

**Gesture volume and brightness control**

**AIM23CL201 – Artificial Intelligence Laboratory**

***Submitted by***

**M.Abeejay – E0123013**

***In partial fulfillment for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

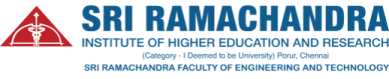
**in**

**COMPUTER SCIENCE AND ENGINEERING**

**(Artificial Intelligence and Machine Learning)**

**Sri Ramachandra Faculty of Engineering and Technology, Sri Ramachandra Institute of Higher Education and Research, Porur, Chennai -600116**

**MARCH, 2025**



**BONAFIDE CERTIFICATE**

Certified that this project report **“GESTURE VOLUME AND BRIGHTNESS CONTROL”** is the bonafide record of work done by **“M.ABEEJAY -E0123013”** who carried out the project work under my supervision.

**Signature of the Supervisor**

**Mr. Rajive Gandhi C,**

**Assistant Professor,**

Sri Ramachandra Faculty of Engineering and Technology,

SRIHER, Porur, Chennai-600 116.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE**  **NO.** |
|  | **ABSTRACT** | **4** |
| **1** | **INTRODUCTION**  1.1 Overview  1.2 Motivation  1.3 Problem Statement  1.4 Objectives | **5** |
| **2** | **SYSTEM ARCHITECTURE** | **8** |
| **3** | **METHODOLOGY** | **10** |
| **4** | **IMPLEMENTATION and RESULTS**  4.1 Code and Output | **12** |
| **5** | **CONCLUSION and FUTURE WORK** | **17** |
|  | **REFERENCES** | **18** |

**ABSTRACT**

In the era of smart human-computer interaction, touchless control systems are gaining significant traction. This project presents a gesture-based volume control system using computer vision and artificial intelligence. The primary goal is to enable users to adjust system audio volume intuitively by using simple hand gestures captured through a webcam.

By leveraging **MediaPipe** for hand tracking and **OpenCV** for image processing, the system identifies the distance between key hand landmarks—specifically, the thumb and index finger. Based on this distance, the volume level is dynamically increased or decreased in real-time. Additionally, the system can be extended to control other functionalities such as screen brightness or media playback, offering a comprehensive touch-free user interface.

This project demonstrates how AI-powered vision systems can enhance user experience in both personal computing and smart environments by providing a natural, hands-free method of interaction. It is especially beneficial in scenarios where physical contact with devices may be inconvenient or undesirable.

**Keywords:**

**Gesture Recognition, Hand Tracking, Volume Control, MediaPipe, OpenCV, Human-Computer Interaction (HCI), Touchless Interface, Computer Vision, Artificial Intelligence, Real-time Control.**

**CHAPTER 1**

**INTRODUCTION**

**1.1 Overview**

The Gesture and Volume Control project is a smart system designed to enhance user interaction with computers using hand gestures instead of traditional input devices like keyboards or mouse. The system utilizes a webcam to capture real-time video and applies computer vision techniques, primarily using **MediaPipe** for hand detection and **OpenCV** for image processing. By detecting the relative distance between the thumb and index finger, the program adjusts the system’s audio volume dynamically.

**1.2 Motivation**

The idea of controlling a computer system using simple hand gestures brings us closer to seamless human-computer interaction. Inspired by the convenience and futuristic appeal of gesture-controlled systems in smart devices and modern vehicles, this project aims to provide a low-cost, easy-to-use solution for gesture-based volume control.

**1.3 Problem Statement**

In conventional computing systems, volume control is typically achieved using physical buttons, keyboard shortcuts, or mouse interactions. These methods can be inconvenient or inaccessible in certain situations, such as when the user's hands are occupied, during presentations, or in touchless environments like medical facilities.

