Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2025) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

10th International Conference on Computer Science and Computational Intelligence 2025 (ICCSCI 2025)

Vision-Based Hand Gesture Recognition as an Accessible Alternative to Conventional Game Input for Players with Disabilities

Virly Karaniyametta Arista[a], Putri Khairani Azzahra[a], Colin Wilson[a], Muhammad Fikri Hasani[a]*, Ayu Maulina[a]

[a]*Computer Science Department, School of Computer Science, Bina Nusantara University, Jakarta, Indonesia, 11480*

Abstract

This paper presents a vision-based hand gesture recognition system designed as an accessible game input alternative for individuals with motor and speech impairments. This system utilizes computer vision techniques to detect and classify hand gestures, which are then translated into keyboard inputs using PyAutoGUI. By mapping predefined gestures to directional keys (← ↑ ↓ →), this system allows users to interact with games without relying on physical controllers or modifying game code. Usability testing was conducted using the System Usability Scale (SUS), yielding an average score of 65.62, indicating marginal acceptability and highlighting areas for refinement. The results demonstrate the potential of gesture-to-keyboard translation as a non-invasive, scalable, and low-cost approach to improve accessibility in digital environments, particularly gaming. Future work will focus on improving recognition accuracy and response time.

Keywords: hand gesture recognition; accessibility; computer vision; game input system.

1. Introduction

Video games have significantly evolved over the past few decades, shifting from simple arcade systems to complex and immersive experiences powered by artificial intelligence, virtual reality, and cloud-based gaming. Along with these developments, the industry emphasizes game accessibility, ensuring that players of all abilities can also engage in the digital experience.

* Corresponding author. Tel.: +62-804-169-6969.
E-mail address: Muhammad.fikri003@binus.ac.id

In recent years, accessibility in gaming has emerged as a critical area of research, with developers, researchers, and organizations striving to create more inclusive technologies. Major industry players, such as Microsoft and Sony, have introduced adaptive controllers and software-based accessibility features, while academic studies keep looking for alternative input methods that can be used for gamers with disabilities. Beyond gaming, accessibility has become a major focus in various fields, including healthcare, human-computer interaction, and artificial intelligence. This aligns with global initiatives such as the United Nations Sustainable Development Goals (SDGs), particularly Goal 10: Reducing Inequality, which encourages the removal of barriers preventing individuals with disabilities from fully participating in society [1]. Organizations such as the World Health Organization (WHO) also highlight the significance of accessibility, reporting that over 1 billion people worldwide live with disabilities, many of whom have trouble accessing digital services, education, and entertainment [2].

Accessibility is still an issue for individuals with speech and motor impairments, especially those who are bedridden, regardless of technological advances. For individuals with disabilities, conventional input devices like touch screens, keyboards, and controllers offer significant difficulties because they require long-term physical contact and fine motor skills [3,4]. Voice commands and adaptive controllers may provide some accessibility improvements, but they are not practical for people with severe motor or speech impairments because they often need precise hand movements or voice recognition. Accessibility research has introduced several kinds of assistive technologies, such as motion sensors, eye-tracking devices, and brain-controlled interfaces (EEG-based systems). Even though these techniques offer alternative inputs, their potential for broad use is limited because they often need costly hardware, complex calibration, or invasive procedures [5–8].

Alternative input methods that require little physical strain and no fine motor control are required to make gaming truly inclusive. This study proposes a vision-based hand gesture recognition system as a more accessible alternative to traditional game controls or input methods, which allows players to control games with simple hand movements. This system uses computer vision techniques to identify hand landmarks and categorize predefined gestures [3], including numbered gestures, closed fists, and open palms. These gestures are mapped to real-time keyboard inputs (e.g., "←↑↓→" for movement). This method is easily integrated into existing games, allowing users to interact with them without the need for physical controllers or game code modifications, providing a non-invasive and adaptable solution for players with motor and speech impairments.

Hand gesture recognition systems, which use computer vision, provide a more accessible and non-invasive alternative that only requires a standard camera. Computer vision is a more affordable and scalable solution than sensor-based methods like gloves or motion-tracking devices because it doesn't require extra physical devices.

