Mini Project Work-Phase 1 Review/22IDT28

Project Review Presentation (25 Mks) – CIE

28 July 2023, First Year (2nd Semester)

Dayananda Sagar College of Engineering, Bangalore, Karnataka

Department of Electronics & Communication Engineering

HAND GESTURE BASED VOLUME CONTROL USING AIML

SECTION : N

Section Co-ordinator
Dr. PAVITHRA G.

Mini Project Guide
Prof. VIBHA T.G.

MINI PROJECT
BATCH NO : 18

USN : 1DS22EC061 (SEC-N) NAME : DHEERAJ R NAIK

USN : 1DS22EC163 (SEC-N) NAME :PRITISH PRIYADARSHI PATRA

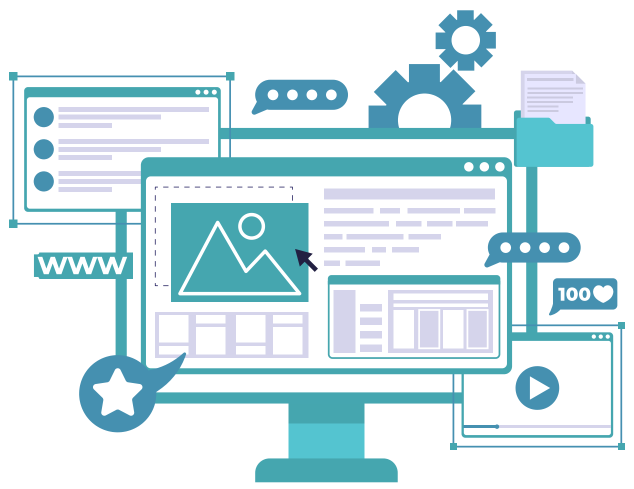
USN : 1DS22EC177 (SEC-N) NAME :ROHAN V NAIK

USN : 1DS22EC239 (SEC-N) NAME :UJJWAL KUMAR SINGH

CONTENTS

INTRODUCTION/ABSTRACT

- This project presents a system for controlling audio volume through hand gestures using the Python programming language.
- The primary objective is to develop an intuitive and hands-free approach to adjust the system's audio output without the need for physical input devices.
- The implementation involves capturing real-time video feed from the device's camera, and then processing each frame to identify and track the hand region.
- The gesture analysis enables the determination of the desired volume adjustment, whether to increase or decrease the audio output.
- To control the volume, the system calls the library, which emulates keypress events to manipulate the system's volume controls.



LITERATURE SURVEY

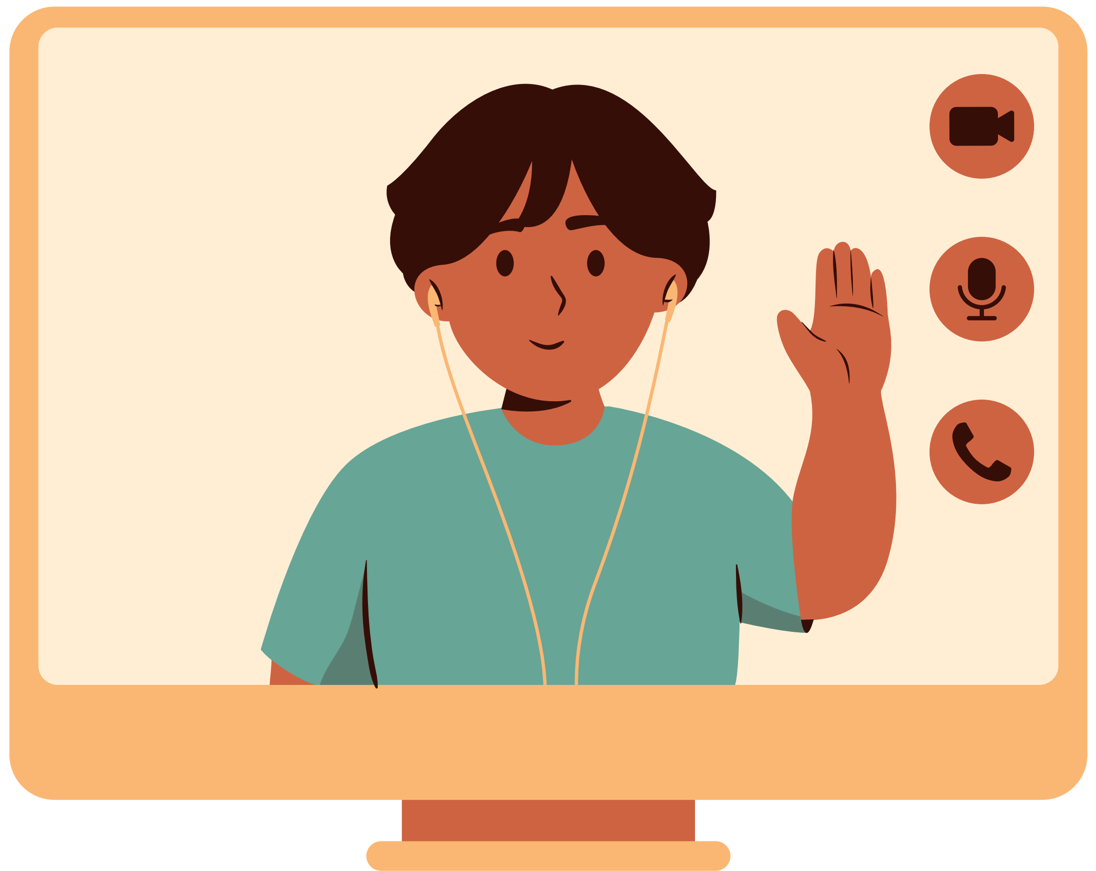
- A non-local algorithm for hand gestures was proposed by A. Buades, B. Coll, and J. Morel. At the moment, finding finger movement algorithms remains a valid task. Functional analysis and statistics collide.

