International Journal of Current Science Research and Review
ISSN: 2581-8341
Volume 08 Issue 05 May 2025
DOI: 10.47191/ijcsrr/V8-i5-57, Impact Factor: 8.048
IJCSRR @ 2025
www.ijcsrr.org

Gesture Plus: A Novel Approach to Enhance Interactive Media

Savita Lade[1], Uzair Afnan Shaikh[2], Arya Sahane[3], Kasturi Pednekar[4], Akansha Wankhade[5]
[1]Professor, Dept. of Computer Engineering MCT's Rajiv Gandhi Institute of Technology Mumbai, India
[2,3,4,5] Student, Dept. of Computer Engineering MCT's Rajiv Gandhi Institute of Technology Mumbai, India

ABSTRACT: GesturePlus is a comprehensive human- computer interaction system that integrates hand gestures, voice commands, and a chatbot for seamless handling. Through the use of computer vision and machine learning, GesturePlus recognizes hand gestures through image pre- processing, fingertip detection, and real-time classification with high accuracy. GesturePlus finds great use within sterile environments or for people with limited mobility in an intuitive manner that replaces the traditional input devices.

KEYWORDS: Human-Computer Interaction (HCI), Gesture Recognition, Machine Learning, Computer Vision, Chatbot, Retrieval-Augmented Generation (RAG), Multi- modal Interaction, Voice Command.

I. INTRODUCTION

GesturePlus is a leading-edge technology for inter- acting with computer interfaces using voice commands, predefined hand-gesture recognition, and a chatbot in- tegrated with the voice module, bridging the gap be- tween human users and digital interfaces in an era where seamless human-computer interaction is a requisite. GesturePlus combines innovative technologies such as machine learning, computer vision, natural language processing (NLP) models to generate a multi-functional, user-friendly, multi-platform, and adaptable solution that raises the bar for accessibility, usability, and interactivity across diverse disciplines.

The GesturePlus system is composed of three predominant modules. First, the Hand Gesture Module, which enables users to control and operate mouse commands through ambidextrous hand gestures, using computer vision and deep learning methodologies to analyze hand movements and map them to their specific commands, providing a touchless and interactive interface. This is incredibly beneficial in sanitary environments, including healthcare, and for people with mobility difficulties.

One of its most notable features is GesturePlus's multi-platform ability to run seamlessly. The system sup- ports inter-modality integration, allowing users to switch between all three modes. Thus, the system is set to revolutionize human-computer interaction by combining cutting-edge innovations to generate a more natural, touch-free, and user-friendly interface that responds to a variety of real instances.

II. LITERATURE REVIEW

A. Survey of Existing System

Human-computer interaction (HCI) has undergone tremendous growth, most notably in virtual mouse systems that allow users to maneuver computers using hand gestures. [1] examined a real-time fingertip tracking interface that compensates for occlusions and lighting conditions and is thus more practically applied to in- teractive systems. In the same vein, [2] presented an extensive overview of hand gesture recognition (HGR) technology, including detection methods, challenges, and potential opportunities to support improved accuracy in human-machine interfaces.

Gesture control has also been extensively investigated, with [3] presenting a system that uses hand gestures to manipulate virtual mouse and keyboard operations, noting limitations in gesture recognition accuracy and environmental dependency. In addition, [4] presented chatbot-based human-computer interactions, with an emphasis on natural language processing methods in Python to improve conversational agents. The Touchless Systems Virtual Palm Pointer Mouse uses machine learn- ing and computer vision to monitor palm movement, allowing hands-free control of the cursor. This increases accessibility for mobility-impaired users and provides an intuitive alternative to conventional input devices in [5].

B. Survey of Technology Stack

Gesture recognition has been a critical area of research in computer vision-based human-computer interaction.

International Journal of Current Science Research and Review
ISSN: 2581-8341
Volume 08 Issue 05 May 2025
DOI: 10.47191/ijcsrr/V8-i5-57, Impact Factor: 8.048
IJCSRR @ 2025
www.ijcsrr.org

[6] conducted a detailed review of computer vision techniques for HGR, analyzing strengths and limitations under varying conditions. Further [7] expanded on vision-based HGR by reviewing advances in methods, databases, and applications for HCI systems, emphasizing improvements in accuracy and real-world usability. Conversational AI systems have evolved with signifi- cant contributions from knowledge-powered chatbots. As introduced by [8], the ”Wizard of Wikipedia” dataset, which enables chatbots to retrieve and utilize factual information from Wikipedia, leading to more informed and contextual conversations.