Through this study, we assess the accuracy, responsiveness, and usability of the gesture-to-keyboard input translation system to evaluate its efficiency. With this approach, gaming accessibility could be improved, and individuals with disabilities could interact with digital systems more freely and independently even with their physical limitations.

2. Related Works

2.1. Alternative Input Methods for Accessibility

Nowadays, individuals with impairments also can interact with digital systems by using assistive technologies like conventional controllers, conventional keyboards, voice commands, and many other assistive technologies as alternative input methods. However, each of the technologies comes with its own disadvantage. Voice commands or speech-based systems are used in audio-based recognition to allow users to operate interfaces by using voice, so they are not effective for individuals with speech impairments or individuals in noisy environments.

Another innovation is an EEG-based system for brainwave-controlled interfaces. This system converts thoughts or brainwaves into commands by detecting electrical activity in the brain [4]. EEG-based system for brainwave-controlled interfaces. This system converts thoughts or brainwaves into commands by detecting electrical activity in the brain. Regardless of their potential, these systems' widespread usage is limited because of their costly hardware and extensive calibration conditions. As an alternative, there are eye-tracking devices that track users' gaze to interact with elements on the screen. Although eye-tracking works well, it needs to be calibrated precisely and can be less comfortable for long-term use. Motion sensors and adaptive controllers, such as the Xbox Adaptive Controller, offer customizable features but have platform limitations and are expensive [5]. Considering these disadvantages, individuals with severe motor and speech impairments require an alternative input method that is hands-free, straightforward, and minimizes fatigue.

2.2. Vision-Based Hand Gesture Recognition

Accessibility has been helped by computer vision, especially when it comes to recognizing human actions and interpreting them into system commands. Unlike sensor-based solutions, which require specialized hardware, vision-based systems track and interpret gestures with standard cameras. Deep learning models, including convolutional neural networks (CNNs) and other modified CNNs, have increased the accuracy of hand gesture recognition by allowing real-time tracking and categorization [6].

Vision-based systems have several benefits over wearable motion sensors, including a simpler setup process, the elimination of the need for extra hardware, and the capability to offer natural interaction without the need for direct physical contact. Studies have shown that vision-based gesture recognition works well for accessibility applications, such as navigating virtual environments and controlling smart devices. With these advantages, vision-based hand gesture recognition offers a good alternative for traditional game input methods, especially for individuals with disabilities.

2.3. Hand Gesture Recognition Techniques

Hand landmark identification is an important component of hand gesture recognition because it identifies key points on the hand to track movement and orientation.

Many studies have been conducted on sensor-based tracking of hand skeletons, offering a framework for gesture interpretation. Real-time gesture-to-keyboard mapping is achieved using vision-based tracking systems, such as MediaPipe Hand Tracking, which identify and categorize 21 hand landmarks [3]. The illustration for the 21 hand landmarks is as illustrated on Figure 1 with each point initialization stated in Table 1.

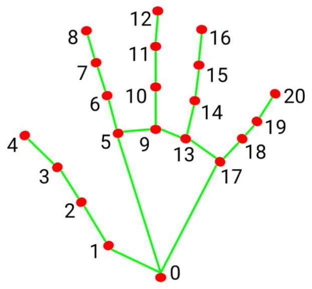


Fig. 1. Sensor-based tracking by 21 hand skeleton landmarks.

Table 1. the 21 hand landmarks for sensor-based tracking

The 21 landmarks of hand skeleton for sensor-based tracking					
0.	Wrist	7.	Index_Finger_DIP	14.	Ring_Finger_PIP
1.	Thumb_CMC	8.	Index_Finger_TIP	15.	Ring_Finger_DIP
2.	Thumb_MCP	9.	Middel_Finger_MCP	16.	Ring_Finger_TIP
3.	Thumb_IP	10.	Middel_Finger_PIP	17.	Pinky_MCP
4.	Thumb_TIP	11.	Middel_Finger_DIP	18.	Pinky_PIP
5.	Index_Finger_MCP	12.	Middel_Finger_TIP	19.	Pinky_DIP
6.	Index_Finger_PIP	13.	Ring_Finger_MCP	20.	Pinky_TIP