- For the no required elements in the video frame, Golam Moktader Daiyan et al. (2012) suggested a high performance decision based median filter. This technique detects noise pixels iteratively over numerous phases before replacing them with the median value.

- Rajeshwari Kumar Dewangan et.al proposed accurate object information and obtain a location using a deep learning object recognition technique. Object recognition algorithms are designed based on the Single Shot MultiBox Detector (SSD) structure, an object recognition deep learning model, to detect objects using a camera.

SCOPE AND OBJECTIVES

The scope of the project is to construct a synchronous gesture classifying system that can recognize gestures in lighting circumstances spontaneously. To achieve this goal, a synchronous gesture which based on real time is generated to recognize gestures. An intention of this project is to generate a complete system which can identify, spot and explain the hand motioning through computer sight. This structure will work as one of the envisioning of computer sight and AI with user interaction. It create function to identify hand motion based on various arguments. The topmost preference of the structure is to make it easy to use and simple to handle.



METHODOLOGY

Image Capturing

In this initial phase we used a webcam to get the RGB image (frame by frame) using bare hand gestures only.

Pre-Processing

Image processing works to manipulate over the color images into a grayscale image to progress the processing and after completing the processing it restores the images to its initial color space, in this way accordingly, we convert region of interest into a grayscale image.

Hand region Segmentation

This phase is important in any process to hand gestures recognition and facilitate in developing the working of the system by eliminating the unnecessary data within the video stream.

