भारतीय प्रौद्योगिकी संस्थान
काशी हिन्दू विश्वविद्यालय

IIT INDIAN INSTITUTE OF TECHNOLOGY
BANARAS HINDU UNIVERSITY

Exploratory Project Report
SEMESTER -IV 2021-22

VOLUME CONTROL USING HAND GESTURE AND FINGER COUNTER IN HAND

Project By:-

Avinash Singh (20095020)

Korivi Vedarshini (20095055)

Nipun Khanduja (20095073)

Under the guidance

Dr. Satyabrata Jit

Department of Electronics Engineering

IIT BHU (Varanasi)

Department of Electronics Engineering IIT (BHU) Varanasi

CERTIFICATE

This is to certify that this project report "**Volume Control using Hand Gesture and Finger Counter in Hand**" is submitted by ***Avinash Singh, Korivi Vedarshini and Nipun Khanduja*** who carried out the project work under the supervision of **Dr. Satyabrata Jit.**

We approve this project for submission of the Exploratory Project, IIT(BHU) Varanasi.

Signature of Supervisor
Dr.Satyabrata Jit
Department of Electronics Engineering
IIT (BHU) Varanasi

ACKNOWLEDGEMENT

It gives us immense pleasure to express our deepest sense of gratitude and sincere thanks to our highly respected and esteemed guide, **Dr. Satyabrata Jit**, for his valuable encouragement, and help for accomplishing this work.

We would also like to express our sincere thanks to all others who helped us directly or indirectly during this project work.

STUDENTS NAME:

Avinash Singh (20095020)

Korivi Vedarshini (20095055)

Nipun Khanduja (20095073)

Table Of Contents

Introduction

The ability to perceive the shape and motion of hands can be a vital component in improving the user experience across a variety of technological domains and platforms. For example, it can form the basis for sign language understanding and hand gesture control, and can also enable the overlay of digital content and information on top of the physical world in augmented reality.

While coming naturally to people, robust real-time hand perception is a decidedly challenging computer vision task, as hands often occlude themselves or each other (e.g. finger/palm occlusions and hand shakes) and lack high contrast patterns.

As we know, the vision-based technology of hand gesture recognition is an important part of human-computer interaction (HCI). In the last decades, keyboard and mouse play a significant role in human-computer interaction.

However, owing to the rapid development of hardware and software, new types of HCI methods have been required. In the faster growing world of automation technologies such as gesture recognition receive great attention in the field of HCI and it also seeks great demand as well.

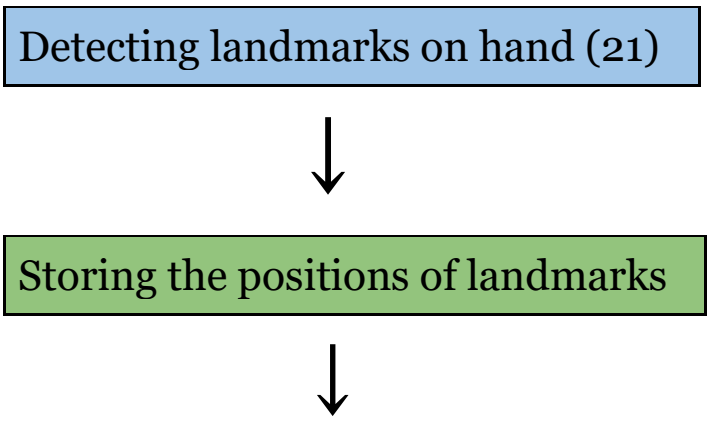
Aim and Objectives

The project aims to create interaction between machine hardware and hand gestures using landmarks on hand to minimise the use of hardware.