2.4. Hand Gesture Recognition in Gaming

Hand gesture recognition has become a game-changing method in computer vision-based gaming that allows for more natural and engaging user interactions. Real-time computer vision algorithms allow gaming systems to identify and interpret certain hand gestures. This allows users to modify gameplay commands without using traditional input methods. Digital image processing methods that extract important elements like shape, texture, and motion from video, such as motion tracking and object recognition, improve the accuracy of gesture-based controls. These developments establish hand gesture detection as an essential part of modern gaming technology, redefining interactive entertainment by providing more responsive and natural gameplay experiences [7].

2.5. Gesture-Based Accessibility for Individuals with Disabilities

Hand gesture recognition has the potential to improve accessibility for individuals with impairments. Previous studies have explored gesture-controlled interfaces for assistive communication devices, which let users interact with computers and smart devices using predefined hand gestures. Research has also proven the effectiveness of gestures to control interfaces to let users with mobility impairments perform keyboard-based interactions without physical input devices [8,9].

2.6. Motor and Speech Impairments in Gaming Accessibility

Individuals with severe motor and speech impairments face difficulty in gaming due to the reliance on traditional controllers and fine motor skills. Some conditions that affect amyotrophic lateral sclerosis, cerebral palsy, stroke, and spinal cord injury [10,11].

For these individuals, prolonged use of standard controllers can cause discomfort and difficulty in playing or maintaining grip strength [5]. Gesture-based recognition systems can help mitigate these challenges by providing an intuitive, non-contact method of interaction, enabling disabled gamers to engage with digital experiences more comfortably and independently.

3. Methodology

There are four phases that are passed through before doing the evaluation. The four phases are system design, implementation, testing, and data collection. The flow of the phases is illustrated in Figure 2.

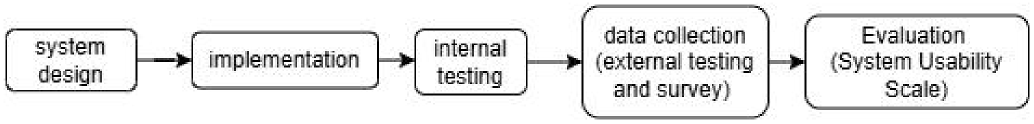

system design
implementation
internal testing
data collection (external testing and survey)
Evaluation (System Usability Scale)

Fig. 2. phases of methodology

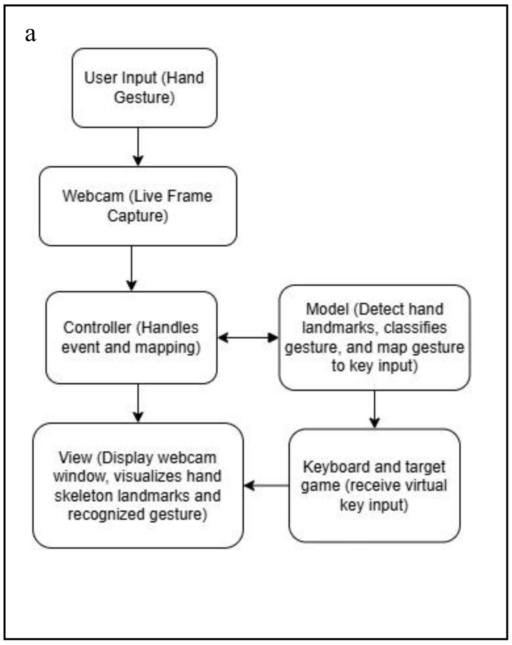
3.1. System Design

The system captures live video through the webcam, where each frame is processed using MediaPipe to identify 21 hand landmarks. Based on the finger positions detected, gestures are categorized (e.g., open palm, closed fist, two fingers). These gestures are then translated into corresponding keyboard events using PyAutoGUI—for example, an open palm gesture is mapped to the 'W' key. This keyboard event is passed to the operating system and interpreted by any game that supports traditional keyboard input, allowing seamless integration without modifying the game source code. The architecture for this system as stated above is illustrated in Figure 3. (b) with the architecture model similar to MCV (Model-Controller-View).

