VOLUME CONTROL
USING HAND GESTURES

PROJECT

Submitted by

Sanjana Ravishankar (20BCE1820)

VIT
Vellore Institute of Technology

TABLE OF CONTENTS

ABSTRACT:

Hand gesture recognition is one of the active research areas in the field of human-computer interface due to its flexibility and user friendliness.
The gesture recognition technique is used to develop a system that can be used to convey information among disabled people or for controlling device.

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 recognize the gesture taken as input by the user.

This Vision Based method requires a web camera, so that one can realize natural interaction between humans and computer without using any other devices.

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 have a wide area of application in human computer interaction.

LIST OF FIGURES

LIST OF ABBREVIATIONS

NOTATION	DESCRIPTION
OpenCV	Open Source Computer Vision Library
Pycaw	Python Core Audio Windows Library
HCI	Human Computer Interaction
OCR	Optical Character Recognition
INS	Inertial navigation system
NumPy	Numerical Python

MDA	Model Driven Architecture
EM	Expectation-Maximization algorithm
HGR	Hand Gesture Recognition
RGB	red, green and blue
MPHANDS	MediaPipe-hands
MPDRAW	MediaPipe-draw
FEMD	Finger-Earth Mover's Distance

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW:

- In this project, we are developing a volume controller in which we are using hand gestures as the input to control the system wherein OpenCV module is basically used to implement the control 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.

1.2 CHALLENGES:

- Not all humans are able to access the technology, computer systems, operating systems due to the complexities of the functionalities of these systems.
- If we investigate the problem, particularly we'll notice that senior citizens, people with vision problems, kids, people with disabilities, or general people with less dexterity in computer applications are not able to perform even the simple functions of an operating device.
- Studies also show that people with disabilities are less likely to use technology than people who do not have disabilities.
- This happens due to the non-inclusivity of measures into normal day to day technology which prevents especially abled citizens from using the technology that a normal person uses not only for leisure but also to access fundamental right benefits that are necessary so this makes it a serious issue to provide a solution for this problem.

1.3 PROBLEM STATEMENT:

- People will not have to learn machine-like skills which are a burden most of the time, especially the blind or aged people with poor eye-sight.
- This type of hand gesture system provides a natural and innovative modern way of non-verbal communication.

1.4 OBJECTIVE:

- The purpose of this project is to discuss a volume control system using hand gesture recognitions application based on detection of hand gestures.
- In this, the system consists of a high-resolution camera to recognize 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 the same input as the information for controlling the device and by using real time gesture recognition specific users 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, the volume can be controlled by hand gesture without making use of the keyboard and mouse.

1.5 SCOPE:

- Hand gestures is the powerful communication medium for Human Computer Interaction (HCI).
- Several input devices are available for interaction with computer, such as keyboard, mouse, joystick and touch screen, but these devices do not provide easier ways to communicate.
- In this, the system which is proposed will consist of a desktop and laptop interface; hand gesture can be used by the users where they need to wear data gloves and they also can use the web camera or separate cameras for recording the hand gestures.
- The first and most important step toward any hand gesture recognition system is to implement hand tracking system.
- Sensors used in this system will collect hand configuration and hand movements. The Vision Based method requires a web camera so that one can realize natural interaction between humans and computer without using any other devices.

1.6 ORGANISATION OF REPORT:

The organization of report is as follows:

Chapter 2 analyses various methods and notions from a number of international conference papers and journal papers. Chapter 3 describes System architecture and design explaining the parts present in the architecture. Chapter 4 describes the implementation. Chapter 5 describes the results and its discussion. Chapter 6 presents conclusion of the project along with the future enhancement that can be applied to the existing work.

Chapter 7 contains the appendix and screenshots. Finally, it is followed by a Reference Section containing a catalogue of the papers listed in thesis along with the authors and the year of publication

CHAPTER 2

LITERATURE SURVEY

2.1 METHODOLOGIES AND DRAWBACKS:

[1] Hand gesture recognition based on accelerometer sensors (2011)

Methodologies:

This paper represented two implementations of gesture recognizing systems: sensor-based and vision-based. The sensor-based solutions use accelerometers or gyros for detecting the gestures. This paper presents a hand gesture recognition system built around an accelerometer sensor. The system is made by a sensing and transmitting part and a computer. The computer receives the data describing the motion trajectory and interprets it, through OCR, as a letter. It can be used in a more complex human-computer friendly interface, in which the commands are given by gestures. Without the OCR part, the system can be also used in gaming systems, embedded systems, intelligent peripherals and so on.

Drawbacks:

The use of physical sensors makes the system design more complicated and becomes expensive.

[2] Hand Gesture Recognition Using Hidden Markov Model Algorithm (2017)

Methodologies:

The objective of this project is to recognize human hand gesture through the use of INS sensor with the approach of statistical pattern recognition tool. It is assumed that hand gesture would be detected by the fluctuation of acceleration and the difference of hand-position in 3-axis. Therefore, this project employed INS sensor to admit the information of human hand gesture.

Drawbacks:

It becomes very complicated when more states and more interactions among states are included. This complexity becomes particularly problematic in presence of time-dependent probabilities.

[3] Robust Part-Based Hand Gesture Recognition Using Kinect Sensor (2013)
Methodologies:

This paper focuses on building a robust part-based hand gesture recognition system using Kinect sensor. To handle the noisy hand shapes obtained from the Kinect sensor, we propose a novel distance metric, Finger-Earth Mover's Distance (FEMD), to measure the dissimilarity between hand shapes. As it only matches the finger parts while not the whole hand, it can better distinguish the hand gestures of slight differences.

Drawbacks:

The hand detection provided from Kinect sensor has errors since it is usually in the inferred state. Also, it has sensitivity to sunlight factors.

[4] Brightness factor matching for gesture recognition system using scaled normalization (2011)
Methodologies:

In this paper, they have introduced a new gesture recognition system based on image blocking and the gestures are recognized using their suggested brightness factor matching algorithm and they have applied two different feature extraction techniques, the first one based on features extracted from edge information and the other one based on a new technique for center of mass normalization based on block scaling instead of coordinates shifting. Also, they have achieved 83.3% recognition accuracy in first technique with significant and satisfactory recognition time of 1.5 seconds per gesture and 96.6 % recognition accuracy with recognition time less than one second by eliminating the use of edge detector which consumes time.

This paper focused on appearance-based gestures.