C. Limitation of Existing system

- Limited to basic functions such as cursor moving, clicking, and scrolling.
- Depending on controlled light and simple back- ground for detection.
- There may be a noticeable lag in gesture recognition, making it unsuitable for real-time usage.
- Requires good cameras and optimized setups, making it inaccessible to low-end devices.
- Existing systems do not have a combination of the gesture recognition, the chatbot module, and the voice module.
- Traditional systems are OS dependent, functioning only on one operating system, while cross-platform compatibility is missing.
- Many traditional systems only allow one-handed gestures, affecting accessibility and usability.

D. Problem Statement

Existing gesture-based HCI systems have limited functions, rely on predefined one-handed gestures, only work on certain operating systems, and lack integra- tion with other interaction modalities. These constraints reduce their adaptiveness, accessibility, and accuracy, rendering them implausible in real-life applications. GesturePlus addresses these challenges through an AI- supported cross-platform system capable of using two- handed gestures, speech commands, and a chatbot module. Initiating computer vision, machine learning, and natural language processing, GesturePlus smoothens the user experience by focusing on more intuitive and seam- less human-computer interactions.

E. Objectives

- Create an AI-based recognition system for real- time hand gestures—one that depicts and deciphers complex hand gestures.
- Introduce the two-handed gesture methodology, enhancing the experience in this area to be more comfortable and intuitive.
- Achieve cross-platform compatibility so that the system works well across different OS and devices.
- Improve precision and adaptability of the software to achieve error minimization during lighting and background variations by recognizing hand gestures made by human beings.
- Optimize response time and system efficacy to accommodate all inputs efficiently.

III. PROPOSED SYSTEM

Proposed system is a multimodal interaction combining gesture and voice-based interaction. Aimed at enhancing user interaction with the computer by using alternative input modalities. Designed to detect easy natural movements of the palm and the instructions by the spoken voice in real time, the system responds to movement, clicks, changes of volume and brightness, traversing the files, and searching the web. It achieves this through a combination of cutting-edge computer vision, speech recognition, and text-to-speech technologies, combined with a broadly cross-platform implementation that adapts the features to the respective OS. It’s modular architecture, so each component operates independently but in concert, providing a responsive and adaptive interface.

A. Analysis

The study aims to point the need for an intuitive user-friendly system offering an alternative input device apart from the traditional keyboard and mouse. The system is flexible, reliable, rugged, with low latency, and responsive to light levels and background noise. Some challenges include accurate hand landmarks, gesture classification, and managing OS specific functions. Thus this study led to using time-tested open-source libraries like OpenCV, MediaPipe, Speech Recognition, and pyttsx3 for a modular architecture.

B. Framework

The framework is built on a modular architecture separating the gesture recognition, voice-based chatbot, and the system monitoring module. In gesture mod- ule, a live video feed captured via OpenCV is processed by MediaPipe to detect hand

International Journal of Current Science Research and Review
ISSN: 2581-8341
Volume 08 Issue 05 May 2025
DOI: 10.47191/ijcsrr/V8-i5-57, Impact Factor: 8.048
IJCSRR @ 2025
www.ijcsrr.org

landmarks. Then, a customized gesture recognition algorithm interprets the spatial relationships between key points for gesture classification. These mapped gestures determine the actions of the system in mouse-movement, click, scroll, and adjust settings such as volume and brightness via pyautogui. OS-specific functionalities are managed by libraries like osascript for MacOS, and pycaw along with screen_brightness_control for Windows. On the other hand, the voice-based chatbot module utilizes PyQt6 to create a graphical user interface that accommodates both text and voice inputs. It uses SpeechRecognition to capture the audio input and convert it to text, pyttsx3 for text-to-speech conversion of the bot's response while a command processing engine interprets commands using a user rule-based parser and invokes the corresponding system operations. Although System Monitoring and Parallel Execution is not the core interaction module, the system monitoring component gathers performance metrics using psutil and visualizes data with Plotly. The run_parallel.py script ensures that the gesture and voice modules can run concurrently using Python's multiprocessing. Integration of these modules with parallel processing to allow simultaneous operation, forms a comprehensive framework that can cater to a wide range of user needs. the said text through a series of conditional statements that get commands like "search" or "launch gesture recognition" linked to the desired functions. The voice assistant providing instant feedback through the pyttsx3 library updated accordingly to the GUI. The emphasis throughout the processes is on real-time performance and reliability to make the system remain responsive to changes in the aforementioned conditions.