This research utilizes an open-source hand gesture recognition system obtained from a Discord community. The system is based on vision technology, utilizing OpenCV (cv2), MediaPipe, and PyAutoGUI to provide an accessible gaming input method using visual hand gestures.

The system operates through a structured process consisting of three main stages as illustrated in Figure 3. (b):

1. Hand Detection & Tracking: The system captures real-time video frames from a connected webcam. By using MediaPipe Hands, the system detects and tracks 21 specific hand landmarks in each frame. These landmarks enable precise recognition of finger movements and hand positioning.
2. Gesture Classification: After detecting the hand, the system analyzes the relative positioning of fingers to classify gestures. The system supports a predefined set of gestures, such as open palm (up or jump), closed fist (down or roll), index finger gesture (left), index and middle fingers gesture (right), pinky finger gesture (exit), and other custom hand signs (which can be assigned to specific actions).
3. Command Execution & Interaction with Games: The recognized gestures are mapped to specific keyboard inputs using PyAutoGUI. Since PyAutoGUI simulates keyboard presses, the system can be used with any game or application that supports conventional keyboard input without requiring modifications to the source code.


a
User Input (Hand Gesture)
Webcam (Live Frame Capture)
Controller (Handles event and mapping)
Model (Detect hand landmarks, classifies gesture, and map gesture to key input)
View (Display webcam window, visualizes hand skeleton landmarks and recognized gesture)
Keyboard and target game (receive virtual key input)

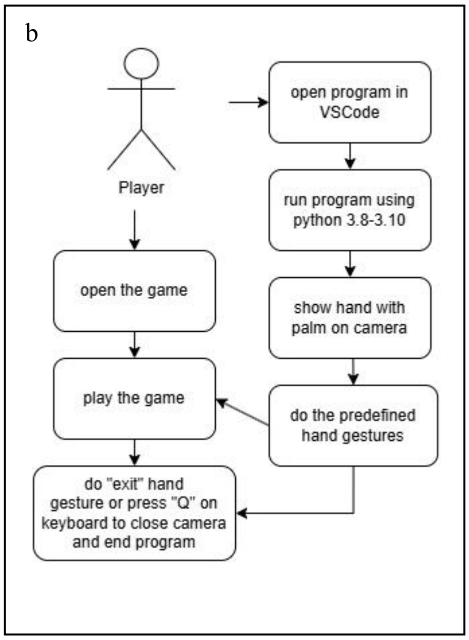

b
Player
open program in VSCode
run program using python 3.8-3.10
show hand with palm on camera
do the predefined hand gestures
open the game
play the game
do "exit" hand gesture or press "Q" on keyboard to close camera and end program

Fig. 3. (a) system structure; (b) system operation.

3.2. Implementation

The designed system will be implemented by running the program in Visual Studio Code using Python versions 3.8 to 3.10, ensuring compatibility with OpenCV, MediaPipe, and PyAutoGUI. Once executed, the system will open the webcam using OpenCV and detect the user's hand via the camera, utilizing MediaPipe to track hand landmarks in real time. When users perform predefined hand gestures, the system will map them to corresponding keyboard inputs, which will be emulated in real-time by PyAutoGUI, allowing interaction with the game without modifying its code. After launching the game while the program and camera are running, the player's recognized hand gestures will control the game as if using a keyboard. To close the webcam and terminate the program, the user can either raise their pinky finger or press 'Q' on the keyboard. The implementation of the gesture-to-keyboard mapping is available in our GitHub repository (Hinakhina/gesture-accessibility-game-input). The flow of the system operation is illustrated in Figure 4, showing how hand gestures are detected and mapped to keyboard inputs in real time.

