VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering
VIT Chennai
Vandalur - Kelambakkam Road, Chennai - 600 127

Programme: Mtech Business Analytics

Course: Computer Vision

Slot: F1

Faculty: Dr. Bharadwaja Kumar G

Component: J-Component

Title: Hand Gesture Volume Controller

BY

S. Roshan Kumar 20MIA1156

Objective

The objective of this report is to explore the technology behind hand gesture volume control and analyze its potential impact on user experience and accessibility in audio devices. The report will cover the basics of hand gesture recognition and volume control algorithms, explore the advantages and limitations of this technology in different scenarios, discuss the current state of research and development in this field, and provide recommendations for further improvement and commercialization.

Additionally, the report will analyze the market potential and economic feasibility of hand gesture volume control technology, and assess the impact it could have on user experience and accessibility in the audio industry.

In this paper we are developing a volume controller in which we are using hand gestures as the input to control the system , Opencv module is basically used in this implementation to control

the gestures.This system basically uses the web camera to record or capture the images /videos and accordingly on the basis of the input, the volume of the system is controlled by this application. The main function is to increase and decrease the volume of the system. The project is implemented using Python, OpenCV. We can use our hand gestures to control the basic operation of a computer like increasing and decreasing volume. Therefore ,people will not have to learn machine-like skills which are a burden most of the time. This type of hand gesture systems provides a natural and innovative modern way of non verbal communication. These systems has a wide area of application in human computer interaction. The purpose of this project is to discuss a volume control using hand gesture recognition system based on detection of hand gestures. In this the system is consist of a high resolution camera to recognise the gesture taken as input by the user.

The main goal of hand gesture recognition is to create a system which can identify the human hand gestures and use same input as the information for controlling the device and by using real time gesture recognition specific user can control a computer by using hand gesture in front of a system video camera linked to a computer. In this project we are developing a hand gesture volume controller system with the help of OpenCV , Python. In this system can be controlled by hand gesture without making use of the keyboard and mouse.

Methodology

- Detect hand landmarks
- Calculate the distance between thumb tip and index finger tip.
- Map the distance of thumb tip and index finger tip with volume range. For my case, distance between thumb tip and index finger tip was within the range of 30 – 350 and the volume range was from -63.5 – 0.0.
- In order to exit press ‘q'

In this project we are using python technology to develop the project , the code is written and designed in python language using Opensv and NumPy modules. In this project firstly we import the libraries which are to be used for further processing of the input and the output. The libraries which are used in this project which needs to be imported are OpenCV, mediapipe, math, ctypes, pycaw and numpy. We get video inputs from our primary camera.

Now, here mediapipe is used to detect the video as the input from our camera and use mphand.hands module to detect the gesture .Then , in order to access the speaker we have used the pycaw and we have provided the range of the volume from minimum volume to maximum volume.

Next step is to convert the input image to rgb image to complete the processing of the input captured. Then its turn to specify the points of thumb in input and fingers.

A. OPEN CV

Open CV is a library of python which tackle PC vision issue. It is used to detect the face which is done using the machine learning .It is a very important library and is used in several projects to detect the face and recognize the several frames also it supports several programming languages. It also performs object detection and motion detection. It also support several type of operating system and can be used to detect the face of the animals also.

B. NUMPY

NumPy is the module of the Python. The numpy word basically shows Numerical Python and it is utilized. This is the module which is basically written in c language and is said as expansion module . Numpy guarantee remarkable execution speed. Numpy is mostly used for performing calculations, tasks using certain functions it provides like multiply, divide, power etc.

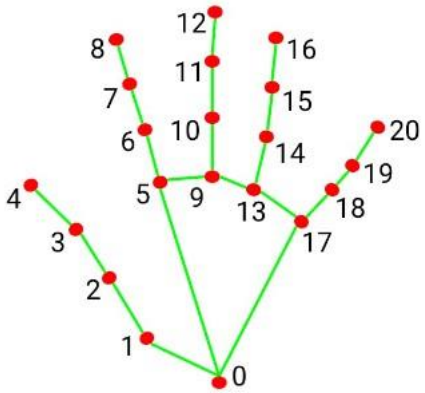
C. IMAGE FILTERING –HISTOGRAM

Histogram is a type of graph which represents the movement of the pixels power in the portrayal.In this we use to filter the images using histogram and convert them into the rgb in order to process the image in our system . Consequently the power of a pixel is in the range [0,255].

