

A Mini Project with Seminar On

VOLUME ADJUSTMENT

-by an Action

Submitted in partial fulfillment of the requirements for the award of the

Bachelor of Technology

In

Department of Computer Science and Engineering

By

| | |
|-------------------------|-------------------|
| Korra Rakesh | 19245A0519 |
| Mohammed Asif | 19245A0520 |
| Gujja Sai Nikhil | 19245A0522 |

Under the Esteemed guidance of

Dr.Ashlin Deepa R. N

Assistant professor



Department of Computer Science and Engineering

**GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND
TECHNOLOGY (Approved by AICTE, Autonomous under JNTUH,
Hyderabad, Bachupally, Kukatpally, Hyderabad-500090)**



**GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND
TECHNOLOGY(Approved by AICTE, Autonomous under JNTUH,
Hyderabad, Bachupally, Kukatpally, Hyderabad-500090**

CERTIFICATE

This is to certify that the major project entitled “VOLUME ADJUSTMENT” is submitted by **Mohammed Asif(19245A0520)**, **Gujja Sainikhil (19245A0522)**, **Korra Rakesh (19245A0519)** in partial fulfillment of the award of degree in BACHELOR OF TECHNOLOGY in Computer Science and Engineering during academic year 2020-2021.

INTERNAL GUIDE

DR.ASHLIN DEEPA R. N

Assistant Professor

HEAD OF THE DEPARTMENT

Dr. K. MADHAVI

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

There are many people who helped us directly and indirectly to complete our project successfully. We would like to take this opportunity to thank one and all. First, we would like to express our deep gratitude towards our internal guide **Dr. Ashlin Deepa RN, Assistant Prof. Department of CSE** for her support in the completion of our dissertation. We wish to express our sincere thanks to **Dr. K. Madhavi, HOD, Department of CSE** and to our principal **Dr. J. Praveen** for providing the facilities to complete the dissertation. We would like to thank all our faculty and friends for their help and constructive criticism during the project period. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

Mohammed Asif(19245A0520)

Gujja Sai Nikhil(19245A0522)

Korra Rakesh(19245A0519)

DECLARATION

We hereby declare that the industrial major project entitled “**VOLUME ADJUSTMENT**” is the work done during the period from **8th March 2021 to 8th July 2021** and is submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering from Gokaraju Rangaraju Institute of Engineering and Technology (Autonomous under Jawaharlal Nehru Technology University, Hyderabad). The results embodied in this project have not been submitted to any other university or Institution for the award of any degree or diploma.

Mohammed Asif

(19245A0520)

Gujja Sai Nikhil

(19245A0522)

Korra Rakesh

(19245A0519)

ABSTRACT

Nowadays most people are using computers instead of TV. In this project, we are focusing on novice users. We took a real-time problem faced by novice users while using computers and we are trying to resolve one of the issues i.e., VOLUME ADJUSTMENT. Novice users: Those users who don't know much about computers and how to operate. They were afraid of operating small things such as volume, brightness...etc. for better human-computer interaction (HCI) where we are using hand gesture recognition using the real-time camera to overcome the issues faced by novice users and make the system cheaper and more user-friendly. Our method is to use a camera and computer vision technology, such as image segmentation and gesture recognition, to control the system's volume. So that novice users can easily control the volume of their system using fingers within the radius and they no need to use the Mouse and keyboard of the system. Using this project novice users can able to interact with the computers in an easier way just by hand gestures.

TABLE OF CONTENTS

| CONTENTS | Page No. |
|--|----------|
| Title Page | i |
| Certificate by the supervisor | ii |
| Acknowledgment | iii |
| DECLARATION | iv |
| ABSTRACT | v |
| Chapter 1: INTRODUCTION | 1 |
| 1.1 Rationale | |
| 1.2 Goal | 2 |
| 1.3 Objective | 2 |
| 1.4 Existing Systems | 2 |
| 1.5 Methodology | 2-3 |
| 1.6 Roles and Responsibilities | 4-6 |
| 1.7 Contribution of Project..... | 6 |
| 1.7.1 Market potential | |
| 1.7.2 Innovativeness | 7 |
| 1.7.3 Usefulness | 7 |
| 1.8 Report Organization | 8 |
| Chapter 2 : REQUIREMENT ENGINEERING | |
| 2.1 Functional Requirement | 9-10 |
| 2.1.1 Interface Requirement | 10 |
| 2.2 Non Functional Requirement | 10 |
| Chapter 3 : ANALYSIS AND DESIGN | |
| 3.1 Use-case Diagram | 11 |
| 3.2 Activity Diagram | 12 |
| 3.3 Sequence Diagram | 13 |

| | |
|--|-------|
| Chapter 4 : CONSTRUCTION | |
| 4.1 Implementation | 14 |
| 4.2 Modules..... | 14 |
| 4.2.1 Hand Captures | 14 |
| 4.2.2 Hand Tracking | 15-21 |
| 4.2.3 Volume Control | 22-26 |
| 4.3 Datasets | 27-28 |
| 4.4 Software Details | 28 |
| 4.5 Hardware Details | 28-29 |
| 4.6 Testing | 29-31 |
| 4.6.1 Test Cases | 32-35 |
| Chapter 5: Conclusion and Future Scope | |
| 5.1 Conclusion | 36 |
| 5.2 Future Scope | 36 |
| Chapter 6: Appendixes | |
| 6.1 APPENDIX I | 37-40 |
| 6.2 APPENDIX II | 41-44 |
| References | 45-46 |

CHAPTER-1

INTRODUCTION

The main goal of gesture recognition is to create a system that can detect human gestures and use them to send information to the control device, as well as by the implementation of gesture recognition in real time, the user can control the computer by means of a particular gesture in front of a camera connected to your computer. The setup consists of a single camera to capture the gesture formed with the user and take this as input to the system. In this project we will develop a hand gesture volume control system with the help of OpenCV module. Here the system can be operated during hand movement(action) without using keyboard and mouse. Nowadays we control volume using various devices like a remote for the TV and a mouse for Laptop or pc. Those users who don't know much about computers and how to operate. These users who don't know how to use or are new to these kinds of things are called novice users. As we tested with novice users for controlling volume by using a mouse it took approximately equal to 1 minute. But now in our project, the novice users can use their fingers to control the volume which takes less than 30 seconds. We use Gesture control to change the volume of a computer. We use hand tracking and then we will use the hand landmarks to find the gesture of our hand to change the volume. This project is not only for novice users but for all users who know how to control the volume.

1.1 Rationale

With the rapid growth in new technology and gadgets, there is a requirement to ease the process of using the latest gadgets. Not only for novice users but also for normal users there is a requirement to ease the usage of gadgets. This simplifying the usage of gadgets requires modern technology i.e Gesture recognition. In gesture recognition, enables real-time data to the computer to perform tasks. The motion sensors in your device, you are able to follow and interpret the gestures and use them as a primary source. In contrast to the traditional buttons, knobs and menus; it can be done does not interrupt the activity of the user, forcing him to put his hand in the direction of the commission. Instead, they can be carried out starting from the current cursor position. In addition, they do not require any additional devices in the order of the parameters that can be configured with a simple swipe of your hand. The Input devices can reduce the user

experience for example, a pen or the mouse, to a limited extent as input to a touch screen. These are the words to create this design be more useful for all types of people have to adjust the volume more easily.

1.2 Goal

Most of the work is divided into three sections:

1. Capture Hand
2. Hand Tracking
3. Volume control

1.3 Objective

The main objective of our project is to:

1. With the help of the OpenCV library, we need to capture the Images through the Camera
2. By using the Media Pipe Framework, we should recognize the Hand and Finger movement/actions to control the system's volume.
3. With Pycaw library, we can control volume according to the finger movements.

1.4 Existing Systems:

1. Hand Gesture Recognition Using Kinect Sensor(Robust Part-Based) [4]:

Inexpensive depth cameras, The Kinect sensor can be used to make a reliable part-based hand gesture recognition. As with the kinect sensor, with the low-resolution, it can be difficult to recognize, but it can be a simple large object. In order to combat the tough movements will be recorded by the kinect sensors, the authors propose a new distance metric is known as the finger to dig the distance. Just the fingers, to respond to FEMD, but it's not the whole hand .Raucous hand, can be better controlled, as 0FEMD able to differentiate between the gestures, with minor differences. This system works well and is efficient for an uncontrolled environment. The accuracy of 93.2%, is achieved by the experimental results.

2. Daily Information Retrieval from Internet by Real-Time Hand Gesture Recognition System[3]:

In this paper, a system is proposed that, in such a way that it is the everyday information that was downloaded from the Internet with the help of the movements of the hand. To determine the hand, the analysis of the main components. With the help of the YCbCr color space for skin detection and CAMSHIFT algorithm can be used to detect and track hand gestures. To turn on the CAMERA algorithm that will be used when all of the terms and conditions for the start of the track, are simple and easy .The segmentation and normalization were carried out with the help of ATP. The experiment shows that the accuracy of hand gesture recognition is achieved 93.1%.

3. For hand gesture recognition Combining multiple depth-based descriptors [5]

On the basis of information about the depth of the image from the camera to the authors of a plan known as the novel hand gesture recognition scheme. In order to correctly identify the complex movements, using the three-dimensional information, we use a set of three-dimensional objects. The SVM classifier based on the feature vectors in order to determine the one hand, the gesture is performed in front of the camera for .95% of accuracy is achieved by the combination of a set of grades, and the SVM classifier.

4. Voice Controlled System based on Arduino [1]:

In this system, a voice recognition module is used for the recognition of the voice of the user in order to determine the direction of the wheelchair. The promotion will be used in this project is to help the management of the household, and with the help of voice commands, so that this project can also act as a home automation system.

5. Using Accelerometer Data, ANN for Gesture Recognition [12]:

The authors introduced an Artificial Neural network application used for the classification and gesture recognition. The gesture recognition is done through the Wii remote, this remote will rotate in X,Y,Z directions. To reduce the computational cost and memory consumption the gesture recognition is processed in two levels. In first level User Authentication is done for gesture recognition. Accelerometer- Based gesture recognition method is used .In second level without any kind of signal processing for gesture recognition Fuzzy automata algorithm has been proposed. After recognizing the data of the gestures, the data was normalized and filtered by k-means and Fast Fourier transform algorithm. Using this Dynamic Bayesian Network The recognition accuracy has increased up to 95%.

1.5 Methodology

Open CV

This is a machine learning approach where a cascade function is to learn more of the positive and negative images, and then used to detect objects in other images.

They are just like our convolutional kernel. Each feature is a single value obtained by subtracting the sum of pixels under the white rectangle from the sum of pixels under the black rectangle.

The core functions of the OpenCV library cover the basic data structures such as Scalar, Point, Range, etc. To store images, it has the multidimensional array Mat. This project is developed using python technology, the code is arranged and designed in python using OpenCV and NumPy modules.

we have to capture live stream with the help of a webcam. OpenCV provides a very simple interface to this. Let's make a video out of the camera (I am using the in-built webcam of my laptop), and we converted it to grayscale and full names.