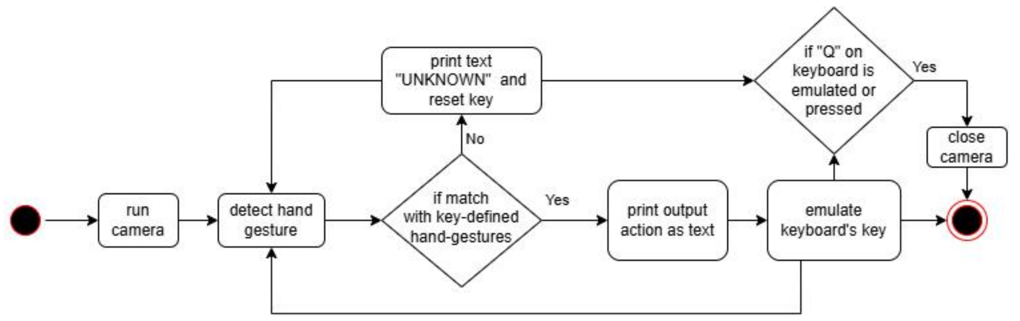

print text "UNKNOWN" and reset key
if "Q" on keyboard is emulated or pressed
Yes
No
close camera
run camera
detect hand gesture
if match with key-defined hand-gestures
Yes
print output action as text
emulate keyboard's key

Fig. 4. system flowchart.

3.3. Testing

The system will be tested using a standard webcam with a minimum resolution of 360p to ensure accurate hand tracking. The testing environment consists of a mid-range PC and Visual Studio Code that is equipped with Python version 3.8 to 3.10, OpenCV, MediaPipe, and PyAutoGUI. The accuracy of gesture classification will be evaluated in real-time scenarios where users navigate movements within a basic 2D game to go up, left, right, and down. The testing will be conducted internally and externally. Internal testing was conducted by the developers to assess the core functionalities of the system. This phase focused on evaluating the gesture recognition and program functionalities to ensure the system worked and ran properly before proceeding to external testing conducted by survey participants.

3.4. Data Collection

A qualitative and quantitative research approach is used to assess the vision-based hand gesture recognition system's accuracy, usability, and effectiveness. The study is divided into three stages: participant selection, data collection, and data analysis. A survey will be taken by approximately 30 participants to collect a broad range of insights. Participants will include both people with and without motor impairments for a comparative evaluation of usability across different user groups. Participants must test the system and watch a demonstration video that demonstrates its features before filling out a survey. During external testing, participants run the system and use hand gestures to control in-game actions. After the test, participants will be interviewed and asked to complete a System Usability Scale (SUS) questionnaire to provide feedback on their experience. The questionnaire will be distributed digitally to ensure consistency and ease of data collection. The survey will evaluate various aspects, such as perceived usability, accuracy, learning ease, comfort and fatigue levels, possible accessibility improvements, and user feedback in general.

3.5. Expected Result

This research focuses on creating a vision-based hand gesture recognition system that converts hand gestures into keyboard commands in real time. The goal is to provide a more accessible gaming experience for users with motor and speech disabilities, eliminating the need for extra hardware or altering existing games.

4. Results and Discussion

4.1. System Usability Scale (SUS) Evaluation

The usability evaluation was conducted using the System Usability Scale (SUS), involving 32 respondents, both impaired and non-impaired people who answered 10 standard questions using a 5-point Likert scale. The analysis consists of two parts, which are per-item statistical summary and overall SUS score and interpretation. The detailed user responses for each SUS question are shown in Table 2, capturing individual participant ratings across all usability statements.

Table 2. SUS evaluation from users for each question.

Questions	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Max	5	5	5	5	5	5	5	5	5	5
Min	1	1	2	1	2	1	2	1	2	1
Average	3.5625	2.6875	4.0312	2.8125	4.0000	2.5000	4.2187	2.5000	4.0265	3.0625
Standard Deviation	1.26841	1.17603	1.03126	1.46876	0.87988	1.24434	0.87009	1.24434	0.94825	1.47970

Positively framed items (Q1, Q3, Q5, Q7, and Q9) yielded higher mean scores, indicating that users generally perceived the system as easy to use, well-integrated, and confidence-enhancing. Conversely, negatively worded items (Q2, Q4, Q6, Q8, and Q10) received lower ratings, suggesting potential usability concerns related to system complexity and the degree of user autonomy.

The scatterplot in Figure 5 illustrates each user's SUS score. A line is added to show the sequence trend across participants.

