Journal of Transactions in Systems Engineering
https://journals.tultech.eu/index.php/jtse
ISSN: 2806-2973
Volume 2, Issue 2
DOI: https://doi.org/10.15157/jtse.2024.2.2.256-264
Received: 04.07.2024; Revised: 16.08.2024; Accepted: 30.08.2024

Technical Paper

Real-Time Volume Control Using OpenCV

Amol Rakh*, Vaishnavi Dherange, Vivek Shelke, Siddhesh Londhe, Pragati Akhare*

Department of Artificial Intelligence and Data science, Ajeenkya D.Y Patil School of Engineering, Pune, India.

***amol.rakh@dypic.in, pragati.akhare@dypic.in**

Abstract

This paper introduces a novel method for controlling computer volume through hand gestures using OpenCV, a computer vision library. Instead of traditional input methods like buttons or a mouse, users can adjust volume levels by waving their hands in front of a webcam. OpenCV tracks these hand movements and interprets them in real-time, providing accurate and responsive volume control. Our experiments demonstrate the effectiveness of this approach, offering an intuitive solution that enhances user interaction, particularly for individuals with mobility impairments. Furthermore, OpenCV's compatibility with various platforms and programming languages increases the system's versatility. This research advances gesture-based interaction techniques and underscores the potential of computer vision technology to improve user experience across different applications, making computing more interactive and accessible.

Keywords: OpenCV; gesture control; mediapipe; paycaw volume control; numpy.

INTRODUCTION

In today's digital era, controlling electronic devices has become an integral part of our daily life. Traditionally, volume control on computer and audio devices has relied on physical buttons or software interfaces [1-4]. However, advancements in technology have opened up new possibilities for more intuitive and interactive control methods. This research explores the implementation of volume control using Open Source Computer Vision Library (OpenCV), a powerful computer vision library, to track hand gestures and adjust audio levels without the need for physical input devices.

Traditional volume control mechanisms typically involve physical buttons, sliders, or software-based interfaces [5-7]. While functional, these methods may not always be convenient or accessible, particularly in situations where users have limited mobility or are engaged in hands-on tasks. Additionally, with the increasing popularity of voice-controlled assistants and smart home devices, there is a growing demand for more natural and intuitive interaction methods [8-10].

To implement volume control using OpenCV, we utilized a standard webcam connected to the computer. The OpenCV library was employed to process the video feed from the webcam, enabling real-time hand detection and gesture recognition. Various image processing techniques, including background subtraction, contour detection, and

convex hull analysis, were applied to identify and track the user's hand movements accurately. Once the hand gestures were recognized, corresponding volume adjustment commands were sent to the computer's audio system.

METHODOLOGY AND SIMULATION RESULTS

OpenCV

In our real-time volume control research work, OpenCV plays a crucial role, particularly in the following areas that we have used in our methodology:

- Gesture Recognition: OpenCV has been used to detect and track hand gestures. By identifying specific gestures, such as raising or lowering your hand, the system can interpret these movements as commands to increase or decrease the volume.
- Object Detection: OpenCV has been utilized to detect specific objects, like a hand or fingers, within the video stream. The position and movement of these objects can then be used to adjust the volume accordingly.
- Face or Hand Tracking: In our case to control the volume based on the position or movement of your face or hands, OpenCV's tracking algorithms has been used to follow these features and determine the appropriate volume level.

In essence, OpenCV is employed to process the video input, recognize or track relevant features, and then translate those visual cues into volume control commands.

Algorithm

In the following steps we have implemented the algorithm for real time volume control by using OpenCV.

1. initialize frame size

   640*480, and take camera acces using cv2.VideoCapture(0)

2. by using img.shape function we get landmark co-ordinate

   x and y, also we give id to each landmark using enumerate function

   i.e. making list of x, y and id lmlist = [id, x , y]

3. using x and y co-ordinate we plot dots on hand in frame

   with radius 7 and color (255, 0, 0) i.e. Red.

