Student's first and last name, Year

# CS4043D Image Processing

## Course Project

**Date: 06/04/24**

# Gesture Volume Control

**GROUP: A**


| | |
|---|---|
| Lokam Teja Ram | B200793CS |
| E Krishna Prasad | B200803CS |
| P Gnana Prakash | B200840CS |
| Anagha M V | B200762CS |
| Abhina Sunny | B200692CS |

Department of Computer Science and Engineering

National Institute of Technology, Calicut

# Declaration

This report has been prepared on the basis of our group’s work. All the other published and unpublished source materials have been cited in this report.

| Student Name: | Signature: |
|---|---|
| Lokam Teja Ram | B200793CS |
| E Krishna Prasad | B200803CS |
| P Gnana Prakash | B200840CS |
| Anagha M V | B200762CS |
| Abhina Sunny | B200692CS |

# Table of Contents

# Abstract

Gestures play a significant role in human communication and are increasingly integral to Human-Computer Interaction (HCI) technologies. These evolving HCI systems enable users to convey instructions to machines through hand movements, facial expressions, voice commands, and touch interactions. Gesture recognition has become a focal point of research in fields such as computer vision, pattern recognition, and HCI. The primary objective of this research paper is to explore the utilization of hand gestures as a vital mode of machine control. In this pursuit, OpenCV libraries are employed to address the challenges associated with gesture recognition in the context of Human Computer Interaction.

# Project Specification

The Project Specification for creating a volume control system using hand gestures is :

## Hardware Requirements :

1. Camera: HD resolution (720p or higher), capable of capturing high-quality video feed and mounted securely for effective hand tracking.

2. Computer: Multi-core processor (e.g., Intel Core i5 or higher), with at least 8 GB of RAM and available USB ports for connecting peripherals.

## Overview and Technical Specifications:

1. Develop a volume control system that allows users to adjust audio volume using handgestures.

2. Utilize OpenCV for image processing, MediaPipe for hand tracking and gesture recognition, and Pyautogui for controlling audio volume.

3. Utilize Python for coding the application due to its compatibility with OpenCV, MediaPipe, and PyAutogui.

4. Aim to enhance Human-Computer Interaction (HCI) experience by providing an intuitive and hands-free control mechanism.

## Functional Requirements:

1. Gesture Detection: Implement hand gesture detection using OpenCV and MediaPipe to recognize predefined gestures for volume control.

2. Volume Adjustment: Integrate PyCaw to adjust the system's audio volume based on recognized gestures.

3. User Interface: Develop a simple graphical user interface (GUI) to display real-time camera feed, recognized gestures, and current volume level.

## Testing and validation:

1. Validate gesture recognition accuracy, volume adjustment functionality, and error handling capabilities through simulated scenarios

2. Gather feedback on usability, responsiveness, and performance from diverse users, refining the system based on observations and recommendations.

By adhering to these specifications, we can develop a comprehensive volume control system using hand gestures with the help of OpenCV, MediaPipe, and PyAutogui to enhance the Human-Computer Interaction experience for users.

# Chapter 1: **Introduction**

## 1.1 Background

In recent years, Human-Computer Interaction (HCI) technology has emerged as an exciting frontier for human-computer communication. Human-Machine Interaction (HMI) is the key to bridging the gap between humans and machines through a user interface. Within this landscape, hand gesture recognition has gained prominence as a novel and intuitive technique for HCI, offering automation and ease of interaction without relying on traditional input devices such as keyboards and mouse.

The unique advantage of hand gestures lies in their ability to convey a wealth of information in a shorter time, presenting a compelling case for enhancing the interface between users and computers.Our project is centered on harnessing the potential of hand gestures, employing technologies like OpenCV, MediaPipe, and PyCaw to create a volume control system that enables users to interact with their devices using intuitive hand gestures.

## 1.2 Challenges

Real-Time Processing: Achieving real-time processing of video feed for gesture recognition while maintaining system responsiveness.