**1.4 Objectives**

The objective of this project is to create a gesture volume and brightness control that enhances accessibility and inclusivity. The key goals are:

* To develop a real-time gesture recognition system using a standard webcam.
* To implement hand tracking using **MediaPipe** for accurate detection of both left and right hands.
* To use **OpenCV** for video capture, gesture analysis, and UI display.
* To control **system volume** using gestures from the **right hand** (e.g., distance between thumb and index finger).
* To control **screen brightness** using gestures from the **left hand**.
* To provide a smooth and intuitive touchless interface for daily computer interactions.

**1.5 Implementation Approach**

The system is implemented using **Python**, with the integration of **MediaPipe**, **OpenCV**, and system-level control libraries such as **pycaw** (for audio control) and **screen\_brightness\_control** (for brightness control). The approach follows a step-by-step process outlined below:

### 1. ****Video Capture****

* A webcam is used to continuously capture live video frames.
* Frames are flipped horizontally to provide a mirror-like experience for natural interaction.

### 2. ****Hand Detection using MediaPipe****

* The MediaPipe Hands module is used to detect and track hand landmarks in real time.

### 3. ****Landmark Processing and Gesture Calculation****

* From the detected hand landmarks, the positions of the **thumb tip** and **index finger tip** are extracted.

### 4. ****Volume Control****

* If the **right hand** is detected, the system interprets the gesture to adjust the **audio volume**.
* The mapped value is passed to **pycaw** to set the system's master volume.

### 5. ****Brightness Control****

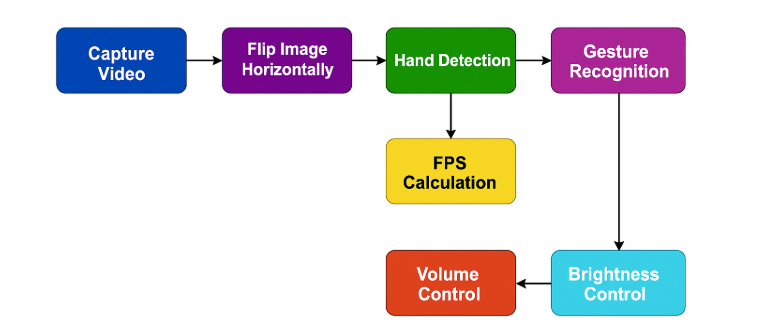
* If the **left hand** is detected, the gesture is interpreted to control **screen brightness**.
* The brightness level is adjusted using the **screen\_brightness\_control** library.

### 6. ****Visual Feedback****

* Circles and lines are drawn between the thumb and index finger to provide visual cues.

**CHAPTER 2**

**SYSTEM ARCHITECTURE**

****

The system architecture for the Gesture-Based Volume and Brightness Control project is designed around a modular and real-time processing approach. At its core, the system utilizes a webcam or built-in camera to continuously capture live video feed, which is then processed using OpenCV for image handling and MediaPipe for efficient and accurate hand tracking. The captured frames are analyzed to detect specific hand landmarks, from which gestures are interpreted based on the relative position and distance of fingers. These gestures are mapped to corresponding system control commands — such as adjusting the system volume or screen brightness — using platform-specific libraries like pycaw for audio control and screen\_brightness\_control for brightness adjustment on Windows. The architecture ensures smooth integration between gesture recognition and system control modules, with a feedback loop that allows real-time adjustments based on user input.

**Workflow Explanation**

1. **Video Capture:**
   * The system starts by capturing real-time video using a webcam or built-in camera.
2. **Hand Detection:**
   * Each video frame is processed using **MediaPipe** to detect and track hand landmarks.
3. **Landmark Extraction:**
   * Key points (fingertips, joints) on the hand are extracted for further analysis.
4. **Gesture Recognition:**
   * Gestures are interpreted by calculating the distance between specific landmarks (e.g., thumb and index finger).
   * Specific gestures are mapped to control volume or brightness levels.
5. **Control Mapping:**
   * Recognized gestures are converted into system-level commands:
     + **Volume Control:** Using the pycaw library.
     + **Brightness Control:** Using the screen\_brightness\_control library.
6. **Command Execution:**
   * The system executes the mapped commands to adjust the volume or screen brightness in real-time.
7. **Visual Feedback (Optional):**
   * The system displays feedback on the screen (e.g., volume level bar or brightness level indicator).
8. **Continuous Loop:**
   * The process repeats continuously as long as the camera is active, allowing real-time control with gestures.