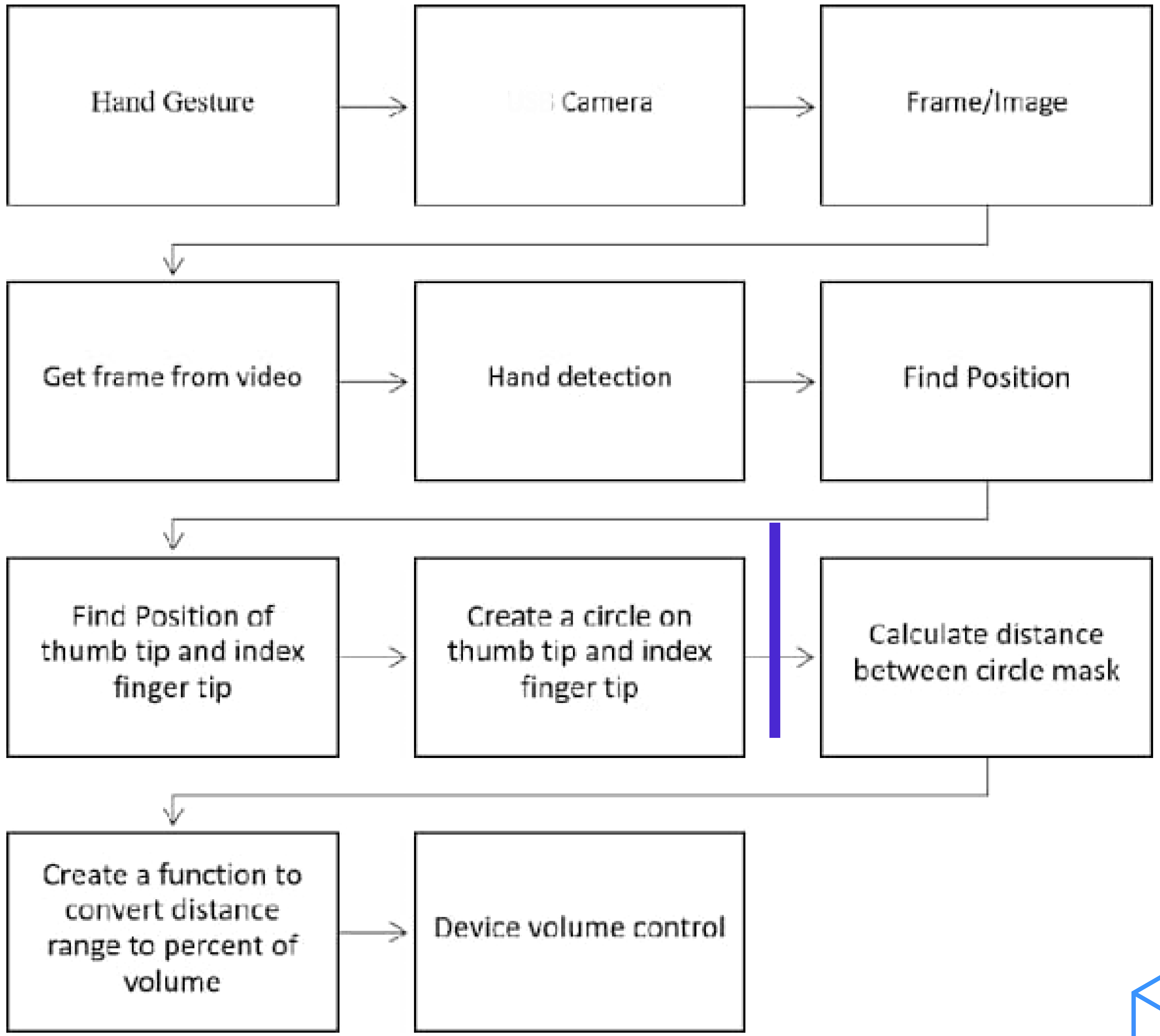
Contour Extraction

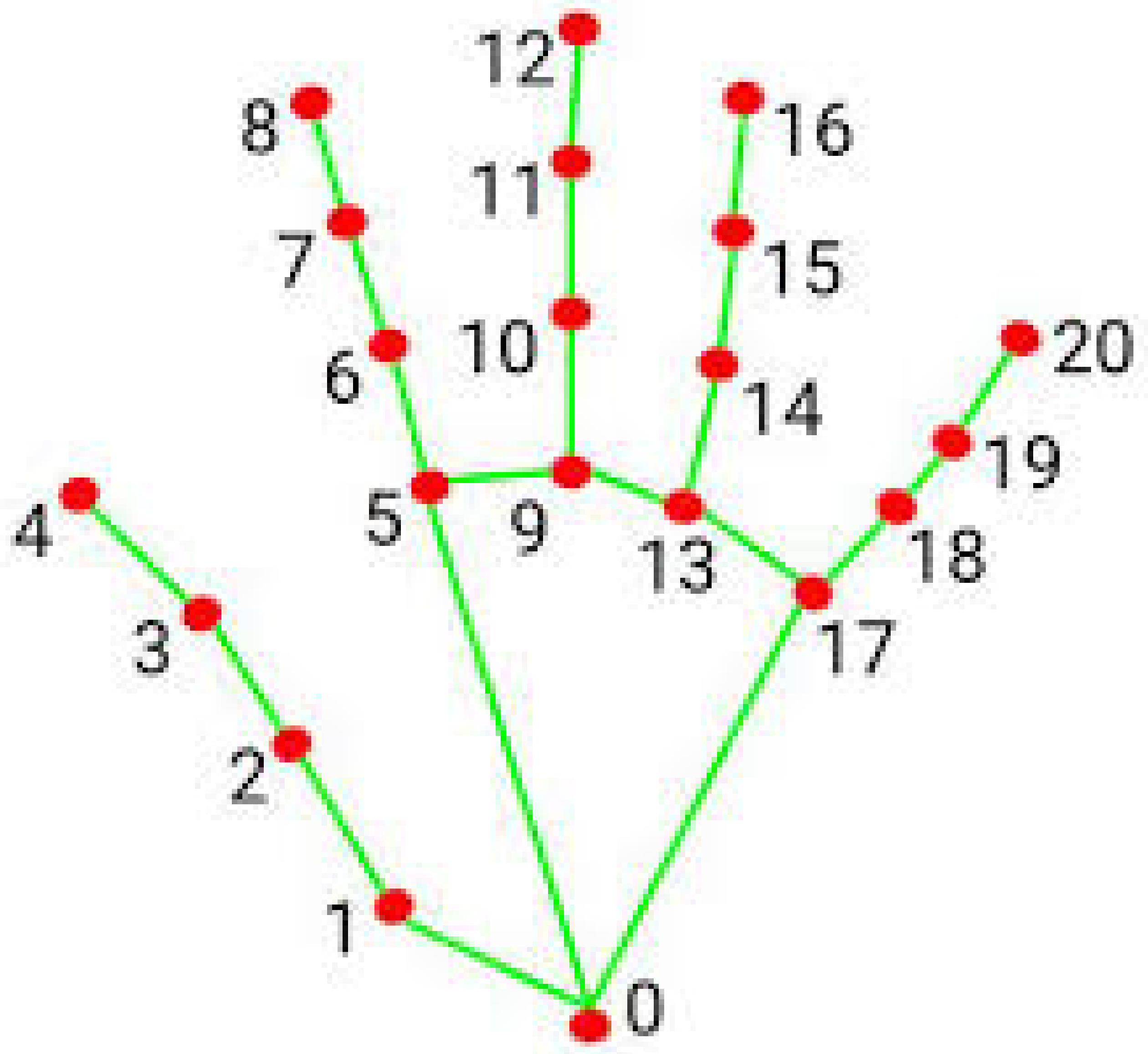
The contour is outlined as object's (hand) boundary that can be seen in the image. The contour can also be a wave connecting points that has the similar color value and is important in shape analyzing, objects identification method.

