AI-Powered Browser-Based Gesture Control: A Touchless Interface for Accessibility and Public Use

Rohith Deshamshetti, Zelvin Elsafan Harefa, Dhanyaa K S Alok Kumar Singh

Abstract

The research explores the development of a browser-based, AI-powered hand gesture recognition system that functions as a touchless interface for human-computer interaction. The motivation for this project comes from two significant global needs: the demand for inclusive technologies that allow differently-abled individuals to access digital systems, and the growing awareness of hygiene and public safety in shared environments following the COVID-19 pandemic. By combining OpenCV for video processing, MediaPipe for robust hand landmark detection, and PyAutoGUI for simulating mouse actions, we created a prototype that accurately interprets gestures such as cursor movement, clicking, double clicking, and scrolling. Unlike specialised hardware solutions such as Kinect or Leap Motion, our system requires only a standard webcam, ensuring low implementation cost and wide accessibility. Through a series of design, programming, and testing stages, the project demonstrates that gesture recognition can be performed in real time with minimal latency, while offering the flexibility to extend into domains like education, healthcare, and public kiosks. The findings suggest that this approach has the potential to transform human–computer interaction by making it more natural, hygienic, and universally accessible.

Keywords: accuracy; touchless interactions; gesture recognition; Kinect; Leap Motion; human motion tracking

1. Introduction

Touchless interaction is becoming a popular way to interact with devices, with no need for physical contact[1]. This is especially true in public spaces, where touchless interaction can be faster than using a mouse or a keyboard, also due to the frequent use of large displays(Ardito, Buono, Costabile, & Desolda, 2015)[5]. Web browsing is

one of the most common tasks that users perform on their devices. However, touchless web browsing can be
challenging, as it can be difficult to move a cursor or carry out other tasks without traditional input devices.[2]
The idea of controlling computers without physical touch has been around for years, but it gained new urgency during the COVID-19 pandemic, when hygiene and safety became global priorities. Shared devices such as kiosks, ATMş, and public information screens became potential points of disease transmission[7]. This context pushed researchers and developers to think more seriously about touchless technology as both a safety measure and a step toward more natural ways of interacting with machines.

Another important motivation for touchless interaction is accessibility. Traditional input devices like keyboards and mice can be barriers for differently-abled individuals. Gesture-based control offers an alternative that is often more intuitive and inclusive. For example, someone with limited mobility might find it easier to point or move their hand in front of a camera than to press keys or grip a mouse. Thus, gesture recognition not only enhances convenience but also makes computing more universal.

Research in Human–Computer Interaction (HCI) has shown that gestures can serve as a “natural language” between humans and computers. Early systems such as Microsoft Kinect and Leap Motion proved this idea, but were dependent on costly, specialised hardware. This limited their real-world applications[4]. Several researchers have attempted to overcome these limitations by creating low-cost gesture-based virtual mice using webcams, such as the system designed by Shibly et al. (2019), which demonstrated the feasibility of real-time hand tracking for cursor control.[] With the rise of computer vision tools like OpenCV and Google’s MediaPipe, it is now possible to achieve accurate hand tracking using nothing more than a standard webcam. This represents a turning point, making touchless technology affordable and widely accessible.

Studies such as Pasquale et al. (2024) confirm that certain gestures are more intuitive and widely understood. Their work with 56 designers and 248 users showed that the index finger is the most natural way to control a cursor, pinch gestures work well for clicks, and open palm movements are preferred for scrolling. Our project builds on this knowledge, deliberately choosing gestures that are both practical and supported by user research[3].

In this paper, we present our browser-based gesture control system. The project was divided into four stages:

1. researching gesture-based interaction and ideating use cases,
2. designing the system's architecture and UI
3. Programming the recognition system using OpenCV, MediaPipe, and PyAutoGUI,
4. Presenting the outputs with the ethical and social reflections.