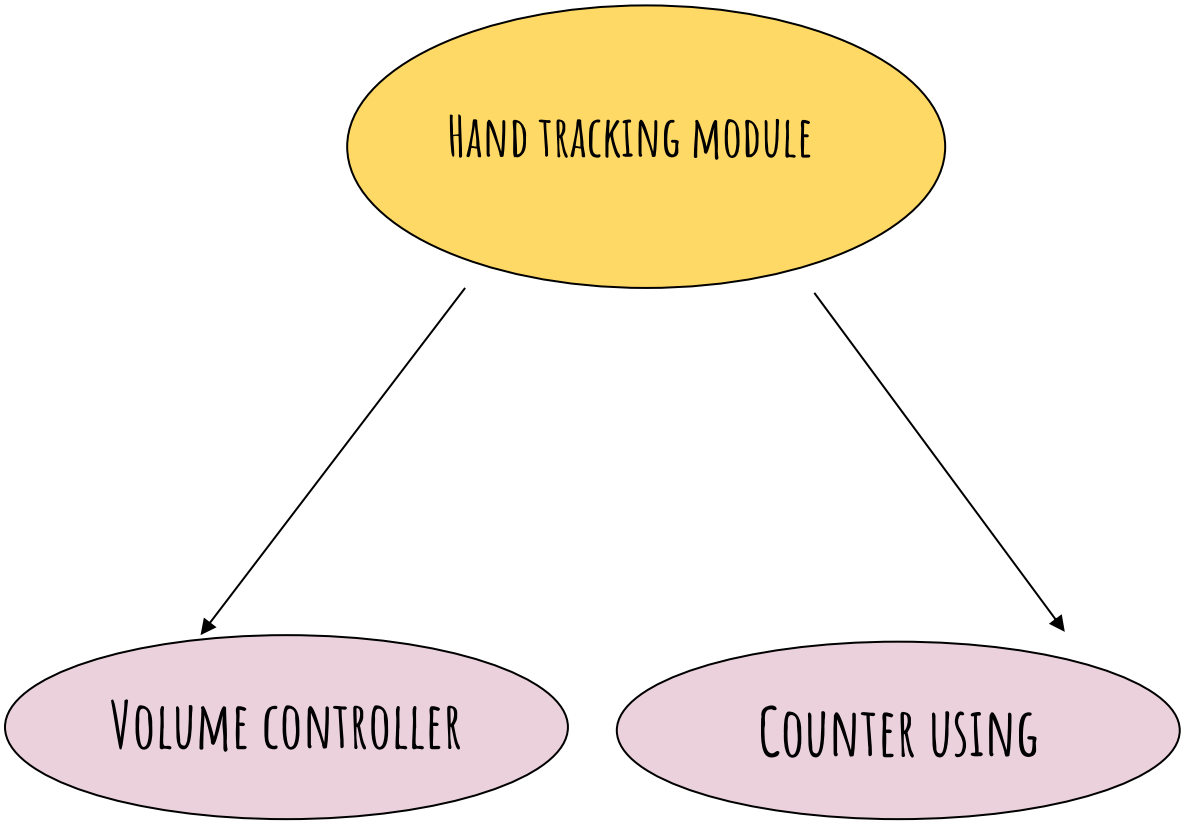
So our objective is to create some applications of this through which we can illustrate the use of our project.

This project includes:-

- Volume controller using hand movement.
- Counter using hand landmarks of fingers.



Creating a hand module which will give positions of landmarks in live camera feed.


Hand tracking module
Volume controller
Counter using

Theory and Working

Hand Module

Libraries Used:

- **OpenCV**

OpenCV is the huge open-source library for computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it is integrated with various libraries, such as Numpuy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

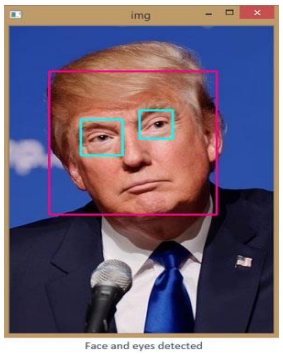


Fig : - OpenCV used to detect eyes and face of a person .