Feature Extraction and recognition

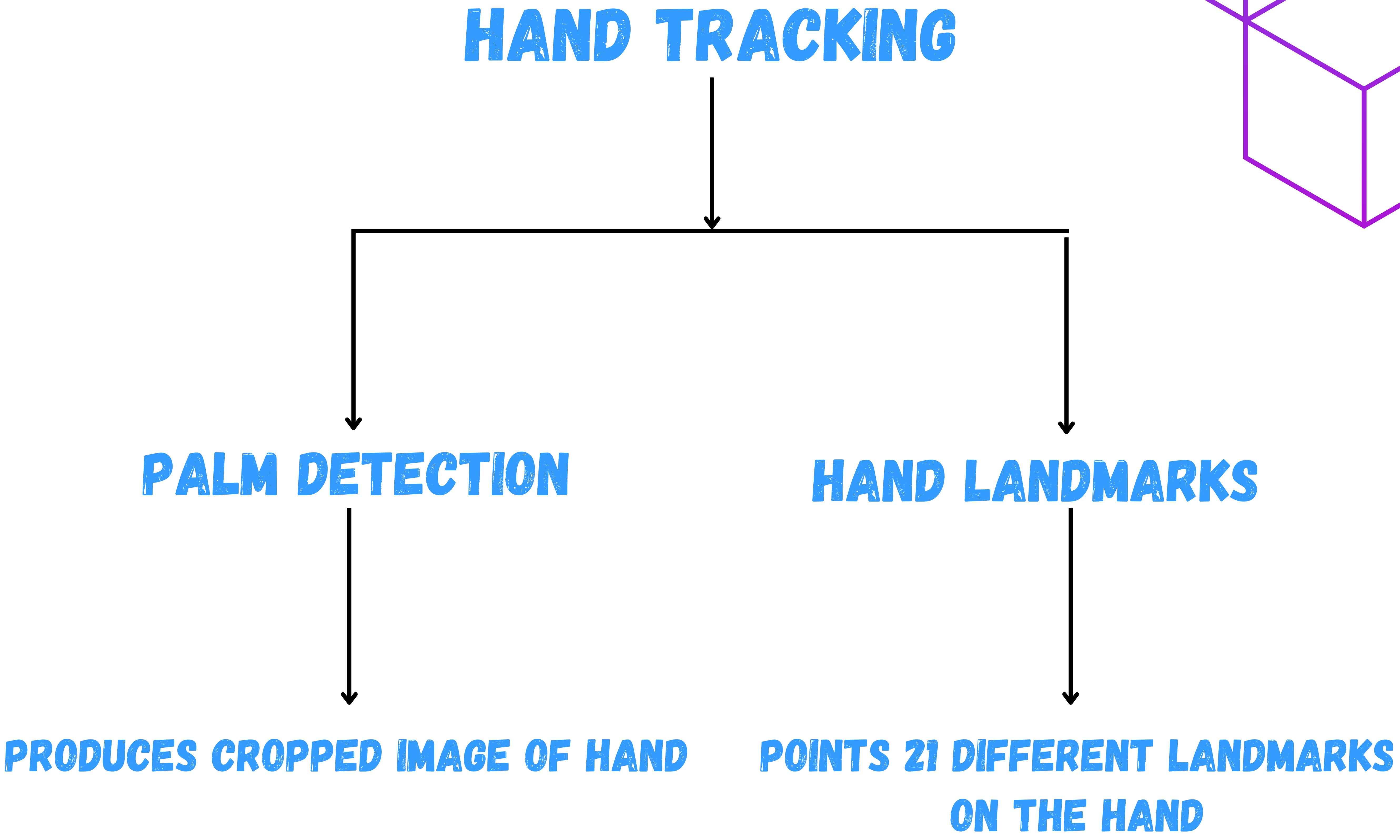
After determining the gesture, the specific functioning is performed. The method of recognizing the movement is a dynamic process.

BLOCK DIAGRAM AND FLOW CHARTS


Hand Gesture
Camera
Frame/Image
Get frame from video
Hand detection
Find Position
Find Position of thumb tip and index finger tip
Create a circle on thumb tip and index finger tip
Calculate distance between circle mask
Create a function to convert distance range to percent of volume
Device volume control



ALGORITHMS


HAND TRACKING
PALM DETECTION
HAND LANDMARKS
PRODUCES CROPPED IMAGE OF HAND
POINTS 21 DIFFERENT LANDMARKS ON THE HAND

SOFTWARE TOOLS USED

1. OpenCV: OpenCV is a powerful computer vision library that allows you to capture and process video streams from a webcam or any other video source. You can use it to detect and track hand gestures in real-time.

2. MediaPipe: MediaPipe is a library developed by Google that provides pre-trained models for hand and pose detection. It can be used to detect and track hand landmarks, which is essential for recognizing hand gestures.

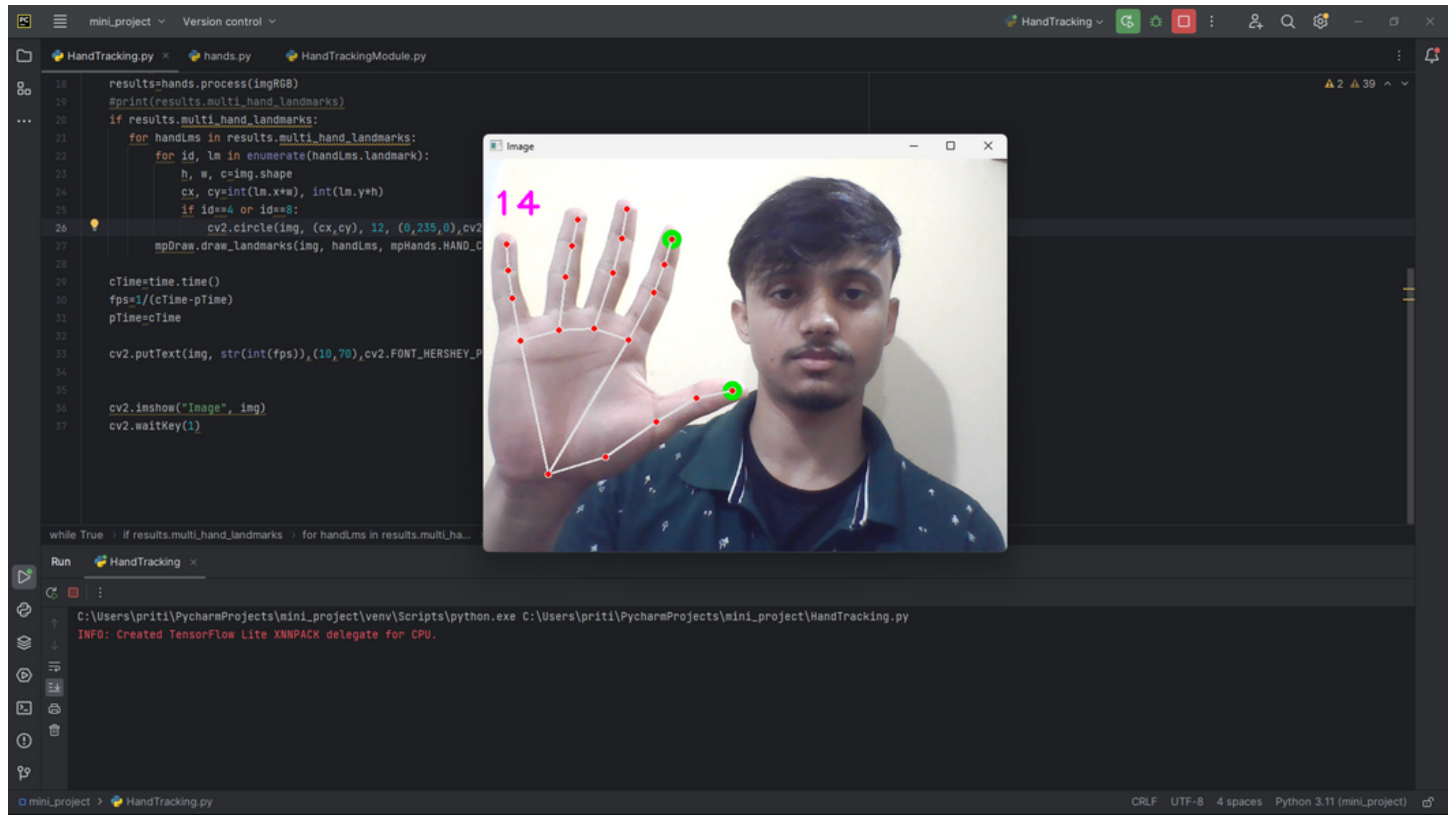
3. NumPy: NumPy is a fundamental library for scientific computing in Python. It is useful for performing numerical operations and data manipulation, which will be helpful in processing the hand gesture data.