To capture a video, you will need a video Capture object. The argument may be a device, a pointer, or a video file. The device index is just a number, which indicates the camera. As a rule, it is a camera that is connected to the system (as in my case). So, I'm going to just have to be 0 (or -1). You can select a different camera, the following is 1, and so on. After that, you can capture frame-by-frame. However, in the end, don't forget to check the input.

Media Pipe

The Media pipe is, it is a framework that is used for constructing ML pipelines. It works with a variety of solutions, such as face recognition, Hand Detection, object recognition, and Holistic Approach to the Front to Form, etc., etc. Media-Pipe Hands to make use of an ML pipeline which is composed of a number of models, to work together in A palm-detection model, which works on the entire image, and it gives the entire border of the frame to the hand. As a reference, a model of the hand of the image, crop the area that is specified in the palm of your hand and it gives you precise, 3D-most points in the hand.

Palm Detection Model Palms are smaller objects, the non-maximum suppression algorithm works well even for two-hand self-occlusion cases, like handshakes. Moreover, palms can be modeled using square bounding boxes (anchors in ML terminology) ignoring other aspect ratios. With the above techniques, we achieve an average precision of 95.7% in palm detection.

A reference Arm and a Model For the detection of the hand across the entire image, check out our next reference of the arm model is run with the precise location of 21 in a three-dimensional, hand, joint coordinates are within the approved arm with the help of the regression model, i.e., direct, coordinate prediction.

MediaPipe

Live ML anywhere

MediaPipe offers cross-platform, customizable ML solutions for live and streaming media.




| | |
|---|--|
|  <p>End-to-End acceleration: Built-in fast ML inference and processing accelerated even on common hardware</p> |  <p>Build once, deploy anywhere: Unified solution works across Android, iOS, desktop/cloud, web and IoT</p> |
|  <p>Ready-to-use solutions: Cutting-edge ML solutions demonstrating full power of the framework</p> |  <p>Free and open source: Framework and solutions both under Apache 2.0, fully extensible and customizable</p> |

Fig No: 1.1 MediaPipe overview



Fig 1.2-Hand Landmark Model

- **Detection.** With the help of a digicam, a tool detects hand or frame moves, and a machine gaining knowledge of set of rules segments the image to locate hand edges and positions.
- **Tracking.** A tool video display units moves body via frame to capture each motion and offer accurate input for records evaluation.
- **Recognition.** The device attempts to discover patterns primarily based on the accrued records. While the gadget reveals a fit and interprets a gesture, it performs the action related to this gesture. Function extraction and class within the scheme under implements the recognition functionality.

1.6 Role and Responsibilities

| Name | Responsibilities |
|------------------|--|
| Korra Rakesh | <ul style="list-style-type: none">● Reading Research papers● Implementation● Testing |
| Mohammed Asif | <ul style="list-style-type: none">● Implementation● Documentation● Testing |
| Gujja Sai Nikhil | <ul style="list-style-type: none">● Reading research papers● Implementation● Documentation |

1.7 Contribution of Project

1.7.1 Market potential

In computer science and language technology, gesture recognition is a hot topic that interprets human gestures using computer-vision algorithms. There have been various movements of the body, which can limit motion, the general form of the character of the development of the face and hands. The whole movement process is to be represented by them, and if you want to set a specific target group, it is known as the recognition of the gesture. Several different technologies have been used in the development and deployment of such devices, communication technologies, and the visual interference of the two main types of technologies used for the creation of a reliable, accurate, and reliable hand gesture recognition systems. To Contact devices, such as accelerometers⁷, multi-touch screens, data, gloves, ⁹, etc., etc. it is based on the physical interaction of the users, who need to learn how to use them. In view of the fact that the development of a vision-based devices, such as cameras, have a significantly different language. Gesture recognition involves processing of degrees of freedom, 4, 10 (DOF), two-dimensional symptom of a variable at different levels of spatial resolution, and size (i.e., changes in the movement speed. The gesture recognition is based on the vision for the future. It can be divided into two broad categories: methods that are based on a three-dimensional model and the other methods, which are based on their appearance.¹, the 3D model of the hands⁴, and a description of the shape of the hand, and they are the first choice for the simulation of the hand gestures that are recognized in the surround sound analyzer. In video-based models, including the appearance of the hands, and the movement of the hands, which is directly linked with the image of a particular movement can be shown in 4. This group includes a wide range of models. If any of these models, i.e., models that are based on the geometry of its form, it had been determined that it is a symptom

1.7.2 Innovativeness

Behind this project the idea is to ease the controlling of volume by using hand gestures. Usually, we control volume by the remote whereas here we use hand gestures to control the volume up or down. The novice users who don't know how to operate these kinds of things can easily use these things in his day to day life.

1.7.3 Usefulness

Gestures provide the user with a new form of interaction that mirrors their experience in the real world. They feel natural and require neither interruption nor an additional device. Furthermore, they do not limit the user to a single point of input, but instead offer various forms of interaction. Unlike traditional buttons and menus, gestures do not interrupt the user's activity by forcing him to move his hand to the location of a command. Instead, they can be performed directly

1.8 Report Organization

The remaining part of the project report is structured as the following:

- In **Chapter 2** detailed business and technical requirements are provided
- Analysis and design of the project is mentioned in **Chapter 3**
- provides implementation, Construction details of this project is provided in **Chapter 4**
- In **Chapter 5** Future scope and Conclusion as well as future application of this project are provided

CHAPTER-2

REQUIREMENT ENGINEERING

2.1 Functional Requirement



Fig no: 2.1 Flow Chart

In our project, we are divided into three modules as shown in figure 2.1.

- In module1, we are accessing System's webcam to capture the objects with help of the OpenCV library and send them to module2.
- Module2, after capturing the objects, has to remove other objects except for the palm and by mideapipe framework helps to detect the palm and also points the 21 key points of the palm.
- Module3 with the key points will control the system's volume with the pycaw library by moving the palm throughout the system's webcam.

Model Workflow

- At first, the webcam captures the Palm using Palm Detector Model and draws a bounding box around the hand.
- Next, the hand landmark model locates 21 keypoint 2D hand coordinates.
- Then, these hand landmarks are captured by the model which is preprocessed further and sent to the Volume control module, it will control the system's volume while moving the palm through of the webcam.

Media Pipe Hand is a machine-learning employed high-fidelity hand and finger tracking solution. It detects 21 Landmark points as shown in Fig. are recorded from a hand in a single frame with the help of multiple models which are working simultaneously.

Media pipe Hands consists of two different model's hand-Palm Detection Model in which a full image is identified and it draws a box around the hand, and Hand - Land-mark Model operates on this boxed image's formed by Palm-Detector and provides high fidelity 2D hand keypoint coordinates. (As shown in above fig. 1.1)

To obtain ground truth knowledge, we've manually annotated ~30K real-world pictures with 21 3D coordinates, as shown below (we take Z-value from image depth map, if it exists per corresponding coordinate). to raised cowl the attainable hand poses and supply further management on the character of hand pure geometric we have a tendency to conjointly render a high-quality artificial hand model over various backgrounds and map it to the corresponding 3D coordinates.

2.1.1 Interface Requirement:

- The user needs to run the application.
- The play's the Audio or video.
- According to users needs to increases or decrease of system's volume by just moving the palm in front of the webcam(show the palm such as that must be capture in webcam) without using a mouse, keyboard..etc.

2.2 Non Functional Requirement

non-functional requirement is a specification that describes the system's operation capabilities and constraints that enhance its functionality. These may be speed, security, reliability, etc

- To provide maximum accuracy
- Mainly useful for novice users
- Ease of use.
- Reliability
- No need to buy any devices (Maximum all computers have a built-in webcam).

CHAPTER - 3

ANALYSIS AND DESIGN

3.1 Use Case diagram

Use case diagrams to symbolize the general situation of the system. A situation is not anything however a collection of steps describing an interplay among a person and a system. Thus a use case is a set of situations tied collectively with the aid of using a few goal. The use case diagrams are drawn for exposing the functionalities of the system..

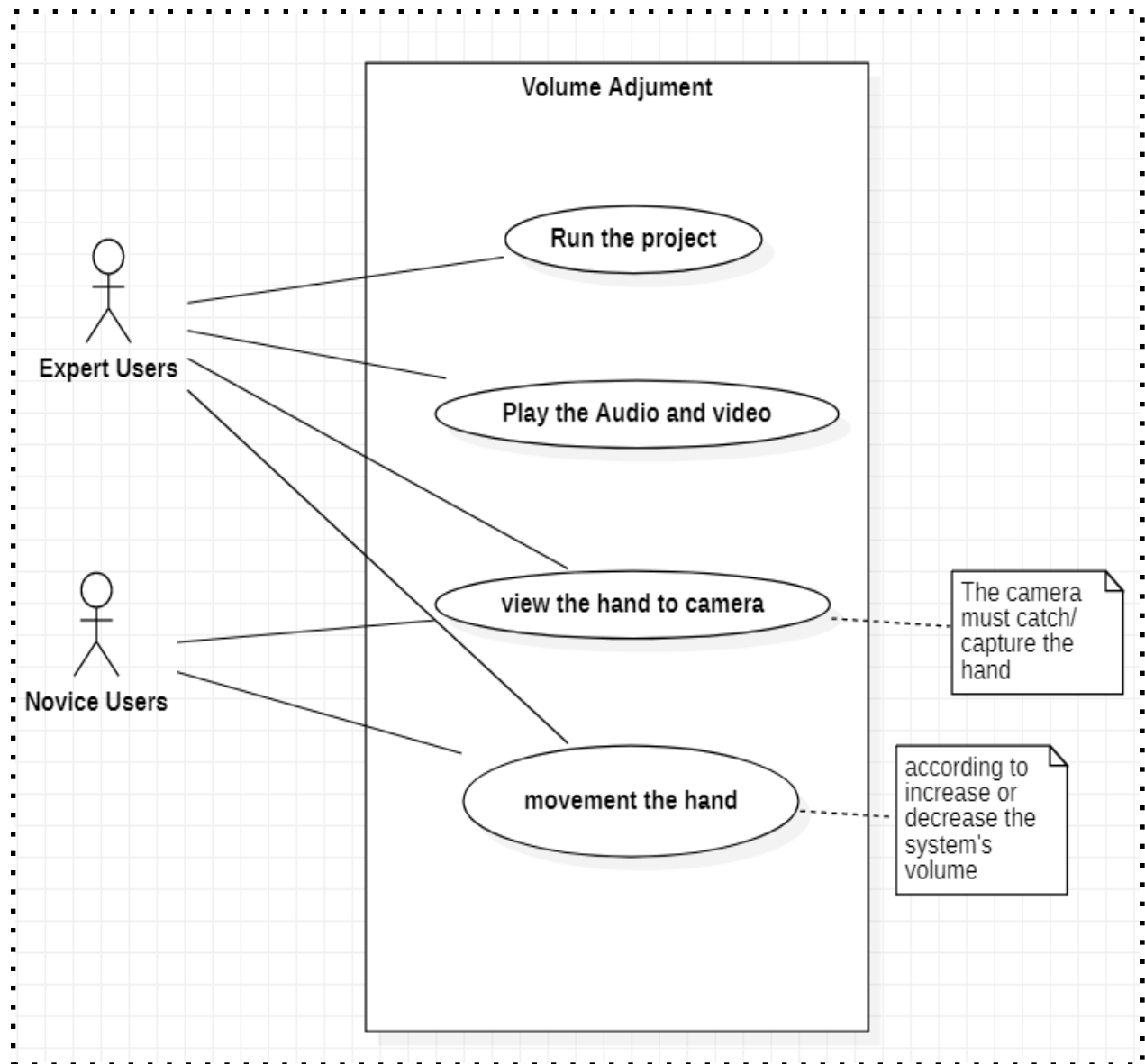


Fig no: 3.1 Use-case diagram

3.2 ACTIVITY DIAGRAM

An activity diagram is a clear representation of the flow of communication within certain contexts. It is similar to a flowchart in which the various functions that can be performed in a system are represented.