Noise in the video feed, such as background clutter or hand movements unrelated to the intended gesture, can introduce inaccuracies in distance measurements. Additionally, in cases like fingers partially overlapping can complicate the measurement process.

Camera clarity should be good so that the hands landmarks are perfectly extracted.

# Chapter 2: **Literature Review**

In recent years, there has been a growing focus on developing hand gesture recognition systems using computer vision and machine learning techniques. These systems offer a more intuitive way for users to interact with devices like volume controllers. Numerous research studies have explored creating volume controllers based on hand gestures, employing various methods including vision-based techniques like color and motion detection, depth-sensing cameras, and machine learning algorithms.One particular study showcases the use of an Artificial Neural Network (ANN) for gesture recognition using an accelerometer, specifically the Wii remote which registers movement along its X, Y, and Z axes. To optimize system performance and memory usage, the author adopts a two-tiered approach. The first tier involves user authentication through gesture recognition based on accelerometer data. The second tier employs (Fuzzy) automata for gesture recognition, with data normalization using k-means and a fast Fourier algorithm.[1]

Drawback for this paper is its limited accuracy and sensitivity in capturing fine-grained hand movements. Unlike vision-based techniques that can precisely track hand gestures in real-time, accelerometers may struggle to capture subtle nuances in hand movements, leading to potentially lower recognition accuracy and reliability, especially for complex gestures or in dynamic environments

In this research paper the author discussed a novel method of hands gestures recognition based on the identification of certain shape-based elements. A single camera is used to record the user's gesture and feed it into the system. The development of a system that can recognise certain human gestures and use them to transfer data for device control is one of the main goals of gesture recognition. Real-time gesture recognition enables users to control computers by making certain gestures in front of a video monitor. The development of a system that can recognise certain human gestures and use them to communicate data for device control is one of the main objectives of gesture.[2]

Drawback of this paper is the potential limitation in accurately capturing and distinguishing between complex or dynamic hand movements. Shape-based methods may struggle to effectively differentiate between similar gestures or gestures with overlapping shapes, leading to lower recognition accuracy

In Another study the volume control is done using hand gestures and incorporates the use of OpenCV. By utilizing a computer's webcam, the module captures images or videos, processes them to extract relevant data, and adjusts the volume based on recognized gestures. Users can manipulate the volume without needing physical touch or input devices like mouse or keyboards. Using OpenCV and Python, the module recognizes specific human gestures to enact the desired adjustments in device settings. By intercepting video input and analyzing gestures within a specified range, the module successfully alters the computer's volume.[3]

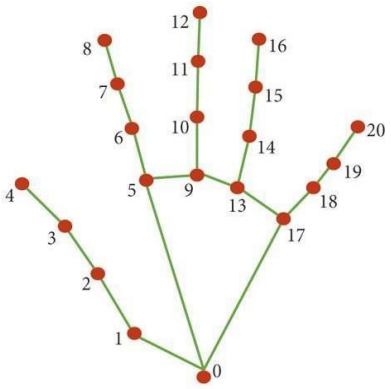
Drawbacks in this paper are Real-Time Processing: Achieving real-time processing of video feed for gesture recognition while maintaining system responsiveness and Noise in the video feed, such as background clutter or hand movements unrelated to the intended gesture, can introduce inaccuracies in distance measurements. Additionally, in cases like fingers partially overlapping can complicate the measurement process.

# Chapter 3: **Proposed Method**

## Mediapipe:

MediaPipe is a software framework developed by Google that helps computers see and understand the world through video input. It's like giving computers the ability to analyse and interpret images and videos in real-time. MediaPipe is often used for tasks like recognizing and tracking objects, detecting hand movements, or identifying facial expressions. It's a powerful tool for creating applications that involve computer vision and understanding the visual world.

Single-shot detector model is used for detecting and recognizing a hand or palm in real time. single shot detector model is used by the MediaPipe. First, in the hand detection module, it is first trained for a palm detection model because it is easier to train palms. Furthermore, the non-maximum suppression works significantly better on small objects such as palms or fists. A model of hand landmark consists of locating 21 joint or knuckle co-ordinates in the hand region,as shown in figure.