D. MEDIAPIPE

MediaPipe is a module for processing video, audio and several types of related data across platform like Android, iOS, web, edge device and several applied ML pipeline. Several types of functions are performed with the help of this module , we have used this module in our project to recognize the hand gesture and detect the input from it.

- Face Detection
- Multi-hand Tracking
- Segmentation
- Object Detection and Tracking



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

```
cap = cv2.VideoCapture(0)
```

We then get the video input from our computer's primary camera. If you are using any other camera, replace the number 0 with that of the camera you are using.

Detecting, initializing, and configuring the hands

```
mpHands = mp.solutions.hands hands
= mpHands.Hands() mpDraw =
mp.solutions.drawing_utils
```

In the code above, we are calling on the mediapipe hand module to detect the hands from the video input we got from our primary camera. MpHands.Hands() then completes the initialization and configuration of the detected hands. We finally draw the *connections* and *landmarks* on the detected hand using mp.solutions.drawing_utils.

Accessing the speaker using pycaw

```
devices = AudioUtilities.GetSpeakers()
interface = devices.Activate(IAudioEndpointVolume._iid_, CLSCTX_ALL,
None) volume = cast(interface, POINTER(IAudioEndpointVolume))
```

These are the initializations we need for pycaw to run smoothly. The developer provides this library together with the initializations. We are not going to change anything.

Finding the volume range between the minimum and maximum volume

```
volMin, volMax = volume.GetVolumeRange()[:2]
```

The code above finds the volume range between the minimum and maximum volume. We place it outside the while loop because we need to find the volume range once.

Capturing an image from our camera and converting it to an RGB image

```
while True:
    success, img = cap.read()
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
results = hands.process(imgRGB)
```

The code above checks whether the camera we have specified works. If it works, we will capture an image. We then convert the image to RGB and complete the processing of the image.

We now need to check whether we have multiple hands in the image we captured.

Checking whether we have multiple hands in our input

```
lmList = [] if
results.multi_hand_landmarks:
```

This code creates an *empty list* that will store the list of elements of the hands detected by the mediapipe hand module, i.e., the number of points on the hand. It also checks whether the input has multiple hands.

We will now create a for loop to manipulate each hand present in the input.

Creating a for loop to manipulate each hand

```
for handlandmark in results.multi_hand_landmarks:
for id, lm in enumerate(handlandmark.landmark):
        h, w, c = img.shape
        cx, cy = int(lm.x * w), int(lm.y * h)
lmList.append([id, cx, cy])
   mpDraw.draw_landmarks(img, handlandmark, mpHands.HAND_CONNECTIONS)
```

- In the code above, we use the first for loop to interact with each hand in the results. We use the second for loop to get the id (id number) and lm (landmark information) for each hand landmark. The landmark information will give us the x and y coordinates. The id number is the number assigned to the various hand points.
- h, w, c = img.shape: this line of code checks the height, width, and channels of our image. This will give us the width and height of the image.
- cx, cy = int(lm.x * w), int(lm.y * h): this line of code will find the central position of our image. We will achieve this by multiplying *lm.x by the width* and assigning the value obtained to cx. Then multiply lm.y by the height and assign the value obtained to cy. lm stands for landmark.

- lmList.append([id, cx, cy]): we will then use this line to add the values of id,cx and cy to lmList.
- We will finally call *mpDraw.draw_landmarks* to draw all the landmarks of the hand using the last line of code.

Specifying the points of the thumb and middle finger we will use

```
if lmList != []:    x1, y1 =
lmList[4][1], lmList[4][2]    x2, y2 =
lmList[8][1], lmList[8][2]
```

In the code above, we specify the number of elements in lmlist. It should not be null. We assign variables x1 and y1 the x and y coordinates of point 4 respectively. This is the tip of the thumb. We then repeat the same for the index finger in the last line.