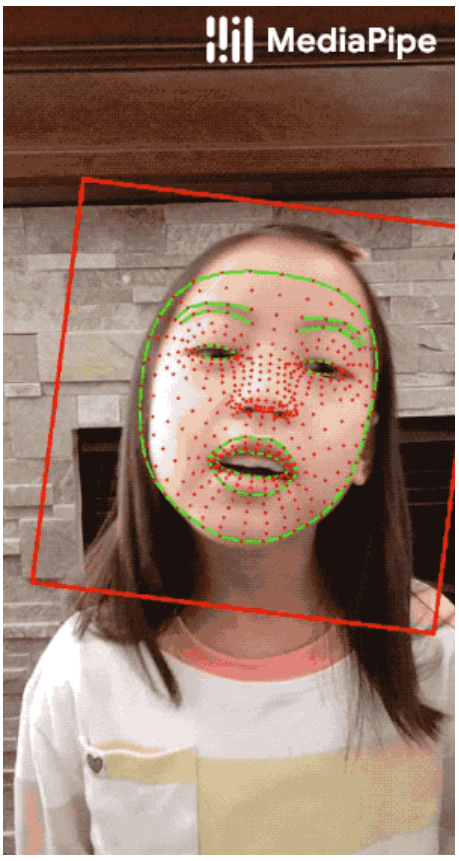
OpenCV Functionality

1. Object/feature detection (objdetect, features2d, nonfree)
2. Image/video I/O, processing, display (core, imgproc, highgui)
3. Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab)
4. Computational photography (photo, video, superres)
5. Machine learning & clustering (ml, flann)
6. CUDA acceleration (gpu)

Application of OpenCV

1. face recognition
2. Automated inspection and surveillance
3. number of people – count (foot traffic in a mall, etc)
4. Vehicle counting on highways along with their speeds
5. Interactive art installations
6. Anomaly (defect) detection in the manufacturing process (the odd defective products)
7. Video/image search and retrieval
8. Robot and driver-less car navigation and control

- **MediaPipe**



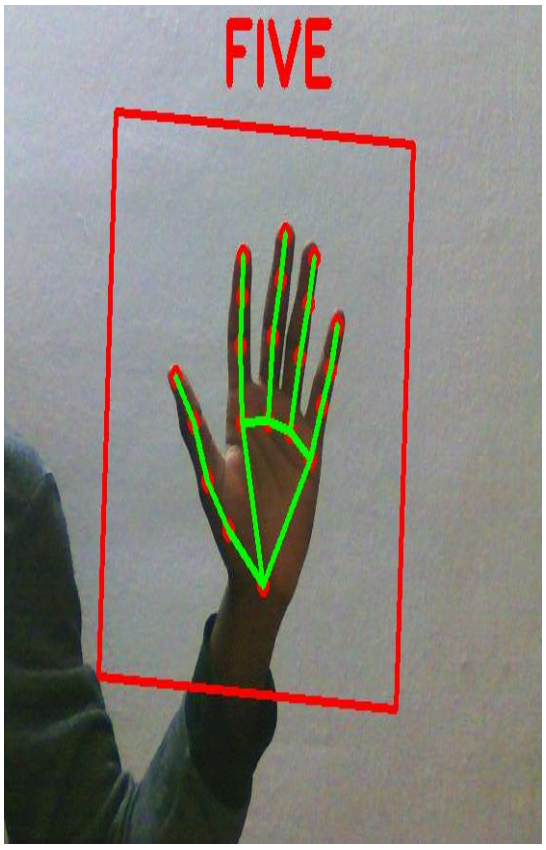


Fig1 - face mesh (468 landmarks) Fig2 - hand tracking (21 landmarks)

MediaPipe Hands is a high-fidelity hand and finger tracking solution. It employs machine learning (ML) to infer 21 3D landmarks of a hand from just a single frame. Whereas current state-of-the-art approaches rely primarily on powerful desktop environments for inference, our

method achieves real-time performance on a mobile phone, and even scales to multiple hands. We hope that providing this hand perception functionality to the wider research and development community will result in an emergence of creative use cases, stimulating new applications and new research avenues.

Application of MediaPipe

1. Human Pose Detection And Tracking
2. Hair Segmentation
3. Hand Tracking
4. Object Detection and Tracking
5. Face Detection
6. Face Mesh

Hand Landmarks:

- **Palm Detection Model**

To detect initial hand locations, we designed a single-shot detector model optimized for mobile real-time uses in a manner similar to the face detection model in MediaPipe Face Mesh. Detecting hands is a decidedly complex task: our model has to work across a variety of hand sizes with a large scale span (~20x) relative to the image frame and be able to detect occluded and self-occluded hands. Whereas faces have high contrast patterns, e.g., in the eye and mouth region, the lack of such features in hands makes it comparatively difficult to detect them reliably from their visual features alone. Instead, providing additional context, like arm, body, or person features, aids accurate hand localization.
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.