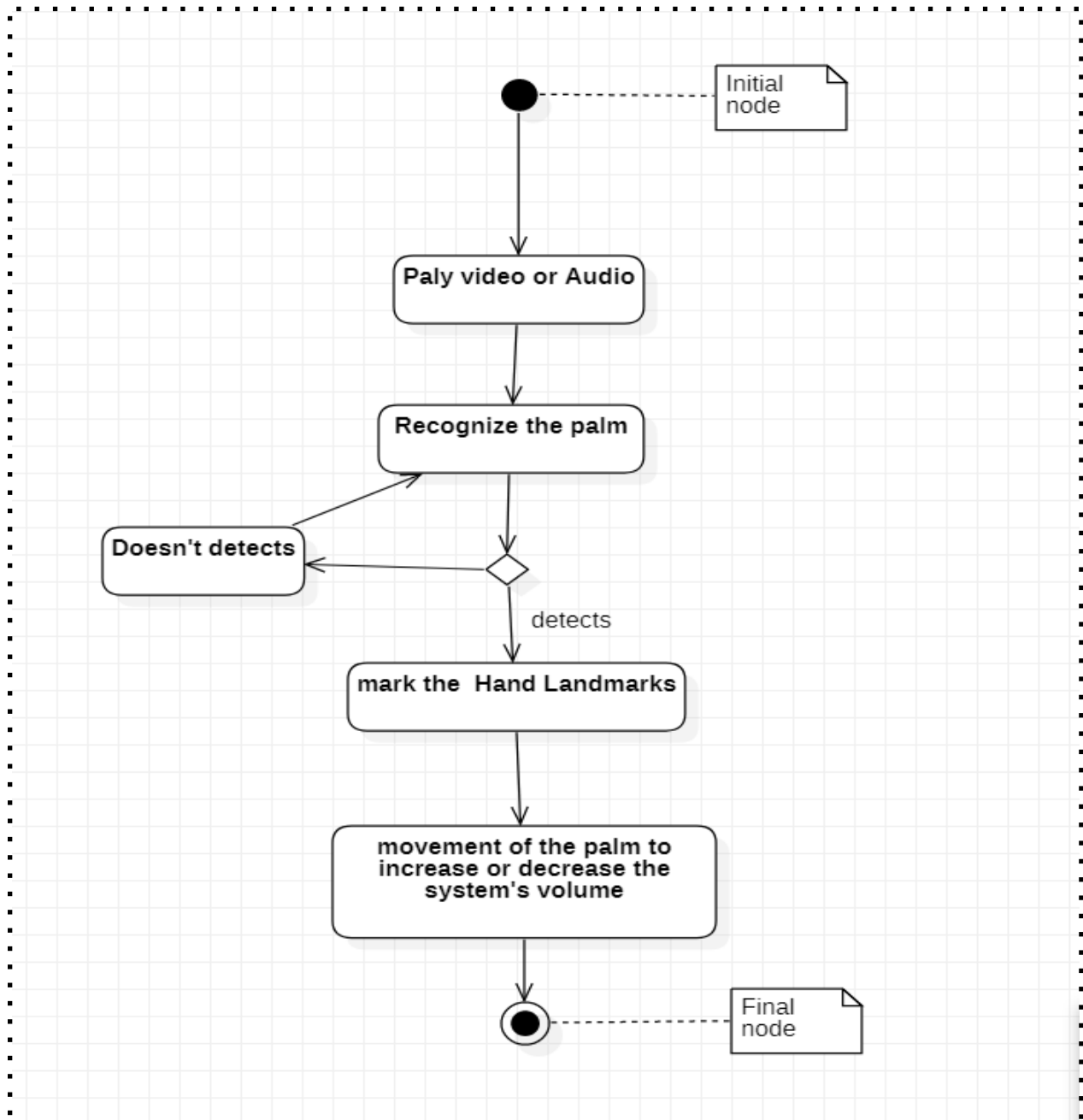


Fig no: 3.2 Activity Diagram

3.3 Sequence Diagram :

In the sequence diagram how an object interacts with another is shown. There is a sequence of events represented.

It is a period situated perspective on the association between items to achieve a conduct objective of the framework

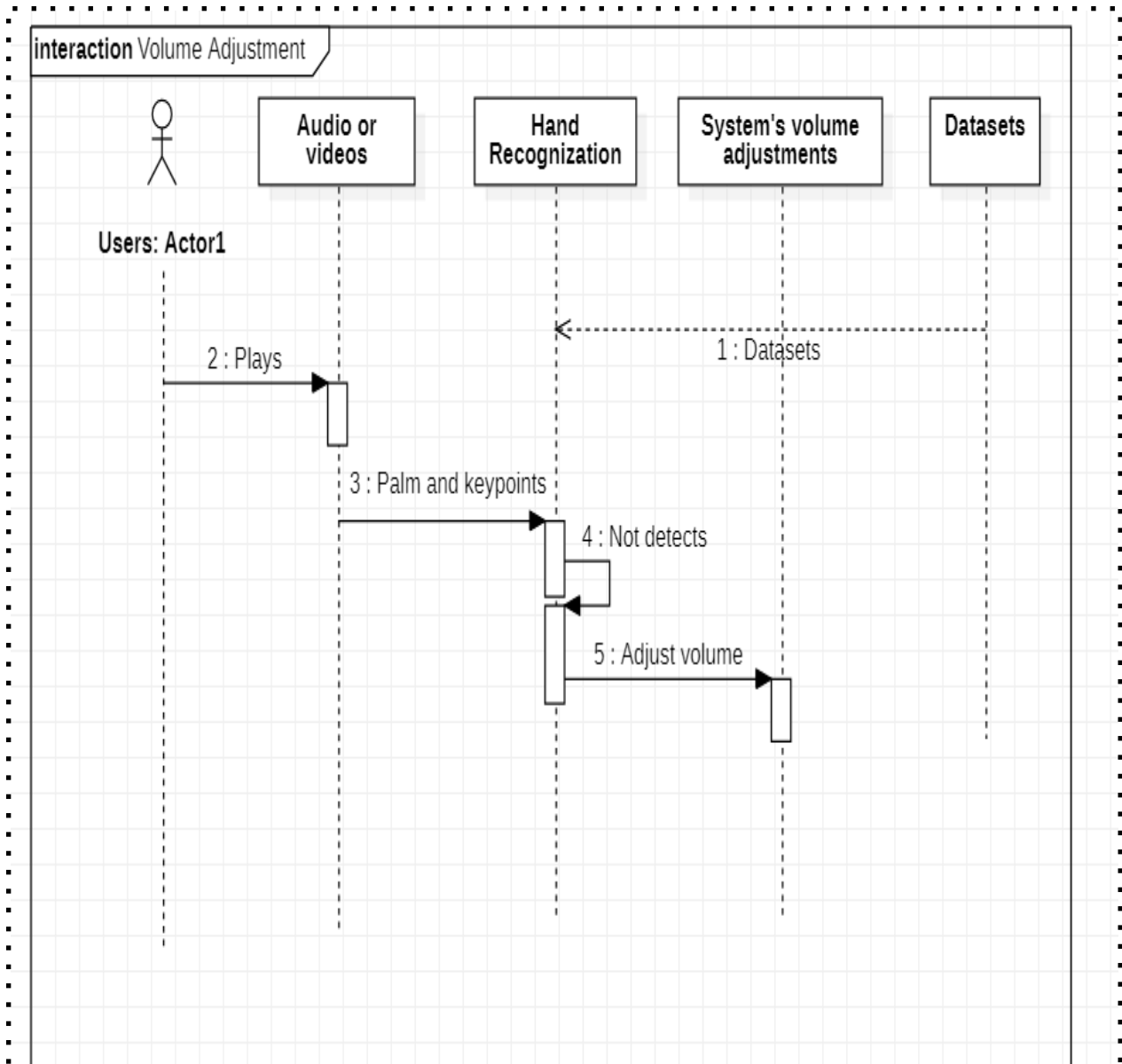


Fig no:3.3 Sequence Diagram

CHAPTER - 4

CONSTRUCTION

4.1 Implementation

The Project divided into three modules:

1. Capture the Hand (Capturing the Frames from a camera)
2. Hand Tracking (Recognizing the Palm and tracking the fingers movement)
3. Volume Control (Controlling the volume)

4.2 Models

4.2.1 Capture the Hand

In this module, we capture the images through the webcam of the system, With the help of the OpenCV library. Video Capture function is used to access the webcam of the system. The read function is used to capture the images from the webcam. Continuously it captures the images and sends them to Module2 for Hand Recognition. In this module, a camera, and the image of the hand and can be obtained from a variety of sources, such as a camera, a depth camera, and an infrared camera system. It is the image of the hand is to be divided, so that it can be the splitting of the image into its respective sections of this document. It's going to be the removal of the user's hand from the background to capture the gestures, and without noise. However, both formations (static) and hand movements (dynamic) in the various segments of the processes depends on the nature of the act. If it is a dynamic way, it has to go through two stages before it is sliced. First of all, the hand gesture to be detected and tracked by the shell, and then use the gesture in the video has to be divided into frames and each frame has to be processed individually . A perfect segmentation, it is difficult to obtain, because of the layering of the skin color, with a complex background. However, due to the sensitive nature of the interior lighting changes, a lot of studies to try to exclude, from the clarity of the item. The most important thing in the hand segmentation is the threshold that will be used for the separation of the hand from the background. The threshold value can be used as a method for the extraction of the background to the setting up of the values of each pixel as a threshold value [13]. The threshold will have to create a binary image with all of the 1 pixel by the hand. This occurs when a pixel with an intensity lower than the threshold value is set to 0, while all pixels with an intensity greater than the threshold value will be set to 1. [13].