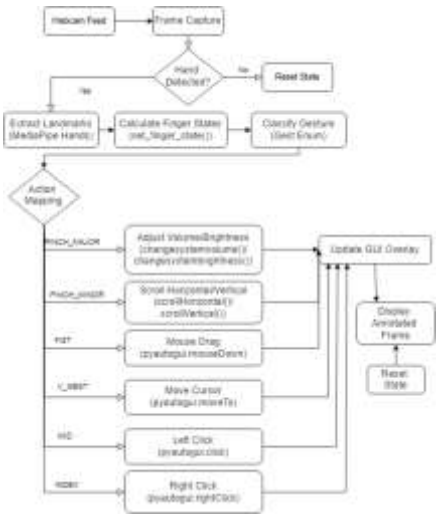

Fig. 1. Gesture Controller Module

IV. ALGORITHM

The processes in the algorithms implemented were set in a way that inputs from both video and audio sources would be processed efficiently. For gesture recognition, each webcam frame has to be preprocessed first (flipped and color converted), and then passed on to MediaPipe for hand landmark extraction. The different distances and ratios between the various landmarks will be calculated. The states of individual fingers will be computed accordingly. An enumeration-based mapping will be applied to link these computed metrics to specific gestures, such as a fist, palm, or pinch. Temporal smoothing is used to ascertain the consistency of gestures over several frames before applying the respective system action. The voice command processing part of the system has its dedicated thread that runs all the time, capturing audio through the microphone. Google Speech API provides a conversion of the audio into text, followed by parsing of

International Journal of Current Science Research and Review
ISSN: 2581-8341
Volume 08 Issue 05 May 2025
DOI: 10.47191/ijcsrr/V8-i5-57, Impact Factor: 8.048
IJCSRR @ 2025
www.ijcsrr.org

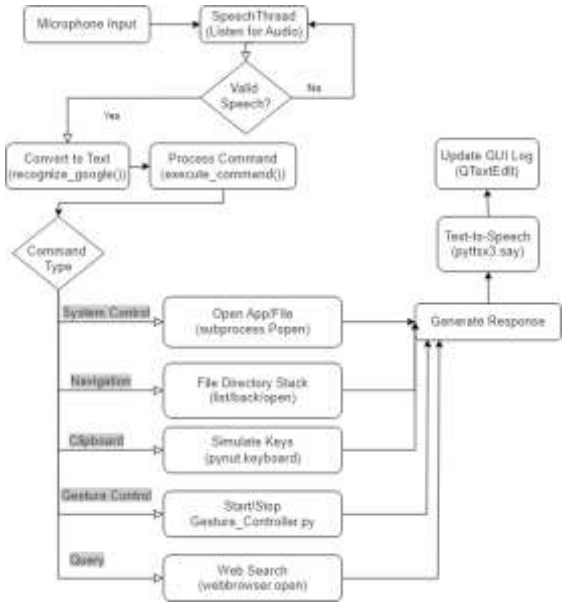

Fig. 2. Voice Assistant Module

V. EXPERIMENTAL SETUP

This section of the report focuses on the experiments performed to process, apply and understand the system's compatibility, effectiveness and interoperability over two major operating systems. The setup consists of two operating systems, macOS and Windows, which are used the most worldwide.

A. System Input and Data Selection

There are three major types of data input used here that are audio, video, and system metrics. The system metrics data is processed while the program starts and ends. This data in this project has a variety and thus needs to be handled properly. However, this is not the problem, as the data is taken by standard Python libraries and efficiently multiprocessed using the Go language.