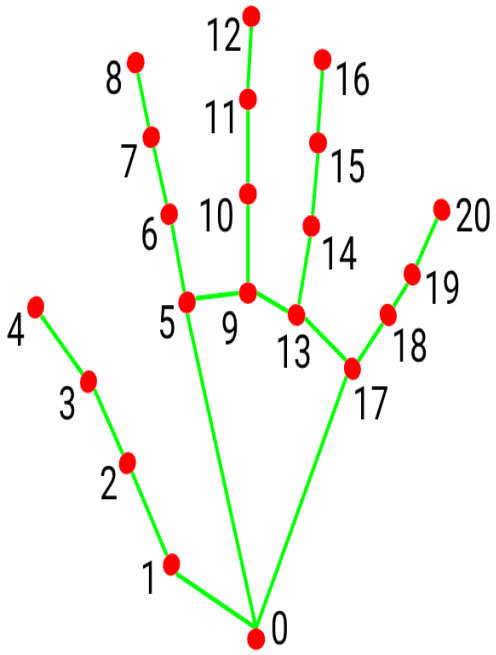
We used function **handsFinder** that is shown below which detects palm from the frame of the live video.

```
def handsFinder(self,image,draw=True):
    imageRGB = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
    self.results = self.hands.process(imageRGB)

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

            if draw:
                self.mpDraw.draw_landmarks(image, handLms, self.mpHands.HAND_CONNECTIONS)
    return image
```

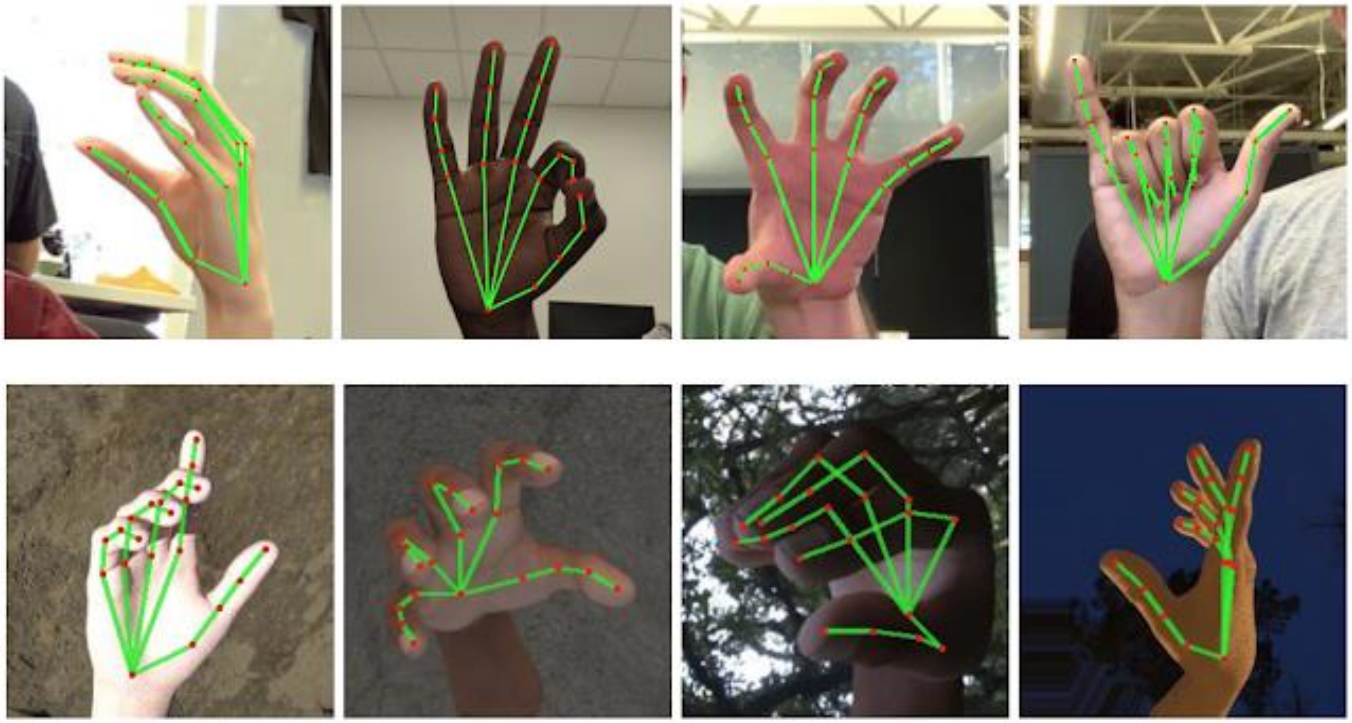
• Hand Landmark Model

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.