**CHAPTER 3**

**METHODOLOGY**

The methodology of this AI-powered gesture control system involves several stages, from real-time video capture to the execution of system-level commands. The project integrates computer vision, hand tracking, and gesture classification to enable intuitive control of volume and screen brightness through hand movements.

#### ****1. Real-Time Video Capture and Preprocessing****

* The system begins by accessing a live video feed using a webcam through OpenCV (cv2.VideoCapture).
* Each frame from the video stream is converted to RGB format for compatibility with MediaPipe’s hand tracking solution.
* Frames are resized or cropped if needed to optimize performance and focus on hand regions.

#### ****2. Hand Landmark Detection****

* MediaPipe Hands is utilized to detect and extract **21 hand landmarks** (x, y, z coordinates) in real time.
* These landmarks provide precise data about finger positions and movements, which form the foundation for gesture recognition.
* The hand tracking model ensures high-speed and accurate detection even with dynamic backgrounds.

#### ****3. Gesture Recognition and Feature Calculation****

* Distances between specific landmarks (e.g., between the thumb tip and index finger tip) are calculated using Euclidean distance.

**4. Control Mapping and Command Execution**

* Recognized gestures are mapped to system-level controls using:
  + pycaw library for **volume control**.
  + screen\_brightness\_control library for **brightness adjustment**.
* The mapped values are scaled proportionally to the hand gesture’s distance for smooth control.

#### ****5. Visual Feedback and User Interaction****

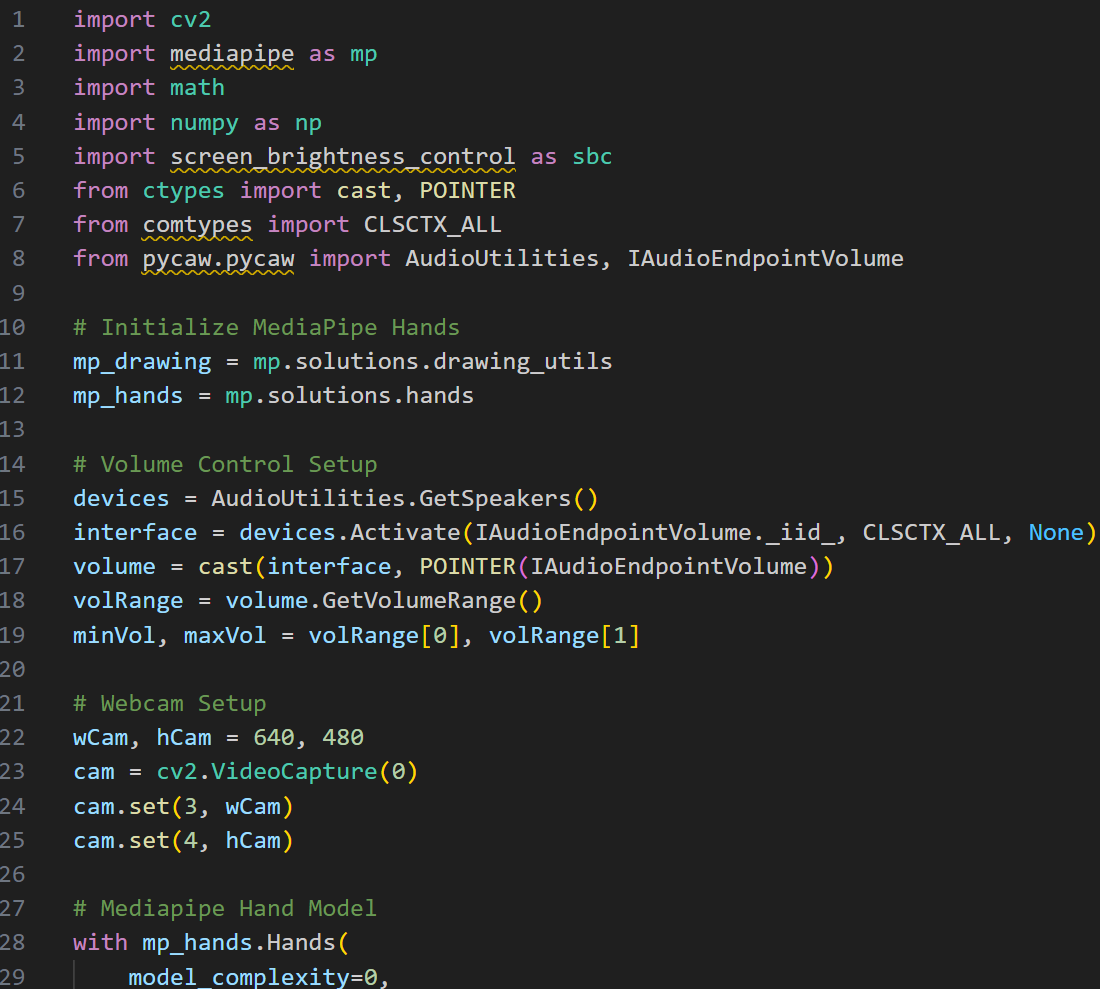
* The system provides on-screen feedback such as volume/brightness bars or percentage values for better user interaction.
* Optional overlays are rendered using OpenCV to show detected landmarks and gesture status.

