##### A Project report

##### on

HAND GESTURE VOLUME CONTROLLER

###### A Dissertation submitted to JNTU Hyderabad in partial fulfillment of the academic requirements for the award of the degree.

**Bachelor of Technology**

**in**

**Computer Science and Engineering**

Submitted by

Team Members: Under the esteemed guidance of

##### (N.Raju-19H51A05E1) (Mr.V.Narasimha)

##### (P.SaiCharan-19H51A05E2) (Assistant Professor)

##### (R.Adithya-19H51A05E6)



**Department of Computer Science and Engineering**

**CMR COLLEGE OF ENGINEERING & TECHNOLOGY**

(An Autonomous Institution under UGC & JNTUH , Approved by AICTE, Permanently Affiliated to JNTUH,Accredited by NBA)

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

2019-2023

**CMR COLLEGE OF ENGINEERING & TECHNOLOGY**

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



#### CERTIFICATE

This is to certify that the Mini Project-1 report entitled **HAND GESTURE VOLUME CONTROLLER** being submitted by **N.Raju *(19H51A05E1),* P.SaiCharan *(19H51A05E2)*, R.Adithya *(19H51A05E6)*** in partial fulfillment for the award of **Bachelor of Technology in Computer Science and Engineering** is a record of bonafide work carried out his/her under my guidance and supervision.

###### The results embodies in this project report have not been submitted to any other University or Institute for the award of any Degree.

**Mr.V.Narasimha Dr. K Vijaya kumar**

**Assistant professor Professor and HOD**

**Dept. of CSE Dept. of CSE**

**TABLE OF CONTENTS**

**CHAPTER NO TITLE PAGE NO**

**1 INTRODUCTION**  05

1.1 Hand Tracking in Real Time 06

|  |  |
| --- | --- |
|  | * 1. Importance of Media Pipe 06 |

**2 BACKGROUND WORK**

2.1 Domain Introduction 06

2.1.1 PyCharm IDE

2.2.2 Essential Libraries

2.2 Media Pipe 07

* + 1. Palm Detection

2.2.2 Hand LandMarking

2.3 Open CV 08

**3 EXISTING SOLUTIONS**

3.1 ANN for Gesture Recognition using 09

Accelerometer Data

3.2 Combining multiple depth-based 09

descriptors for hand gesture recognition

**4 PROPOSED SOLUTIONS**

4.1Hand Gesture Volume Controller 10

**5** **DESIGNING**

5.1 SYSTEM ARCHITECTURE 11

**6 SOURCE CODE**

6.1 Hand Tracking Module 12

6.2 Node Tracking Module 14

6.3 Volume Controller Module 16

**7** **RESULTS AND DISCUSSION**

6.1 Implementation 17

**8** **CONCLUSION AND FUTURE WORK**  20

**9 REFERENCES** 21

#### Abstract

There are several ways of communication for understanding the information and processing. For human interaction we use verbal and non verbal communication. Non verbal communication include sign language. Hand gesture is also one of the sign language. It has a wide area of application in human computer interaction and sign language. The intention of this project is to recognize the hand gesture moments and utilize those moments for human computer interaction. The main goal of gesture recognition is to create a system which can identify specific human gestures and use those detections by device for understanding , recognizing ,performing tasks on the information for human computer interaction .

In this project we are going to create a system which could be used to control the volume of a device by using simple hand gestures without using mouse or keyboard.

# **1. INTRODUCTION**

**1.1 Hand Tracking in Real Time**

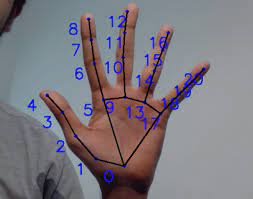
Hand gestures are powerful and spontaneous powerful mode of communication which are useful for human computer interaction .Several input devices are available for interaction with computer, such as keyboard, mouse, joystick as well as touch screen; however they are not advanced interface. Here we proposed system which acts as a interface, through which hand gesture used as input tasks instead of input devices. Web camera is used capturing the hand image and tracking it. The hand gesture should be recognized for hand tracking and segmentation.

This is a Vision Based method which require only a camera for human computer interaction without the use of any extra devices. Here Computer Vision acts as an artificial vision source or sensor for the device for tracking the hand posture and gesture recognition which can be implemented with various algorithm software and/or hardware.

**1.2 Importance of Media pipe**

It is a difficult task for shape and motion detection but it can be easy task by improving the user experience across a variety of methodologies and new domain technology .For understanding the sign language ,the device should have basic information of the hand. Basic information includes palm/hand knuckles. There should be digital content and information on top of the physical world in Augmented reality. This is the real-time and perception challenging task for computer vision as hands often include themselves or each other( Example finger/palm occlusions and handshakes).