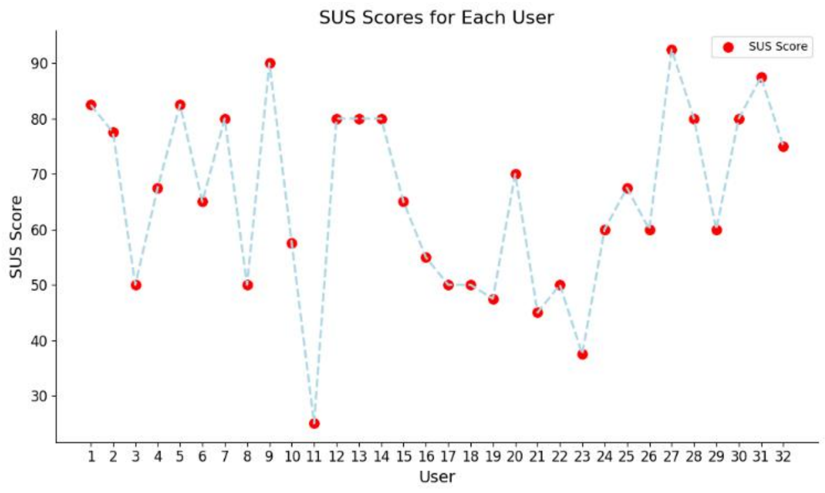


Fig. 5. SUS scores for each user.

Using the standard SUS calculation formula, the total SUS score is 65.62. This average falls within the "Marginal Acceptable" range of usability. It indicates that the system is generally functional but still has areas that need improvement to enhance the overall user experience. The standard deviation of the SUS scores was 16.61, indicating moderate variability in user responses. A 95% confidence interval shows that the true average usability perception likely lies between 57.8 and 73.4, reinforcing the marginal acceptability of the system, with room for usability improvement especially in gesture accuracy.

While many users rated the system favorably, the data also revealed some notably low individual scores, which pulled the average down. These lower scores were associated with concerns about ease of use, gesture recognition consistency, and the perceived need for technical assistance. Importantly, the standard deviation of the SUS scores was 16.61, which indicates a high level of variability in user experiences. This suggests that the average score alone may not fully represent the overall usability of the system. Some users had positive experiences, but others encountered significant difficulties.

Common feedback highlighted the need for improved gesture recognition accuracy and more intuitive control mapping. These insights provide a strong basis for refinement in future development, especially to support better onboarding and accessibility for broader user groups.

4.2. Gesture Accuracy and System Responsiveness

In addition to the SUS assessment, the evaluation also included measurements of gesture recognition accuracy and system responsiveness. The distributions of user feedback are presented in the bar charts below, with the gesture recognition accuracy stated in Figure 6. (a) and the system's responsiveness stated in Figure 6. (b).

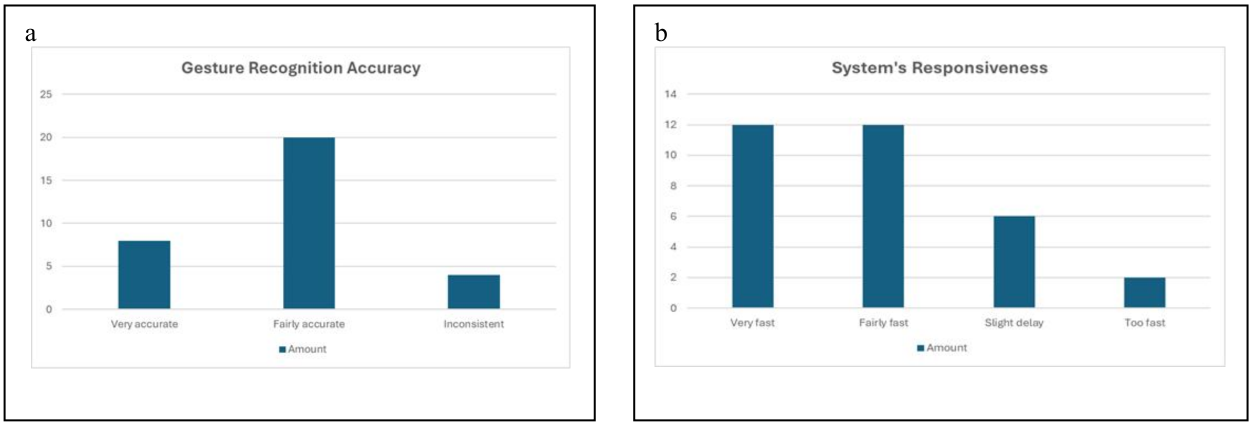


Fig. 6. (a) users' response of gesture recognition accuracy; (b) users' response of system's responsiveness.