The data presented in video format is taken through a webcam, which is already installed in the camera of the personal computer of the user. The mediapipe library is very efficient in this paper [9].

Similarly, the audio data is taken from the mic of the PC used by the user. The "SpeechRecognition" library is also very useful, as we can see in this paper here [10]. The Python text-to-speech library is very useful for converting and commanding as per the input given by the user.

The logs generated will be stored in a folder with the date and time, which can be accessed at any point in the future. Thus, the data presented has no issues related to the task of storing and using dynamic data for processing.

B. System Performance Metrics

During the performance test of the gesture recognition system utilizing varying commands, some essential properties became apparent. We monitored system ac- tivity throughout the following operations: running the binary 'go' 'go run rp.go', invoking the gesture detector via the command './gestureplus', executing '/python3 run_parallel.py', on both MacOS and Windows.

We can observe from Table I and II that the system properly handles the resource consumption in all the mentioned operations. As per the operating systems of MacOS and Windows, the overall logs generated and the actual recording of the data are different, as we can see from the table; however, this data does give a broad overview of the overall usage, through which we can extract relevant conclusions and also visualize the results.

International Journal of Current Science Research and Review
ISSN: 2581-8341
Volume 08 Issue 05 May 2025
DOI: 10.47191/ijcsrr/V8-i5-57, Impact Factor: 8.048
IJCSRR @ 2025
www.ijcsrr.org

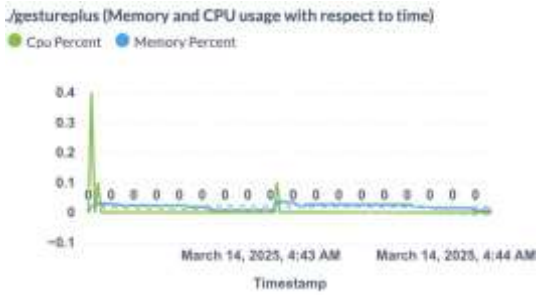

Fig. 3. CPU and Memory Usage Over Time for gestureplus

Fig. 4. CPU and Memory Usage Over Time for go run rp.go

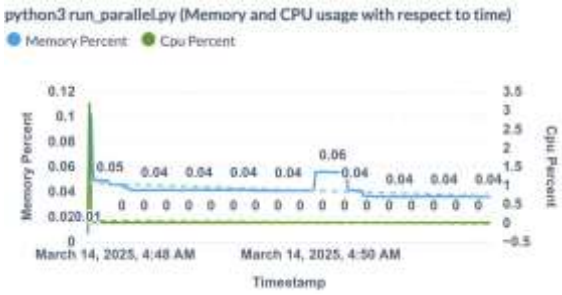

Fig. 5. CPU and Memory Usage Over Time for python3 run parallel.py

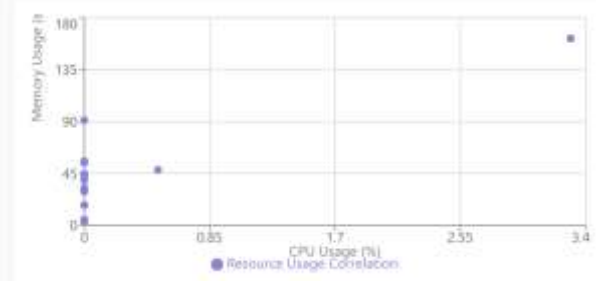

Fig. 6. Resource Usage Correlation

International Journal of Current Science Research and Review
ISSN: 2581-8341
Volume 08 Issue 05 May 2025
DOI: 10.47191/ijcsrr/V8-i5-57, Impact Factor: 8.048
IJCSRR @ 2025
www.ijcsrr.org

TABLE I. SYSTEM RESOURCE UTILIZATION DURING DIFFERENT OPERATIONS

System Resource Utilization During Different Operations						
Operation	***Peak CPU (%)***	***Avg CPU (%)***	***Peak Memory (MB)***	***Avg Memory (MB)***	***Peak RSS (MB)***	***Duration (s)***
go run rp.go	0.5%	0.0%	0.048	0.031	3.9	32
./gestureplus	0.5%	0.0%	0.048	0.029	2.7	701
run parallel.py	3.3%	0.1%	0.162	0.054	13.3	187