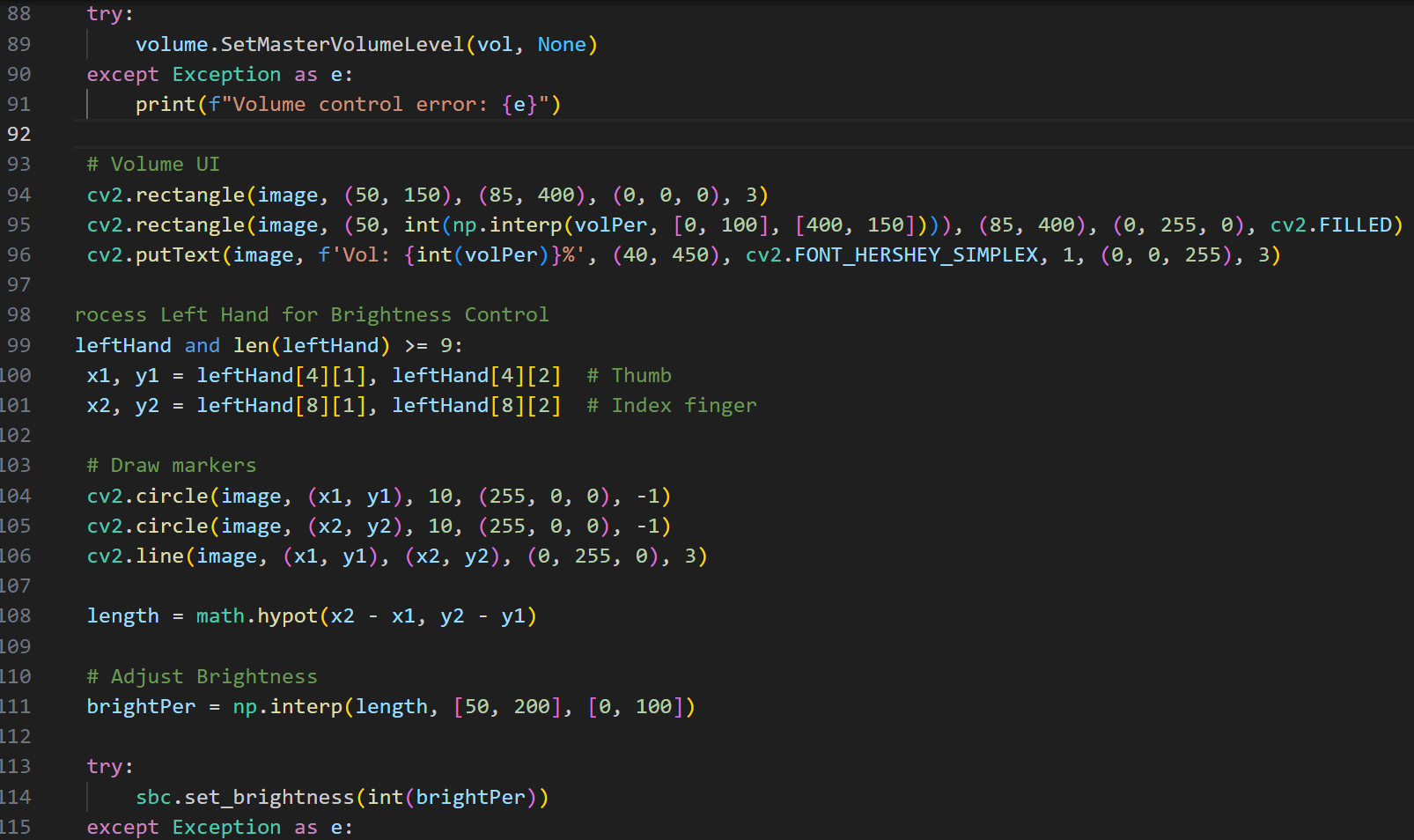
