Gesture Controlled Media Player Using MediaPipe for Hand Gesture Recognition and Python-VLC for Media Playback

John Maverick Robias
College of Computer Studies
Technological Institute of the Philippines
Quezon City, Philippines
qjmbrobias@tip.edu.ph

Abstract—**This project explores the implementation of gesture control for a customized VLC media player using MediaPipe for hand gesture detection and Python-VLC for media control. The system allows users to control media playback through hand gestures, including play/pause, volume adjustment, fast-forward, rewind, and track navigation. The gesture recognition system utilizes real-time hand tracking and landmark detection, translating hand movements into commands with high accuracy. Performance evaluations indicate robust gesture detection accuracy under optimal lighting conditions, with slight performance degradation under low lighting and lens flare. This paper demonstrates the potential of gesture-based interfaces to enhance user interaction with multimedia applications, reducing reliance on traditional control mechanisms and paving the way for more intuitive human-computer interactions.**

Keywords—Gesture control, MediaPipe, Hand gesture recognition, Python-VLC, Human-computer interaction, Real-time video processing, Media player, Computer vision, Gesture-based interface, User experience, Multimedia control, Volume control, Fast-forward, Rewind.

I. Introduction

Computer vision is a field in computer science focused on enabling computers to identify and gather information from images. With the continuous advancement of technology, computer vision has surged in popularity due to its wide range of applications, including image classification, object detection, and face recognition.

Gesture recognition is a subset of computer vision that allows machines to interpret human gestures, which can be used as inputs or shortcuts to command applications to perform tasks. As this field advances, our interactions with computers will evolve to become more efficient [1]. Similar to how touch screens replaced keypads on phones, gesture recognition could revolutionize how we interact with technology.

This project aims to develop a gesture-controlled application to demonstrate the potential of using hand gestures to execute commands. By implementing this technology in one of the most widely used applications on our devices—the media player—the project aims to create a gesture-controlled media player application that enhances the user experience and reduces reliance on traditional control mechanisms.

II. Review of Related Studies

Gesture recognition has been extensively explored as a means to control and command applications, performing tasks and executing commands in innovative ways. This technology has been applied in various fields, each leveraging unique methods to integrate gestures as inputs.

For instance, the study conducted by Lokesh S. Khedekar and Manas Patil explores the use of gesture commands to adjust and manage volume control systems, providing an immersive and captivating alternative to traditional volume control methods. By utilizing the OpenCV library for real-time video capture, machine learning models to detect hands, and the MediaPipe library to recognize specific gestures, this study builds on established techniques. It addresses common challenges, such as lighting variability and computational limitations, to deliver an intuitive and user-friendly interaction framework [2].

Furthermore this can also be seen in the study conducted by Yash Gajanan Pame and Vinayak G. Kottawar explores the use of gesture commands to control mouse movements, providing an intuitive and natural alternative to traditional input devices. By utilizing the OpenCV library for real-time video capture and the MediaPipe library for hand detection and tracking, this study builds on established techniques in gesture recognition. It addresses common challenges, such as mouse pointer displacement and abrupt hand movements, by introducing an algorithm that calculates the new pointer position based on the movement of the hand rather than its absolute location. This approach ensures smoother cursor movement and enhances the overall user experience [1].

However, gesture recognition also faces its own set of challenges. Research conducted by B. Kareem Murad and A. H. Hassin Alasadi stated that gesture recognition struggles with handling scenarios that have lighting variations and complex backgrounds. Additionally, the method struggles with noise management and has issues performing in real-time processes. These limitations hinder its broader adoption as a reliable method of input [6].

In conclusion, gesture recognition has proven to be an innovative way to control applications. Studies by Lokesh S. Khedekar and Manas Patil, as well as Yash Gajanan Pame and Vinayak G. Kottawar, illustrate the potential of using gestures for volume control and mouse movement. Building on these, this project aims to create a gesture-controlled media player that enhances user experience and reduces reliance on traditional controls.

III. Methodology

The utilization of gestures as an alternative for media player input involves several key steps, including selection of tools and technologies, development of gesture recognition algorithms, implementation of gesture-to-command mapping, and integration with the media player.

A. System Development

To achieve the desired output the project was divided into two modules: Hand Gesture Detection Module and Media Control Module. The Hand Gesture Detection Module is responsible for capturing and interpreting hand movements, this is done by tracking handlandmarks dictated by the hand tracking module and mapping distinct poses as inputs. Leveraging the capabilities of Mediapipe for robust hand detection and gesture recognition [3].