4.2.2 Hand Tracking [6]

In module 2, we took the captured images from module 1 and with the help of the **Media pipe** framework will detect the palm(Hand) and mark the coordinates of the fingers of the palm.

The palms can be modeled using square bounding boxes ignoring other aspect ratios. After the palm detection over the whole image our subsequent hand landmark model performs precise key point localization of 21 3D hand-knuckle co-ordinates inside the detected hand regions via regression, that is direct coordinate prediction.

Hand Tracking module further divided into two modules:

1. Palm Detection
2. Hand landmarks

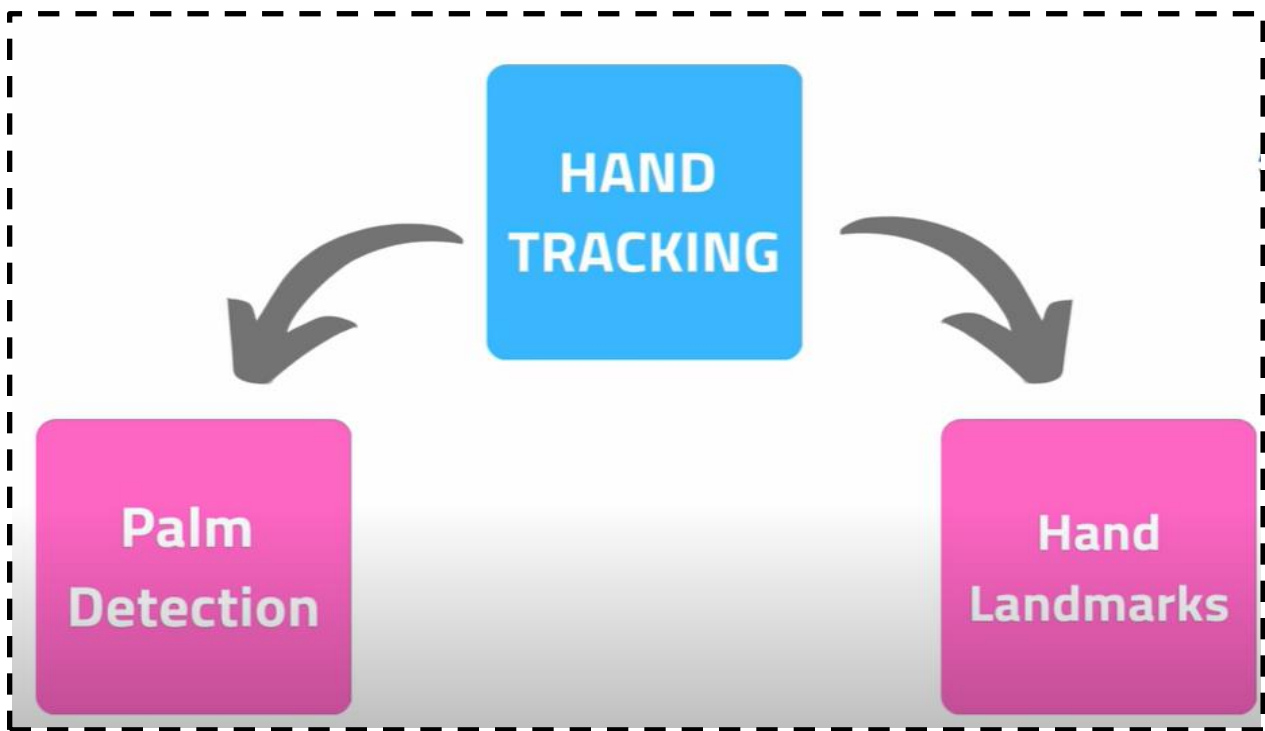


Fig no: 4.1 Hand Tracking Model

1) Palm Detector

- a. The palm branch the detector model, the so-called BlazePalm works on the entire image, and it gives the entire border of the frame to the hand.

- b. Detecting objects in images using an **SSD: Single Shot MultiBox Detector** [11]. First of all, we'll work out a palm to the detector in place of a hand, a detector, since the evaluation of the bounding boxes of rigid objects, such as hands and wrists, it is so much easier than the detection of hands with articulated fingers. Also, since the hands are of smaller objects, the non-maximum suppression system is working well, even in the case of the two-hand, self-occlusion, such as in a handshake. In addition, palm trees, and can be modeled with the aid of only a square bounding box, and ignore the other aspects, and, as a result of the reduction in the number of posts, 3 to 5 times.
- c. As shown in figure 4.2, (b) 8x8 feature map shows the bounding boxes of objects. For a single object, there are many bounding boxes, which use the non-maximum suppression algorithm to get the more accurate bounding box. As in (c) 4x4 feature map selects the one bounding box that is more accurate compared to other bounding boxes of the object.

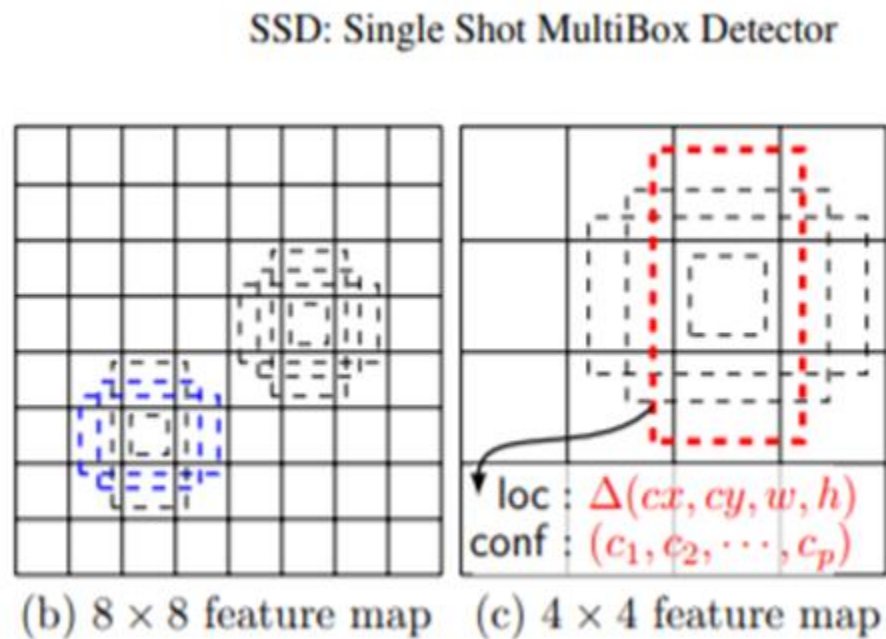


Fig no: 4.2 Object detection

Non - Max Suppression

- The objects in the picture may be of different sizes and shapes and are perfect to capture each and every one of them, the object detection algorithms in the form of a number of rigid frames. (the photo on the left). Best of all, each object in the image, we need to have a border, frame.

- To select the best bounding box , from the multiple predicted bounding boxes , these object detection algorithms use non-max suppression . This technique is used to “suppress” the less likely bounding boxes and keep only the best one.
- The purpose of non-max suppression is to select the best bounding box for an object and reject or “suppress” all other bounding boxes.
- You can see the image above fig no:4.2, along with the bounding boxes, the model returns an objectiveness score. This score denotes how certain the model is, that the desired object is present in this bounding box.
- You can see all the bounding boxes have the object, but only the red bounding box one is the best bounding box for detecting the object.
- The non-max suppression will first select the bounding box with the highest objectiveness score. And then remove all the other boxes with high overlap. So here, in the above image,
- We will select the red bounding box (since it has the highest objectiveness score of 98%)
- And remove other colored boxes (because they have a high overlap with the green box)

2)Hand landmarks [9]

A hand landmark model that operates on the cropped image region defined by the palm detector and returns high fidelity 3D hand keypoints.

The hand landmark model performs precise landmark localization of 21 3D coordinates(shown in figure 4.3) inside the detected hand regions via regression. The model learns a consistent internal hand pose representation and is robust even to partially visible hands and self-occlusions.

Researchers manually annotated around 30K real-world images with 21 coordinates to detect key points on the palm images. They also generated a synthetic dataset to improve the robustness of the hand landmark detection model.

To better cover the visibility of the hand and provide additional protection in the form of hand geometry, we also provide a high-quality hand model in a variety of domains and map to compatible 3D links.

Hand landmarks localization is performed in many gesture recognition systems, but usually it is limited to detecting fingertips of extended digit as well as finding the wrist and palm region . This is certainly helpful for estimating a hand pose, when combined with the analysis of some shape feature extracted from the hand silhouette or contour. Importantly, retrieving more information on hand landmarks location may improve the accuracy of the state-of-the-art technique, especially if it were possible to find the position of the landmarks located inside the hand silhouettes. It has been achieved only for the depth maps acquired using the Kinect sensor cameras , and this has substantially improved the accuracy and reliability of gesture-controlled interfaces.

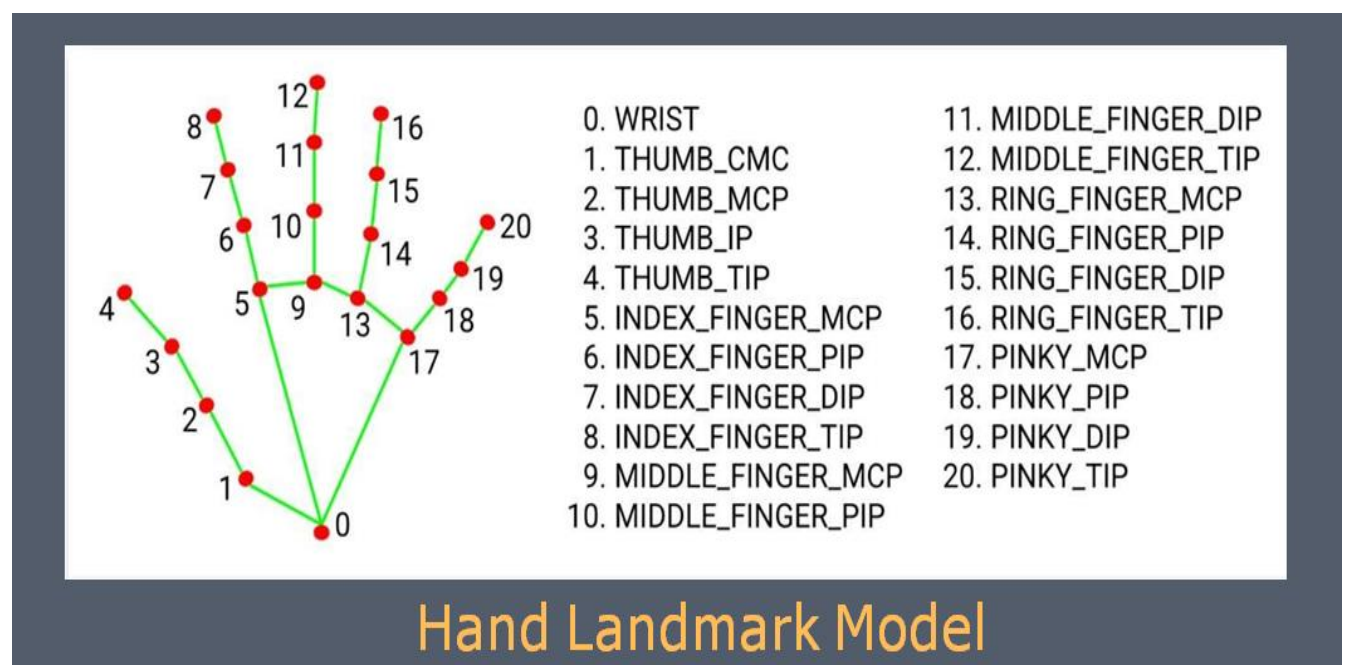


Fig no: 4.3 Hand landmark

Hand Tracking module overview [7]