Through this process, we aimed to demonstrate that touchless interfaces are not just futuristic experiments, but real and usable systems that can be deployed in schools, hospitals, and other public spaces.

2. Related Work On This Theory

Gesture recognition has long been a focus of Human–Computer Interaction (HCI) research[4]. Early commercial systems such as the Kinect and Leap Motion Controller (LMC) demonstrated the effectiveness of mid-air gestures but had limitations in cost and accessibility[4]. The Kinect is designed for full-body motion tracking by using an infrared emitter (v1) to project a dot pattern and calculate distances. At the same time, the Kinect v2 works by Time-of-Flight (ToF) technology to measure distances based on the speed of light. Microsoft's SDK provided a "skeleton stream," enabling real-time tracking of user joints. The Leap Motion Controller, focused on hand tracking, uses two cameras and three infrared LEDs to generate stereo infrared images, from which hand skeleton data is calculated via its API[4].

Before systems proved the concept, their hardware requirements were limited to widespread use. Today, webcam-based solutions such as OpenCV and MediaPipe enable accurate, real-time hand tracking without specialized devices[3]. Our project works with this approach, selecting gestures that balance the technical Suitability and user comfort, ensuring a system that is functional, intuitive, and broadly usable.

3. Our Approach

Our approach to the Virtual Hand Gestures project is centered on a visual framework that enables us to give accurate and real-time gesture recognition. A conventional webcam, combined with robust libraries such as OpenCV and MediaPipe, will be utilized to detect and track the hand using 21 landmark points. The proposed methodology consists of three stages:

(i) hand detection and landmark extraction

(ii) feature analysis and gesture classification through a resource-friendly machine learning model

(iii) mapping the recognized gestures to meaningful actions for interaction with digital systems.

This methodology eliminates the dependency on specialized hardware, maintains low implementation costs, and offers the flexibility to extend the system towards advanced domains such as virtual/augmented reality and human–robot interaction.

Our Gesture Set is when the index finger is up, it is pointing, and the cursor can be moved, and when the middle finger's tip touches the index finger's tip, it is taken as a single click, and when the index finger tip and middle finger's tip, and ring finger's tip are up, it is taken as the double click. Our performance goals are to run the program at 30+ FPS with low latency, ensuring smooth and responsive interaction suitable for real-time applications. evaluates false clicks and accuracy. To find the accuracy, the target will shuffle randomly after every 5 seconds and calculate the average response. Accuracy is measured by this formula

$$Accuracy = \frac{Total\ Number\ of\ Predictions}{Number\ of\ Correct\ Predictions} \times 100$$