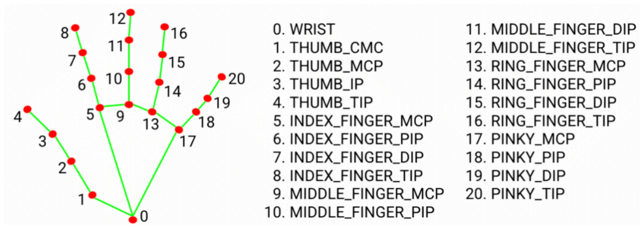


Fig. 1. Hand Landmark Regions

The system also utilizes OpenCV for real-time video processing and by leveraging both libraries, it ensures that gestures are accurately identified and translated into actionable commands.
On the other hand the Media Control module encapsulates the execution of these commands within the media player. By utilizing python-vlc for media playback functionalities, this module allows for seamless control over media operations such as play, pause, volume adjustment, and track navigation. Additionally, Tkinter is used to build a graphical user interface (GUI) that provides users with a visually intuitive way to interact with the media player.

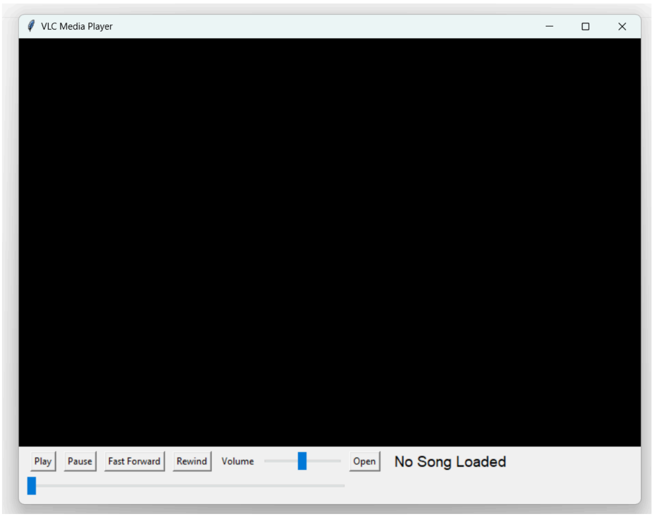


Fig. 2. Media Player GUI

B. Algorithm Design

Traditional control mechanisms, such as those found on remote access devices, offer a wide range of commands through their numerous and varied buttons. To replicate this versatility, the algorithm is designed to support four distinct modes, each with its own unique functionality and control scheme. This ensures that the gesture-controlled media player can effectively mimic the comprehensive set of commands available through traditional control mechanisms, providing an intuitive and seamless user experience.

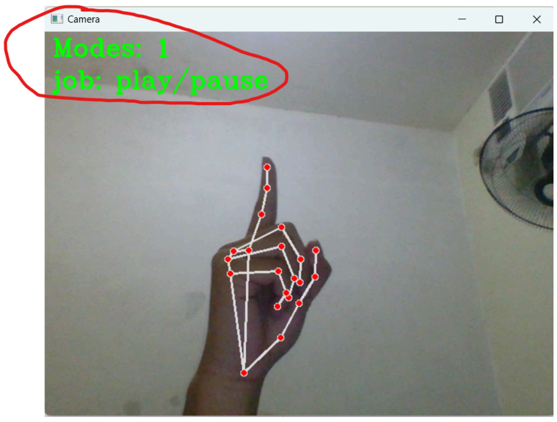
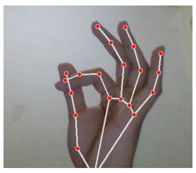
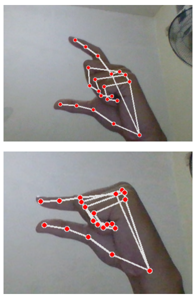
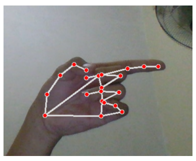
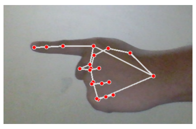
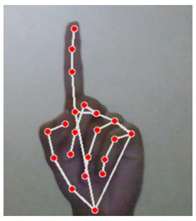


Fig. 3. Media Player Modes

The mode status will be displayed at the top left of the camera screen, controlled by the user's left hand. Each raised finger will change the mode, with a maximum of four modes available. Additionally, each mode will have specific "jobs" or commands that can be executed based on the active mode. This approach ensures a clear and intuitive interface, allowing users to seamlessly switch between modes and perform various commands using simple hand gestures.