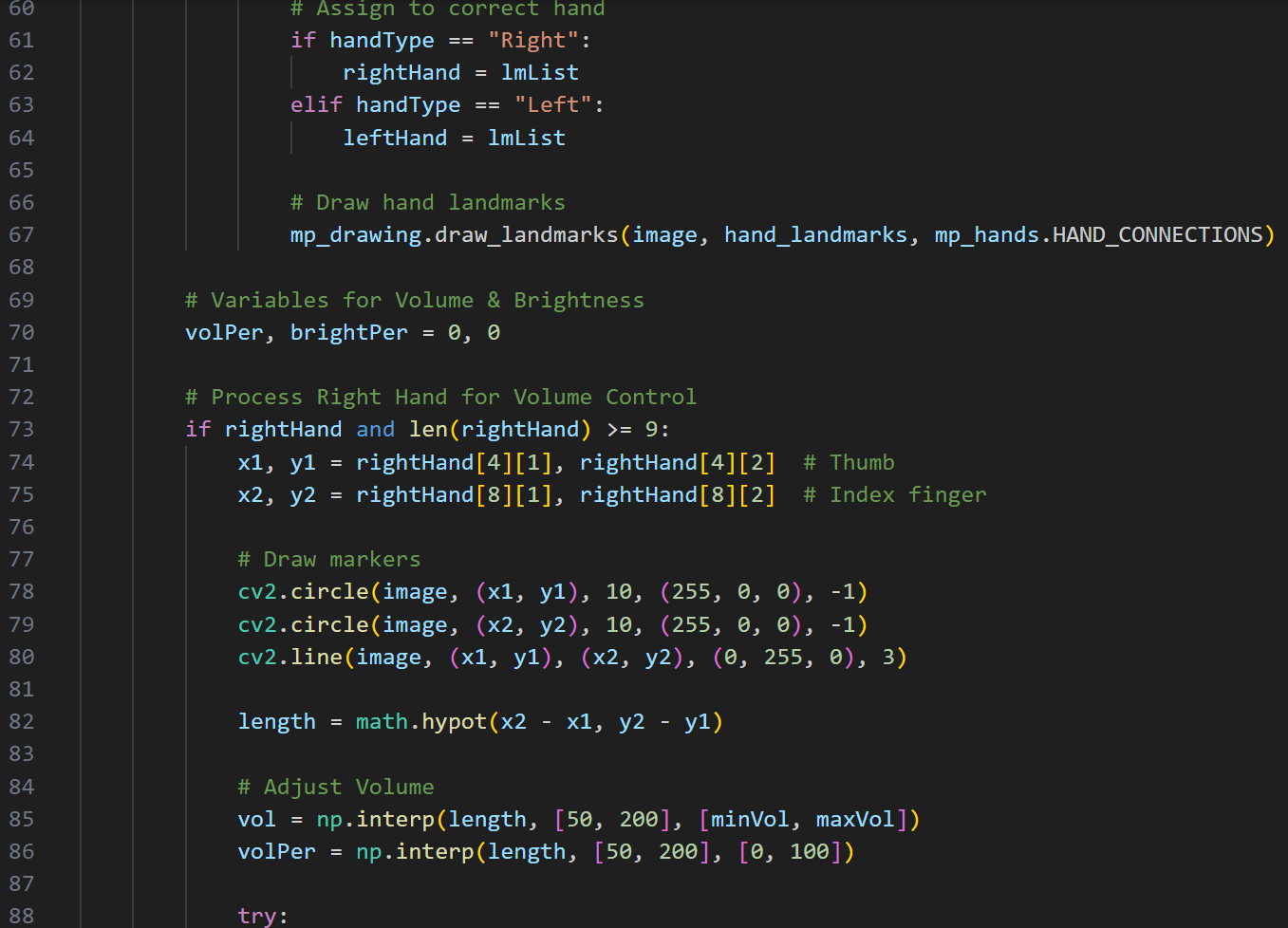
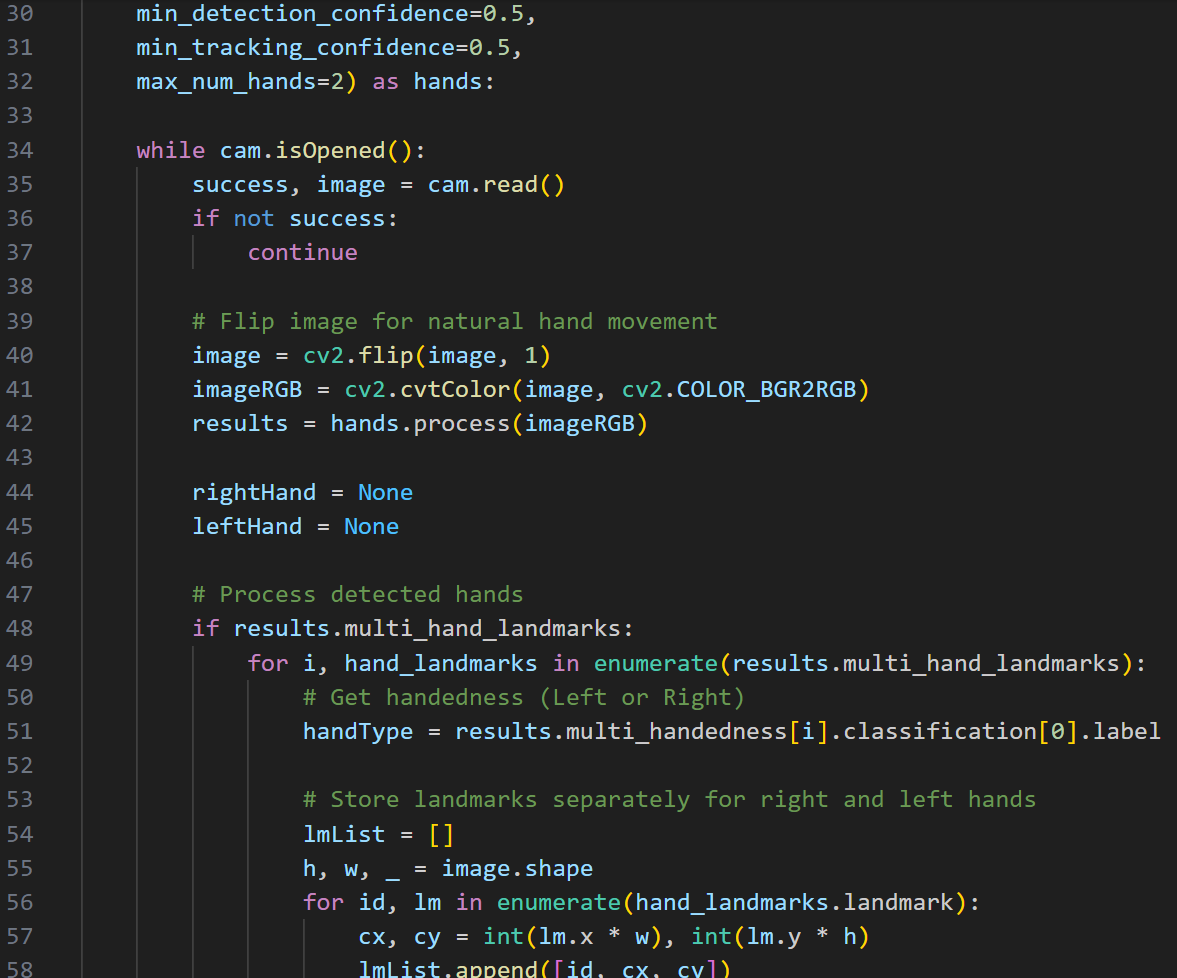
#### ****6. Continuous Processing and Looping****