4. next we have performed imp operation for real time fps calculation with the help of current time and previous time, see equation (1),

   fps = 1 / (cTime - pTime) (1)

   where, value of fps is display in frame using cv2.putText() function,

5. now initialize volume min and max range 0 and 1 respectively

6. we have taken landmark id and location co-ordinate which we use for controlling volume

x1, y1 = lmList[4][1],lmList[4][2] (landmark of tip of thumb)

x2, y2 = lmList[8][1],lmList[8][2] (landmark of tip of index finger)

7. draw a line segment between this two-landmark using function as follows:

cv2.line()

then we can calculate length of line using math function, see equation (2)

length = math.hypot(x2-x1, y2-y1) (2)

8. with respective length of line we set volume using numpy

vol = np.interp(length, [18, 170], [minVol, maxVol])

and get volume.SetMasterVolumeLevel(vol, None)

9. these entire steps is in infinite loop for reading frame.

System Architecture

The volume control system using OpenCV consists of several key components: a webcam for capturing video input, a computer running the, and the audio system for adjusting volume levels. The system architecture can be divided into three main stages: hand detection, gesture OpenCV library for image processing recognition, and volume control.

Equations (3) and (4) have been used to measure distance between index figure tip and thumb tip. Also, according to distance volume range is set with the help of OpenCV Python Packages.

$$\text{Length} = \sqrt{(x2 - x1)^2 + (y2 - y1)^2} \quad (3)$$

$$\text{length} \propto \text{volume} \quad (4)$$

Furthermore, equation (4) indicate length between fingers is directly proportional to the device volume.

Modules

1. Video capture module:

The video capture module is a part of devices like computers and smartphones that lets them record video from sources like webcams. It works by capturing frames of video, converting them into digital data, and allowing users to record, stream, or view the footage. This module includes components like a lens, image sensor, and processing circuitry. Users can control settings such as resolution, frame rate, and exposure through software

interfaces. Overall, the video capture module enables devices to capture video content, making it possible for users to create, share, and enjoy multimedia experiences.

2. Pre-processing module:

The pre-processing module is a part of software or hardware systems that prepares data for further processing by cleaning, enhancing, or transforming it. It includes various techniques like filtering, normalization, and feature extraction. For example, in image processing, preprocessing might involve adjusting brightness, removing noise, or resizing images. In natural language processing, it could involve tokenization, removing stop words, or stemming. Preprocessing helps improve the quality and usability of data, making it easier to analyze and extract useful information. Overall, the preprocessing module plays a crucial role in preparing data for tasks like machine learning, pattern recognition, and data analysis.

3. Hand detection module:

The hand detection module is a part of software systems that identifies and locates human hands in images or video footage. It uses algorithms to analyze the visual characteristics of hands, such as color, shape, and motion. This module helps in applications like gesture recognition, virtual reality, and human-computer interaction. By accurately detecting hands, it enables devices to interpret hand movements and gestures, allowing users to interact with technology in intuitive ways. Overall, the hand detection module plays a vital role in enabling computers and devices to recognize and respond to human hand movements in various applications and scenarios.

4. Gesture recognition module:

The gesture recognition module is a component of software systems that identifies and interprets human gestures, such as hand movements or body gestures, captured through cameras or sensors. It uses algorithms to analyze patterns in the captured data and classify them into predefined gestures. This module enables devices to understand and respond to user gestures, allowing for intuitive interaction in applications like gaming, virtual reality, and human-computer interfaces. By recognizing gestures accurately, it enhances user experience and enables more natural and immersive interactions with technology. Overall, the gesture recognition module facilitates seamless communication between humans and computers through intuitive gestures.

5. Volume adjustment module:

The volume adjustment module is a feature found in audio systems that allows users to control the volume of sound output, see Figure 1. It typically consists of controls like sliders, buttons, or software interfaces that enable users to increase or decrease the volume level. Some systems may also include features like mute buttons or presets for different audio profiles. The volume adjustment module is essential for customizing audio playback to suit individual preferences and environments, providing users with a comfortable listening experience.