4. PyAudio: PyAudio is a Python library for working with audio streams. It allows you to capture and play audio from a microphone or other audio devices

5. Hand Gesture Dataset: You may need a labeled dataset of hand gestures to train your model. You can create your own dataset or use publicly available datasets for this purpose.

6. Machine Learning Libraries: You might use machine learning algorithms to recognize specific hand gestures. Libraries like scikit-learn or TensorFlow can help you build and train a gesture recognition model.

RESULTS(SIMULATION)

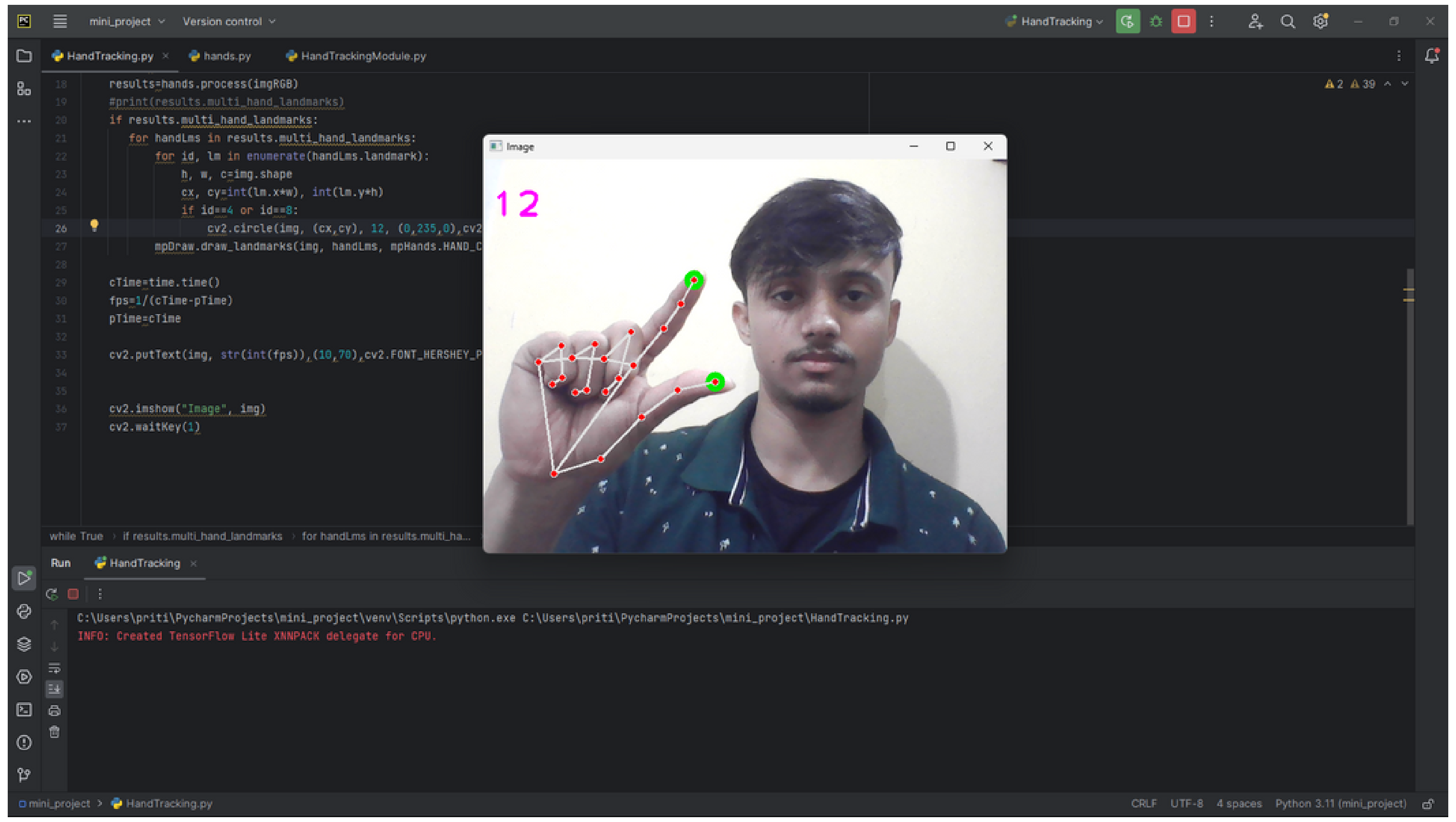


```
results=hands.process(imgRGB)
#print(results.multi_hand_landmarks)
if results.multi_hand_landmarks:
    for handLms in results.multi_hand_landmarks:
        for id, lm in enumerate(handLms.landmark):
            h, w, c=img.shape
            cx, cy=int(lm.x*w), int(lm.y*h)
            if id==4 or id==8:
                cv2.circle(img, (cx,cy), 12, (0,235,0),cv2
        mpDraw.draw_landmarks(img, handLms, mpHands.HAND_C

cTime=time.time()
fps=1/(cTime-pTime)
pTime=cTime

cv2.putText(img, str(int(fps)),(10,70),cv2.FONT_HERSHEY_P


cv2.imshow("Image", img)
cv2.waitKey(1)
```