Table I. Available Modes and Functionalities

Mode	Function	
1	**Play/Pause:** Ability to play or pause the media player.	
2	**Volume Control:** Ability to adjust the volume of the media player.	
3	**Skip/Rewind:** Ability to skip forward or	

		
4	**Load Song:** Ability to load a new song.	

User controls are executed through the user's right hand and interaction with the media player involves gestures such as pinching, pointing, raising, and intersecting fingers. To detect these gestures, we utilize the landmarks plotted by the Mediapipe model as reference points. We then apply mathematical equations, such as the Euclidean distance formula:

$$d = \sqrt{(x_2 - x_1)^2 + (y_1 - y_2)^2}$$

This formula helps determine the distance between two points on a graph. By calculating the distance between the tips of the thumb and index finger during a pinching motion, we can provide a value that adjusts the volume accordingly. Furthermore, we can apply another mathematical concept known as the cross product method to determine if two line segments intersect. This is done by considering the orientation of line segment AB to line segment CD. By computing the orientation of these triplets (A,B,C), (A,B,D), (C,D,A), and (C,D,B) and by applying this formula:

$$Orientation(P_1, P_2, P_3) = (P2[0] - P1[0])\cdot(P3[1] - P1[1]) - (P_2[1] - P_1[1])\cdot(P_3[0] - P_1[0])$$

This allows us to determine if the line segments intersect by checking the condition: If the orientations of (A,B,C) and (A,B,D) are different, and the orientations of (C,D,A) and (C,D,B) are also different, then the line segments intersect.

C. Evaluation Metrics

To thoroughly evaluate the performance of the system, a comprehensive series of tests will be conducted. These tests aimed to rigorously assess various aspects of the system's functionality and reliability, ensuring that all components work seamlessly together and meet the project's specified requirements.

(1) Functional Testing - This testing involves evaluating Gesture Detection Accuracy by subjecting the program to various lighting conditions to assess the system's accuracy and reliability. By testing in different environments, we aim to ensure that the system consistently recognizes and interprets gestures accurately, regardless of lighting variations. Additionally, the test includes validating the Media Player Integration to verify that gestures are correctly mapped to the appropriate commands, ensuring seamless interaction and functionality.

(2) Performance Testing - This testing includes evaluating the program's capability to handle simultaneous inputs, specifically focusing on the smooth transition between modes and the execution of available commands. The aim is to ensure that the program remains stable and efficient even when switching modes while performing actions such as raising the volume, pausing and playing, or fast-forwarding and rewinding. By assessing these aspects, we aim to ensure that the system performs seamlessly under various conditions and remains scalable for future expansions. This comprehensive evaluation helps identify potential performance bottlenecks and ensures that the system can manage increased workloads and more complex interactions without compromising efficiency or reliability.

IV. Results

From the functional testing conducted, the following findings were observed:

Table II. Gesture Detection Accuracy

Conditions	play/pause	volume	ff/rewind	load
normal lighting	96%	92%	96%	100%
low lighting	84%	82%	80%	88%
lens flare	66%	60%	68%	70%

To compute the values found within the table the proponent utilized accuracy formula:

$$Accuracy = \frac{Correctly\ Recognized\ Gestures}{Total\ Gestures\ Performed} \times 100$$

The table demonstrates the accuracy of gesture recognition for a media player under different lighting conditions: normal lighting, low lighting, and lens flare. Each condition was tested across 50 iterations, evaluating the system's ability to perform play/pause, volume adjustment, fast-forward/rewind, and load functions. Under normal lighting, the system performs exceptionally well, with accuracies ranging from 92% to 100%, indicating optimal performance in ideal conditions. In low lighting, the accuracy decreases slightly, with play/pause at 84%, volume at 82%, fast-forward/rewind at 80%, and load at 88%, showing a moderate impact. The most significant drop is observed under lens flare conditions, where accuracies range from 60% to 70%, highlighting the system's difficulty in handling intense lighting interference.