Drawbacks:

Scaled normalization increases the complexity and number of tables and relationships, which can make the data model harder to understand and manage.

[5] A Gesture Interface for Manipulating On-Screen Objects (2007)
Methodologies:

This paper presented a gesture-based interaction system which provides a natural way of manipulating on-screen objects. They generate a synthetic image by linking images from two cameras to recognize hand gestures. The synthetic image contains all the features captured from two different views, which can be used to alleviate the self-occlusion problem and improve the recognition rate. The MDA and EM algorithms are used to obtain parameters for pattern classification. To compute more detailed pose parameters such as fingertip positions and hand contours in the image, a random sampling method is introduced in this system. They describe a method based on projective geometry for background subtraction to improve the system performance.

Drawbacks:

The EM algorithm has slow convergence thus making the system not that efficient.

[6] Hand Gestures Recognition Using Radar Sensors for Human-Computer-Interaction: A Review (2021)
Methodologies:

In this article, they present the first ever review related to HGR using radar sensors. They review the available techniques for multi-domain hand gestures data representation for different signal processing and deep-learning-based HGR hand gestures data representation for different signal processing and deep-learning-based HGR hand gestures data representation for different signal processing and deep-learning-based HGR.

Drawbacks:

The radar sensor has a limited field of vision in the systems.

2.2 SUMMARY

Thus, different methodologies and their disadvantages of various papers on volume control using hand gestures are discussed in this literature survey.

CHAPTER 3

PROPOSED WORK

3.1 INTRODUCTION:

- In this project we are using python technology to develop the project; the code is written and designed in python language using Opencv 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 used in this project that 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 we use mphand.hands module to detect the gesture .
- Then, in order to access the speaker, we have to use the pycaw and provide 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 next is to specify the points of thumb in input and fingers. Volume range is processed using the hand range and Numpy is used to convert this process and process the required output. NumPy package is the fundamental package for computing in Python language.

3.2 SYSTEM DESIGN:

Description of each of the modules:

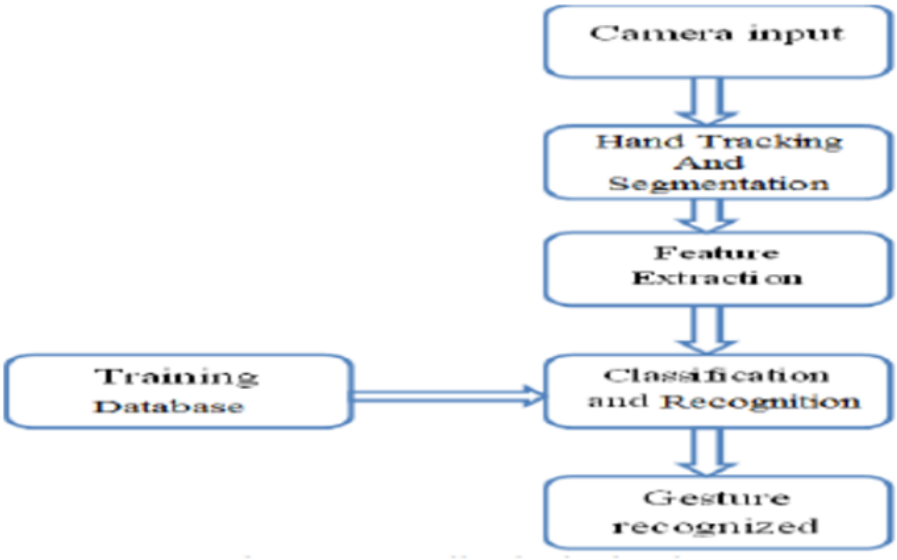
Open CV - Open CV is a library of python which tackles PC vision issue. It also performs object detection and motion detection. It also supports several types of operating systems.

NumPy - NumPy is the module of the Python. The “numpy” word basically shows Numerical Python and it is utilized. Numpy guarantees remarkable execution speed. Numpy is mostly used for performing calculations, tasks using certain functions it provides like multiply, divide, power etc.

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].

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 , to recognize the hand gesture and detect the input from it as in Face Detection,Multi-hand Tracking,Segmentation and Object Detection and Tracking.

3.3 SYSTEM ARCHITECTURE:



Source: https://ijisrt.com/assets/upload/files/IJISRT22MAY250_(1).pdf

In this project, some algorithms and some modules have been used to detect the gestures of the person and these gestures are taken as the input in the system. Here, several modules are used like OpenCV-python, mediapipe, NumPy etc., for the purpose of tracking the hand gestures.
After capturing the input from the user, the image is used in the hand tracking system to check the dimensions and shape of the gesture which is received in the system.
Hand tracking module plays an important role in identifying the input recorded in the system, after that classification and segmentation process is used to classify the gestures in the system. Deep learning and Machine Learning are also used to identify the training data from the system and identify it according to the requirement of the system.
After this the gestures are identified from the trained data and on the basis of that data the gestures are recognized and is used for processing of the system to implement the functions like increase and decrease in volume.

Here, we have performed the Hand Gestures recognition system to produce the better output, webcam is enabled while executing the program, also the type of gesture used is static to recognize the shape of the hand and it provides us the required output.
In this project the volume is controlled based on the shape of hand.
The system takes input and will capture the object, detects and after that hand gesture recognition is performed.

3.4 SUMMARY:

The chapter proposed work contains architecture of proposed system and brief explanation of the proposed system.

CHAPTER 4

IMPLEMENTATION

4.1 VOLUME CONTROL SYSTEM USING HAND GESTURES WITH CAMERA

PROCEDURAL STEPS:

Input: The camera in our device.
Output: A hand image showing the numbers of the points that MediaPipe uses to refer to different points of the hand.

Step 1: Importing the python libraries we will need – CV2, mediapipe, pycaw, python-math, Ctypes, Comtypes and NumPy.

Step 2: 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.

Step 3: We then detect, initialize and configure the hands from the video input we got from our primary camera. For this, we are calling on the mediapipe hand module. 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.

Step 4: Access the speaker using pycaw.

Step 5: Finding the volume range between the minimum and maximum volume.

Step 6: Capturing an image from our camera and converting it to an RGB image.

Step 7: Checking whether we have multiple hands in our input:

We create 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.

<u>Step 8:</u> Creating a for loop to manipulate each hand:

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.

1. **h, w= img.shape:** This line of code checks the height and width of our image. This will give us the width and height of the image.
2. **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.