while True › if results.multi_hand_landmarks › for handLms in results.multi_ha...
Run
HandTracking

```
C:\Users\priti\PycharmProjects\mini_project\venv\Scripts\python.exe C:\Users\priti\PycharmProjects\mini_project\HandTracking.py
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
```

mini_project › HandTracking.py
CRLF UTF-8 4 spaces Python 3.11 (mini_project)



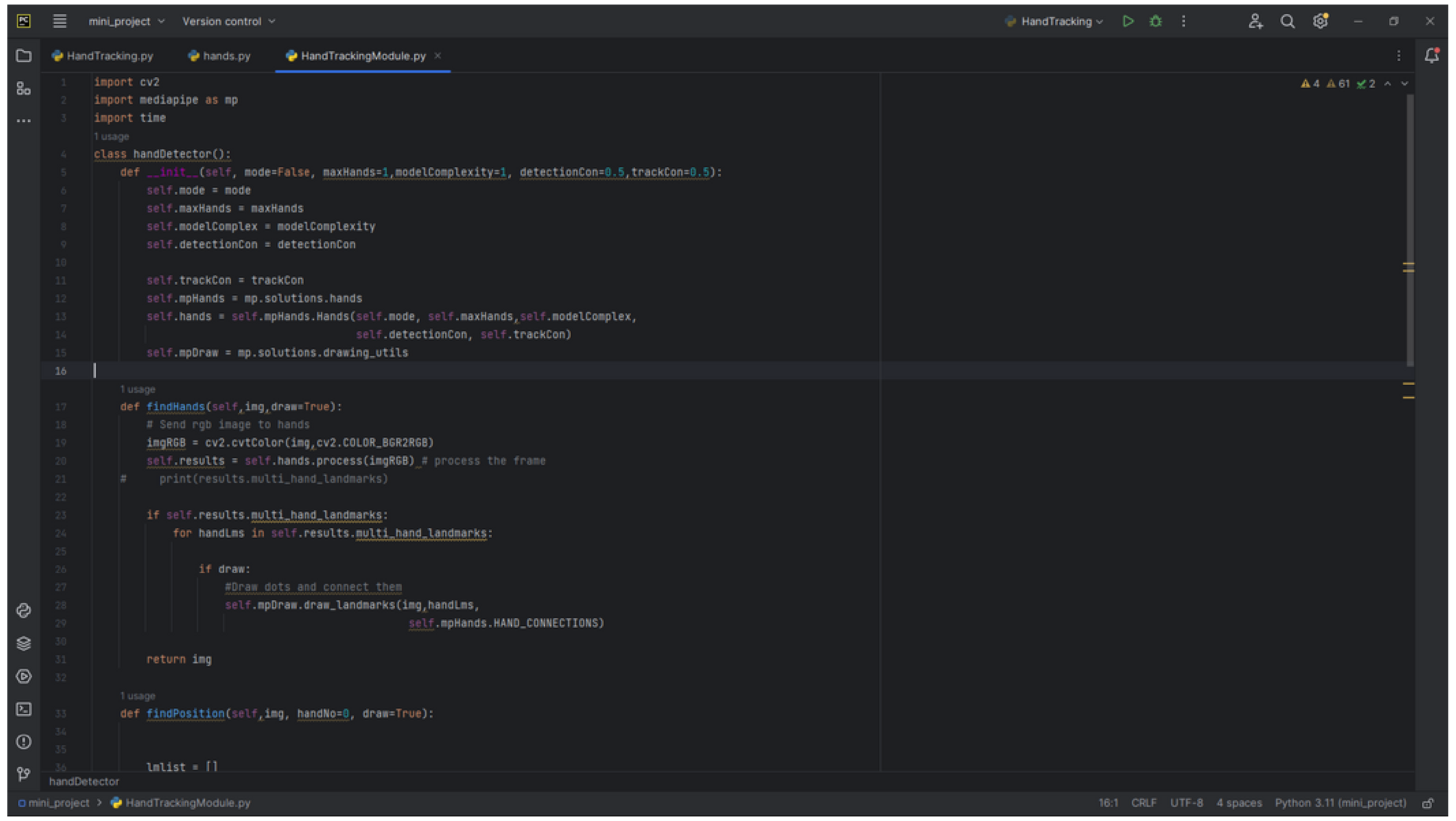
```
results=hands.process(imgRGB)
#print(results.multi_hand_landmarks)
if results.multi_hand_landmarks:
    for handLms in results.multi_hand_landmarks:
        for id, lm in enumerate(handLms.landmark):
            h, w, c=img.shape
            cx, cy=int(lm.x*w), int(lm.y*h)
            if id==4 or id==8:
                cv2.circle(img, (cx,cy), 12, (0,235,0),cv2
        mpDraw.draw_landmarks(img, handLms, mpHands.HAND_C

cTime=time.time()
fps=1/(cTime-pTime)
pTime=cTime

cv2.putText(img, str(int(fps)),(10,70),cv2.FONT_HERSHEY_P


cv2.imshow("Image", img)
cv2.waitKey(1)
```