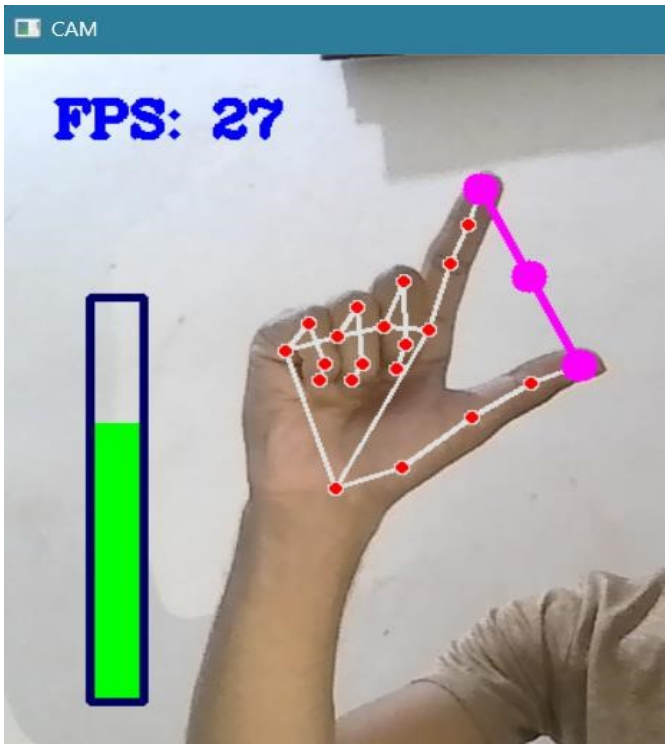
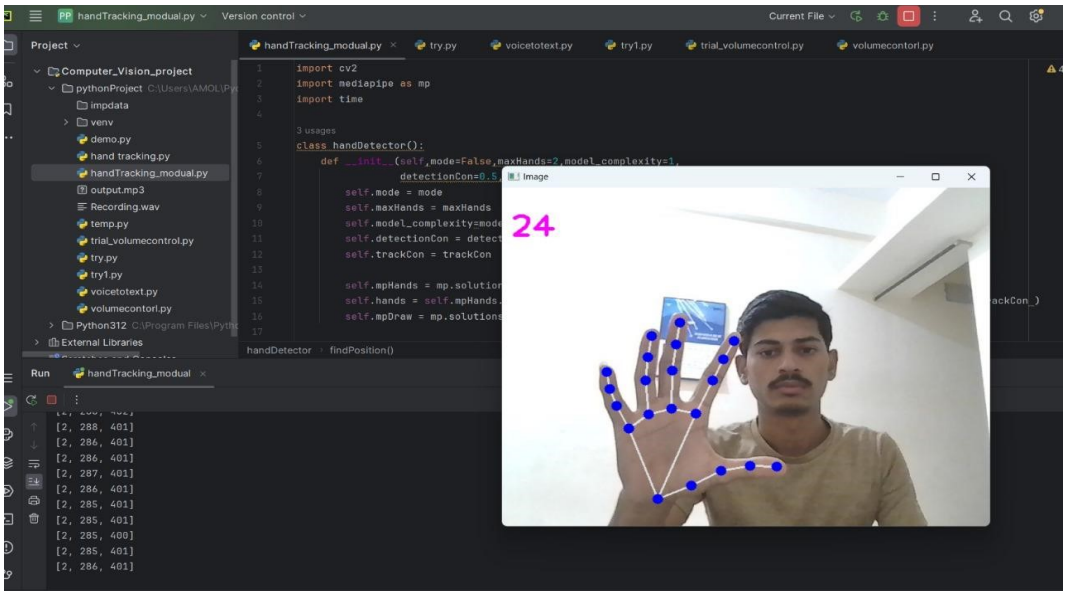