3. **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.

<u>Step 9:</u> Specifying the points of the thumb and middle finger we will use:

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 of the code.

Step 10: Drawing a circle between the tip of the thumb and the tip 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 colour of the circle. cv2.FILLED refers to the thickness of -1 pixels which will fill the circle with the colour we specify.

Step 11: 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.

Step 12: To find the distance between the tip of the thumb and the index finger (points 4 and 8) 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***.

Step 13: Converting the hand range to the volume range:

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.
- [30- 350]: This is the hand range.
- [volMin, volMax]: Giving the range to which we want to convert.

Step 14: Setting the master volume level.

Step 15: Displaying the video output used to interact with the user.

Step 16: Terminating the program:

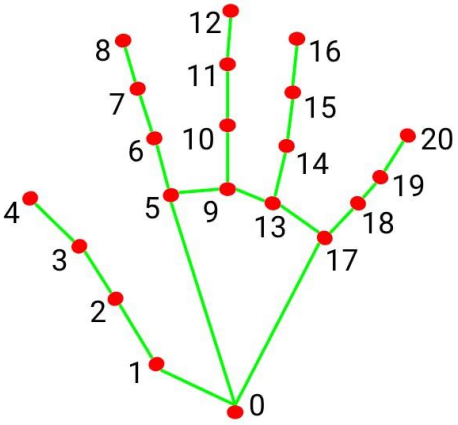
When the user presses the space bar, the program will terminate.

4.2 PERFORMANCE ANALYSIS METRICS:

For analyzing and performing models and methods certain performance metrics are used.

4.2.1 PERFORMANCE PARAMETERS:

A. APPEARANCE OF THE BELOW IMAGE ON THE SCREEN:



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

Image – MediaPipe hand landmarks

Source: https://developers.google.com/static/mediapipe/images/solutions/hand-landmarks.png

If the above image appears for our hand on the screen on running the program and the system volume can be seen increasing and decreasing according to the finger movements and the logic of the code, then the system is performing well and this can be considered as a performance metric.

B. DETECTION OF THE HAND LANDMARKS AND CALCULATING APPROPPRIATE DISTANCES:

The system maps the distance of thumb tip and index finger tip with volume range correctly.

The distance between the points 4 and 8 is directly proportional to the volume of device and was calculated accurately.

In this 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.

C. EXITING THE PROGRAM:

The program terminates on clicking the spacebar.

D. OVERALL ACCURACY:

The system has 100% accuracy.

E. PRECISION

The system calculates parameters like hand range, volume range, etc., to 100% precision and also the length of the hypotenuse can be seen directly proportional to the volume of the device.

4.3 SUMMARY:

This chapter contains the implementation the model and analyzes the performance of the implemented model.

CHAPTER 5

EVALUATION AND RESULTS

5. 1 VOLUME CONTROL USING HAND GESTURES

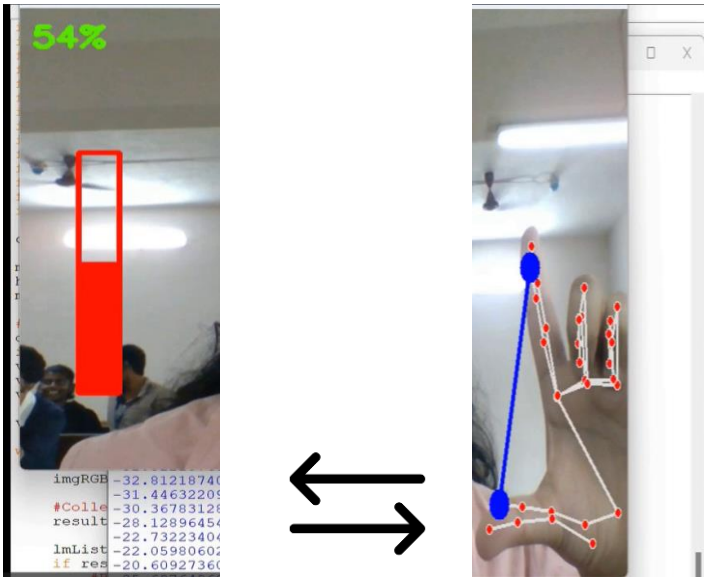


Image: Percentage bar based on hand movement

Result: The percentage bar can be seen in the above image and it was calculated according to the movement of my finger gestures which can be seen on the right-hand side image (the distance between the points 4 and 8 is directly proportional to the volume of device).

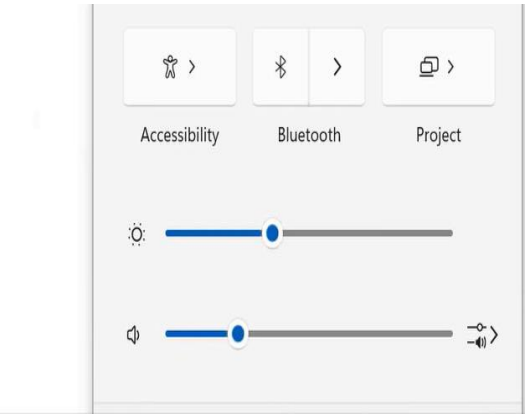


Image – System speaker volume

Result: In order to access the speaker, we used the pycaw library and provide the range of the volume from minimum volume to maximum volume.

In order to test the whole system, a song was played in the background and the volume of the song kept either increasing or decreasing based on the hand movement.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION:

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 (PCs) with a very low-cost cameras as this not requires very high-definition cameras to detect or record the hand gestures.

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

6.2 FUTURE WORK:

The model successfully controls the volume of the system by hand gesture, that means the volume of the device is controlled without opening the volume-settings or the volume icon in the system. This model successfully controls the volume of a device without a mouse, and without having to open the volume icon or the volume settings. The model uses the index finger and the thumb of our hand, and a camera. The model uses hand landmarks defined by Google‟s Mediapipe library, and the ctypes library, for seamless execution. After which we can access the audio functions inside the system and based on the distance between the thumb and the index finger there will be alteration in the volume.

The scope of this project can be extended to be used for longer distances between the user and the system.