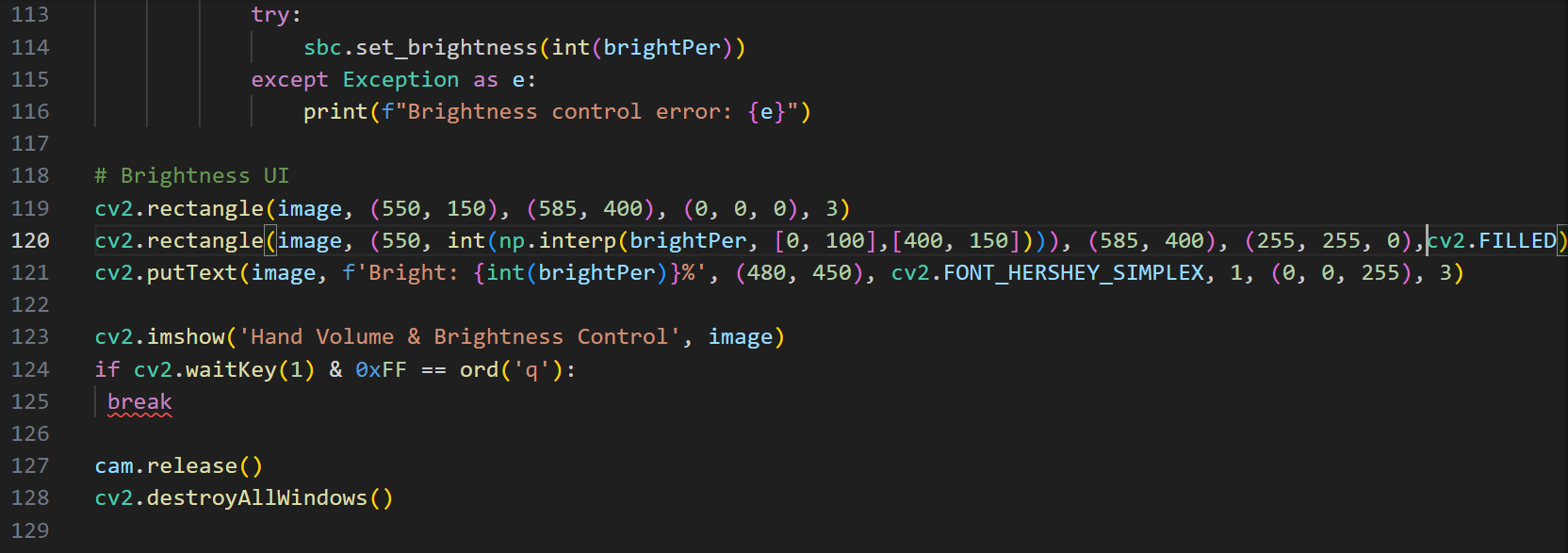
* The entire process runs in a continuous loop to provide real-time responsiveness.
* The system adjusts volume and brightness dynamically as the user changes hand gestures in front of the camera