Figure 1. Volume Adjustment Module

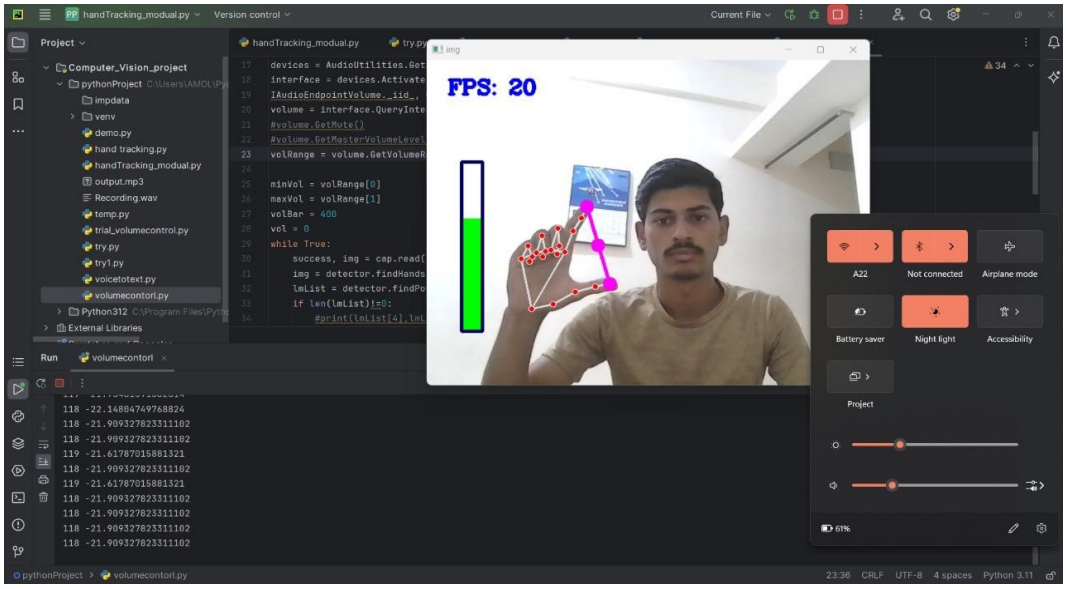
In the existing system, the biggest problem we face is that when the camera captures a random hand gesture involving the index and thumb fingers, it adjusts the volume without the user's intention. To address this, we propose a new system where the volume can only be adjusted when a fixed hand position is captured by the camera, allowing the user to control the volume as desired. Based on it, from the image above, the three fingers (little, ring, and middle fingers) are folded down towards the palm, while the other two (thumb and index fingers) are positioned upward to adjust the system's volume.

6. Continuous loop module:

The continuous loop module is a component of software or hardware systems that repeats a sequence of instructions or data continuously without interruption. It allows tasks, such as playing music or displaying animations, to repeat indefinitely until manually stopped. This module is commonly used in applications like media players, presentations, and simulations to create seamless and repetitive experiences. By looping content continuously, it ensures consistent playback or display, enhancing user engagement and providing a continuous flow of information or entertainment. Figure 2 shows the simulation results of hand detection using the Mediapipe Python package, along with the volume adjustment function, which operates based on the fingertip landmarks.



(a)



(b)

Figure 2. Simulation results; (a) It shows hand detection using the Mediapipe Python package, which enables the detection of landmarks on the human hand. Here, all points on the hand are defined as landmarks and will be used for further functions. (b) The volume adjustment function operates based on the fingertip landmarks. The green-coloured container indicates the volume adjustment range

Limitations for Volume Control Using OpenCV

1. Lighting Sensitivity:

- Limitation: OpenCV's performance can degrade in low-light or uneven lighting conditions.

- Impact: Gesture recognition and object detection may become less accurate, leading to incorrect volume adjustments or failure to detect gestures altogether.

2. Complex Backgrounds:

- Limitation: OpenCV may struggle to distinguish hand gestures from complex or cluttered backgrounds.
- Impact: The system might falsely detect objects in the background as gestures, causing unintended volume changes.

3. Occlusion:

- Limitation: If the user's hand is partially obscured or occluded, OpenCV may fail to recognize the gesture correctly.
- Impact: This can result in missed inputs or incorrect gesture detection, affecting the user experience.

4. Processor Demand:

- Limitation: Real-time video processing for gesture recognition can be computationally intensive.
- Impact: On lower-end hardware, this may lead to lag or reduced frame rates, making the system less responsive.

5. Limited Gesture Recognition:

- Limitation: OpenCV's gesture recognition capabilities may be limited to basic gestures.
- Impact: The system may not support more complex gestures or may require significant customization to recognize a broader range of inputs.

6. Dependency on Camera Quality:

- Limitation: The quality and resolution of the camera significantly affect OpenCV's ability to detect gestures accurately.
- Impact: Lower-quality cameras may result in poorer gesture detection performance, especially in terms of precision and responsiveness.

DATA AVAILABILITY

For a project on volume control using OpenCV, data availability is crucial for training and testing machine learning models, as well as for evaluating the effectiveness of the volume control in algorithm. Here are some considerations regarding the data availability:

1. Training the Data: High-quality datasets containing images or videos of individuals interacting with volume control interfaces are essential for training computer vision models. These datasets may need to be collected or sourced from existing repositories.

2. Audio Data: In addition to visual data, audio recordings of individuals speaking or making gestures to control volume are necessary for training and testing the volume control algorithm. These recordings should cover a variety of voices, accents, and environmental conditions.