APPENDIX 1

PROGRAM CODE

```
import cv2
import mediapipe as mp
from math import hypot
from ctypes import cast, POINTER
from comtypes import CLSCTX_ALL
from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume
import numpy as np
import cv2
import mediapipe as mp
from math import hypot
from ctypes import cast, POINTER
from comtypes import CLSCTX_ALL
from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume
import numpy as np

cap = cv2.VideoCapture(0) #Checks for camera

mpHands = mp.solutions.hands #detects hand/finger
hands = mpHands.Hands()   #complete the initialization configuration of hands
mpDraw = mp.solutions.drawing_utils

#To access speaker through the library pycaw
devices = AudioUtilities.GetSpeakers()
interface = devices.Activate(IAudioEndpointVolume._iid_, CLSCTX_ALL, None)
volume = cast(interface, POINTER(IAudioEndpointVolume))
volbar=400
volper=0

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

while True:
```

```
    success,img = cap.read() #If camera works capture an image
    imgRGB = cv2.cvtColor(img,cv2.COLOR_BGR2RGB) #Convert to rgb

    #Collection of gesture information
    results = hands.process(imgRGB) #completes the image processing.

    lmList = [] #empty list
    if results.multi_hand_landmarks: #list of all hands detected.
        #By accessing the list, we can get the information of each hand's corresponding flag bit
        for handlandmark in results.multi_hand_landmarks:
            for id,lm in enumerate(handlandmark.landmark): #adding counter and returning it
                # Get finger joint points
                h,w,_ = img.shape
                cx,cy = int(lm.x*w),int(lm.y*h)
                lmList.append([id,cx,cy]) #adding to the empty list 'lmList'
            mpDraw.draw_landmarks(img,handlandmark,mpHands.HAND_CONNECTIONS)

    if lmList != []:
        #getting the value at a point
                    #x      #y
        x1,y1 = lmList[4][1],lmList[4][2]  #thumb
        x2,y2 = lmList[8][1],lmList[8][2]  #index finger
        #creating circle at the tips of thumb and index finger
        cv2.circle(img,(x1,y1),13,(255,0,0),cv2.FILLED) #image #fingers #radius #rgb
        cv2.circle(img,(x2,y2),13,(255,0,0),cv2.FILLED) #image #fingers #radius #rgb
        cv2.line(img,(x1,y1),(x2,y2),(255,0,0),3)  #create a line b/w tips of index finger and thumb

        length = hypot(x2-x1,y2-y1) #distance b/w tips using hypotenuse
    # from numpy we find our length,by converting hand range in terms of volume range ie b/w -63.5 to 0
        vol = np.interp(length,[30,350],[volMin,volMax])
        volbar=np.interp(length,[30,350],[400,150])
        volper=np.interp(length,[30,350],[0,100])



        print(vol,int(length))
        volume.SetMasterVolumeLevel(vol, None)
```

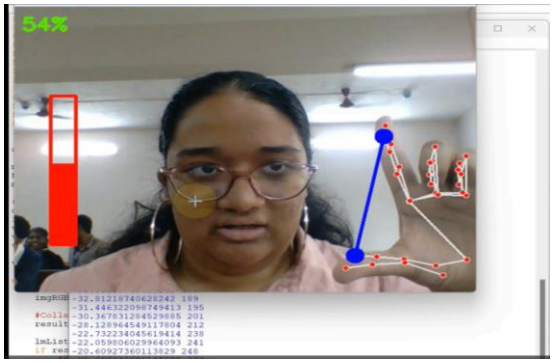
```
        # Hand range 30 - 350
        # Volume range -63.5 - 0.0
        #creating volume bar for volume level
        cv2.rectangle(img,(50,150),(85,400),(0,0,255),4) # vid ,initial position ,ending position ,rgb ,thickness
        cv2.rectangle(img,(50,int(volbar)),(85,400),(0,0,255),cv2.FILLED)
        cv2.putText(img,f'{int(volper)}%',(10,40),cv2.FONT_ITALIC,1,(0, 255, 98),3)
        #tell the volume percentage ,location,font of text,length,rgb color,thickness
    cv2.imshow('Image',img) #Show the video
    if cv2.waitKey(1) & 0xff==ord(' '): #By using spacebar delay will stop
        break

cap.release()     #stop cam
cv2.destroyAllWindows()          #closewindow
```

APPENDIX 2

SCREENSHOTS

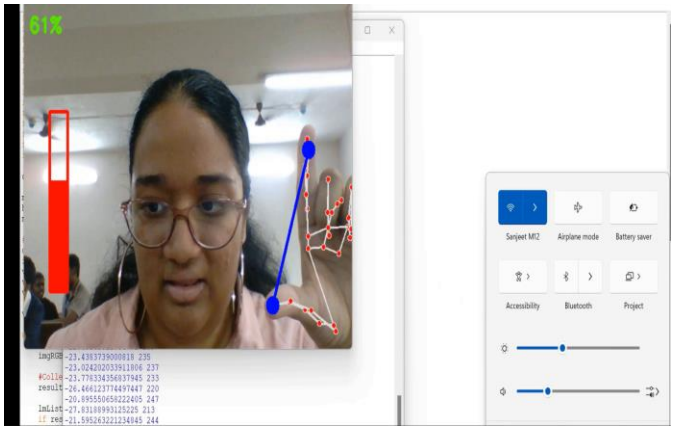
Image - 1



Library used: Mediapipe

Process that happened: Video inputs were given from our primary camera. Mediapipe detected the video as the input from our camera and mphand.hands module was used to detect the gesture, here.

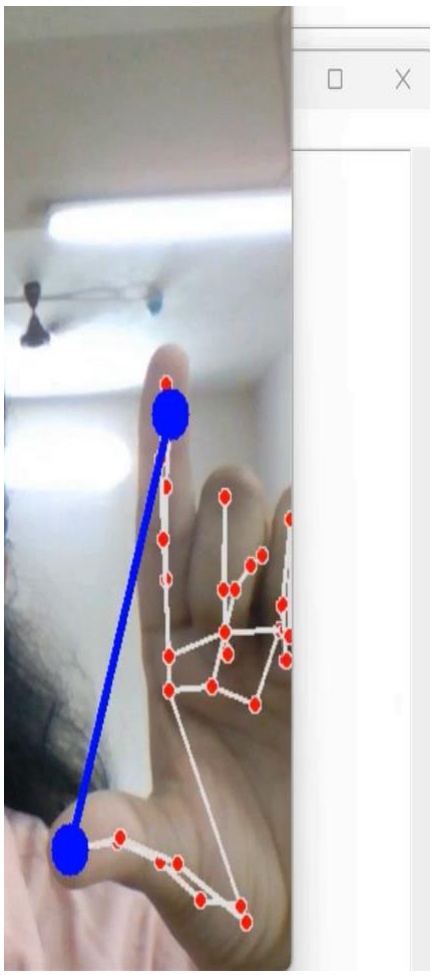
Image - 2



Library used: pycaw

Process that happened: In order to access the speaker we used the pycaw library and provide the range of the volume from minimum volume to maximum volume. Next step was to convert the input image to rgb image to complete the processing of the input captured.