This effort can be easily implemented by Media pipe. Media pipe provides the basic data for hand and finger tracking solution. It is an approach of machine learning for detecting landmarks of a hand form by just a single frame. This method achieves real-time performance on a mobile phone and even skills to multiple hands.



**2. BACKGROUND WORK**

**2.1 Domain Introduction**

**2.1.1 PyCharm**

PyCharm is a cross-platform IDE that provides consistent experience on the Windows, macOS, and Linux operating systems.

PyCharm is available in three editions: Professional, Community, and Edu. The Community and Edu editions are open-source projects and they are free, but they have fewer features. PyCharm Edu provides courses and helps you learn programming with Python. The Professional edition is commercial, and provides an outstanding set of tools and features

**2.1.2 Numpy**

NumPy (Numerical Python) is an open source Python library that’s used in almost every field of science and engineering. It’s the universal standard for working with numerical data in Python, and it’s at the core of the scientific Python and PyData ecosystems. NumPy users include everyone from beginning coders to experienced researchers doing state-of-the-art scientific and industrial research and development. The NumPy API is used extensively in Pandas, SciPy, Matplotlib, scikit-learn, scikit-image and most other data science and scientific Python packages.

The NumPy library contains multidimensional array and matrix data structures (you’ll find more information about this in later sections). It provides ndarray, a homogeneous n-dimensional array object, with methods to efficiently operate on it. NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

**2.1.3 MediaPipe**

MediaPipe is a framework for building multimodal (eg. video, audio, any time series data), cross platform (i.e Android, iOS, web, edge devices) applied ML pipelines. With MediaPipe, a perception pipeline can be built as a graph of modular components, including, for instance, inference models (e.g., TensorFlow, TFLite) and media processing functions.

**Cutting edge ML models**

Face Detection

Multi-hand Tracking

Hair Segmentation

Object Detection and Tracking

Objectron: 3D Object Detection and Tracking

AutoFlip: Automatic video cropping pipeline

**Cross Platform ML solutions**

Build once, deploy anywhere. Works optimally across mobile (iOS, Android), desktop server and the Web

**Ondevice ML Acceleration**

Performance optimized end-to-end ondevice inference with ML acceleration for mobile GPU & EdgeTPU compute

**2.2 MediaPipe**

**2.2.1 Palm Detection**

Similar to the face detection model in MediaPipe Face Mesh ,we need to create a software for palm detection. It is a complex task for detection of palm,hands .We need to create a image frame and be able to detect occluded and self-occluded hands.The detection should be restricted to only palm and hands instead of arms ,face etc.Each and every moment of the palm/hand should be tracked inside the frame.

Inorder,to over come above headover obstacles we come up with a solution with efficient methodology. As the first step, we trained a palm detector instead of a hand detector, since estimating bounding boxes of rigid objects like palms and fists is significantly simpler than detecting hands with articulated fingers.As palms are smaller objects, the non-maximum suppression algorithm works well even for two-hand self-occlusion cases, like handshakes. Moreover, palms can be modelled using square bounding boxes (anchors in ML terminology) ignoring other aspect ratios, and therefore reducing the number of anchors by a factor of 3-5. Second, an encoder-decoder feature extractor is used for bigger scene context awareness even for small objects (similar to the RetinaNet approach). Lastly, we minimize the focal loss during training to support a large amount of anchors resulting from the high scale variance.

With the above techniques, we achieve an average precision of 94% in palm detection. Using a regular cross entropy loss and no decoder gives a baseline of just 84.21%.

**2.2.2 Hand Landmark**

After the palm detection over the whole image our subsequent hand landmark model performs precise keypoint localization of 21 3D hand-knuckle coordinates inside the detected hand regions via regression, that is direct coordinate prediction. The model learns a consistent internal hand pose representation and is robust even to partially visible hands and self-occlusions.

To obtain ground truth data, we have manually annotated ~30K real-world images with 21 3D coordinates, as shown below (we take Z-value from image depth map, if it exists per corresponding coordinate). To better cover the possible hand poses and provide additional supervision on the nature of hand geometry, we also render a high-quality synthetic hand model over various backgrounds and map it to the corresponding 3D coordinates.

**2.3 OpenCV**

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV’s deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux,Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms.

**3. EXISTING SOLUTIONS**

**3.1 ANN for Gesture Recognition using Accelerometer Data**

The Artificial Neural network application used for the classification and gesture recognition.