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



We used the function **positionFinder** that detect 21 landmarks from the hand and store the position of the landmark with corresponding ids(as in above image) of landmarks.

```
def positionFinder(self, image, handNo=0, draw=True):
    lmlist = []
    if self.results.multi_hand_landmarks:
        Hand = self.results.multi_hand_landmarks[handNo]
        for id, lm in enumerate(Hand.landmark):
            h, w, c = image.shape
            cx, cy = int(lm.x * w), int(lm.y * h)
            lmlist.append([id, cx, cy])
        if draw:
            cv2.circle(image, (cx, cy), 15, (255, 0, 255), cv2.FILLED)

    return lmlist
```

Volume Control using Hand Movement

We imported the hand module which we created above. Then using that module we created an instance of the module.

This line of code creates the instance of module “handDetector” having parameter “detectionCon” which signifies the detection confidence of hand in the frame of live video.

```
detector = htm.handDetector(detectionCon=0.7)
```

Now for using the speaker of the machine(laptop here) we used an open-source library.

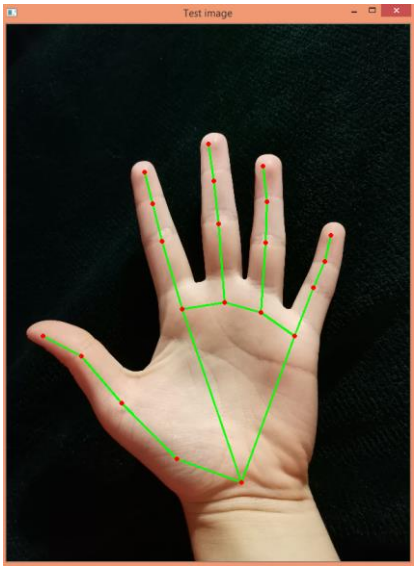
```
from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume
```

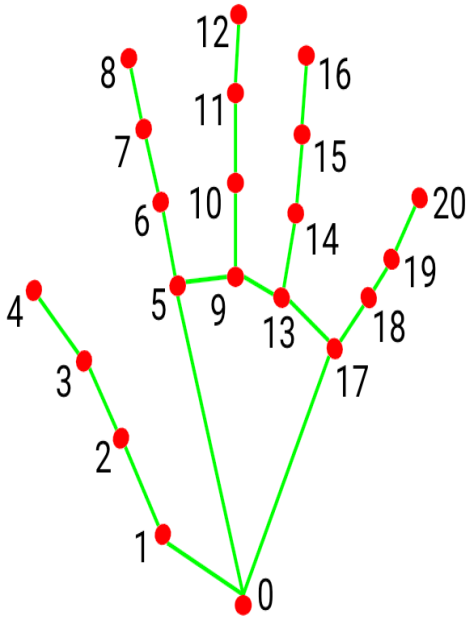
Using functions in this library we linked our speaker with our code.

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

This function finds 21 positions on palm and stores the coordinate in LmList.

```
img = detector.handsFinder(img)
lmList = detector.positionFinder(img, draw=False)
```





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

From the above landmarks to points chart we can see that the index finger has index 8 and thumb have index 4. So our idea was

to see the distance between thumb and index finger and adjust volume according to it.

From the LmList we calculated the distance between landmarks with index “4” and “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
```

Now below code synchronise the distance between index finger and thumb with the volume of the device.

```
vol = np.interp(length, [20,275], [minVol, maxVol])
volBar = np.interp(length, [20, 275], [400, 150])
volPer = np.interp(length, [20,275], [0, 100])
#print(int(length), vol)
volume.SetMasterVolumeLevel(vol, None)
```

This code takes live feed from the front camera of the laptop and “img” have each frame of feed.