Image - 3

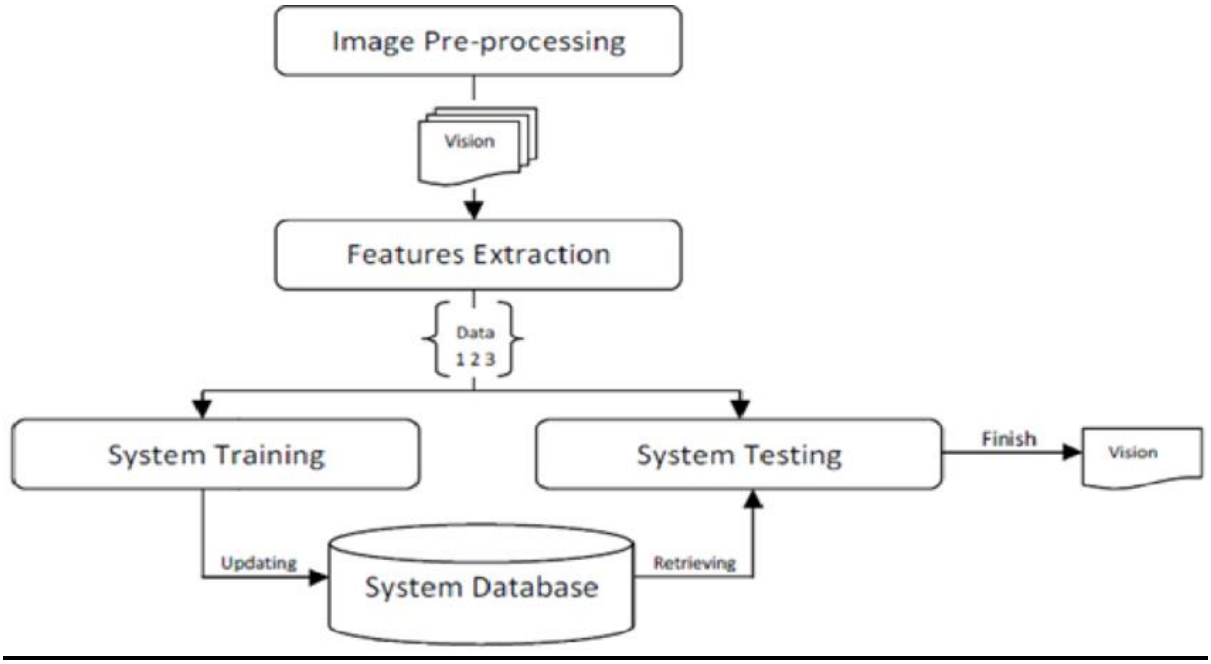


Library used: Numpy

Process that happened: Volume range was

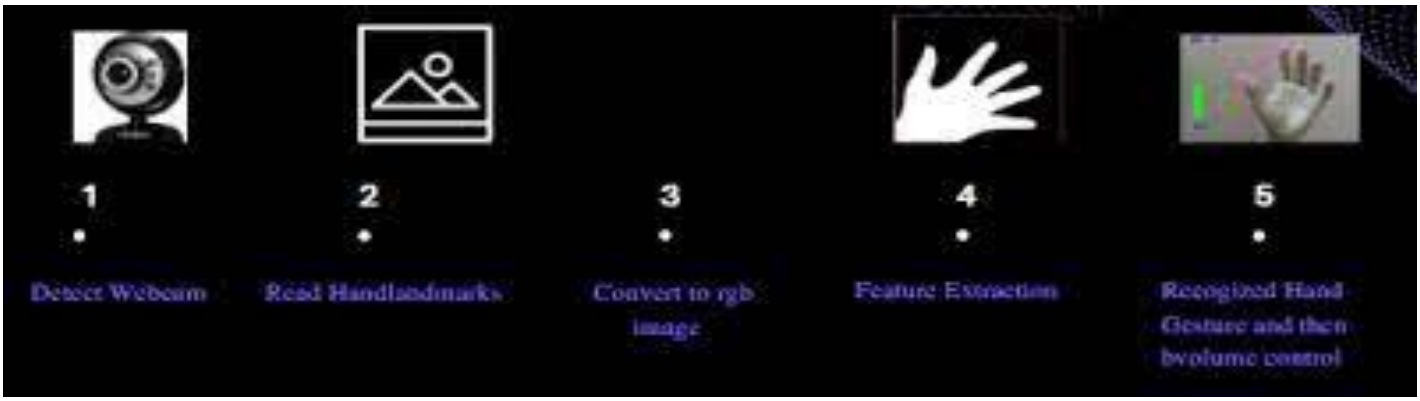
processed using the hand range and Numpy was used to convert this process and process the required output.

Flow diagram



Source: *https://www.researchgate.net/publication/284626785/figure/fig4/AS:393491688509453@1470827137439/Architecture-of-gesture-recognition-system-5.png*

System design

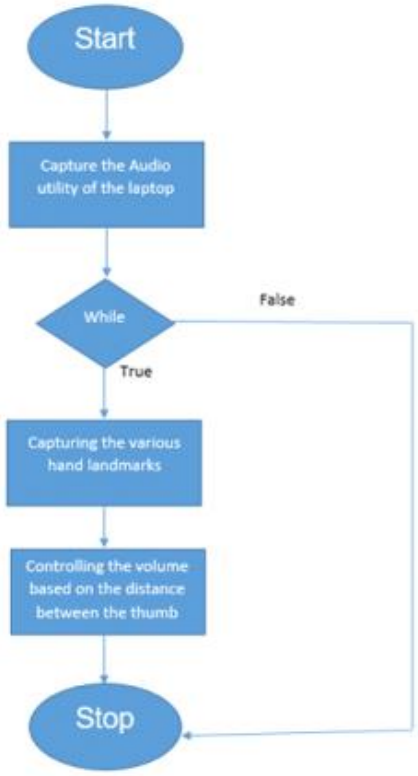


Source: https://ijcrt.org/papers/IJCRT22A6059.pdf

Description:

Here our input is hand gesture which is captured using web camera. Then the GUI (graphical user interface) helps to display the hand that conveys information and it processes the actions of the hand gestures of the user. By recognizing the gestures, the user is able to control the volume of the system which is final output.