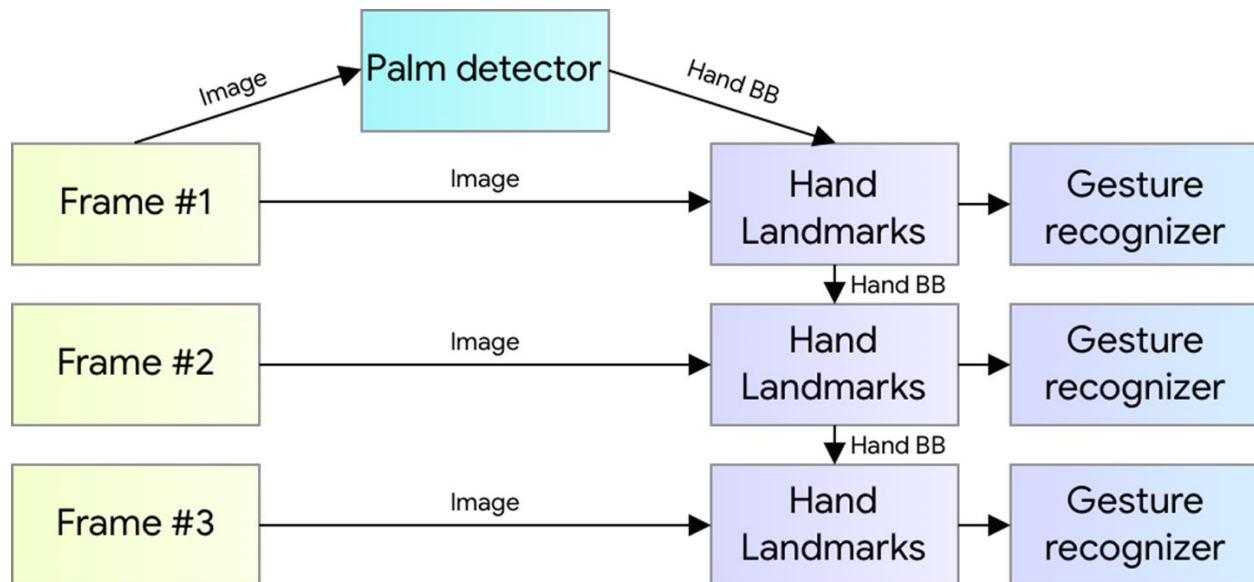


Fig no: 4.4 Overview of Hand Tracking

- A palm detector model (called BlazePalm) that operates on the full image and returns an oriented hand bounding box.
- A hand landmark model that operates on the cropped image region defined by the palm detector and returns high fidelity 3D hand keypoints.
- A gesture recognizer that classifies the previously computed keypoint configuration into a discrete set of gestures.

In this figure 4.4, can see the overview of the Hand tracking module. The Frames are taken from module 1 and those frames given to the palm detector, The palm detector detects the palm with the best accurate value of the bounding box. Then hand landmarks will point to the 21 key points on the palm.

Note: All frames will not be sent to the palm detector module, if and only when in Hand landmark module unable to detect the palm.

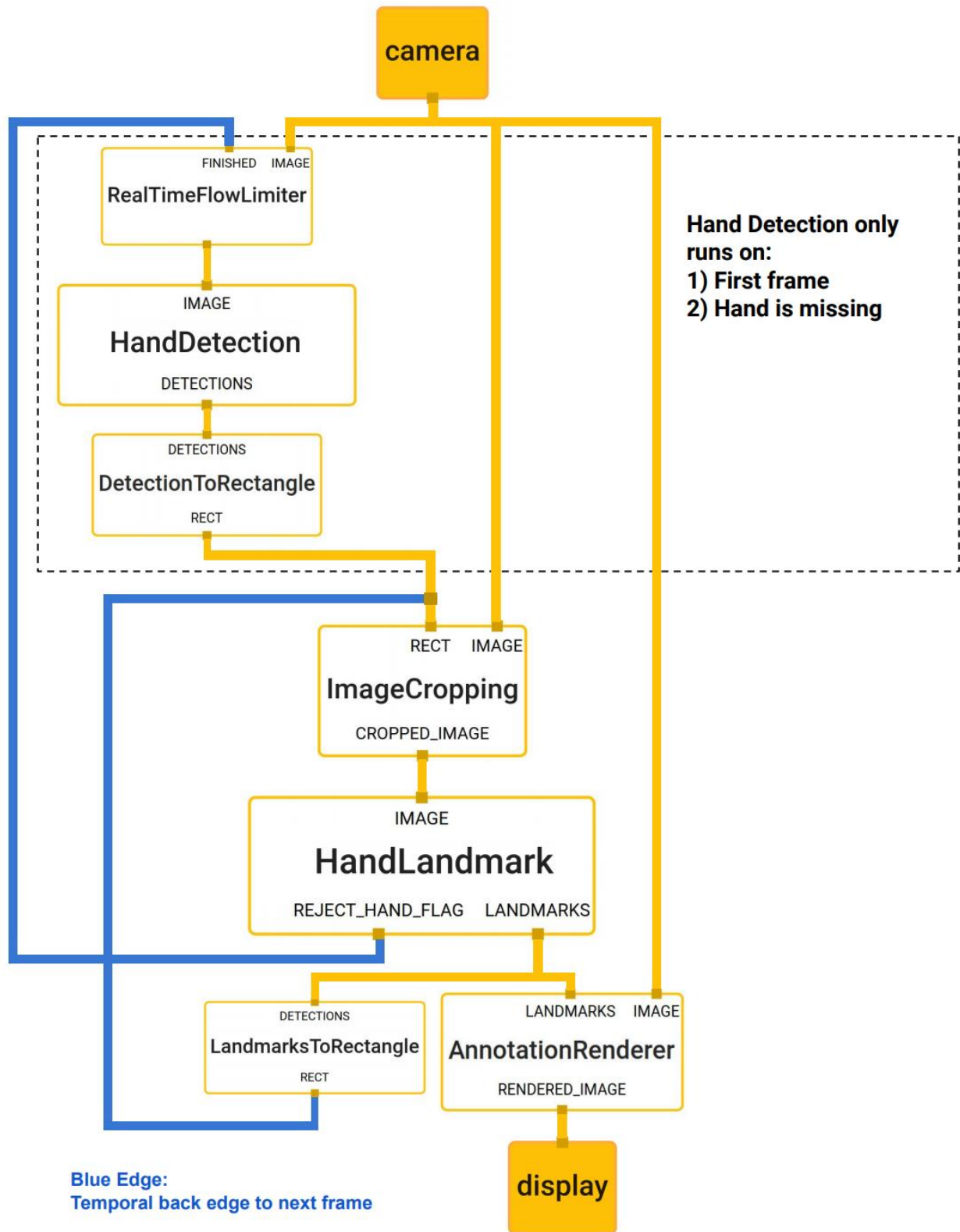


Fig no: 4.5 Flow chart

In figure 4.5, it captures the 30 frames per second (**FPS**) from the webcam. For the first frame, it will send to the palm detector module and if the palm is not detected then it goes to the second frame, it continuously checks the frames until it detects the palm. After detecting the palm then it crops the image and marks the 21 landmarks points in the hand landmark module. If it fails to detect the palm in-between frames then it again goes to the palm detector module.

Note: All frames will not continuously send the palm detector module, if and only if the frame is unable to detect the palm [7].

The most effective ML solution that works in real time and across all different platforms and form features including a much more complex difficulty than the one described above. To date, we open access to the above handwriting pipeline and touch direction in the MediaPipe framework, which is compatible with the appropriate end-to-end use and source code. This gives researchers and developers a complete stack of experiments and speculation of novel ideas according to our model.

With MediaPipe, our hand-tracking pipeline can be constructed as a direct graph of modular components, called Calculators. Mediapipe comes with an expandable set of Calculators to solve tasks such as model tracking, media processing, and data conversion on various devices and platforms. Individual calculators such as cropping, supplying and neural network integration are also improved to use GPU acceleration. For example, we use TFLite GPU capture on most modern phones. Our MediaPipe hand tracking graph is shown in Figure 4.5. The graph consists of two phases, one for manual acquisition and one for the calculation of land symbols. The key functionality of MediaPipe which provides that the palm detector only works as required (rarely), saves important calculations. We do this by getting a handful of current video frames from the world symbols integrated into the previous frame, eliminating the need to install a palm detector across all frames. Firmly, the hand tracker model also releases an additional scale for gaining confidence that the hand is present and well aligned to the input yield. Only when confidence falls below a certain threshold when a hand-to-hand acquisition model is applied to the following frame.

4.2.3 Volume Control

- Controlling the system's volume with a finger, After Detecting the palm and pointing the landmarks of the palm in Module2. In “Hand Landmark Model” has 21 landmarks. Here we focused on the 8th ("INDEX_FINGER_TIP") landmark to control the volume.
- While moving the Palm, will get the landmark number, X-axis, and y-axis in the list format. Here we ignore the y-axis point, just selecting the x-axis and landmark values to control the system's volume.
- Here we used the Pycaw library, the library is fully controlling the system volume. Mapping the ranges of the x-axis and the system's volume. Finally while moving the 8th landmark through the x-axis to control the system's volume.