3. Labelling's: Data labeling is important for supervised learning tasks, such as training a model to recognize hand gestures or facial expressions associated with volume control. Manual annotation or crowdsourcing may be required to label the data accurately.

4. Privacy Considerations: When collecting or using data containing images or audio recordings of individuals, it's important to respect privacy rights and comply with relevant regulations, such as GDPR or HIPAA. Anonymizing or de-identifying personal data can help mitigate privacy risks.

5. Open Source Datasets: Look for existing open datasets that may be suitable for the project, such as datasets of hand gestures or facial expressions. Websites like Kaggle, Open Images, or OpenCV datasets may have relevant resources.

6. Simulated Datasets: If real-world data is scarce or difficult to obtain, consider generating synthetic data using simulation tools or graphics software. Simulated data can help augment training datasets and improve model generalization.

7. Data Splitting: Divide the available data into training, validation, and testing sets to evaluate the model's performance accurately and avoiding overfitting.

8. Data Sharing: If possible, consider sharing the collected or generated data with the research community to facilitate further research and development in field.

CONCLUSION

In conclusion, using OpenCV for volume control through hand gestures presents a user-friendly and innovative method for adjusting audio levels. By recognizing gestures accurately, this technology offers an intuitive and hands-free interaction with audio devices, enhancing user experience. Its adaptability across various platforms and potential for further development make it a promising solution for improving accessibility and usability in computing. Moving forward, continued research and refinement of this technology will contribute to the advancement of gesture-based interaction in everyday computing tasks. Our future research work will be focused on the effectiveness of gesture recognition and their accuracy in object detection.

CONFLICT OF INTERESTS

The authors confirm that there is no conflict of interest associated with this publication.

REFERENCES

1. Parimala, N., Teja, K.K.S., Kumar, J.A., Keerthana, G., Meghana K. and Pitchai, R. Real-time Brightness, Contrast and The Volume Control with Hand Gesture Using Open CV Python. *10th International Conference on Communication and Signal Processing (ICCSP)*, Melmaruvathur, India, **2024**; pp. 726-730.
2. Tiwari, S., Mishra, A., Kukreja D. and Yadav, A.L. Volume Controller using Hand Gestures. *14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, Delhi, India, **2023**; pp. 1-6.

3. Bhole, G., Bhingare, D., Bhise, R., Bhegade, S., Bhokare, S. and Bhosle, A. System Control using Hand Gesture. International Conference for Advancement in Technology (ICONAT), Goa, India, **2023**, pp. 1-4.
4. Meshram, A., Thakre, C., Rahangdale, C., Chouksey, Swaroop Bhoyar, S. Volume, brightness and curser controller with hand gesture", *International Research Journal of Modernization in Engineering Technology and Science*, **2023**; 5(6); pp. 503-507.
5. Gifta Jerith, G., Neelam, R.E.R., A. Kumar, R., Rohan, P., Rushika, D., Rupa Sri, K. OpenCV and Media Pipe-Based Hand-Gesture Control for Volume and Brightness. *International Journal of Research Publication and Reviews*, **2024**, 5(6); pp. 1334-1339.
6. Sharma, A., Sethiya, A., Ramtek, A., Shinde, A. Volume Control Using Hand Gesture and Mediapipe. *International Journal of Research Publication and Reviews*, **2023**; 4(5); pp. 267-271.
7. Theraja, K., Gupta S. and Gupta, D.K. Automatic Volume Control Using Image Processing and Deep Learning Techniques: A Review. *International Conference on Intelligent Systems for Cybersecurity (ISCS)*, Gurugram, India, **2024**; pp. 1-7.
8. Naveenkumar, M. OpenCV for Computer Vision Applications, *Proceedings of National Conference on Big Data and Cloud Computing (NCBDC'15)*, Tamilnadu, India, **2015**; pp. 1-5.
9. Tomar, Y., Himanshu, S.D. and Kaur, H. Human Motion Tracker using OpenCv and Mediapipe. *3rd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, Bengaluru, India, **2023**; pp. 1199-1204.
10. Vidya, M., Vineela, S., Sathish P. and Reddy, A.S. Gesture-Based Control of Presentation Slides using OpenCV. *2nd International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*, Trichy, India, **2023;** pp. 1786-1791.