To test Media Player Integration the proponent utilized error rate formula:

$$Error\ Rate = (\frac{Number\ of\ Incorrect\ Commands}{Total\ Commands\ Tested}) \times 100$$

This metric helps assess how often the system fails to correctly map gestures to the intended commands. which resulted in the following findings:

Table III. Media Player Integration Error Rate

Command	Error Rate
play/pause	4%
volume	8%
ff/rewind	4%
load	0%

The table shows the error rates for gesture-controlled media player commands. Fast-forward/rewind and load commands have a perfect error rate of 0%, indicating flawless recognition. Play/pause has a 4% error rate, while volume shows an 8% error rate, highlighting some inconsistencies. Overall, the system performs well, with room for improvement in play/pause and volume gestures.

For the performance testing, the proponent evaluates how accurately the system detects and switches modes while a gesture is in progress. Furthermore, the proponent also checks if the command is successfully executed even as the mode is switched. The following are the formulas used to compute the score:

$$Mode\ Switching\ Accuracy = (\frac{Correct\ Mode\ Switches}{Total\ Commands\ Attempted})\ x\ 100$$

$$Command\ Execution\ Success = (\frac{Correct\ Commands\ Executed}{Total\ Commands\ Attempted})\ x\ 100$$

which resulted in the following findings:

Metric	Accuracy Rate
Mode Switching Accuracy	90%
Command Execution Success	98%

The table indicates that the system achieves a 90% accuracy in switching modes during gestures and a 98% success rate in executing commands such as volume adjustments or play/pause during mode transitions. These results demonstrate strong performance. However, the proponents observed that mode switching accuracy diminished due to errors within the hand detection model. Specifically, when both hands intersect or intermingle, the model occasionally fails to correctly identify which hand is the right or left, resulting in mistaken mode switches. This issue is also encountered in the gesture recognition accuracy test, particularly in low lighting conditions. The model struggles to detect fingers due to the low luminosity of the room, causing the landmarks of the undetected fingers to move sporadically and leading to misinputs.

V. Conclusion and Recommendations

The project successfully demonstrated the potential of gesture control in a customized VLC media player application. By utilizing MediaPipe for hand tracking and gesture detection, along with real-time media control through Python-VLC, the system efficiently integrates hand gesture recognition controls into a media player. Various tests resulted in a high gesture detection accuracy rate of 92% to 100%, showcasing the viability of using gestures as commands. Furthermore, performance testing showed that the system achieves a 90% accuracy in mode switching and a 98% success rate in executing commands, even during mode transitions. These results demonstrate a robust system capable of handling complex interactions efficiently.

However, there are still challenges that need to be addressed to further improve the system. Issues such as hand detection inaccuracies during hand intersections, difficulties in low lighting conditions, and detection issues due to light interference need to be resolved. Therefore, further research is necessary to enhance the system’s performance and reliability for practical use.

To further enhance the developed system, it is recommended to expand testing with a more diverse set of testing metrics, more test instances, and complex conditions like varied lighting and environmental conditions. Consider incorporating machine learning to improve gesture recognition and utilizing high-definition cameras to further optimize accuracy and responsiveness. Additionally, expanding the media player’s functionality and supporting more formats would increase versatility, ensuring the system's robustness and practicality for broader real-world applications.

References

[1] Yash Gajanan Pame and V. G. Kottawar, “A Novel Approach to Improve User Experience of Mouse Control using CNN Based Hand Gesture Recognition,” Aug. 2023, doi: https://doi.org/10.1109/iccubea58933.2023.10392164.

[2] L. S. Khedekar and M. Patil, “Gesture-Based Volume Control Using Computer Vision and Audio Processing,” Apr. 2024, doi: https://doi.org/10.1109/iccsp60870.2024.10543441.

[3] “Gesture recognition task guide | Google AI Edge,” Google for Developers. https://ai.google.dev/edge/mediapipe/solutions/vision/gesture_recognizer

[4] Computer Vision Applications - javaTpoint. (n.d.). Www.javatpoint.com. https://www.javatpoint.com/computer-vision-applications

[5] Christian, S.-C., Dan, G., Alexandra, F., Adela, Ovidiu, S., Valean Honoriu, & Miclea Liviu. (2022). Hand Gesture Recognition and Infrared Information System. 113–118. https://doi.org/10.1109/iccc54292.2022.9805963

[6] B. Kareem Murad and A. H. Hassin Alasadi, “Advancements and Challenges in Hand Gesture Recognition: A Comprehensive Review,” Iraqi Journal for Electrical and Electronic Engineering, vol. 20, no. 2, pp. 154–164, Jul. 2024, doi: https://doi.org/10.37917/ijeee.20.2.1