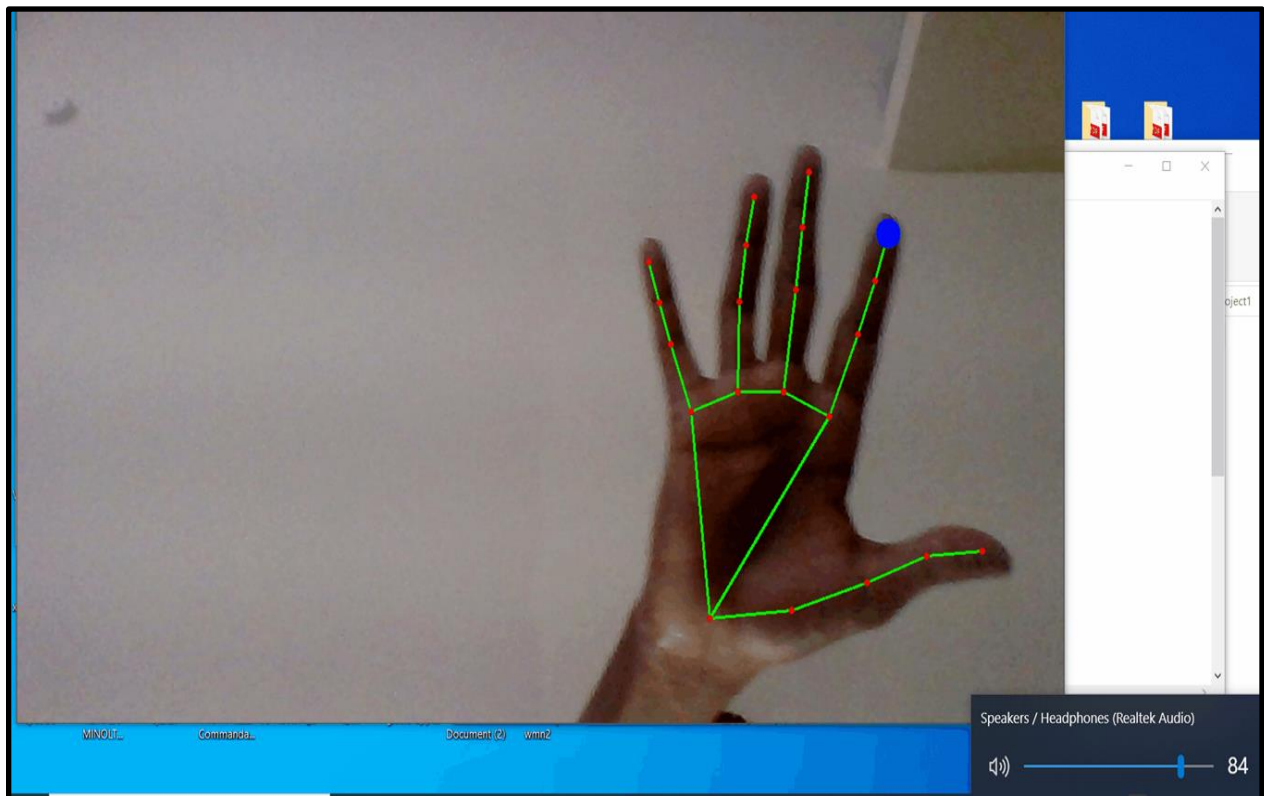


Fig no:4.6 Volume controlled

Implementation Details [9]

- We obtain hand landmarks from an image, using Python, MediaPipe and OpenCV. We will be using OpenCV to read the image and displaying it and MediaPipe to perform the hand detection and landmark estimation.
- MediaPipe is a free and open-source framework that provides cross-platform, customized solutions for live media and streaming [6].
- By importing the **cv2** module, which will allow us to read an image from the file system and display it, alongside the hand detection results, in a window. We will also import the **mediapipe** module, which will expose to us the functionality we need to do the estimation of the hands landmarks.
- we will access two sub modules from **mediapipe**, namely **drawing_utils** and **Hands**. The **drawing_utils** module includes some useful helper functions to draw detections and landmarks over images, amongst other functionalities. The **hands** module contains the **Hands** class that we will use to perform the detection of hand landmarks on an image. We are doing this as a convenience, to avoid using the full path every time we want to access one of the functionalities of these modules.
- It is important to take in consideration that the **process** method receives a RGB image but, when reading images with OpenCV, we obtain them in BGR format. As such, we will convert our original image first to RGB, with a call to the **cvtColor** function from the **cv2** module, and pass the result to the **process** method. Note that we need the image in RGB format just for the landmarks estimation process since we are going to display the result in a OpenCV window, which will display the image correctly in BGR format.
- We then draw the landmarks with a call to the **draw_landmarks** function from the **drawing_utils** module. As first input we pass the image where we want to draw the landmarks. As second input we pass the list of hand landmarks of the hand.
- As third input of the **draw_landmarks** function we will pass an optional parameter that corresponds to a list that indicates how landmarks connect to each other, which will allow to draw these connections between them as lines. We will use this python frozenset exposed by the **mediapipe** library, which already contains all of those connections. If we don't pass this list, then only the landmark points will be drawn on the image.
- Since each hand is composed by a set of well-known points (ex: the wrist, the tip of the thumb, etc...), the **hands** module has this enumerated value, called **HandLandmark**,

containing the 21 hand landmarks indexes. You can check a visual representation of these landmarks as points here [fig no: 1.2].

- When looking into the list of landmarks for a given hand, these indexes are always respected so we know where to look for to find a specific part of the hand (ex: if we know that we have thumbs up and down hand images, we can possibly want to look to the position of the tip of the thumb in comparison to the wrist and identify if it is a thumbs up or down, which explains why it is important to know what each landmark refers to).

- **STATIC IMAGE MODE**

If set to false, the solution treats the input images as a video stream. It will try to detect hands in the first input images, and upon a successful detection further localizes the hand landmarks. In subsequent images, once all max_num_hands hands are detected and the corresponding hand landmarks are localized, it simply tracks those landmarks without invoking detection until it loses track of any of the hands. This reduces latency and is ideal for processing video frames. If set to true, hand detection runs on every input image, ideal for processing a batch of static, possibly unrelated, images. Default to false.

- **MAX NUM HANDS**

Maximum no.of hands to detect. Default to 2.

- **MIN DETECTION CONFIDENCE**

Minimum confidence value ([0.0, 1.0]) from the hand detection model for the detection to be considered successful. Default to 0.5.

- **MIN_TRACKING_CONFIDENCE:**

Minimum confidence value ([0.0, 1.0]) from the landmark-tracking model for the hand landmarks to be considered tracked successfully, or otherwise hand detection will be invoked automatically on the next input image. Setting it to a higher value can increase the durability of the solution, the higher the delay. Ignored if static_image_mode is true, then hand detection only applies to all images. Default to 0.5.

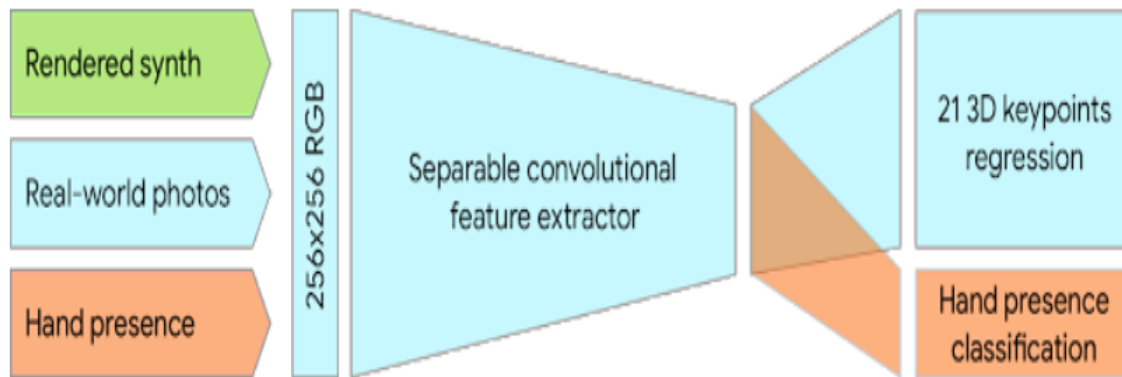
- **MULTI_HAND_LANDMARKS**

Collection of detected/tracked hands, where each hand is represented as a list of 21 hand landmarks, and each landmark is composed of x, y, and z. x and y are normalized to [0.0, 1.0] by the image width and height respectively. z represents the landmark depth with the depth at the wrist being the origin, and the smaller the value the closer the landmark is to the camera. The magnitude of z uses roughly the same scale as x.

➤ MULTI_HANDEDNESS

Collection of handouts received / tracked (e.g. left or right). Each hand is made up of a label and points. The label is a number of values either "Left" or "Right". Points are the estimated probability of a predicted offer and remain large or equal to 0.5 (and the opposite offer has a probability of approximately 1 - Points).

Note that the offer is determined by the photo shoot in the mirror, i.e., taken with a forward / selfie camera with the photos wrapped up. If not, please change the offer offer in the app.



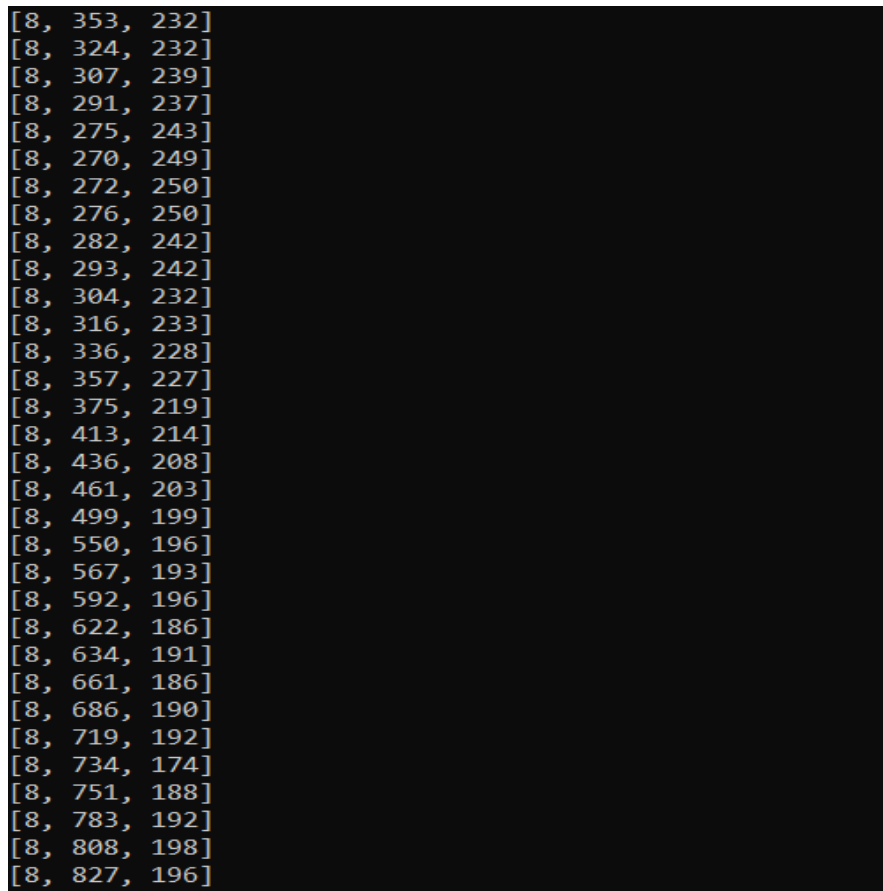
Mixed training schema for hand tracking network. Cropped real-world photos and rendered synthetic images are used as input to predict 21 3D keypoints.