The findings show that gesture recognition was generally accurate, though some inconsistencies occurred, likely due to factors like lighting or hand speed. Responsiveness was mostly rated as fast, with a few users noting slight delays or excessive speed, suggesting a need for minor adjustments.

The system itself is only compatible on PC or laptop with VS Code and libraries installed since for now the code can only be used by running the code in VS Code. Overall, the system is compatible with most PC games that respond to DOM's keyboard events, including browser games and simple desktop applications. However, games using game engine input capture layers (e.g., Unity Input System or raw hardware APIs) are not able to detect emulated keyboard inputs. Unity's legacy input system (Input.GetKeyDown()) only detects physical keyboard inputs, making it incompatible with synthetic events from tools like PyAutoGUI. This differs from browser-based games, which use the DOM (Document Object Model) event and can process simulated inputs, which means PyAutoGUI is mostly used for DOM's keyboard event and not for actual hardware keyboard inputs.

Due to limitations in the study scope and the use of already made models from the library by MediaPipe, quantitative metrics such as precision, recall, and F1-score were not computed. However, these performance indicators are planned for inclusion in future work to better quantify system reliability in various gesture recognition scenarios.

4.3. User Feedback and Insight

Qualitative feedback revealed that users generally found the system intuitive and easy to use, even when operated with only one hand. This suggests a level of accessibility suitable for users with limited mobility or dexterity. Some participants noted the need for a brief tutorial to assist first-time users in understanding the available gestures.

Gesture recognition was generally considered accurate by participants; however, some specific gestures such as open palm, showing no fingers, or two-finger gestures were occasionally difficult to detect. These issues often depend on factors like lighting, hand positioning, and speed of movement. Users recommended enhancing the system's ability to recognize minor variations in hand posture to improve overall accuracy.

Responsiveness was generally well received, though a few participants mentioned response delays during rapid gestures, while others felt that some actions triggered too quickly. Suggestions included the addition of customizable sensitivity settings and clearer visual cues during gesture detection to enhance user control and confidence.

4.4. Combining Quantitative and Qualitative Findings

By integrating the SUS score, gesture accuracy data, and user feedback, the fuller pictures of the system's performance are:

- Usability: The SUS score of 65.62 reflects decent usability, but user comments pointed to challenges, especially with gesture recognition inconsistencies.
- Responsiveness: While most users found the response time acceptable, feedback indicated variability depending on gesture speed and system settings. Some users experienced delays or responses too quickly, indicating a need for further adjustment.
- Improvement Areas: Qualitative insight emphasizes areas for improvement, such as gesture recognition accuracy and response time. Users also expressed interest in more flexibility for gesture action and finer controls to adjust responsiveness.
- Platform Compatibility: The system is compatible with most browser-based and desktop applications that accept DOM's keyboard event. However, games built using engines like Unity may use direct hardware polling, making simulated keystrokes from PyAutoGUI undetectable.

5. Conclusion

This research introduced a vision-based hand gesture recognition system as an accessible alternative to conventional game input methods, aimed at users with motor and speech impairments. By translating simple hand gestures into keyboard inputs using webcam-based detection, the system allows users to interact with games without modifying game code or using physical controllers. Usability testing using the SUS yielded a score of 65.62, indicating marginal acceptability and highlighting potential for improvement. Despite some limitations in accuracy and responsiveness, participants expressed positive responses regarding its usability and potential impact. The findings support the viability of this approach as a non-invasive, low-cost accessibility tool, with future development focused on refining gesture recognition and expanding its compatibility with various digital applications.