0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

## OpenCV:

OpenCV is a computer vision library which contains image-processing algorithms for object detection. OpenCV is a library of python programming language, and real-time computer vision applications can be developed by using the computer vision library. OpenCV library is used in image and video processing and also analysis such as face detection and object detection.

The various functions and conditions used in the system are explained in the flowchart of the real-time gesture volume control system as shown in Figure.

The proposed gesture volume control system is based on the frames that have been captured by the webcam in a laptop or PC. By using the Python computer vision library **OpenCV**, the video capture object is created and the web camera will start capturing video.

## Capturing the Video and Processing:

The Gesture volume control system uses the webcam where each frame is captured till the termination of the program.The video frames are processed from BGR to RGB colour space to find the hands in the video frame by frame.

Once we get the 21 points of the hand, we take co-ordinates of thumb and index finger then find thedistance between those two points using the formula.

$$\sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

Then according to the distance we change the volume of the device using pyautogui .

## Design :

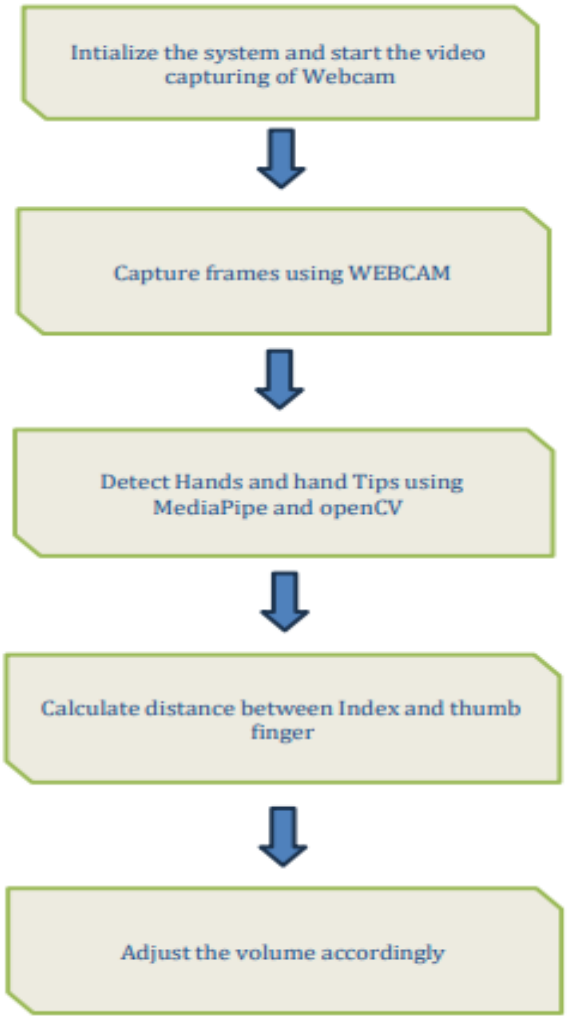



***Figure 1.*** *Flow Chart of the methodology*

## STEPS INVOLVED:

1. **Data Preparation:**
   No specific dataset preparation is required as the system operates in real-time, capturing frames from the webcam.
2. **Feature Extraction:**
   Utilize OpenCV and MediaPipe Hands library to detect hand landmarks in each frame. Extract relevant features such as the distance between specific landmarks representing fingertips.
3. **Gesture Classification:**
   Define thresholds for the distances between landmarks to classify gestures (e.g., volume up, volume down, neutral).
4. **Volume Control:**
   Based on the detected gestures, adjust the system volume accordingly using PyAutoGUI or other appropriate libraries.

5. **Real-time Processing:**
   Continuously capture frames from the webcam.
   Process each frame to detect hand gestures and adjust the volume accordingly.
6. **Performance Evaluation:**
   Validate the system's performance in real-time by interacting with the gesture volume control system.

## 3.1 Tools and Techniques

**OpenCV (cv2):** OpenCV (Open Source Computer Vision Library) is a popular library used for computer vision tasks such as image and video manipulation, object detection, and more.