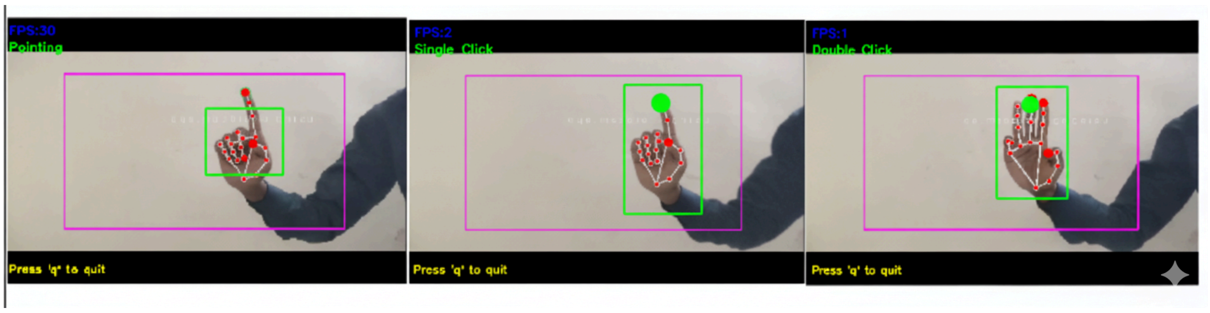
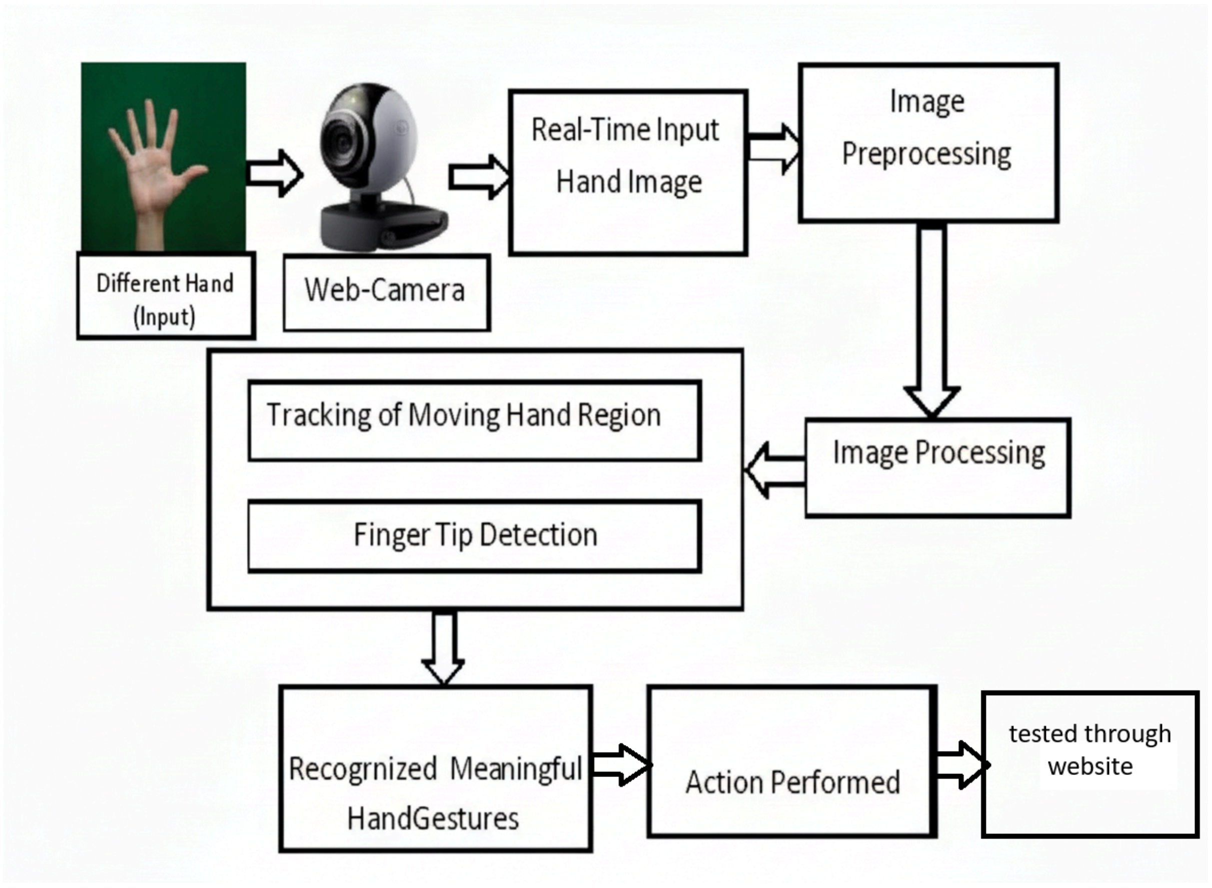


Figure 1: Hand gestures that are used for the testing



Flowchart 1: System Architecture of Hand Gesture Recognition. This figure explains how the real-time input is converted into recognized, meaningful hand gestures