while True › if results.multi_hand_landmarks › for handLms in results.multi_ha...
Run
HandTracking

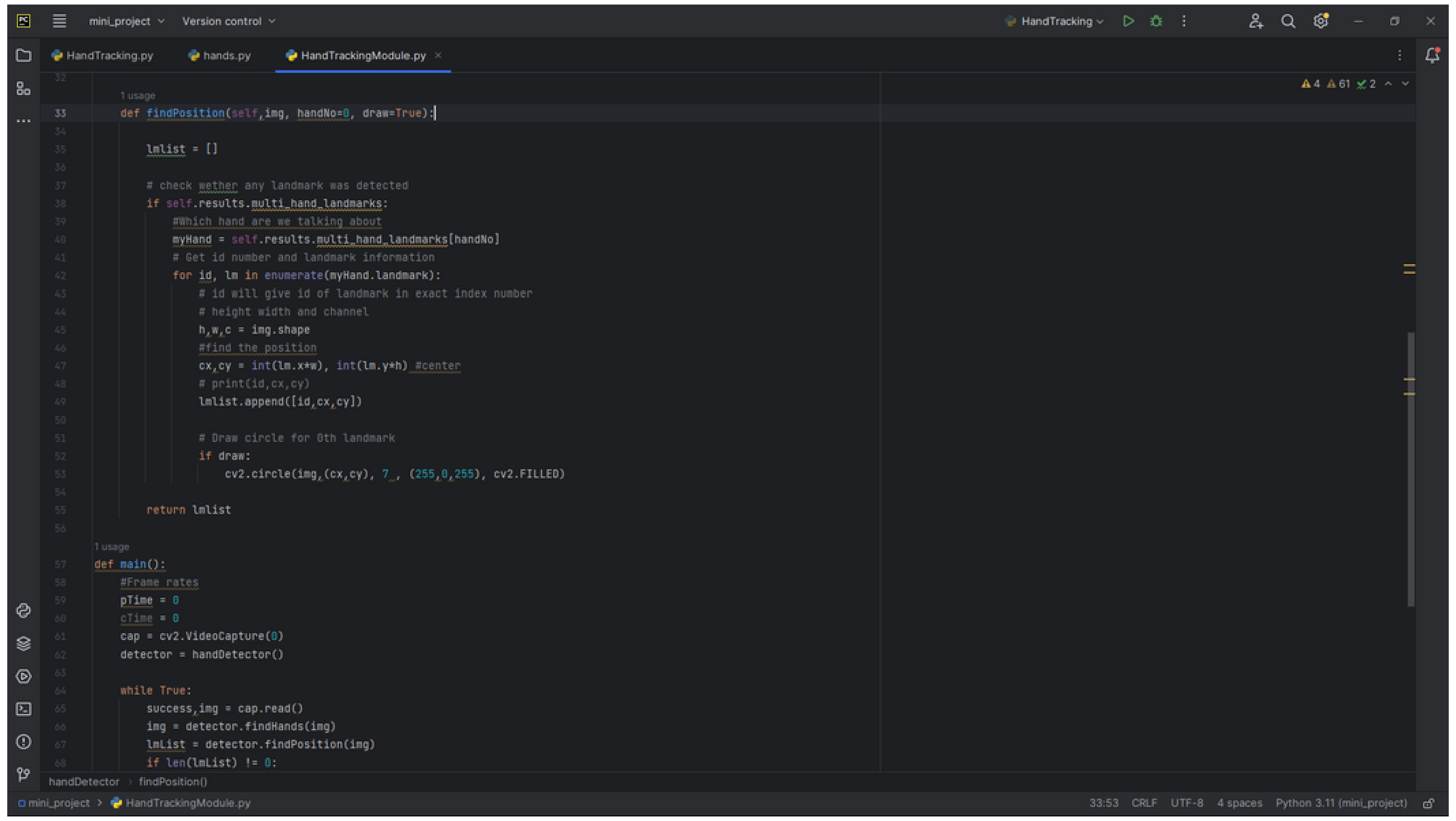
```
C:\Users\priti\PycharmProjects\mini_project\venv\Scripts\python.exe C:\Users\priti\PycharmProjects\mini_project\HandTracking.py
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
```

mini_project › HandTracking.py
CRLF UTF-8 4 spaces Python 3.11 (mini_project)