In this the gesture recognition is done through the Wi-fi remote and this remote will rotate in the

different directions. In this process the computational cost and memory consumption of the gesture recognition is done in two levels. In the first level the User Authentication is done for gesture recognition using the Accelerometer-Based gesture recognition method. In the second level by not using any kind of signal processing the gesture recognition Fuzzy automata algorithm has been proposed. By recognizing the data of the gestures, the data has normalized and filtered by using

the k-means and Fast Fourier transform algorithm.

**3.2 Combining multiple depth-based descriptors for hand gesture recognition**

By using the depth information of the image taken by the depth cameras the author has introduced a scheme called a novel hand gesture recognition scheme to properly recognize complex

gestures by using 3-D information by using 3-Dimensional features.

These hand gesture recognition has done in 3 main steps:

* The first step is based on color and depth information by the hand samples which are segmented from the background. The wrist samples, palm, and the fingers are the subparts of the segmented hand samples and it consists of four types of features.
* The second step is used to extract the features from the segmentation. The first 2 set of features are based on the distance from elevation of finger tips to palm center.
* In the third step it will concentrate on the computed curvature features of hand counter.

**4. PROPOSED SOLUTIONS**

**4.1 Hand Gesture Volume Controller**

The hand gesture is the active area for the research in the vision community and the main purpose

of the hand gesture is the sign language recognition and the human interaction. It is one of the original tracking systems for the hand motion. In this system a 27 degree freedom hand can be tracked at 10Hz by using the grayscale images and requires manual initialization step before beginning the tracking. By the interaction perspective most of the hand tracking work by focusing on 2D interfaces.

Firstly a finger has been tracked by using the low cost web cameras in order to manipulate a traditional graphical interface without using keyboard and mouse. The Fingertip detection was done by fitting a conic to rounded features and local tracking of the tip.

Similarly, the infrared cameras are used to segment skin regions from background pixels in order to track two hands for interaction on a 2D tabletop display. The method they used to template matching approach in order to recognize a small set of gestures that could be interpreted as interface commands. However, no precise fingertip position information was obtained using their technique.

**5. DESIGNING**

**5.1 SYSTEM ARCHITECTURE**

## 

## 

## 6. SOURCE CODE

**6.1 HAND TRACKING MODULE**

import cv2

import mediapipe as mp

import time

cap= cv2.VideoCapture(0)

mpHands=mp.solutions.hands

hands= mpHands.Hands()

mpDraw= mp.solutions.drawing\_utils

pTime=0

cTime=0

while True:

success,img=cap.read()a

imgRGB=cv2.cvtColor(img,cv2.COLOR\_BGR2RGB )

results = hands.process(imgRGB)

#print(results.multi\_hand\_landmarks)

if results.multi\_hand\_landmarks:

for handLms in results.multi\_hand\_landmarks:

for id,lm in enumerate(handLms.landmark):

print(id,lm)

h,w,c=img.shape

cx,cy=int(lm.x\*w),int(lm.y\*h)

print(id,cx,cy)

if id ==4:

cv2.circle(img,(cx,cy),15,(255,0,255),cv2.FILLED)

mpDraw.draw\_landmarks(img,handLms,mpHands.HAND\_CONNECTIONS)

cTime = time.time()

fps = 1/(cTime-pTime)

pTime=cTime

cv2.putText(img,str(int(fps)),(10,70),cv2.FONT\_HERSHEY\_PLAIN,3,

(255,0,255),3)

cv2.imshow("Image",img)

cv2.waitKey(1)