**Mediapipe:** Mediapipe is a framework developed by Google that provides tools for building machine learning pipelines for various perception tasks, including body pose estimation, hand tracking, facial recognition, and more.Here we utilize the mediapipe module for detecting and tracking hand landmarks in the webcam feed.

**Pyautogui:** PyAutoGUI is a Python library for automating keyboard and mouse interactions. In this code, it's used to simulate key presses based on hand gestures detected by the hand tracking model.here it is used to simulate pressing the volume up or volume down keys.

**Image Processing:** Techniques such as converting the color space of the image from BGR to RGB and extracting hand landmarks coordinates from the detected hand landmarks are used for processing the webcam frames.

**Hand Landmarks Detection:** The code detects hand landmarks using the pre-trained hand tracking model provided by Mediapipe.It then extracts the coordinates of specific landmarks (such as fingertips) to determine hand gestures.

# Chapter 4: **Evaluation and Results**

## 4.1 Evaluation

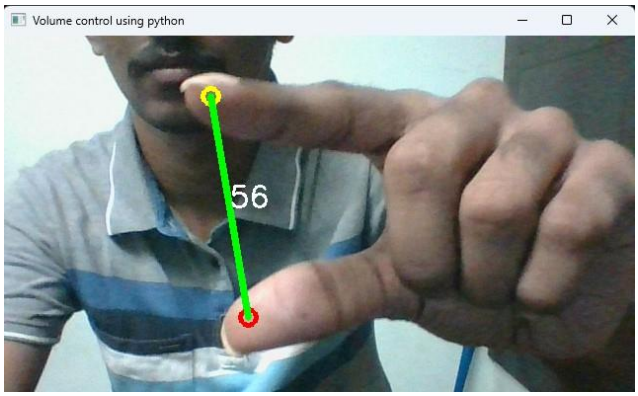
This project is being evaluated by calculating the distance between the index and thumb finger. If the hand that is detected is left then brightness will be changed corresponding to the distance and if hand detected is right then volume is changed corresponding.

If the Hand detected is left then, distance between index and thumb finger is calculated and brightness of device is set to that value using SBC.

If the Hand detected is right then, distance between index and thumb finger is calculated , if the distance is less than 30 then our code keeps on pressing volume down button and if distance is greater than 30 then our code keeps on pressing volume up button.

## 4.2 Comparison of results

1)

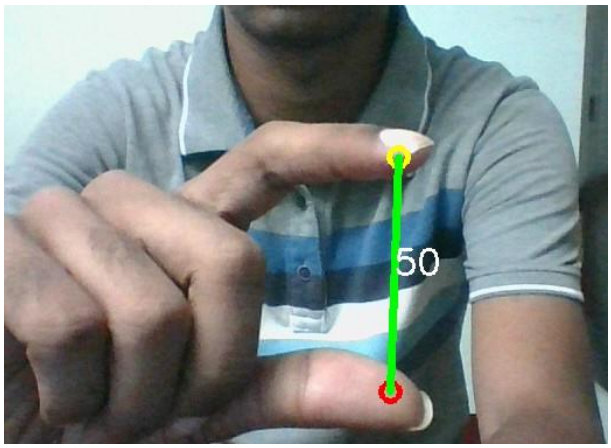



Hand Detected is right and distance is greater than 30 so volume increases.

2)


Hand Detected is Right and distance is less than 30 so volume decreases.

3)




Hand Detected is left and distance is 50 so brightness of device will be set to 50

**Table 1.** Comparision of results

| Cases | Accuracy |
|---|---|
| Right Hand and distance is greater than 30 | 99% Volume is increasing and no change in brightness of device |
| Right Hand and distance is less than 30 | 98% Volume is decreasing and no change in brightness of device |
| Left Hand | 98% Brightness will be set to distance and no change in volume of device |

| Hand Detected | Distance between thumb and index finger | Less/Greater than 30 | Action |
|---|---|---|---|
| Right Hand | 56 | Greater | Code will press volume up button |
| Right Hand | 26 | Less | Code will press volume down button |
| Left Hand | 50 | Greater(Doesn't matter) | Code will set the brightness of device to 50. |

# Chapter 5: **Implementation and Code**

## 5.1 Documentation

Python script is used which utilizes OpenCV, MediaPipe, and PyAutoGUI libraries to implement a volume control system based on hand gestures.