Fig No: 4.7

However, fully customized data does not do well in the wild. To overcome this problem, we use an integrated mixing system. A high-level model training diagram is shown in the following figure 4.6.

After discovering the palm detection over the entire image, our landmark hand model performs a precise local line-making of 21 links within the detached regions. The model learns the internal representation of the inner hand shape and is powerful even in the visible hands in part and in its own way. The model has three effects (see Figure 4.6):

1. 21 earth symbols containing x, y, and related depths.
2. A hand flag indicating the potential for an image in the input image.
3. Binary distribution of offers, e.g. left or right hand.



```

[8, 353, 232]
[8, 324, 232]
[8, 307, 239]
[8, 291, 237]
[8, 275, 243]
[8, 270, 249]
[8, 272, 250]
[8, 276, 250]
[8, 282, 242]
[8, 293, 242]
[8, 304, 232]
[8, 316, 233]
[8, 336, 228]
[8, 357, 227]
[8, 375, 219]
[8, 413, 214]
[8, 436, 208]
[8, 461, 203]
[8, 499, 199]
[8, 550, 196]
[8, 567, 193]
[8, 592, 196]
[8, 622, 186]
[8, 634, 191]
[8, 661, 186]
[8, 686, 190]
[8, 719, 192]
[8, 734, 174]
[8, 751, 188]
[8, 783, 192]
[8, 808, 198]
[8, 827, 196]

```

After done with hand landmarks, while moving the palm throughout the webcam. Here we focused only “INDEX_FINGER_TIP” {8th keypoint} . Have to collect only 8th keypoint values as shown in above figure. We collect the values in the list, the indexes are explained

- [0] – it indicates the keypoint
- [1] – it gives the X –axis value while moving the palm.
- [2] – it gives the Y- axis value while moving the palm.

In the list only x-axis value used to control the systems volume. Here while moving the palm throughout the webcam will get the x-axis values and by using the pycaw library has full of access to control the system volume, here will map the our x-axis values and the system volume values.

4.3 Datasets [10]

- **In-the-wild dataset:** This dataset contains 6K images of large variety, e.g. geographical diversity, different lighting conditions, and hand appearance. The limit of this database is that it does not contain complex hand gestures.
- **In-house collected gesture dataset:** This dataset contains 10K images that include different angles for all physical actions. This data limit is limited to 30 people with limited background variations. The wild and indoor databases are great resources for each other to improve resilience.
- **Synthetic dataset:** The best possible hand cover and provide extra depth in depth, we provide a high-quality hand model in a variety of domains and tag it in 3D-compatible links. We use a 3D commercial model tied with 24 bones and incorporates 36 blendshapes, which control the fingers and palm size. The model also offers 5 make-up with different skin tones. The table below summarizes the accuracy of retrieval based on training data details. You are using both performance and actual data for key performance enhancements.

| Dataset | Mean regression error normalized by palm size |
|------------------------------|--|
| Only real-world | 16.1 % |
| Only rendered synthetic | 25.7 % |
| Mixed real-world + synthetic | 13.4 % |

Fig no: 4.8 Results of different Datasets

In the landmarks model, our experiments show that the integration of real-world data with performance yields the best results. See Figure 4.7 for details. We only look at images of the real world. In addition to quality improvements, training and large-scale archives lead to smaller jitters in appearance across all frames. This view leads us to believe that our real world database can be enriched for better performance. For the palm detector, we use only wild data, which is enough to make a handmade and give a very high look. However, all data sets are used to train the manual model. We describe real world images with 21 world symbols and use real 3D joints of the proposed artificial world images. With the presence of a hand, we select the bottom set of real-world images as good examples and sample in the region with the exception of hand-defined areas as negative examples. Given, we describe the below set of images of the real world with the offer to provide such data.

4.4 Software Details:

- Python
- Mediapipe framework
- Paycaw library

4.5 Hardware Details:

- 2 GB RAM
- 4 GB Internal storage memory

Installation Requirements

- Python
- OpenCV python
- MediaPipe framework
- Pycaw
- screen_brightness_control
- NumPy

Procedure:

step 1: Install the above libraries and frameworks.

step 2: After successfully installing, open any python IDE or Command-Line.

step 3: Open Command-Line, change the directory to the project folder, and then run the VC.py file.

Note: Make sure all files should be in the same folder.

step 4: After successfully done with execution, then it's open the window that accesses the webcam.

step 5: By moving the Palm (Hand) throughout the webcam, can control the system's volume.

while moving the Index finger of the Palm from right to left throughout the screen then it controls the volume in an increase to decrease order.

while moving the index finger of the Palm from left to right throughout the screen then it controls the volume in a decrease to increase order.

step 6: As we implement the controlling of screen brightness also,

Here while moving palm throughout the screen in the x-axis then it controls the system's Volume.

Here while moving palm throughout the screen in the y-axis then it controls the system's Brightness.

step 7: Here we control the system volume and brightness through the index finger of the palm by using the system webcam.

4.6 Testing

Software testing is an examination led to give partners data about the nature of the product item or administration under test. Software testing can likewise give a target, autonomous perspective on the product to permit the business to acknowledge and comprehend the dangers of programming execution. Test strategies incorporate the way toward executing a program or application with the plan of discovering programming bugs (mistakes or different deformities) and checking that the product item is fit for use.

Programming testing includes the execution of a product part or framework segment to assess at least one property of intrigue. When all is said in done, these properties demonstrate the degree to which the segment or framework under test:

- meets the prerequisites that guided its structure and advancement,

- reacts effectively to a wide range of sources of info,
- plays out its capacities inside an adequate time,
- it is adequately usable,
- can be introduced and run in its expected surroundings, and
- Accomplishes the general outcome its partners want.

As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As a result, software testing typically (but not exclusively) attempts to execute a program or application with the intent of finding failures due to software faults. The job of testing is an iterative process as when one fault is fixed, it can illuminate other failures due to deeper faults, or can even create new ones.

Software testing may provide accurate, independent information about the quality of the software and the risk of its failure to users or sponsors.

Software testing can be done as soon as portable (or less complete) software is available. The complete software development method usually determines when and how the test is performed. For example, in a phased process, most tests take place after the system requirements have been defined and then performed on the pilot programs. Conversely, under an agile approach, requirements, plans, and tests are usually performed simultaneously.

Software testing methods are traditionally divided into white tests and black boxes. These two methods are used to describe the point of view an inspector takes when designing test cases. The hybrid method called gray box test can also be used in the software test method. In the sense of gray box testing - starting tests from certain design elements - it gets prominent, this "extreme difference" between black and white box testing is somewhat blurred.

1.White-box testing

White-box testing is a strategy for programming testing that tests the inner structures or operations of an application, instead of its usefulness (for example discovery testing). In white-box testing, an inner perspective of the framework, just as programming abilities, is utilized for

configuration experiments. The analyzer picks contributions to practice ways through the code and decides the normal yields.

The white box test (also known as clear box test, glass box test, clear box test, and structural test) confirms the internal structure or function of the system, in contrast to the performance expressed by the end user. In the white box test, the internal concept of the system (source code), as well as editing skills, are used to design test cases. The tester selects the input to use the methods with the code and determines the appropriate results. This is similar to regional spatial testing, e.g., in-circuit (ICT) testing.

While white box testing can be applied to unit, integration, and software program process standards, it is usually done at unit level. Methods can be tested within unit, inter-unit methods during integration, and between sub-systems during system-level testing. While this test build method may find many errors or problems, it may not find the missing parts for specifications or missing requirements.

2.Black box testing

This is designed to uncover the error in functional requirements without regard to the internal working of the project. This testing focuses on the information domain of the project, deriving the test case by partitioning the input and output domain of programming – A manner that provides thorough test coverage.

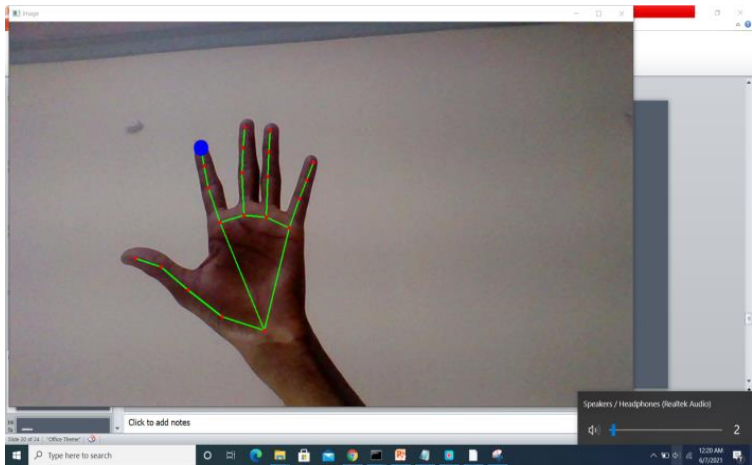
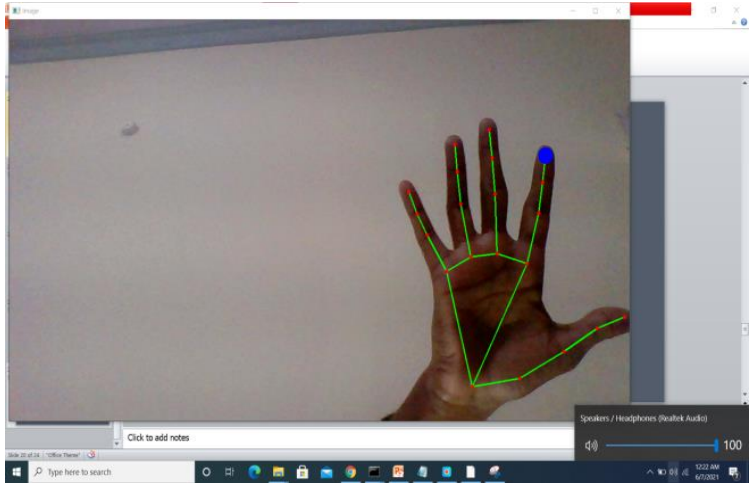
A black box test (also known as a performance test) treats software like a "black box," to test performance without internal use information, without seeing the source code. Testers only know what the software should do, not how it does it. Methods of checking black boxes include: equity classification, boundary analysis, double check, state switch tables, decision table test, fuzz test, model-based test, case test, test test, and specification-based testing.