TABLE II. SYSTEM RESOURCE UTILIZATION DURING DIFFERENT OPERATIONS FOR MAC

System Resource Utilization During Different Operations						
Operation	***Peak CPU (%)***	***Avg CPU (%)***	***Peak Memory (%)***	***Avg Memory (%)***	***Peak RSS (MB)***	***Duration (s)***
go run rp.go	17.7	0.8	0.2	0.02	20.2	151
./gestureplus	0.1	0.0	0.035	0.02	2.8	125
run parallel.py	3.2	0.0	0.01	0.04	8.4	236

From the Fig. 6, we can clearly see that the CPU and memory utilization have been high and closely correlated with each other. This is due to the multithreaded nature of the application as well as the low-level implementation using executables. Also, the deviation in the graph could be seen, which could be explained as a one-time memory and cache loading of the application, which is an overhead cost that is accounted for while the program was started. Thus, frim this we can see that the overall system is running smoothly without any errors and also accounting all the task that are present efficiently without any problem.

From this experiment, we found out that the three files had varying effects on performance concerning their efficiency in CPU and memory utilization when it comes to both MacOS and Windows. The files, when run with the Go compiler, are the slowest of the three, with it taking huge memory and CPU overheads while being used, which in turn also leads to reduced performance and less efficiency. We can also see that after a while, the CPU and Memory consumption have increased. This could be due to the cores having to run first, go, then switching to the Python interpreter for the execution of the files.

The Python interpreter, although it was faster than the Go compiler, was still slower than the executable files. This could be due to the code already being written in the Python language, which could affect on its execution time and also its efficiency. However, the overheads with regard to CPU and memory were consistent; thus, the overall efficiency of the system was maintained at a steady rate rather than the approach we saw beforehand. The executable file was created by the "go build" command, which makes an executable file for any architecture that is possible. The executable, as it runs on the native, gives great results, which was up to 10 times more efficient than the Go compiler as per the data that was recorded for both these use cases. The CPU overhead on average was negligible, and the memory overhead was the least as per the data. Thus, the application was also optimized to its most efficient form through this process. Thus, we could see that in both macOS and Windows, the executables have been clearly best at making use of the native architecture while reducing the overheads and thus making a better overall system. Thus, we have also seen that compilers are slower than Python interpreters, but this could be due to the actual code being written in Python itself. Also, the memory and CPU overhead were low as the various Python libraries suited for the input and output were efficient and performant in both Python systems and caused no crash or system problems.

VI. METHODOLOGY

A. Design Details

The proposed architecture is highly modular and layered, separating core functionalities while providing seamless integration. There is no interference between the gesture module and its own class hierarchy, with HandRecog facilitating the processing of raw hand land- mark data with Controller mapping system commands to these gestures. The GestureController class initializes the video capture device and mediates the continuous processing loop. The voice-enabled chatbot module is developed using PyQt6, providing elements like the main window, conversation log, and interactive input. Multi- threading is done in such a way that there is one thread,

International Journal of Current Science Research and Review
ISSN: 2581-8341
Volume 08 Issue 05 May 2025
DOI: 10.47191/ijcsrr/V8-i5-57, Impact Factor: 8.048
IJCSRR @ 2025
www.ijcsrr.org

SpeechThread, that is in charge of audio processing. The interaction of these modules with OS-specific libraries is governed by some condition for requesting system-level operations from the exact APIs. With the help of the multiprocessing module in Python, it is possible to work gesture and voice modules together at the same time.

B. Methodology of proposed system

The methodologies employed for handling multi- modal interaction are based on a user-centric iterative modular mechanism. The development process embarks upon identifying end-user needs and problems, analyzing test runs for defining potential solutions, and selecting open-source libraries having various functions, particularly strong in computer vision processing, speech recognition, and system control. Single-module development and testing shall follow each module, which should allow continuous refinement and rapid troubleshooting. Because of its modular nature, this architecture greatly facilitates its maintenance and scalable capabilities; swapping out a module or setting about to upgrade or do fix-ups on one would not affect the others. Library-based parallel and asynchronous processing will be the general tenets of the methodology that allows easy handling of simultaneous inputs and a more fluid UI ex- perience, including multiprocessing and multithreading. The diverse library approach offers OS-based functional operations thanks to respective coding or conditional logic which runs volume and brightness controls on macOS, and Windows without affecting one another and with cross-platform usability. Overall, this structured way brings about a solid, extensible, and straightforward user interface that could keep pace with multifaceted human–computer interactions that are ever evolving.