Drawing a circle between the tip of the thumb and the tip of the index finger

```
cv2.circle(img, (x1, y1), 15, (255, 0, 0), cv2.FILLED)
cv2.circle(img, (x2, y2), 15, (255, 0, 0), cv2.FILLED)
```

The code above draws a circle at the tip of the thumb and that of the index finger.

- (x1, y1) specifies that we will draw the circle at the tip of the thumb. 15 is the *radius* of the circle. (255, 0, 0) is the *color* of the circle. cv2.FILLED refers to the thickness of 1 pixels which will fill the circle with the color we specify.
- We will repeat the same for the index finger:

Drawing a line between points 4 and 8

```
cv2.line(img, (x1, y1), (x2, y2), (255, 0, 0), 3)
```

In the code above, we use the cv2.line function to draw a line between point four of the hand and point 8. The line will connect point 4 (x1, y1), which is the tip of the thumb, and point 8 (x2, y2), which is the tip of the index finger. (255, 0, 0) is the line color and 3 is its thickness.

Finding the distance between points 4 and 8

```
length = hypot(x2 - x1, y2 - y1)
```

In the code above, we find the distance between the tip of the thumb and the index finger using a hypotenuse. We achieve this by calling the math hypot function then passing the difference between x2 and x1 and the difference between y2 and y1.

Converting the hand range to the volume range

```
vol = np.interp(length, [15, 220], [volMin, volMax]) print(vol, length)
```

We call the NumPy function np.interp, to convert the hand range to the volume range. The arguments used are:

- length: This is the value we want to convert.
- [15 - 220]: This is the hand range.
- [volMin, volMax]: Giving the range to which we want to convert.

Setting the master volume

```
volume.SetMasterVolumeLevel(vol, None)
```

We are setting the master volume level following the hand range. We achieve this by passing vol, which is the value of the hand range we converted to volume range.

Displaying the video output used to interact with the user

```
cv2.imshow('Image', img)
```

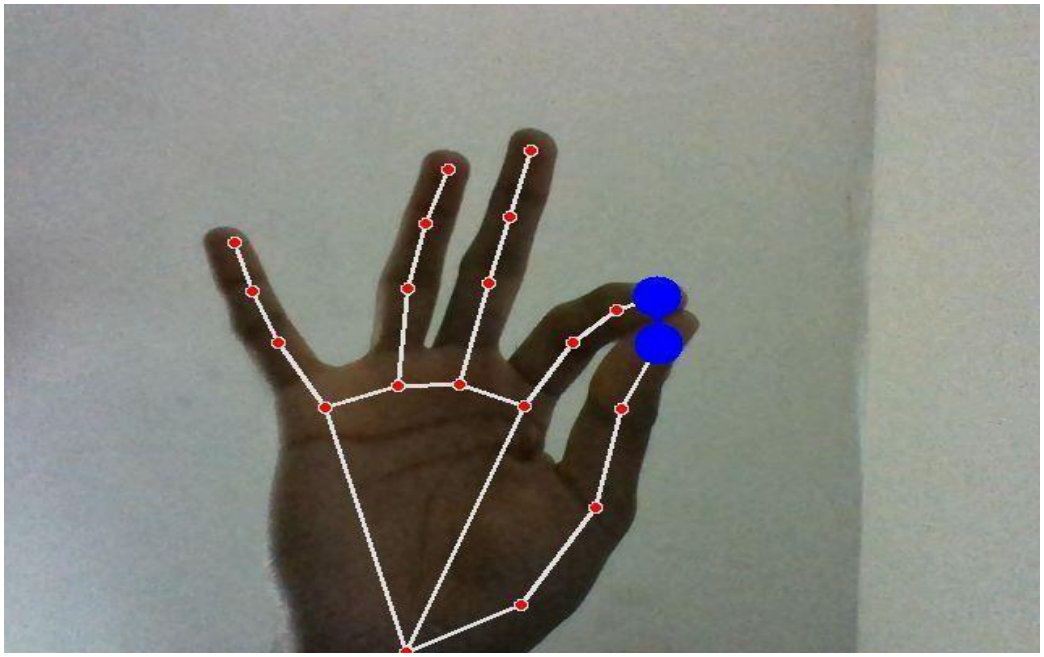
The code above shows the real-time video of the user interacting with the program, i.e., the user uses the thumb finger and the index finger to control the volume.