```
cap = cv2.VideoCapture(0)
success, img = cap.read()
```

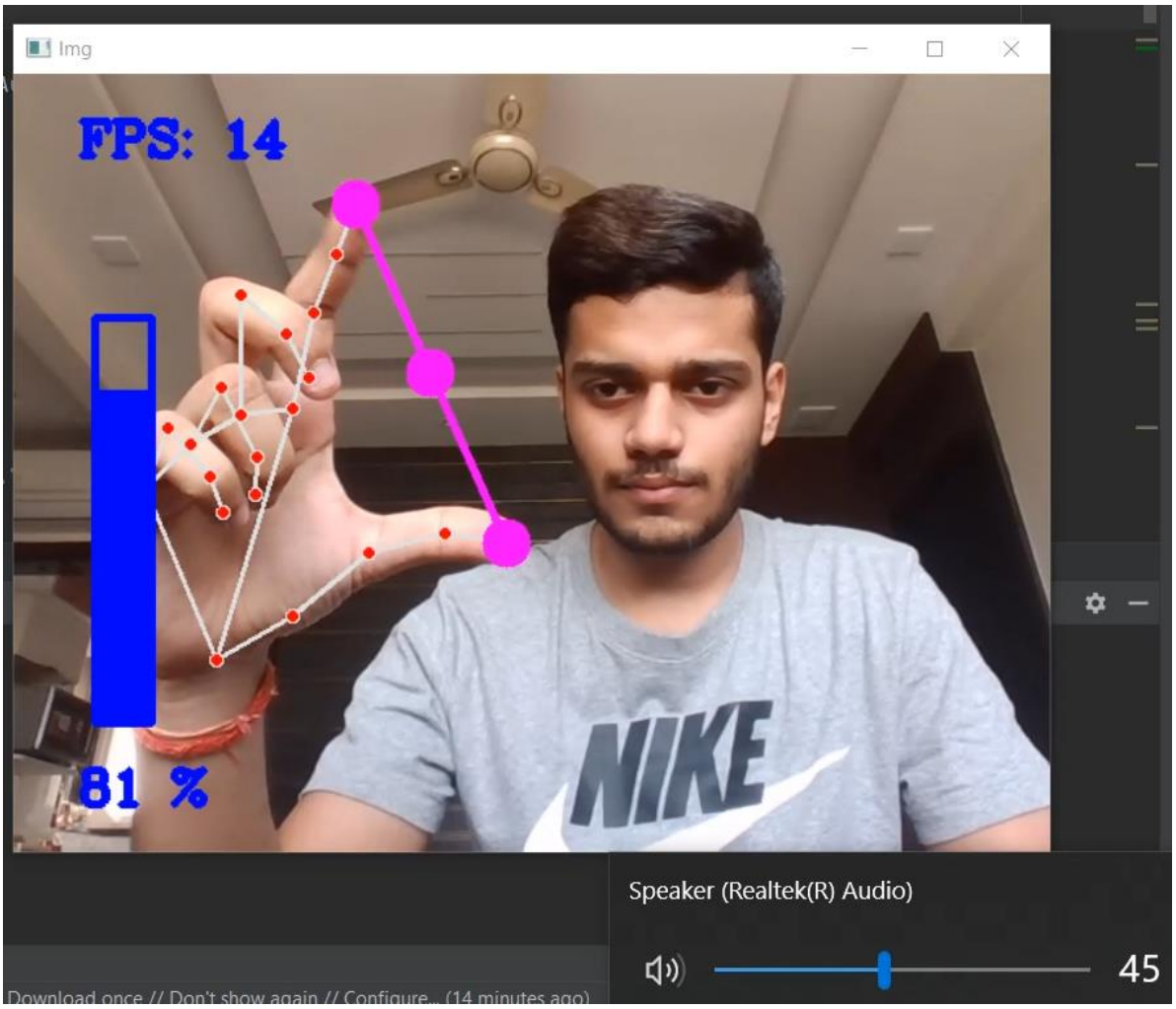
RESULT:-

Now when we run the code we get to see our live feed.

Taking a hand in front of camera shows us 21 landmarks of the palm which we are shown as red dots which are connected by green lines.

The pink line between thumb and index finger shows the distance between those two.

As we can see from the below gif moving finger up and down(i.e. Changing distance) help us to adjust the volume of our laptop.