VII. RESULTS

The findings reveal impressively low resource con- sumption for all operations, with CPU usage consistently remaining low (averaging from 0.0-0.1%) and only experiencing occasional spikes. The Python-based parallel execution had the highest resource demands, peaking at 3.3% CPU usage and 13.3 MB of memory, in contrast to the much lower requirements of the Go-based execution and the gesture recognition system.

A. Gesture Detection Performance

Observations from multiple test sessions reported in the monitoring files, that the gesture recognition system has enabled consistent and efficient performance again reaffirming our work in human-computer interaction. In examining the various CPU resource metrics, the average CPU consumption was found to be remarkably small, generally below 1%, with minor instances in which short spikes occurred repeatedly, confirming the negligible overhead in running the framework. In the same manner, the memory usage was consistently resource-efficient, with average utilization less than 0.2% and with peak usage not exceeding 0.3%, ensuring that the system could run evenly. These results from the monitoring data were very strong in supporting the claim that such a system highly reliably detects exit gestures while being resource efficient statically, thus validating its application in real-time scenarios.

B. Interpretation of Resource Usage Patterns

CPU sustained the very low usage (still under 0.1% on average) throughout all recordings and test runs, indi- cating our implementation's lightweight character. Such efficiency is epitomized, through its gesture recognition system, running with minimum resource usage while undertaking highly complex computer vision tasks. This concurs with findings from [6], who underscored the crucial importance of computational efficiency in gesture recognition systems.

The memory usage patterns show an optimized implementation, with the resource-intensive operation consuming only 13.3 MB at peak. This makes our solution deployable in resource-constrained devices. In other words, it addresses environmental dependencies on gesture recognition systems.

C. Comparative Analysis with Existing Solutions

Some important insights can be derived by comparing the performance metrics of our system against existing solutions in the literature. As per our literature survey, our approach keeps pace with the most recent improvements and thus addresses issues with key limitations:

Our system uses significantly fewer resources than other gesture recognition implementations. [1] noted that developing interfaces involving real-time fingertip gestures run into considerable sensor noise and variable illumination conditions. Our implementation can manage this difficulty amid the much-elevated minimal resource use. The long active time taken for

International Journal of Current Science Research and Review
ISSN: 2581-8341
Volume 08 Issue 05 May 2025
DOI: 10.47191/ijcsrr/V8-i5-57, Impact Factor: 8.048
IJCSRR @ 2025
www.ijcsrr.org

executing the gesture recognition component (701 seconds) indicates capable, stable long-running performance, which is important in continuous interaction scenarios, just as in human- machine interaction systems reviewed by [2] in the area of hand gesture recognition technologies.

Unlike the other approaches discussed by [6] in their survey on computer vision-based approaches for hand gesture recognition, our framework incorporates com- puter vision techniques for gesture recognition while maintaining low resource consumption. In gesture-based control work of PowerPoint presentations in [3], signifies the vast expanding ambit of applications where gesture recognition can fit in. Given the resource efficiency of our framework, it may be applicable in such application settings, with an expected level of performance.

VIII. LIMITATIONS AND FUTURE WORK

- The system produced deprecation warnings related to AVCaptureDeviceTypeExternal and TensorFlow feedback sensor issues. These should be addressed in future versions to ensure continued compatibility and optimal performance.
- Although resource utilization is already minimal, further optimizations could be explored, particularly for the Python-based parallel execution, which showed higher resource demands compared to other components.
- Building on the work of [3], future implementations could extend our gesture recognition capabilities to control additional applications beyond the current scope, such as presentation control or virtual key- board input.
- Additional testing across diverse hardware configurations would help verify the consistent performance of our system in varied deployment environments.
- Given the advancements in natural language processing shown in our literature survey (e.g., [4]), future work could explore multimodal interaction combining gesture recognition with conversational interfaces for more intuitive human-computer interaction.