**CHAPTER 4**

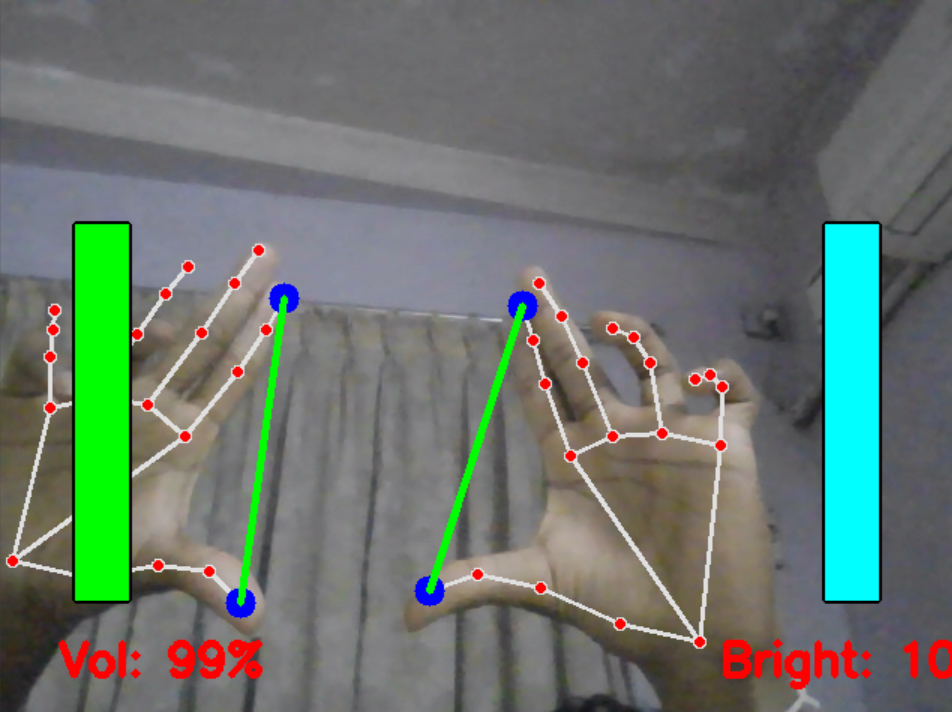
**IMPLEMENTATION & RESULTS**

****





OUTPUT:



**CHAPTER 5**

**CONCLUSION**

The **Gesture-Based Volume and Brightness Control** project demonstrates an innovative, touch-free interaction method using **computer vision and hand tracking**. By leveraging **OpenCV** and **MediaPipe**, the system accurately detects hand gestures and maps them to control system settings such as **audio volume** and **screen brightness.**

**FUTURE WORK**

 **Integrate machine learning models** for more accurate and adaptive gesture recognition.

 **Expand gesture support** to include media control, app switching, and custom shortcuts.

 **Enable smart device and IoT integration** for broader real-world applications.

 **Develop cross-platform and mobile compatibility** to increase accessibility and usability.

**REFERENCES**

 Zhang, Z. (2012). *Microsoft Kinect Sensor and Its Effect*. IEEE MultiMedia, 19(2), 4–10.

 Lugaresi, C., et al. (2019). *MediaPipe: A Framework for Building Perception Pipelines*. arXiv preprint arXiv:1906.08172.

 Bradski, G. (2000). *The OpenCV Library*. Dr. Dobb’s Journal of Software Tools.

**ONLINE REFERENCES**

https://docs.opencv.org/

<https://arxiv.org/>

<https://developers.google.com/mediapipe/solutions/vision/hand_landmarker>