Flow chart of the system



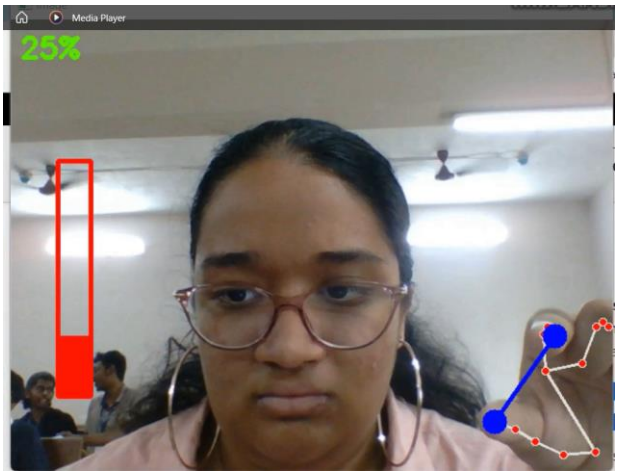
Source: *Google*

Description:

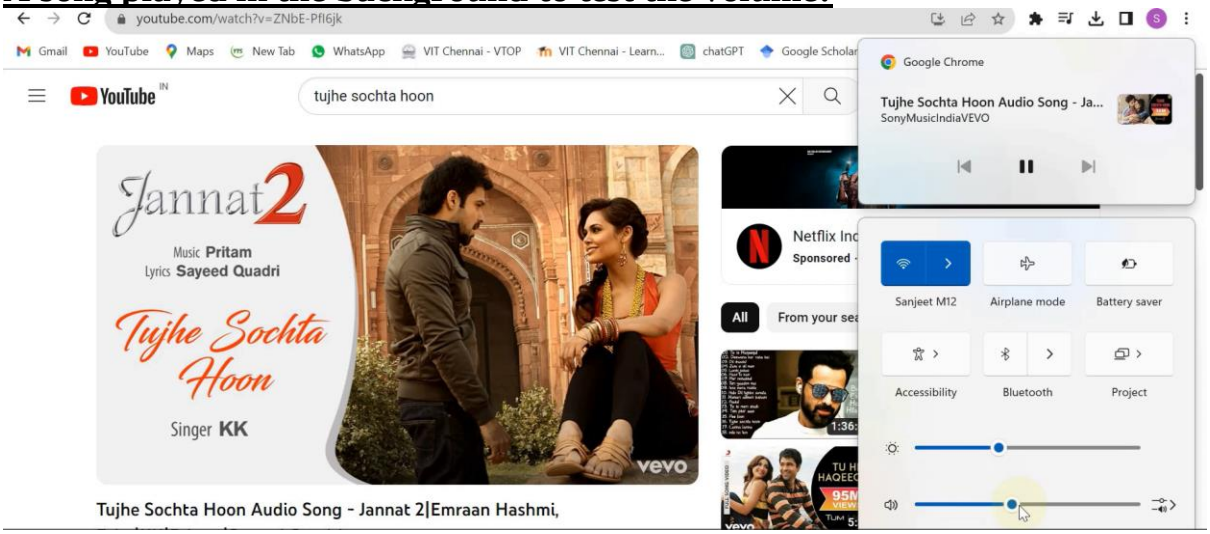
The above flow chart indicates the flow chart of operation. Firstly, initiate start the program and then import various modules used for the AI recognition and various audio utilities which are of main concern. Next, it captures the area of interest by detecting various contours. Later it executes the loop to detect various hand landmarks. After getting the hand land marks, it verifies the distance between the thumb and index figure tip. The frame is displayed giving the final values of the reading with complete decrease and increase in the volume using AI. The program is executed till the loop is iterated. Once it completes the iterations it comes out of the loop and the program stops.

Further experimentation with the system

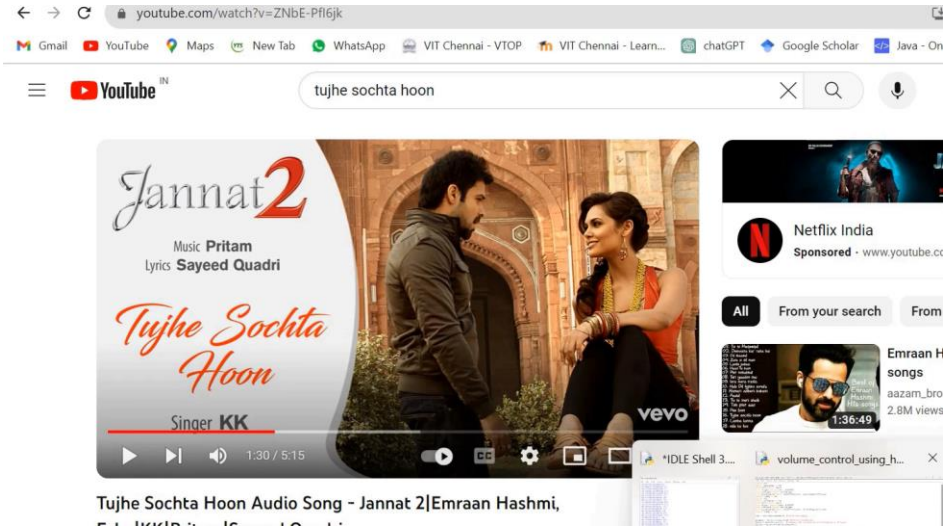
Decrease in the volume:



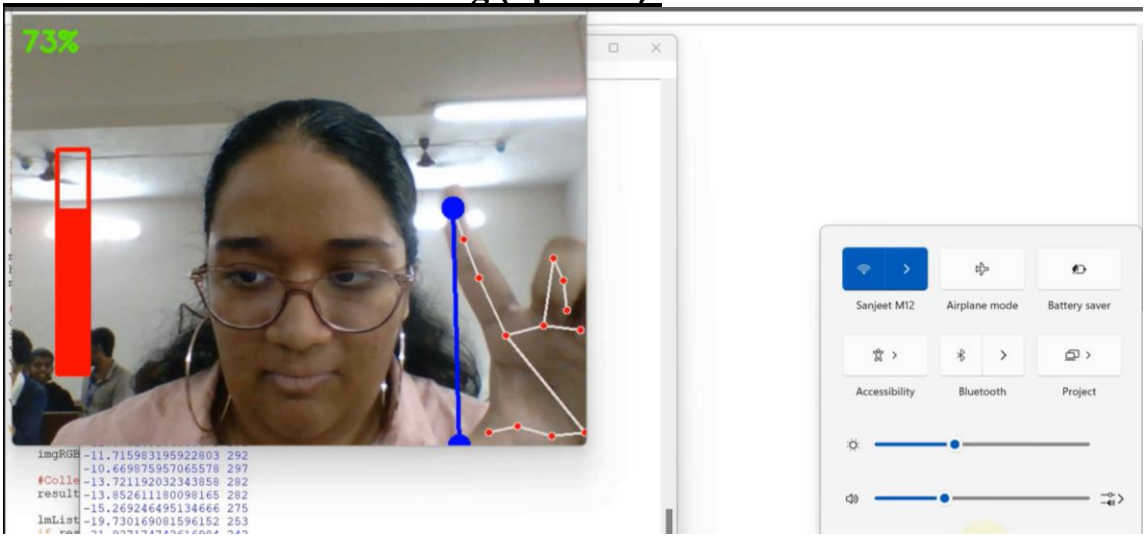
A song played in the background to test the volume:



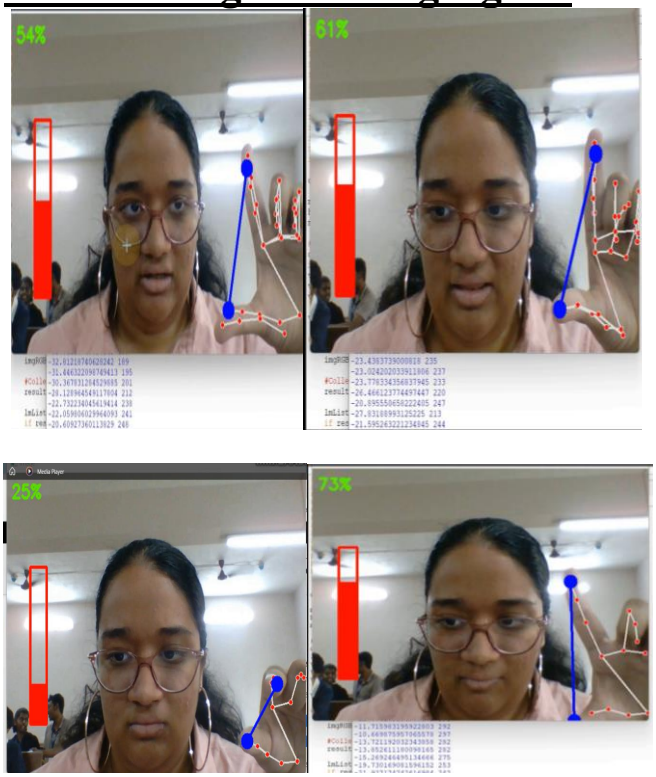
The program can also be seen simultaneously running in the system while the song is played:



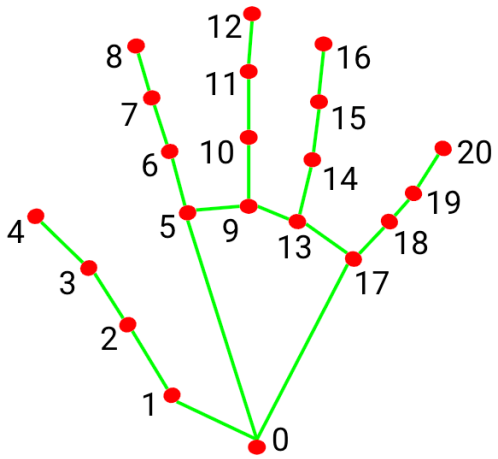
Volume can be seen increasing (upward):



Volume ranges in a single grid:



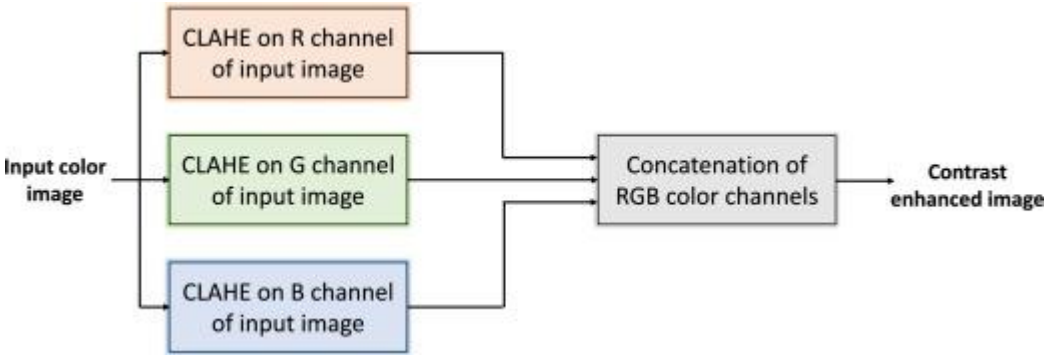
Mediapipe for real-time hand tracking:



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

Source: https://developers.google.com/static/mediapipe/images/solutions/hand-landmarks.png

Image - Histogram Equalization (image filtering)



Source: https://ars.els-cdn.com/content/image/3-s2.0-B9780323858458000137-f08-02-9780323858458.jpg

This was used to filter the images using histogram and convert them into the rgb in order to process the image in our system.