This thing will help us to reduce dependency on hardwares to do our daily task and make life more simpler.

Counter Using Hand Landmarks of Finger

We imported Hand tracking module which we already made above. Then we created an instance with a confidence ratio 0.75 .

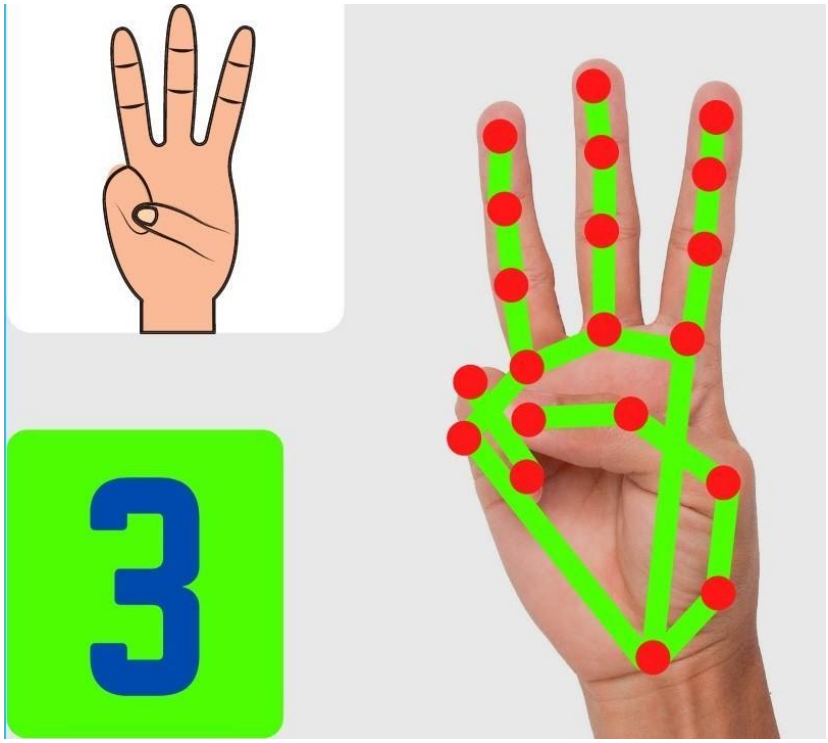
```
import HandTrackingModule as htm
detector = htm.handDetector(detectionCon=0.75)
```

This function finds 21 positions on the palm and stores the coordinate in LmList.