## 6.2 NODE TRACKING MODULE

## import cv2

## import mediapipe as mp

## import time

## class handDetector():

## def \_\_init\_\_(self, mode=False, maxHands=2, detectionCon=0.5, trackCon=0.5):

## self.mode = mode

## self.maxHands = maxHands

## self.detectionCon = detectionCon

## self.trackCon = trackCon

## self.mpHands = mp.solutions.hands

## self.hands = self.mpHands.Hands(self.mode, self.maxHands, self.detectionCon, self.trackCon)

## self.mpDraw = mp.solutions.drawing\_utils

## def findHands(self, img, draw=True):

## imgRGB = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

## self.results = self.hands.process(imgRGB)

## # print(results.multi\_hand\_landmarks)

## if self.results.multi\_hand\_landmarks:

## for handLms in self.results.multi\_hand\_landmarks:

## if draw:

## self.mpDraw.draw\_landmarks(img, handLms, self.mpHands.HAND\_CONNECTIONS)

## return img

## def findPosition(self, img, handNo=0, draw= True ):

## lmList = []

## if self.results.multi\_hand\_landmarks:

## myHand = self.results.multi\_hand\_landmarks[handNo]

## for id, lm in enumerate(myHand.landmark):

## #print(id, lm)

## h, w, c = img.shape

## cx , cy = int(lm.x \* w), int(lm.y \* h)

## # print(id, cx, cy)

## lmList.append([id, cx, cy])

## if draw:

## cv2.circle(img, (cx, cy), 6, (255, 0, 0), cv2.FILLED)

## return lmList

## def main():

## pTime = 0

## cTime = 0

## cap = cv2.VideoCapture(0)

## detector = handDetector()

## while True:

## success, img = cap.read()

## img = detector.findHands(img)

## lmList = detector.findPosition(img)

## if len(lmList) != 0:

## print(lmList[1])

## cTime = time.time()

## fps = 1/(cTime-pTime)

## pTime = cTime

## cv2.putText(img, str(int(fps)), (10, 70), cv2.FONT\_HERSHEY\_PLAIN, 3,

## (255, 0, 255), 3)

## cv2.imshow("Image", img)

## cv2.waitKey(1)

## if \_\_name\_\_ =="\_\_main\_\_":

## main()

**6.3 VOLUME CONTROLLER MODULE**

import cv2

import time

import numpy as np

import RajHandTrackingModule as htm

import math

from ctypes import cast, POINTER

from comtypes import CLSCTX\_ALL

from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume

import numpy as np

####################################

wCam,hCam=640,480

####################################

cap=cv2.VideoCapture(0)

cap.set(3,wCam)

cap.set(4,hCam)

pTime=0

detector = htm.handDetector(detectionCon=0.7)

devices = AudioUtilities.GetSpeakers()

interface = devices.Activate(

IAudioEndpointVolume.\_iid\_, CLSCTX\_ALL, None)

volume = cast(interface, POINTER(IAudioEndpointVolume))

#volume.GetMute()

#volume.GetMasterVolumeLevel()

volRange = volume.GetVolumeRange()

minVol= volRange[0]

maxVol= volRange[1]

vol = 0

volBar = 400

volPer =0

while True:

success,img=cap.read()

img=detector.findHands(img)

lmList=detector.findPosition(img,draw=False)

if len(lmList)!=0:

#print(lmList[4], lmList[8])

x1, y1 = lmList[4][1], lmList[4][2]

x2, y2 = lmList[8][1], lmList[8][2]

cx, cy = (x1+x2)//2,(y1+y2)//2

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

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

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

cv2.circle(img, (cx, cy), 15, (255, 0, 255), cv2.FILLED)

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

print(length)

# Hand range 50 - 300

# Volume Range -65 - 0

vol = np.interp(length,[50,300],[minVol,maxVol])

volBar = np.interp(length, [50, 300], [400, 150])

volPer = np.interp(length, [50, 300], [0, 100])

print(int(length),vol)

volume.SetMasterVolumeLevel(vol,None)

if length<50:

cv2.circle(img,(cx,cy),15,(0,255,0),cv2.FILLED)

cv2.rectangle(img,(50,150),(85,400),(0,255,0),3)

cv2.rectangle(img, (50, int(volBar)), (85, 400), (255, 0, 0), cv2.FILLED)

cv2.putText(img, f'FPS: {int(volPer)}%', (40, 450), cv2.FONT\_HERSHEY\_COMPLEX, 1, (255, 0, 0), 3)

cTime=time.time()

fps=1/(cTime-pTime)

pTime=cTime

cv2.putText(img,f'FPS: {int(fps)}',(40,50),cv2.FONT\_HERSHEY\_COMPLEX,1,(255,0,0),3)