4. The study

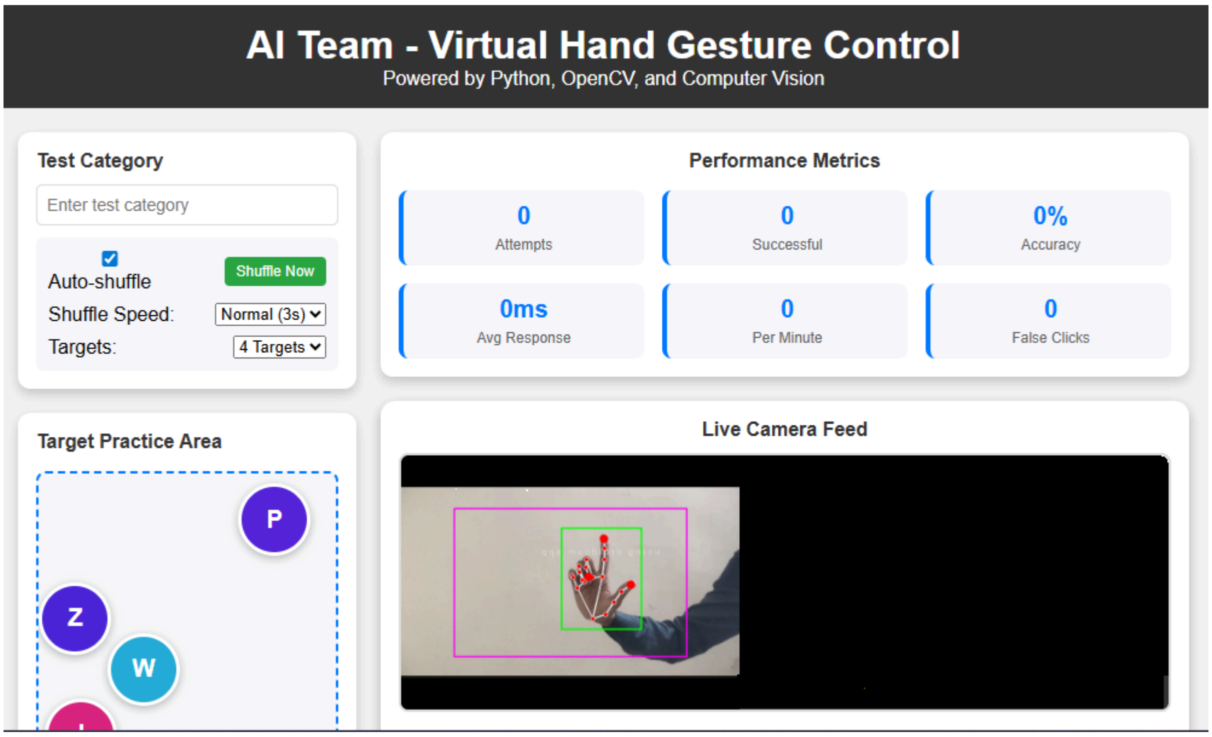
The project followed four stages: (1) Research and Ideation: Studied existing literature and identified real-world use cases like hospital kiosks and classroom boards. (2) Design and Simulation: Created diagrams and UI layouts showing how gestures map to commands. (3) Programming: Implemented gesture detection in Python, tested accuracy in different conditions, and refined timing for reliable recognition. (4) Output and Presentation: Prepared demonstrations, documented findings, and reflected on accessibility and ethical issues.

Apparatus and Materials

The experimental setup consisted of a standard laptop with an Intel Core i5 processor, 8 GB RAM, and a 720p webcam for video capture. The project was implemented using Python 3.11 on Windows 11. The software stack included OpenCV for image processing, MediaPipe for 21-point hand landmark detection, and PyAutoGUI for simulating mouse and scroll actions. All testing was performed through a browser interface(Flask), making the system lightweight and hardware-independent. No external gesture dataset was used; all gestures were captured in real-time via the webcam.