Acknowledgements

We would like to acknowledge the open-source hand gesture recognition project shared by Bagusde within the Discord developer community, which served as the foundation for this research. The original code was adapted with minor revisions to better suit the needs of our study. We are grateful for the shared knowledge that made this research possible.

Data Availability

The data for this study is publicly available at this link

(https://docs.google.com/spreadsheets/d/1D_kSHJkVPqdXHhDg3zENcb8VCou7Fd22lH1PdmFqwJo/edit?usp=drivesdk)

Author Contribution Statement

V. K. Arista contributed to the development and writing of the Methodology section and was responsible for implementing and documenting the code. C. Wilson contributed to the writing of the Results and Discussion section. P. K.Azzahra was responsible for writing the Abstract, Introduction, Related Works, and Conclusion. M. F. Hasani and A. Maulina served as supervisors, providing guidance and feedback throughout the research and writing process.

AI Usage Declaration

We declare that we use AI tools ChatGPT OpenAI in this manuscript to check for grammar and paraphrasing. The result from AI is not used directly and is under human supervision for error checking.

References

[1] United Nations. Goal 10: Reduce inequality within and among countries. United Nations Sustain Dev Goals n.d. https://sdgs.un.org/goals/goal10 (accessed March 11, 2025).

[2] Disability n.d. https://www.who.int/news-room/fact-sheets/detail/disability-and-health (accessed March 11, 2025).

[3] Islam MR, Rahman R, Ahmed A, Jany R. NFS: A Hand Gesture Recognition Based Game Using MediaPipe and PyGame. ArXivOrg 2022. https://doi.org/10.48550/ARXIV.2204.11119.

[4] Moghimi S, Kushki A, Marie Guerguerian A, Chau T. A Review of EEG-Based Brain-Computer Interfaces as Access Pathways for Individuals with Severe Disabilities. Assist Technol 2013;25:99–110. https://doi.org/10.1080/10400435.2012.723298.

[5] Baltzar P, Hassan L, Turunen M. Assistive technology in gaming: A survey of gamers with disabilities 2023.

[6] Shanmugam S, Narayanan RS. An accurate estimation of hand gestures using optimal modified convolutional neural network. Expert Syst Appl 2024;249:123351. https://doi.org/10.1016/J.ESWA.2024.123351.

[7] Choudhary P, Shinde B, Yadav A, Kumayu A, Parmar A, Upadhyay A, et al. Gesture Driven Gaming: A Deep Dive into Computer Vision-Based Hand Gesture Recognition. SSRN Electron J 2024. https://doi.org/10.2139/SSRN.5008717.

[8] Husna R, Candra Brata K, Anggraini IT, Funabiki N, Rahmadani AA, Fan C-P. An Investigation of Hand Gestures for Controlling Video Games in a Rehabilitation Exergame System 2025. https://doi.org/10.3390/computers14010025.

[9] Chen J, Zhao S, Meng H, Cheng X, Tan W. An interactive game for rehabilitation based on real-time hand gesture recognition. Front Physiol 2022;13:1028907. https://doi.org/10.3389/FPHYS.2022.1028907/BIBTEX.

[10] Morris R, Whishaw IQ. Arm and hand movement: Current knowledge and future perspective. Front Neurol 2015;6:129357. https://doi.org/10.3389/FNEUR.2015.00019/BIBTEX.

[11] Common Neurological Disorders Affecting Speech n.d. https://lonestarneurology.net/neurological-disorders/neurological-disorders-affecting-speech/ (accessed March 12, 2025).

[12] Guaman D, Delgado S, Perez J. Classifying Model-View-Controller Software Applications Using Self-Organizing Maps. IEEE Access 2021;9:45201–29. https://doi.org/10.1109/ACCESS.2021.3066348.

[13] Ramirez-Noriega A, Martinez-Ramirez Y, Chavez Lizarraga J, Vazquez Niebla K, Soto J. A software tool to generate a model-view-controller architecture based on the entity-relationship model. Proc - 2020 8th Ed Int Conf Softw Eng Res Innov CONISOFT 2020 2020:57–63. https://doi.org/10.1109/CONISOFT50191.2020.00018.