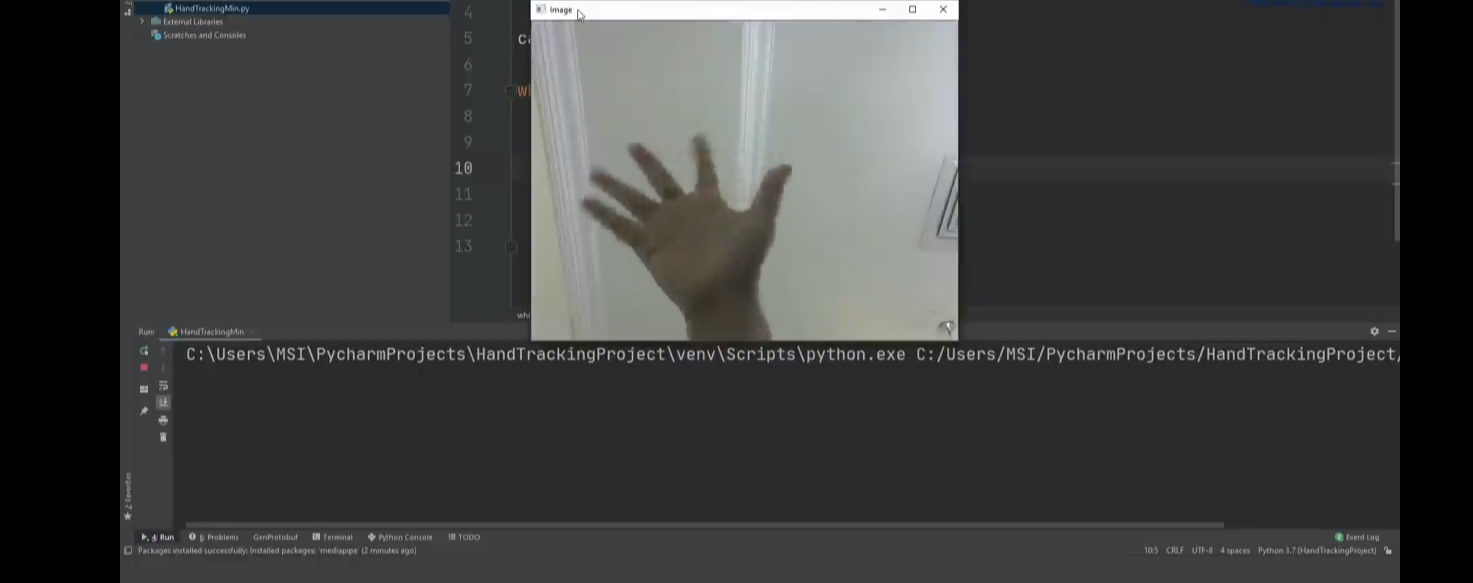
cv2.imshow("img",img)

cv2.waitKey(1)

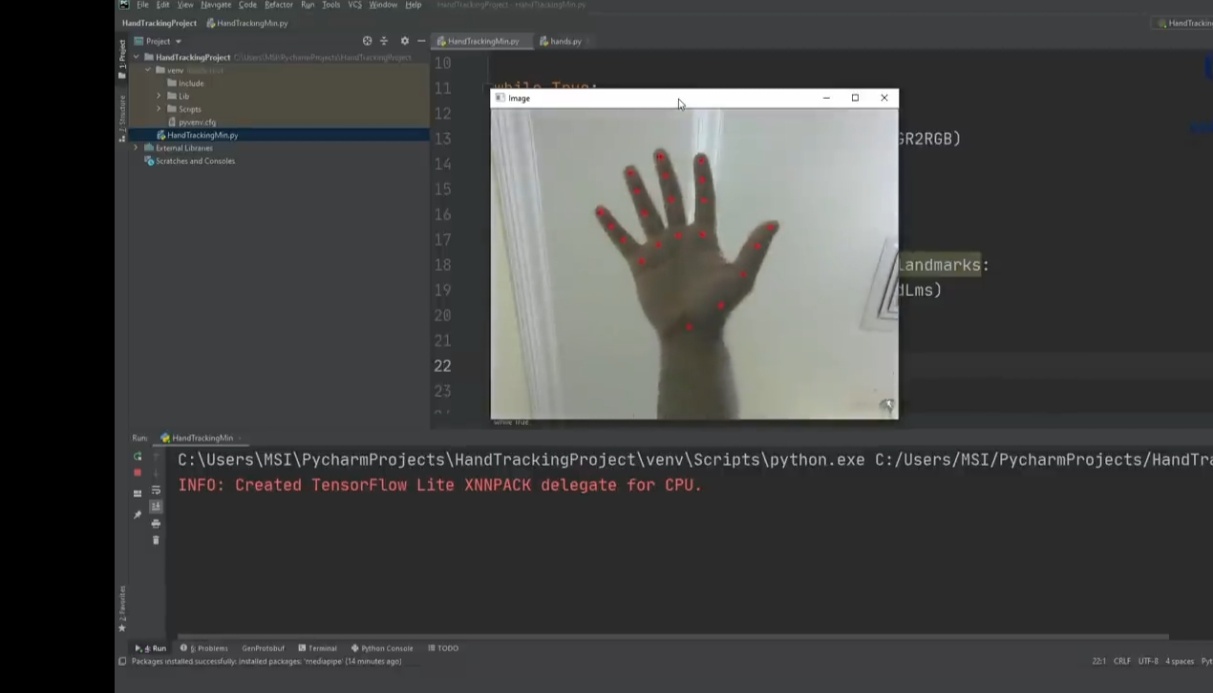
## 7. RESULTS AND DISCUSSION:

**7.1 Implementation**

7.1.1 Created an open frame in PyCharm IDE that could be used for hand detection ,which plays an essential role in our project. The frame size can varied on the preference of programmer.