Implementation and Website Testing: The system was implemented using Python OpenCV and MediaPipe for detection and classification[3], and PyAutoGUI for executing browser actions. A local test website was created to simulate real-world interactions, where gestures controlled clicks, typing, and page navigation. Testing was conducted across multiple browsers (e.g., Chrome, Brave) and under different lighting and background conditions to ensure robustness. Performance was measured in terms of accuracy, latency (average frame rate), smoothness, and stability over extended use sessions.

Output and Presentation: A live demonstration website was prepared, and findings were documented. Considerations such as local-only video processing (no data storage) were highlighted to ensure privacy and security in public use scenarios.


AI Team - Virtual Hand Gesture Control

Powered by Python, OpenCV, and Computer Vision

Test Category

Enter test category

Auto-shuffle: ✓

Shuffle Now

Shuffle Speed: Normal (3s)

Targets: 4 Targets

Performance Metrics

0	0	0%
Attempts	Successful	Accuracy
0ms	0	0
Avg Response	Per Minute	False Clicks

Target Practice Area

Live Camera Feed

Figure 2: This figure shows how the virtual hand gesture control website works with a live camera feed that recognizes our hand gestures and tests on the target practice area to record the attempts, false clicks, successful clicks, accuracy, and average responses

Design and Simulation: System diagrams and interface layouts were developed to visualize gesture-to-command mappings. A shortlist of intuitive gestures was created based on technical feasibility and ease of use.
The features of the website are that we can change the number of targets (3-6 targets), test category section to type virtually with hand gestures, and change the speed of the shuffle (fast-2 seconds, normal-3 seconds, slow-5 seconds), and two graphs that measure the response time trend, and gesture accuracy (pointing, single click, double click).

5. Results

The system successfully recognised three main gestures with an accuracy of pointing is 100%, single click is 98%-99% and double click is 95%-96% in good lighting conditions. Cursor tracking was smooth, with latency under 100 milliseconds. Two fingers up(index, middle) gestures for click worked consistently, while three fingers up(index finger, middle finger, ring finger) recognition required calibration. Typing was effective, but sometimes misinterpreted. Despite these limitations, user tests showed that the systems could replace basic mouse functions in real time.

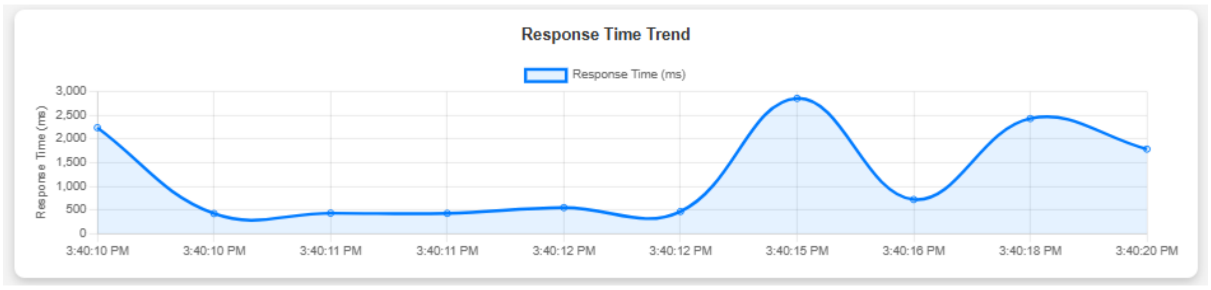


Figure 3:Results of the real experiment as the user tests from 3:40:11 pm to 3:40:20pm, where the 3:40:10 pm response time is consistent, but from 3:40:12 pm, we can see the graph shows an increase in the response time is increasing