### Imported Modules:

OpenCV (cv2) for image processing, MediaPipe (mediapipe) for hand tracking, and PyAutoGUI (pyautogui) for controlling system volume, screen_brightness_control library for adjusting the brightness, numpy library for numerical operations.

**Camera Setup:** Initialize the webcam capture using cv2.VideoCapture(0)

**Image Conversion :** BGR image which is obtained using above camera is converted into RGB format, which is the default input for mediapipe.

**Detection of Hand landmarks :** This RGB formatted image is inputted to Mediapipe and hand landmarks are detected.

**Distance Calculation and Volume Change :** Once hand landmarks are detected, we calculate distance between thumb and index fingers and correspondingly we change the volume or brightness of the device using PYAUTOGUI or SBC.

Variables (x1,y1) and (x2,y2) are the landmarks of thumb and index finger.

**Change in volume or brightness** : Then if the hand that is detected is right hand then volume will be change appropriately as the distance. Else if hand detected is left hand then brightness will be changed appropriately as distance.

**Input:**

- Webcam feed captured by webcam.read()
- Hand landmarks detected by MediaPipe.

**Output:**

- Processed image with hand landmarks and volume adjustment status displayed using cv2.imshow**()**.
- Volume control actions triggered by pyautogui.press().

## 5.2 Source Code

```
import cv2
import mediapipe as mp
import numpy as np
import pyautogui
from math import hypot
```

```
import screen_brightness_control as sbc
import ctypes

webcam = cv2.VideoCapture(0)
my_hands = mp.solutions.hands.Hands()
drawing_utils = mp.solutions.drawing_utils

while True:
    _, image = webcam.read()
    image = cv2.flip(image, 1)
    frame_height, frame_width, _ = image.shape

    rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    output = my_hands.process(rgb_image)
    hands = output.multi_hand_landmarks

    if hands:
        for hand in hands:
            drawing_utils.draw_landmarks(image, hand)
            landmarks = hand.landmark
            x1, y1 = 0, 0
            x2, y2 = 0, 0
            for id, landmark in enumerate(landmarks):
                x = int(landmark.x * frame_width)
                y = int(landmark.y * frame_height)
                if id == 8:
                    cv2.circle(img=image, center=(x, y), radius=8, color=(0, 255, 255), thickness=3)
                    x1 = x
                    y1 = y
                if id == 4:
                    cv2.circle(img=image, center=(x, y), radius=8, color=(0, 0, 255), thickness=3)
                    x2 = x
                    y2 = y

            dist = ((x2 - x1) ** 2 + (y2 - y1) ** 2) ** (0.5) // 4
            cv2.line(image, (x1, y1), (x2, y2), (0, 255, 0), 5)

            if len(output.multi_handedness) > 0:
                label = output.multi_handedness[0].classification[0].label
                if label == 'Right':
                    if dist > 30:
                        pyautogui.press("volumeup")
                    else:
                        pyautogui.press("volumedown")
                elif label == 'Left':
                    sbc.set_brightness(int(dist))

    cv2.imshow("Volume control using python", image)
    key = cv2.waitKey(10)
    if key == 27:
        break

webcam.release()
cv2.destroyAllWindows()
```

# References

[1] H.A JALAB "Static hand Gesture recognition for human computer interaction", 1-72012. JC.MANRESARVARONAR. MASF.

[2] Real-Time Hand Gesture Recognition for Device Control: An OpenCV-Based Approach to Shape Based Element Identification and Interaction by Nishant Kumar, Hridey Dalal, Aditya Ojha , Abhinav Verma, Dr. Mandeep Kaur

[3] Volume Control using Gestures by Martendra Pratap Singh, Arzoo Poswal, Eshu Yadav