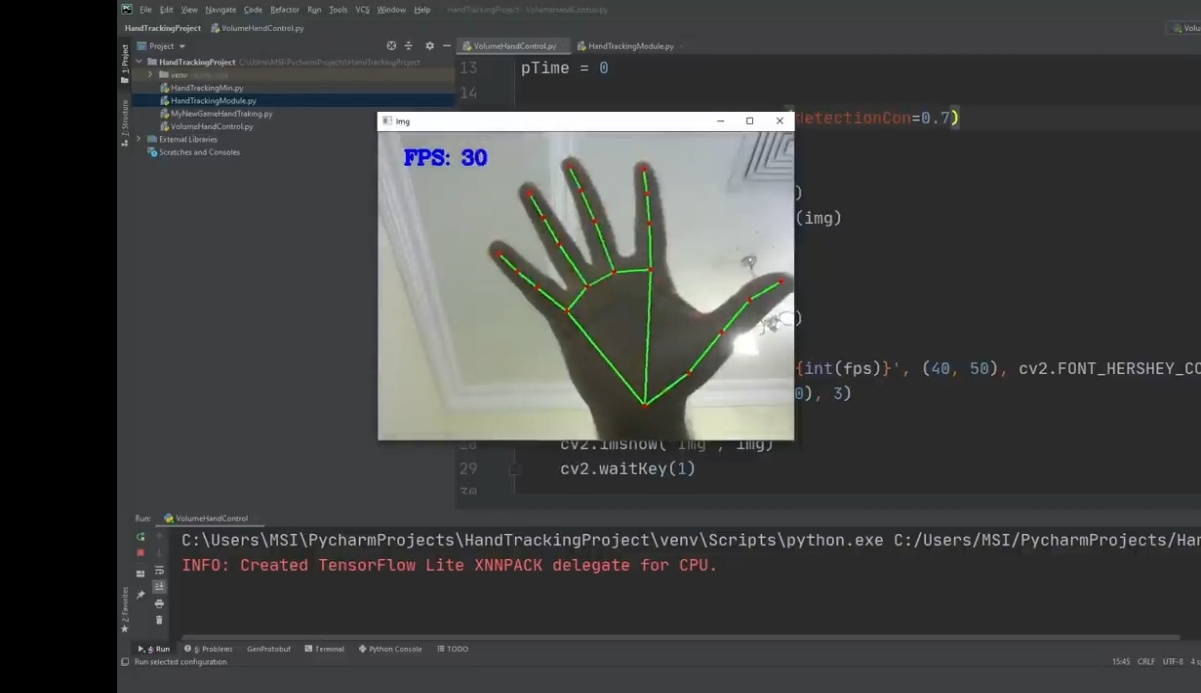


7.1.2 Here palm detection is done by using MediaPipe using MediaPipe the palm is detected with its Landmarks.

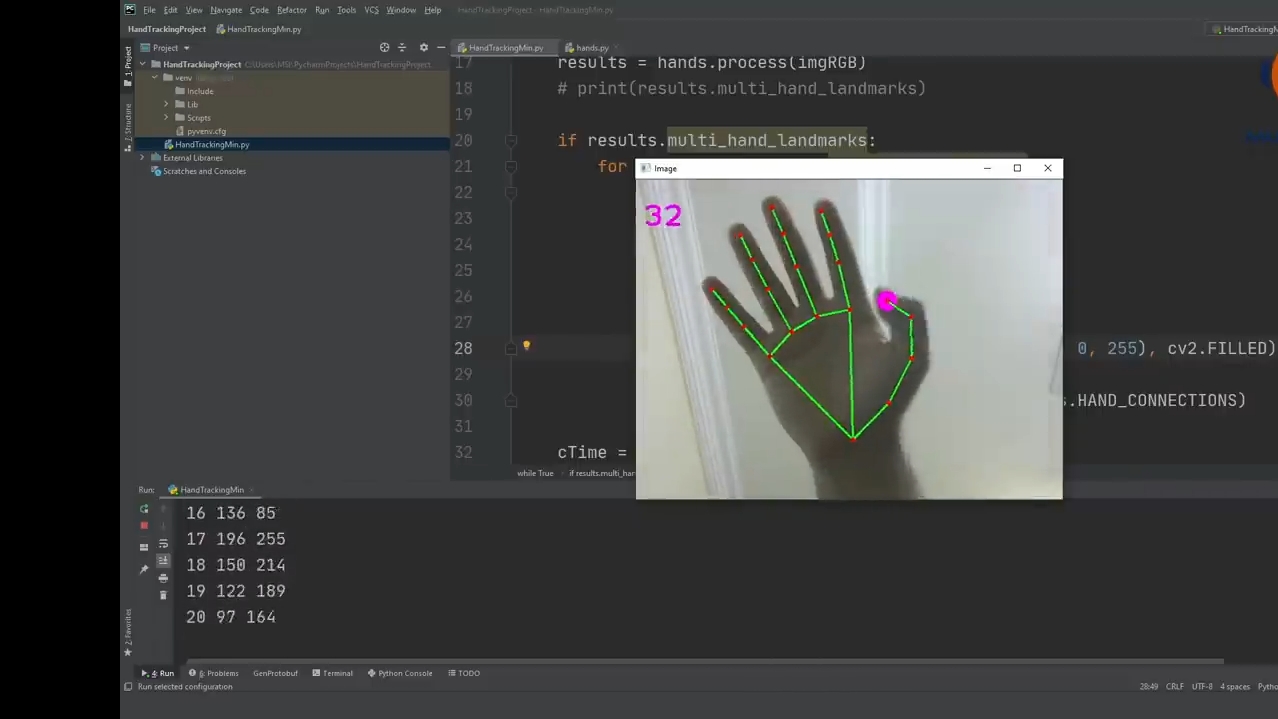