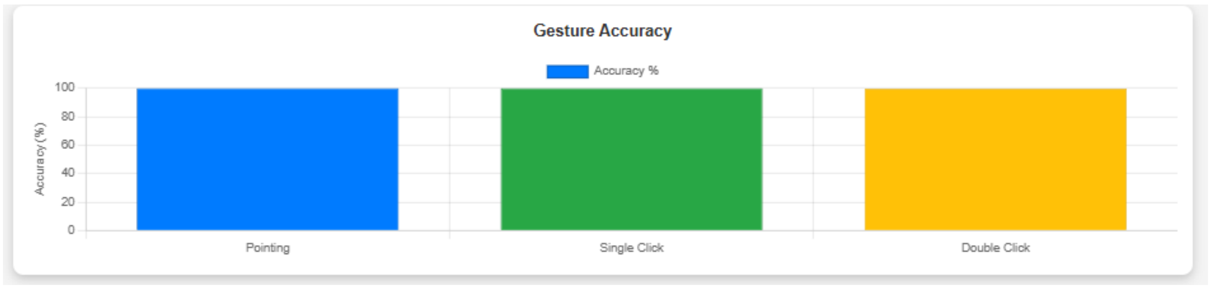
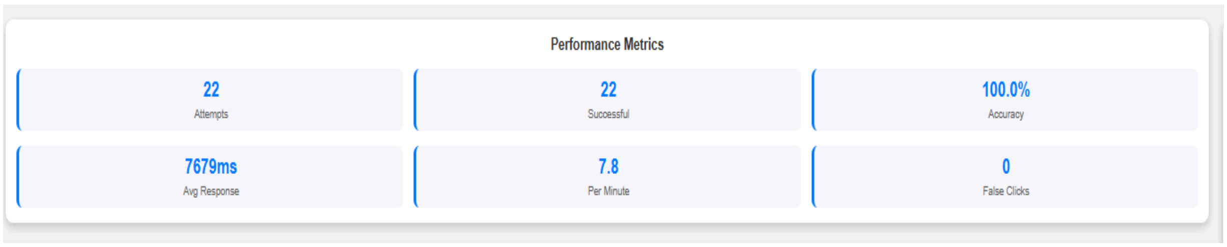


Figure 4:this gesture accuracy graph, which was tested through our local website, where the pointing(index finger) is 100% accurate, single click(index finger, middle finger), which shows 98%-99% accuracy, and double click(index finger, middle finger, ring finger) 95%-96% accuracy



Performance Metrics

22 Attempts	22 Successful	100.0% Accuracy
7679ms Avg Response	7.8 Per Minute	0 False Clicks

Figure 5: Performance metrics are recorded in terms of attempts, successful clicks, accuracy, average response, and false clicks.

From the result of a real experiment, as the user tests the hand gestures by using mediapipe and OpenCV, the results are for the 22 attempts, 22 are successful clicks, and 0 false clicks. The average response speed is 7679ms, which is about 7.6 seconds.

6. Discussion

The results confirm the effectiveness of touchless gesture-based interaction. suggesting that even simple sets of gestures can support intuitive and effective interaction (Pasquale et al., 2024; Wachs et al., 2011)[8]. Our hand gesture model is

browser-based execution. requires no external software installation and runs directly in a browser. Makes it usable in public setups like kiosks, classrooms, or exhibitions.Reduces system setup time and increases accessibility.
Low-Cost, Hardware-Free Approach
Uses only a standard webcam, no Kinect, Leap Motion, or depth sensors. Similar low-cost virtual mouse designs have proven feasible for everyday users[9]. Real-Time and Lightweight Uses MediaPipe's 21-point landmark tracking + OpenCV for smooth detection. Runs on standard laptops/desktops without requiring GPUs.
Deployability & Extensibility: It can be extended to VR/AR, presentations, or assistive technology for people with motor impairments. Easy to add new gestures in the future because of the modular design.
However, challenges remain. Performance decreases in low light, gestures sometimes overlap, and prolonged use may cause hand fatigue. Future improvements include training machine learning models on larger datasets, improving robustness across skin tones and hand sizes, and expanding gesture sets to include drag-and-drop and text input.

