key metrics such as frame rate, latency, and user responsiveness. The system consistently maintained a 30 FPS frame rate for smooth real-time tracking and an average latency of 45 milliseconds, ensuring a responsive user experience.

The experiment involved 600 distinct gestures, manually labeled and tested under varying conditions, such as different lighting levels (normal and low light) and distances from the webcam. This dataset was generated with the participation of four individuals, each performing a series of tests designed to reflect real-world conditions. These tests included the use of the virtual mouse system under varying lighting conditions (normal and low light) and from different distances relative to the webcam (close and far). The results of these tests, capturing the system's performance across multiple scenarios, are summarized in Table 3, providing a detailed overview of the system's accuracy and reliability.

Note: Fingertip IDs correspond to the following fingers: 0 – thumb, 1 – index, 2 – middle, 3 – ring, 4 – little.

Table 3. Summary of gesture recognition experimental results

Gesture	Success	Fail	Accuracy (%)	Response Time (ms)
Mouse Movement	97	3	97.0	25
Left Click	94	6	94.0	30
Right Click	90	10	90.0	35
Scroll Up	94	6	94.0	40
Scroll Down	93	7	93.0	40
Overall	563	37	93.833	34

Table 4 below shows the accuracy and response times for each gesture in graph form, illustrating the system's consistent performance across different actions.

- Mouse movement achieved the highest accuracy at 97%, with minimal latency (25 ms response time), making it ideal for real-time interaction.
- Left click and right click gestures, which involve more complex finger positioning, had slightly lower accuracies but still performed well with 94% and 90% accuracy, respectively.
- The system performed equally well in scroll up and scroll down actions, with accuracy rates of 94% and 93%, respectively.

A confusion matrix and detailed error analysis offer critical insights into where gesture misinterpretations occurred. By analyzing common misclassifications, such as Right Click often being confused with Scroll Down, we can better understand the system's limitations. This helps pinpoint specific gestures that require improvement in recognition.

Table 4. Confusion matrix for gesture recognition

Actual Gesture	Mouse Movement	Left Click	Right Click	Scroll Up	Scroll Down	No Action
Mouse Movement	97	1	1	0	0	1
Left Click	0	94	2	1	1	2
Right Click	1	2	90	3	1	3
Scroll Up	0	0	1	94	2	3
Scroll Down	0	1	2	2	93	2
No Action	0	0	1	0	1	95

In this table, each row represents the actual gesture, and each column represents the gesture as classified by the system. This analysis highlights that the Right Click was correctly identified 90 times but was confused with Scroll Up and Scroll Down in several cases. This helps pinpoint areas where the system struggled, allowing for focused improvements in these areas.

In addition to the technical evaluation, feedback was collected from four participants, who rated the system based on ease of use, responsiveness, and overall satisfaction. The feedback helped assess the user experience, offering insights into potential areas of improvement. The average rating was **4.6/5**, indicating high user satisfaction, with particular praise for responsiveness during cursor control and gesture detection (see Table 5).

Table 5. User feedback on system responsiveness

Participant	Ease of Use (1–5)	Responsiveness (1–5)	Overall Satisfaction (1–5)
1	5	4.5	4.7
2	4.5	4.7	4.6
3	4.7	4.8	4.7
4	4.6	4.6	4.5
Average	4.7	4.65	4.625

When compared to other systems, as detailed in Table 6, the proposed system demonstrated competitive accuracy, closely aligning with models such as the

virtual mouse system using RGB-D images (**96.13%** accuracy), while significantly outperforming others based on palm and finger recognition (78% accuracy).

Table 6. Accuracy comparison of various virtual mouse systems

System	Accuracy (%)	Response Time (ms)
Proposed System (Ours)	93.83	34
Virtual Mouse using RGB-D Images [10]	96.13	45
Hand Gesture-Based Virtual Mouse [9]	78.0	50
Palm and Finger Recognition System [11]	78.0	55

Key insights from the comparison:

- Our system achieved an accuracy of 93.83%, closely matching the more hardware-intensive RGB-D image-based system, which used specialized depth cameras but had a slightly higher accuracy (96.13%).
- Compared to systems using basic webcams, such as hand gesture-based virtual mouse and palm and finger recognition, which achieved only 78% accuracy, our system significantly outperformed them with its superior accuracy and faster response time.
- The response time of our system (34 ms) was faster than all other systems, making it highly responsive in real-time applications.

5 CONCLUSION

In this study, we explored how to detect fingertips and interpret hand gestures in real time to mimic the functions of a traditional mouse. Using Python, we developed a system that converts hand movements into mouse commands by calculating the ratio of the hand's outline to its convex shape and measuring the angles between fingers. This method allows for smooth and accurate mouse operations, providing users with an easy and responsive experience.