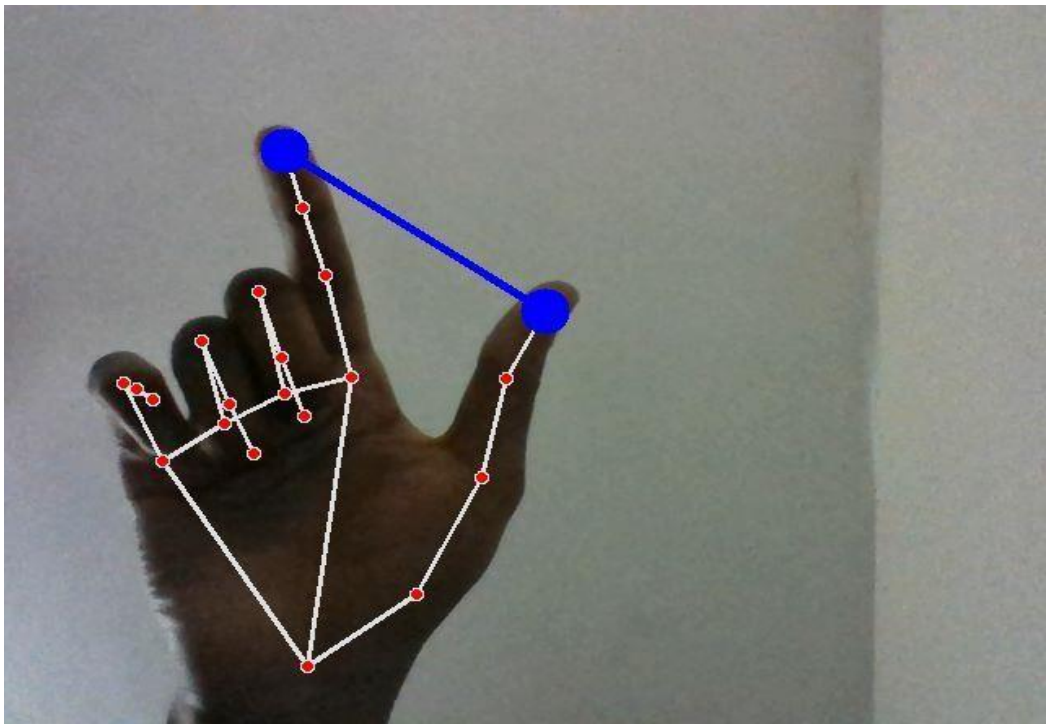
Terminating the program

```
if cv2.waitKey(1) & 0xff == ord('q'):
  break
```

Results and Analysis

As a first step we try the hand detection based on available database of OpenCV. Then for capturing live hand of Camera the initialization has been done. The two gesture detection like palm and fist by green line which is trained by integral images. The second step is the extracted image gestures which are compared with stored positive-negative integral image dataset and perform finger tip tracking by contour detection. The third step Mediapipe locate the palm and detect the 21 hand Landmarks according to Action. Mediapipe tracks the action between thumb and index finger and give command to Pycaw to maximize or Minimize the audio.





Conclusions

This project is presenting a program that allows the user to perform hand gesture for convenient and easier way to control the software .A gesture based volume controller doesn't require some specific type of markers and these can be operated in our real life on simple Personal Computers with a very low cost cameras as this not requires very high definition cameras to detect or record the hand gestures. Specifically, system tracks the tip positions of the counters and index finger of each hand. The main motive of this type of system is basically to automate the things in our system in order to make the things become easier to control. So in order to make it

realiable we have used this system to make the system easier to control with the help of these applications.