7.1.3.By using Mediapipe library , hand landmarks are detected and connected for

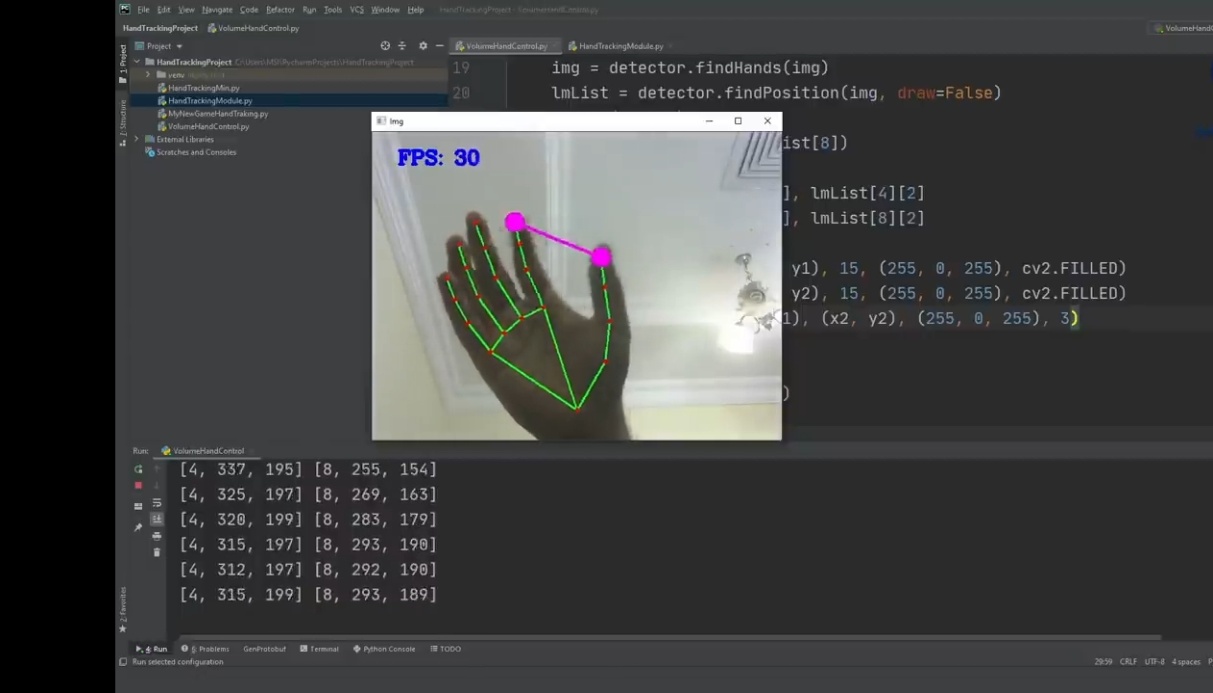
skeleton of a hand.There are 21 hand-knuckle coordinates.