Our approach is designed to be user-friendly, offering quick and precise feedback that makes interacting with the computer feel natural. However, we still face challenges in making the gesture recognition system reliable and accurate in different lighting conditions, hand positions, and for various users. These factors can affect how well the system works.

To improve the technology, future work should focus on enhancing the system's ability to handle different environments and user variations. By developing more adaptable algorithms and expanding the range of recognized gestures, we can make gesture-based mouse control more dependable and accessible for everyone. Ultimately, overcoming these challenges will help integrate gesture recognition smoothly into everyday computer use, making interactions more intuitive and efficient.

6 REFERENCES

[1] J. Katona, "A review of human–computer interaction and virtual reality research fields in cognitive InfoCommunications," *Applied Sciences*, vol. 11, no. 6, p. 2646, 2021. https://doi.org/10.3390/app11062646

[2] S. Shriram, B. Nagaraj, J. Jaya, S. Shankar, and P. Ajay, "Deep learning-based real-time AI virtual mouse system using computer vision to avoid COVID-19 spread," *Journal of Healthcare Engineering*, vol. 2021, pp. 1–8, 2021. https://doi.org/10.1155/2021/8133076

[3] L. Zhang, Q. Tian, Q. Ruan, and Z. Shi, "A simple and effective static gesture recognition method based on attention mechanism," *Journal of Visual Communication and Image Representation*, vol. 92, p. 103783, 2023. https://doi.org/10.1016/j.jvcir.2023.103783

[4] K. K. Biswas and S. K. Basu, "Gesture recognition using Microsoft Kinect®," in *The 5th International Conference on Automation, Robotics and Applications,* Wellington, New Zealand, 2011, pp. 100–103. https://doi.org/10.1109/ICARA.2011.6144864

[5] Y. Liu, M. Y. Chiu, H. L. Li, K. H. Wu, Z. Lei, and Q. Ding, "A real time robust hand tracking method with normal cameras," in *Computer Vision. CCCV 2015. Communications in Computer and Information Science*, H. Zha, X. Chen, L. Wang, and Q. Miao, Eds., Springer, Berlin, Heidelberg, vol. 546, 2015. https://doi.org/10.1007/978-3-662-48558-3_6

[6] Y. Zhang, B. Liu, and Z. Liu, "Recognizing hand gestures with pressure-sensor-based motion sensing," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 6, pp. 1425–1436, 2019. https://doi.org/10.1109/TBCAS.2019.2940030

[7] Y. Li *et al.*, "A dynamic hand gesture recognition model based on the improved dynamic time warping algorithm," in *2019 25th International Conference on Automation and Computing (ICAC)*, Lancaster, UK, 2019, pp. 1–6. https://doi.org/10.23919/IConAC.2019.8895002

[8] B. J. Jo, S.-K. Kim, and S. Kim, "Enhancing virtual and augmented reality interactions with a MediaPipe-based hand gesture recognition user interface," *Ingénierie des Systèmes d'Information*, vol. 28, no. 3, pp. 633–638, 2023. https://doi.org/10.18280/isi.280311

[9] K. H. Shibly, S. Kumar Dey, M. A. Islam, and S. Iftekhar Showrav, "Design and development of hand gesture based virtual mouse," in *Proceedings of the 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, 2019, pp. 1–5. https://doi.org/10.1109/ICASERT.2019.8934612

[10] D.-S. Tran, N.-H. Ho, H.-J. Yang, S.-H. Kim, and G. S. Lee, "Real-time virtual mouse system using RGB-D images and fingertip detection," *Multimedia Tools and Applications Multimedia Tools and Applications*, vol. 80, no. 7, pp. 10473–10490, 2021. https://doi.org/10.1007/s11042-020-10156-5

[11] A. Haria, A. Subramanian, N. Asokkumar, S. Poddar, and J. S. Nayak, "Hand gesture recognition for human computer interaction," *Procedia Computer Science*, vol. 115, pp. 367–374, 2017. https://doi.org/10.1016/j.procs.2017.09.092

7 AUTHORS

Issam El Magrouni is with the RITM-ESTC/CED-ENSEM, Hassan II University, Casablanca, Morocco (E-mail: issam.elmagrouni-etu@etu.univh2c.ma).

Abdelaziz Ettaoufik is with the LTIM, FS ben M'SICK, Hassan II University, Casablanca, Morocco.

Siham Aouad is with the SSL ENSIAS, Mohamed V University, Rabat, Morocco.

Abderrahim Maizate is with the RITM-ESTC/CED-ENSEM, Hassan II University, Casablanca, Morocco.