IX. CONCLUSION

GesturePlus introduces a whole new way of human- computer interaction through gesture recognition, voice commands, and a chatbot for an entirely hands-free, seamless experience on MacOS and Windows. Using advanced machine learning, computer vision, natural language processing, and the RAG model, it also improves accessibility, usability, and adaptability across various aspects. The scalable, modular design allows for seam- less switching between various interaction modes, suiting industries such as healthcare, industrial automation, smart homes, and assistive technology. GesturePlus flips conventional paradigms on their heads, both enhancing user experience and efficiency and, most importantly, opening the door to inclusivity by providing alternative input opportunities for persons suffering from mobility impairments.

REFERENCES

1. 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*, vol. 80, no. 7, pp. 10473–10490, Nov. 2020. doi: https://doi.org/10.1007/s11042-020-10156-5. Available: http://sclab.jnu.ac.kr/wp-content/uploads/2021/03/Tran2021Article Real-timeVirtualMouseSystemUsi.pdf.
2. L. Guo, Z. Lu, and L. Yao, “Human-Machine Interaction Sensing Technology Based on Hand Gesture Recognition: a Review,” *IEEE Transactions on Human-Machine Systems*, vol. 51, no. 4,pp. 300–309, Aug. 2021. doi: https://doi.org/10.1109/thms.2021.3086003.
3. M. J. Vidya, S. Vineela, P. Sathish, and A. S. Reddy, “Gesture- Based Control of Presentation Slides using OpenCV,” *IEEE*, Aug. 2023. doi: https://doi.org/10.1109/icaiss58487.2023.10250520.
4. B. Kohli, T. Choudhury, S. Sharma, and P. Kumar, “A Platform for Human-Chatbot Interaction Using Python,” *IEEE Xplore*, Aug. 01, 2018. doi: https:\//doi.org/10.1109/ICGCIoT.2018.8753031. Available: https://ieeexplore.ieee.org/abstract/document/8753031. [Accessed: Jun. 21, 2021].
5. R. Dudhapachare, M. Awatade, P. Kakde, N. Vaidya, M. Kapgate, and R. Nakhate, “Voice Guided, Gesture Controlled Virtual Mouse,” *IEEE Xplore*, May 2023. doi: https://doi.org/10.1109/ incet57972.2023.10170317.
6. M. Oudah, A. Al-Naji, and J. Chahl, “Hand Gesture Recognition Based on Computer Vision: A Review of Techniques,”

International Journal of Current Science Research and Review
ISSN: 2581-8341
Volume 08 Issue 05 May 2025
DOI: 10.47191/ijcsrr/V8-i5-57, Impact Factor: 8.048
IJCSRR @ 2025
www.ijcsrr.org

Journal of Imaging, vol. 6, no. 8, p. 73, Jul. 2020. doi: https://doi.org/10. 3390/jimaging6080073.

7. D. Sarma and M. K. Bhuyan, "Methods, Databases and Recent Advancement of Vision-Based Hand Gesture Recognition for HCI Systems: a Review," *SN Computer Science*, vol. 2, no. 6, Aug. 2021. doi: https://doi.org/10.1007/s42979-021-00827-x.
8. E. Dinan, S. Roller, K. Shuster, A. Fan, M. Auli, and J. Weston, "Wizard of Wikipedia: Knowledge-Powered Conversational agents," *arXiv:1811.01241 [cs]*, Feb. 2019. Available: https://arxiv.org/abs/1811.01241.
9. C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja,M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, and M. Grundmann, "MediaPipe: A Framework for Building Perception Pipelines," *arXiv preprint arXiv:1906.08172*, 2019.
10. A. Dhakad and S. Singh, "Python-Powered Speech-to-Text: A Comprehensive Survey and Performance Analysis," *International Journal of Engineering Research & Technology (IJERT)*, vol. 12, no. 09, Sep. 2023.

Cite this Article: Lade, S., Shaikh, U.A., Sahane, A., Pednekar, K., Wankhade, A. (2025). Gesture Plus: A Novel Approach to Enhance Interactive Media. International Journal of Current Science Research and Review, 8(5), pp. 2472-2480. DOI: https://doi.org/10.47191/ijcsrr/V8-i5-57