7. Ethical & Social Impact

Gesture-based systems raise important ethical and social considerations. On the positive side, they provide hygienic interaction in shared spaces and greater accessibility for differently-abled users. On the other hand, biases in recognition models may disadvantage certain users if datasets are not diverse. Privacy is another concern, as webcams capture sensitive visual data. To address this, our system processes all video locally and does not store or transmit data.
By considering inclusivity and fairness, touchless interfaces can be developed responsibly and used to benefit society.

8. Conclusion

This project demonstrates that AI-powered gesture recognition can replicate core mouse functions using only a webcam and open-source libraries. The system is cost-effective, responsive, and accessible, making it suitable for applications in education, healthcare, and public environments. While challenges of lighting and ergonomics remain, the prototype highlights the real-world potential of touchless computing.
Future directions include expanding the gesture vocabulary, integrating machine learning for greater robustness, and testing the system in real-world environments

such as schools and hospitals. Touchless interfaces represent not only a step toward futuristic computing but also a practical solution for today's needs.

References

1. Carmelo Ardito, Paolo Buono, Maria Francesca Costabile, and Giuseppe Desolda.2015. Interaction with Large Displays: A Survey. ACM Comput. Surv. 47, 3 (2015),46:1–46:38. https://doi.org/10.1145/2682623
2. Luca Console, Fabrizio Antonelli, Giulia Biamino, Francesca Carmagnola, Federica Cena, Elisa Chiabrando, Vincenzo Cuciti, Matteo Demichelis, Franco Fassio,Fabrizio Franceschi, Roberto Furnari, Cristina Gena, Marina Geymonat, Piercarlo Grimaldi, Pierluigi Grillo, Silvia Likavec, Ilaria Lombardi, Dario Mana, Alessandro Marcengo, Michele Mioli, Mario Mirabelli, Monica Perrero, Claudia Picardi, Federica Protti, Amon Rapp, Rossana Simeoni, Daniele Theseider Dupré, Ilaria Torre, Andrea Toso, Fabio Torta, and Fabiana Vernero. 2013. Interacting with social networks of intelligent things and people in the world of gastronomy. ACM Trans. Interact. Intell. Syst. 3, 1 (2013), 4:1–4:38. https://doi.org/10.1145/2448116.2448120
3. Pasquale, T., Gena, C., & Vernero, F. (2024). Defining a mid-air gesture dictionary for web-based interaction. In the International Conference on Advanced Visual Interfaces (AVI 2024). ACM. https://doi.org/10.1145/3656650.3656661
4. Guzsvinecz, T., Szücs, V., & Sik-Lányi, C. (2019). Suitability of the Kinect Sensor and Leap Motion Controller—A Literature Review. *Sensors (Basel, Switzerland)*, 19. https://doi.org/10.3390/s19051072.
5. Ardito, C., Buono, P., Costabile, M., & Desolda, G. (2015). Interaction with Large Displays. *ACM Computing Surveys (CSUR)*, 47, 1 - 38. https://doi.org/10.1145/2682623.
6. Shibly, K., Dey, S., Islam, M., & Showrav, S. (2019). Design and Development of Hand Gesture-Based Virtual Mouse. *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, 1-5. https://doi.org/10.1109/ICASERT.2019.8934612.
7. Johnson, H., & Saniie, J. (2022). Distributed Gesture Controlled Systems for Human-Machine Interface. *2022 IEEE International Conference on Electro Information Technology (eIT)*, 285-289. https://doi.org/10.1109/eIT53891.2022.9813764.
8. Stern, H., Wachs, J., & Edan, Y. (2008). Designing Hand Gesture Vocabularies for Natural Interaction by Combining Psycho-Physiological and Recognition Factors. *Int. J. Semantic Comput.*, 2, 137-160. https://doi.org/10.1142/S1793351X08000385.

9. Chaudhary, A., Raheja, J., Das, K., & Raheja, S. (2011). A Survey on Hand Gesture Recognition in the Context of Soft Computing. , 46-55. https://doi.org/10.1007/978-3-642-17881-8_5.