Technologies used in this project

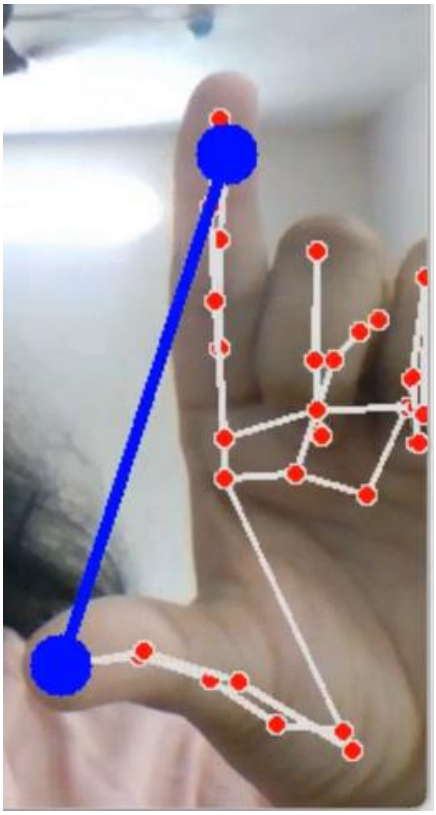
(Images' source: Google)

The following libraries of Python were used:

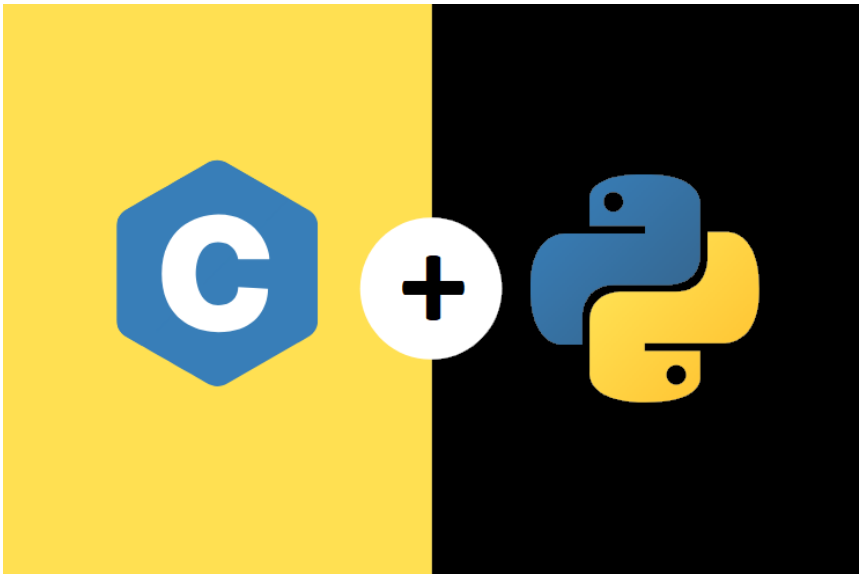
1.MediaPipe:

2.OpenCV:

3.Pycaw:



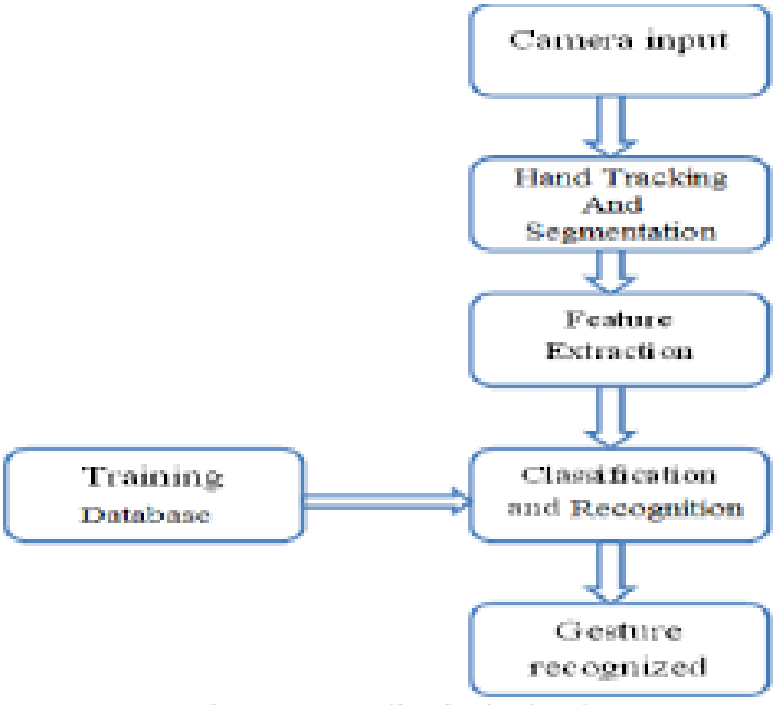
4.ctypes (in python)



5.NumPy

The fundamental package for computing in Python language.

Basic architecture of this system:



Source: Google

CHAPTER 8

REFERENCES

REFERENCES:

[1] M. Popa, "Hand gesture recognition based on accelerometer sensors," The 7th International Conference on Networked Computing and Advanced Information Management, Gyeongju, Korea (South), 2011, pp. 115-120.

[2] Melani, Roro. (2017). Hand Gesture Recognition Using Hidden Markov Model Algorithm. MATICS. 9. 7. 10.18860/mat.v9i1.4126.

[3] Ren, Zhou & Yuan, Junsong & Meng, Jingjing & Zhang, Zhengyou. (2013). Robust Part-Based Hand Gesture Recognition Using Kinect Sensor. Multimedia, IEEE Transactions on. 15. 1110-1120. 10.1109/TMM.2013.2246148.

[4] Hasan, Mokhtar & Misra, Pramoud. (2011). Brightness Factor Matching for Gesture Recognition System Using Scaled Normalization. International Journal of Computer Science and Information Technology. 3. 10.5121/ijcsit.2011.3203.

[5] Li, Shanqing & Lv, Jingjun & Xu, Yihua & Jia, Yunde. (2007). EyeScreen: A Gesture Interface for Manipulating On-Screen Objects. 710-717. 10.1007/978-3-540-73110-8_77.

[6] Ahmed, Shahzad & Kallu, Karam Dad & Ahmed, Sarfaraz & Cho, Sung Ho. (2021). Hand Gestures Recognition Using Radar Sensors for Human-Computer-Interaction: A Review. Remote Sensing. 13. 527. 10.3390/rs13030527.

Other references include:

https://ijisrt.com/assets/upload/files/IJISRT22MAY250_%281%29.pdf

https://www.eurchembull.com/uploads/paper/835bd6081fe42bb2adb7bdc8594c9f8a.pdf

https://ijcrt.org/papers/IJCRT2305507.pdf

https://towardsdatascience.com/histogram-equalization-5d1013626e64

https://ars.els-cdn.com/content/image/3-s2.0-B9780323858458000137-f08-02-9780323858458.jpg

https://developers.google.com/static/mediapipe/images/solutions/hand-landmarks.png

https://www.researchgate.net/publication/284626785/figure/fig4/AS:393491688509453@1470827137439/Architecture-of-gesture-recognition-system-5.png