SOURCE CODE



```python
import cv2
import mediapipe as mp
import time

class handDetector():
    def __init__(self, mode=False, maxHands=1,modelComplexity=1, detectionCon=0.5,trackCon=0.5):
        self.mode = mode
        self.maxHands = maxHands
        self.modelComplex = modelComplexity
        self.detectionCon = detectionCon

        self.trackCon = trackCon
        self.mpHands = mp.solutions.hands
        self.hands = self.mpHands.Hands(self.mode, self.maxHands,self.modelComplex,
                                        self.detectionCon, self.trackCon)
        self.mpDraw = mp.solutions.drawing_utils


    def findHands(self,img,draw=True):
        # Send rgb image to hands
        imgRGB = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
        self.results = self.hands.process(imgRGB) # process the frame
    #     print(results.multi_hand_landmarks)

        if self.results.multi_hand_landmarks:
            for handLms in self.results.multi_hand_landmarks:

                if draw:
                    #Draw dots and connect them
                    self.mpDraw.draw_landmarks(img,handLms,
                                               self.mpHands.HAND_CONNECTIONS)

        return img

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


        lmlist = []
```



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

        lmlist = []

        # check wether any landmark was detected
        if self.results.multi_hand_landmarks:
            #Which hand are we talking about
            myHand = self.results.multi_hand_landmarks[handNo]
            # Get id number and landmark information
            for id, lm in enumerate(myHand.landmark):
                # id will give id of landmark in exact index number
                # height width and channel
                h,w,c = img.shape
                #find the position
                cx,cy = int(lm.x*w), int(lm.y*h) #center
                # print(id,cx,cy)
                lmlist.append([id,cx,cy])

                # Draw circle for 0th landmark
                if draw:
                    cv2.circle(img,(cx,cy), 7 , (255,0,255), cv2.FILLED)

        return lmlist

def main():
    #Frame rates
    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:
```

ADVANTAGES

1. Hands-Free Interaction: Gesture-based volume control allows users to adjust the volume without the need for physical buttons or controls.

2. Intuitive and User-Friendly: Using gestures for volume control makes the user interface more intuitive and user-friendly, particularly for individuals who may have difficulty using traditional interfaces.

3. Inclusive Accessibility: Gesture-based control can improve accessibility for individuals with physical disabilities or impairments that may have difficulty using traditional controls.

4. Novelty and Engagement: It can increase engagement with the application or device, making it more enjoyable to use.

5. Situational Usefulness: Gesture control is particularly useful in scenarios where manual dexterity is limited or when hands are messy.

APPLICATIONS

1. Media Players and Video Streaming: Gesture-based volume control can be integrated into media players, allowing users to adjust the volume while watching videos, listening to music, or streaming content.

2. Virtual Reality (VR) and Augmented Reality (AR): In VR and AR applications, users may not have direct access to physical buttons or controls. Gesture-based volume control can be utilized to adjust the sound in these immersive environments.

3. Home Automation Systems: Hand gesture volume control can be incorporated into smart home systems, enabling users to adjust the volume of smart speakers, TVs, and other audio devices with simple gestures.

4. Gaming: In gaming applications, gesture-based volume control can enhance the gaming experience by allowing users to adjust the in-game sound effects and music using hand gestures.

5. Car Infotainment Systems: Gesture control in car infotainment systems can be safer and more convenient for drivers, as it eliminates the need to look for physical volume knobs while driving.

6. Presentation Tools: Gesture-based volume control can be utilized in presentation tools to control the audio volume during lectures or public speaking engagements.

7. Accessibility Solutions: Gesture-based volume control can be a part of assistive technology solutions for individuals with physical disabilities or motor impairments, making it easier for them to interact with various devices.

CONCLUSION

Overall, this project represents a successful implementation of a hand gesture-based volume control system using Python. The technology exhibits promise not only in audio control but also in diverse applications requiring gesture-based interactions. To further enhance the system's robustness and accuracy, future work may focus on refining the gesture recognition algorithm, exploring deep learning approaches, and expanding the range of supported gestures.

We believe that this project serves as a stepping stone towards more sophisticated and inclusive human-computer interactions, contributing to the advancement of gesture-based control systems in the broader realm of technology.

OUTCOMES

- The outcome of the project is a functional hand gesture-based volume control system developed using Python and computer vision techniques.
- It accurately recognizes and tracks various hand gestures, allowing users to control the volume of an audio system in real-time.
- The system's responsiveness and user-friendly interface have received positive feedback from testers. It has the potential for broader applications in human-computer interaction and can serve as a foundation for future gesture-based control systems.
- The project's codebase is well-documented and open-source, promoting collaboration and further development in the field.

WORKFLOW

June 2023		We decided the final topic of the mini project, discussed the same with our guide and submitted the synopsis to the coordinator.
July 2023		We reffered various papers and documentations and got the idea about the softwares we used in the project. We started with the implementation of the project and successfully built the hand detection module.
August 2023		We will work on issues and try to better the performance of our hand detection model and finally work on volume control aspect of our project.
September 2023		We will give the final touch to the project and work on the optimization part . We will prepare for the presenting our mini project in SEE.

RESOURCES USED

https://docs.opencv.org

https://www.tensorflow.org/api_docs/python/

https://pytorch.org/docs/stable/index.html

https://numpy.org/doc/

RESEARCH GATE, GOOGLE .

H.A JALAB "Static hand Gesture recognition for human computer interaction

PERALES"Hand tracking and gesture recognition for human-computer interaction",2005.

W. T. Freeman and M. Roth, Orientation histograms for hand gesture recognition. International workshop on automatic face and gesture recognition

G. R. S. Murthy, R. S. Jadon. (2009). "A Review of Vision Based Hand Gestures Recognition," Internation Journal of Information Technology and Knowledge Management, vol. 2(2).

Mokhtar M. Hasan, Pramoud K. Misra, (2011). Brightness Factor Matching For Gesture Recognition