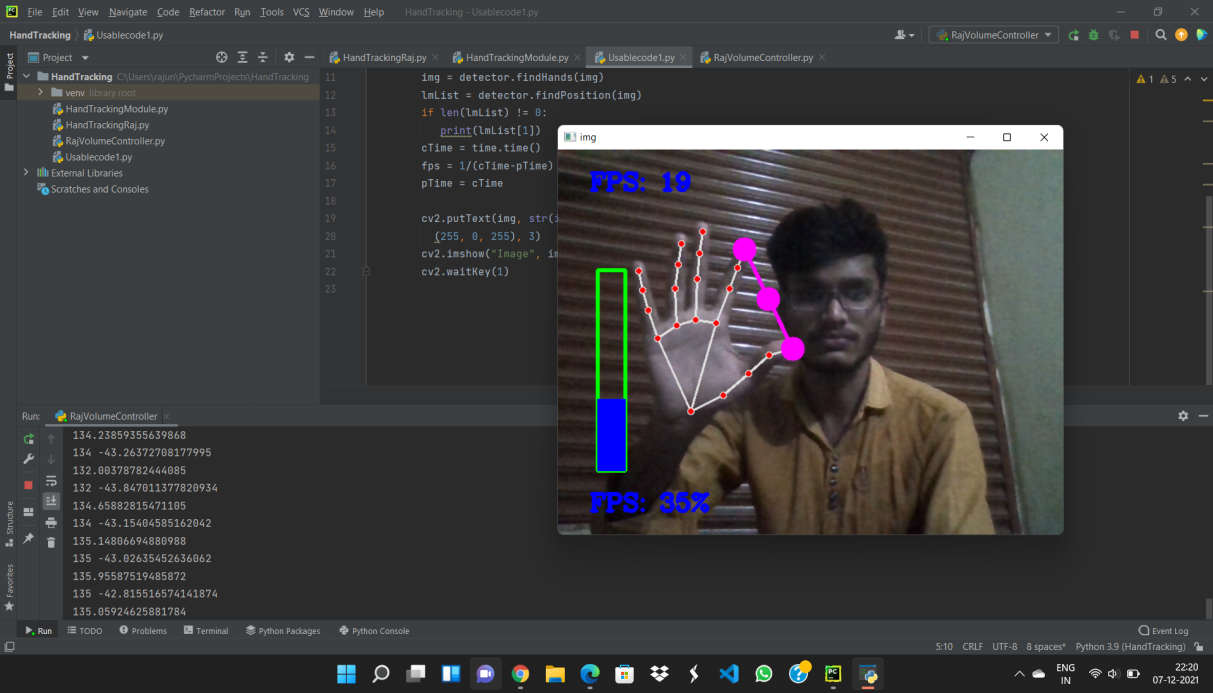
7.1.4.Initialized and detected a hand-knuckle with FPS rate and also detected coordinate individually.



7.1.5.Used 4,8 hand-knuckles for varying the volume and with the help of pycaw volume controlling is done.



7.1.6.Finally the two 4,8 hand-knuckles are detected and pycaw library is implemented for volume controlling. As we vary the distance between the knuckles internal volume of the system get adjusted.



**8. CONCLUSION AND FUTURE WORK**

We developed a vision-based hand Gesture system that does not require any special markers/gloves as input devices and can operate in real-time using cameras. Here , the system can track the positions of the hand and index finger for each hand. The motivation for this hand Gesture was a desktop-based volume control system in which a user can control volume and cursor navigation in real time using natural hand motions. Besides, we propose to employ the motion of the mouse cursor controlled by the hand, and give a suggestion about how to, on the bare hand, position a point through which to control the movement of the mouse cursor.

**9. REFERENCES**

1. G. R. S. Murthy, R. S. Jadon. (2009). “A Review of Vision Based Hand Gestures Recognition,” International Journal of Information Technology and Knowledge Management, vol. 2(2), pp. 405410.
2. Z. Zhang, Y. Wu, Y. Shan, S. Shafer. Visual panel: Virtual mouse keyboard and 3d controller with an ordinary piece of paper. In Proceedings of Perceptual User Interfaces, 2001
3. W.. T. Freeman and M. Roth, Orientation histograms for hand gesture recognition. International workshop on automatic face and gesture recognition. 1995, 12: 296- 301
4. Mokhtar M. Hasan, Pramoud K. Misra, (2011). “Brightness Factor Matching For Gesture Recognition System Using Scaled Normalization”, International Journal of Computer Science & Information Technology (IJCSIT), Vol. 3(2).
5. Akira Utsumi, TsutoniuMiyasato, Fumio KishinoandRyoheiNakatsu, “Real-time Hand Gesture Recognition System,” Proc. of ACCV ’95, vol. 11, pp. 249-253, Singapore,1995