```
img = detector.handsFinder(img)
lmList = detector.positionFinder(img, draw=False)
```

Now we have stores of the ID's of the top landmark of each finger.

```
tipIds = [4, 8, 12, 16, 20]
```



Now we have compared the positions of ID's with other ID's of the same finger if the top ID landmark of the same finger found below the other than the finger is considered as folded .

Example : index finger top has ID as 8, now if in live feed 8 goes below 6 we consider index finger as folded.
But in case of thumb it can't go below its lower bone, so if the thumb top goes below 6 we consider it to be folded.

Then we count the number of non-folded fingers (i.e. tips of that finger above the bone connecting palm and finger) .

```
if lmList[tipIds[0]][1] > lmList[tipIds[0] - 1][1]:
    fingers.append(1)
else:
    fingers.append(0)
```

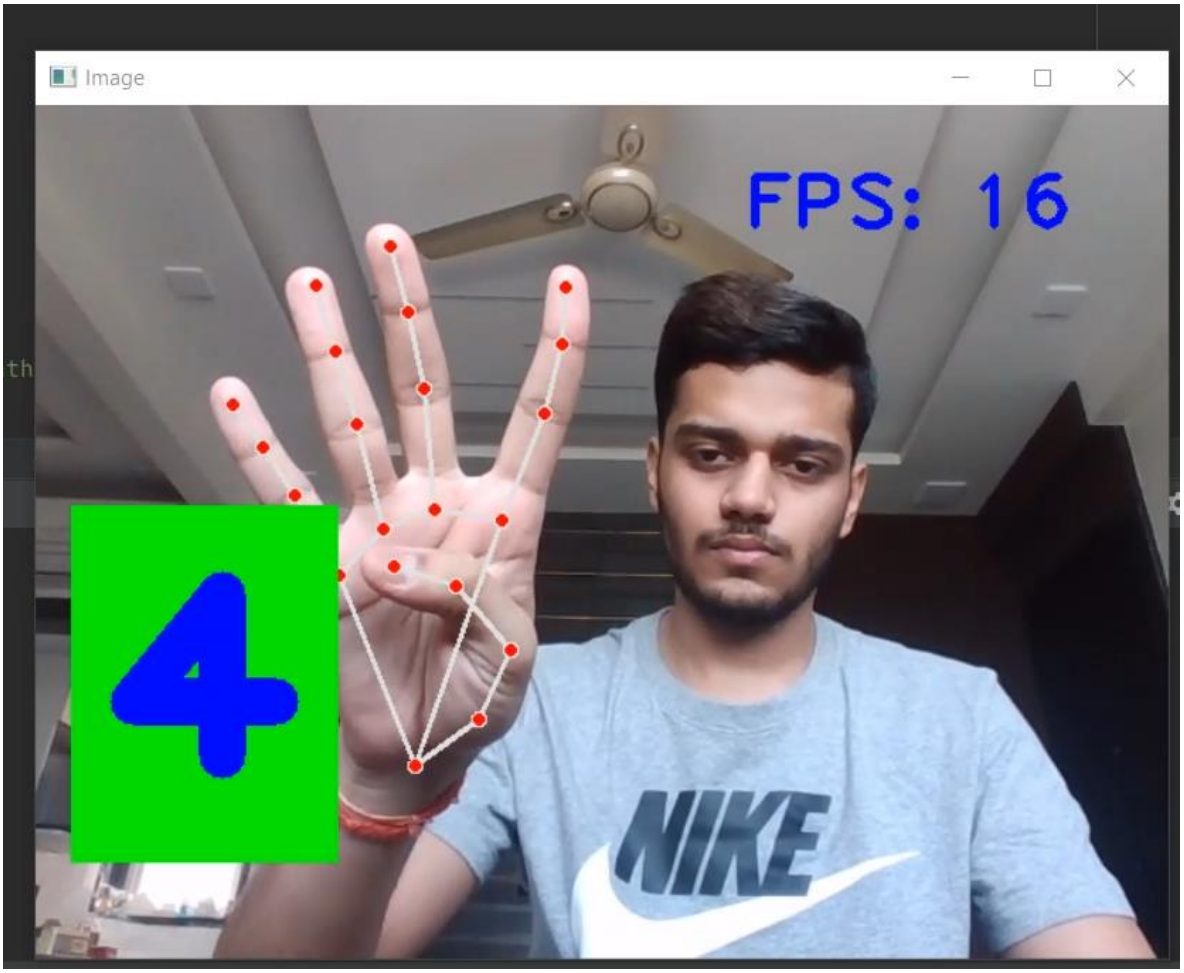
```
# 4 Fingers
for id in range(1, 5):
      if lmList[tipIds[id]][2] < lmList[tipIds[id] - 2][2]:
            fingers.append(1)
      else:
            fingers.append(0)
```

RESULT:-

Now when we run the code we get to see our live feed.

Taking a hand in front of the camera shows us 21 landmarks of the palm which we are shown as red dots which are connected by green lines.

From below gif we can observe that as if ID of the top of finger Goes above the other ID's of the same finger that present below it then it considered as unfolded and is counted.
And when it goes below that it is considered as folded.



This is just an example of what we can do by changing the relative positions of 21 points. We can use the positions to give different gestures of hand as input. These gestures can be feeded as a command to certain devices through which can control hardwares simply by hanging our hand gesture.

Conclusion and future Insights

We can conclude that our exploration of this project can be used in various fields in daily life. This will reduce the human dependency on hardware.

In this time of pandemic we are avoiding touch to any surface. By our project we can make smart control systems which minimizes human touch.

This has tremendous future use like - smart homes, high security systems etc.

This is moving at tremendous speed for futuristic products and services and major companies are developing technology based on the hand gesture system and that includes companies like Microsoft, Samsung, Sony.

It's a brilliant feature turning data into features with a mix of technology and Human wave.This also reduces our dependency on hardware as well.

Smart phones have been experiencing an enormous amount of Gesture Recognition Technology with look and views and working to manage the Smartphone in reading, viewing and that includes what we call touch less gestures.
In the medical fields Hand Gesture may also be experienced in terms of Robotic Nurse and medical assistance. As the Technology is always revolving and changing the future is quite unpredictable

but we have to be certain the future of Gesture Recognition is here to stay with more and eventful and Life touching experiences.

References

- https://www.section.io/engineering-education/creating-a-hand-tracking-module/
- https://www.youtube.com/watch?v=9iEPzbG-xLE
- https://www.youtube.com/watch?v=NZde8Xt78Iw
- https://www.youtube.com/watch?v=p5Z_GGRCI5s