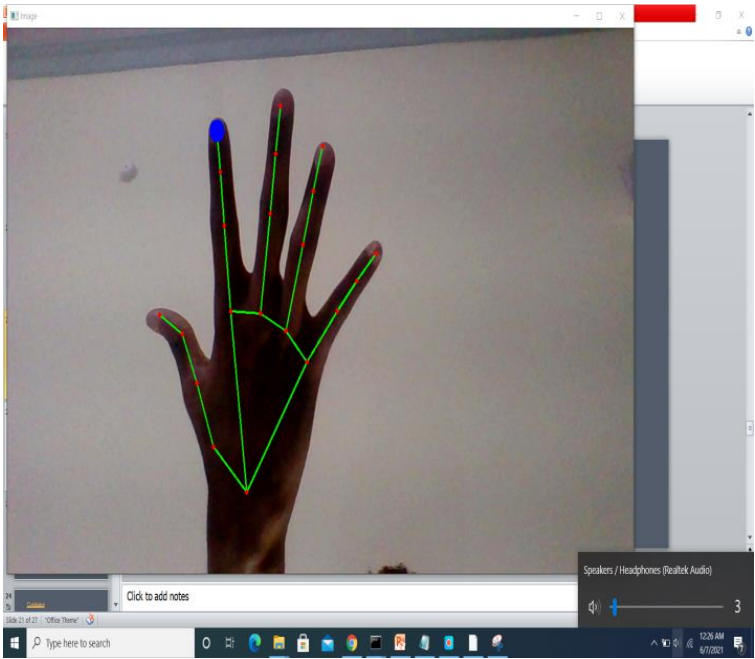
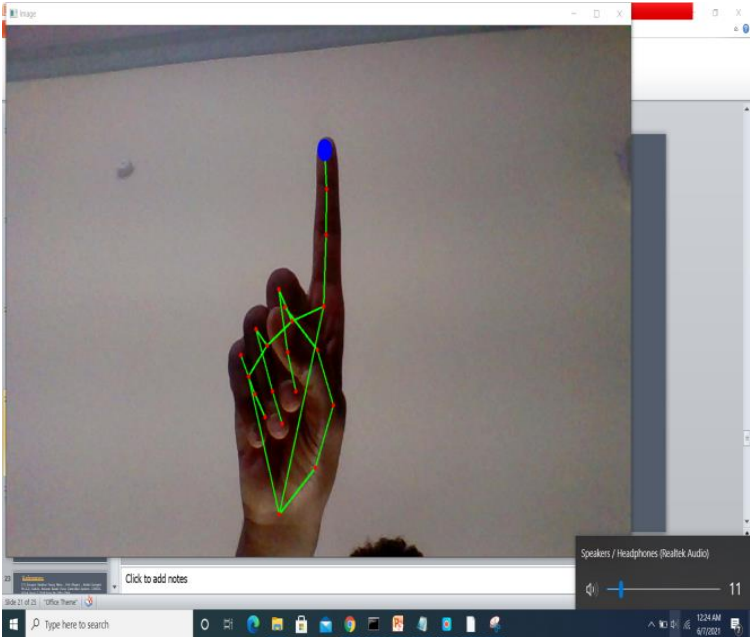
It uses external software definitions, including details, requirements, and configurations to detect test cases. This test may or may not work, or it may often work

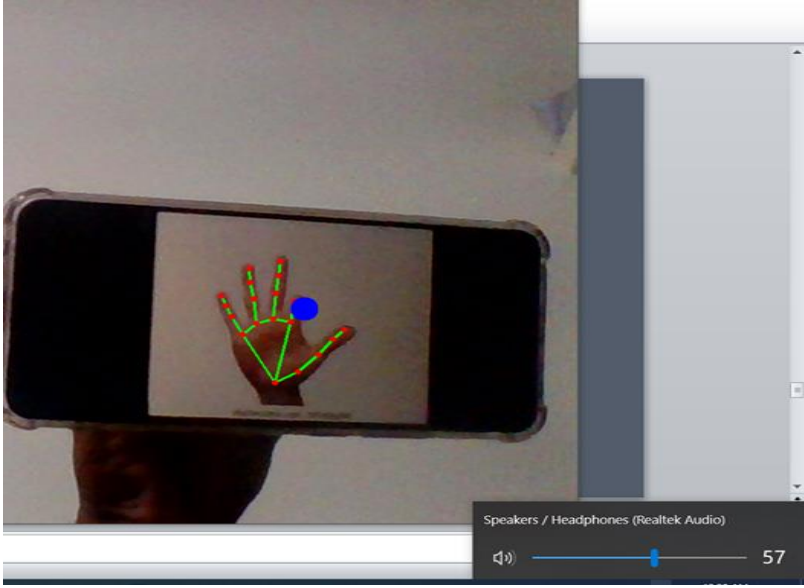
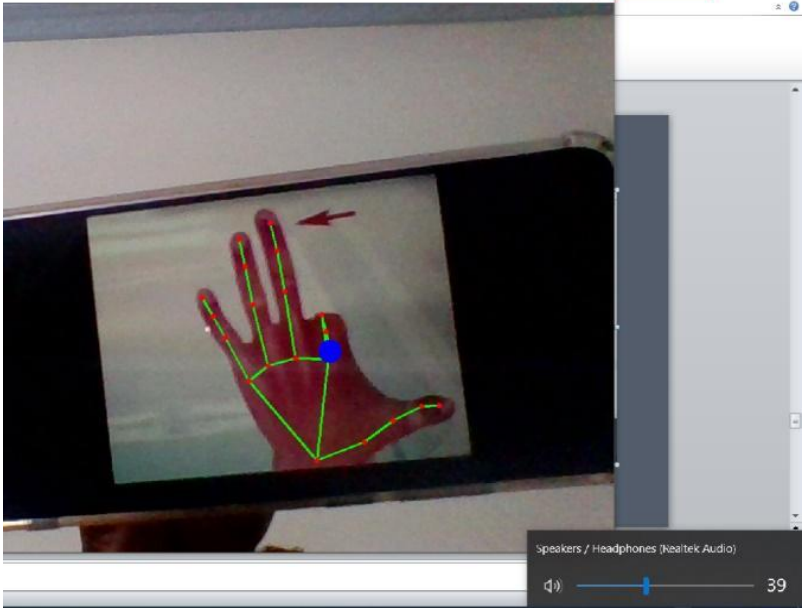
Another advantage of the black box method is that no system information is required. Whatever bias the program organizers may have, the inspector may have a different set and may emphasize different areas of operation. On the other hand, black box tests have been described as "like

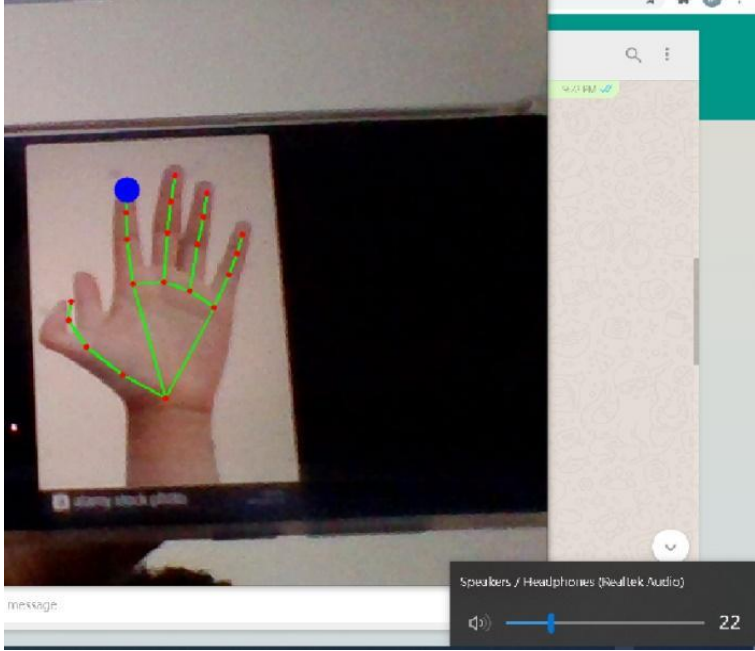
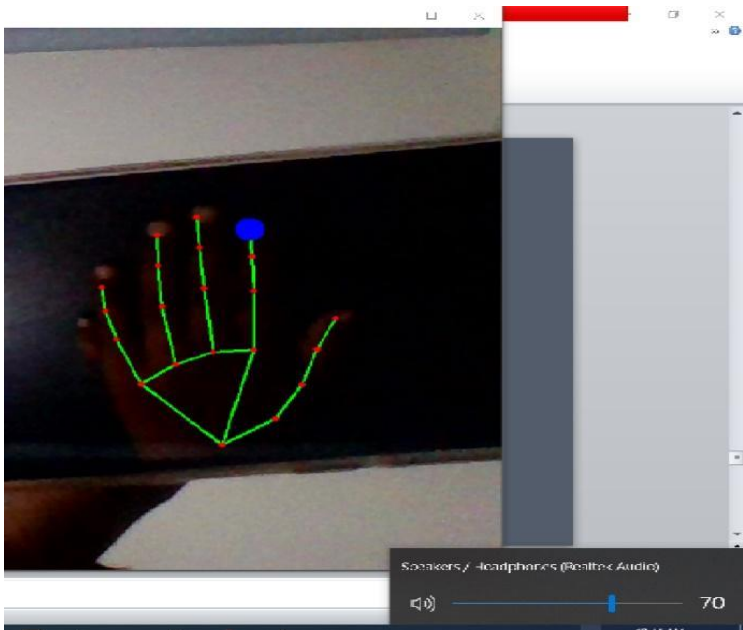
walking on a black labyrinth without a flashlight." Because they do not test the source code, there are cases where the tester writes multiple test cases to look at something that could only be tested in one case or leave other parts of the system unchecked.

4.6.1 Test Cases

| S.no | Figures | Description | Test Cases Status |
|------|---|--|-------------------|
| 1 |  <p>Fig No:4.9 Right Plam</p> | As we can see that, the right hand index finger is tracked and the volume is adjusted. | PASSED |
| 2 |  <p>Fig No:4.10 Left Palm</p> | Here the left hand index finger is tracked and volume is adjusted accordingly. | PASSED |

| | | | |
|---|---|---|---------------|
| 3 |  <p>Fig No: 4.11 Back side Palm</p> | <p>Here we have shown back side of the palm and still it tracks the index finger and volume is adjusted accordingly.</p> | <p>PASSED</p> |
| 4 |  <p>Fig No:4.12 Index finger</p> | <p>Here we have shown only the index figure and remaining fingers are closed but still it tracks the index finger and volume is adjusted accordingly.</p> | <p>PASSED</p> |

| | | | |
|---|--|---|---|
| 5 |  <p data-bbox="440 852 883 888">Fig No: 4.13 Half-cut index finger</p> | <p data-bbox="1084 243 1268 716">Here we have shown half cut index finger (small index figure) to the camera but still our model tracks the index finger and volume is adjusted accordingly</p> | <p data-bbox="1287 243 1422 275">PASSED</p> |
| 6 |  <p data-bbox="467 1671 854 1707">Fig No: 4.14 No Index Finger</p> | <p data-bbox="1084 1020 1268 1566">We have given the image of palm with no index finger but still our model tracks the index finger by Assuming the tip as the index finger and volume is adjusted accordingly.</p> | <p data-bbox="1287 1020 1422 1052">PASSED</p> |

| | | | |
|---|--|--|---------------|
| 7 |  <p>Fig No: 4.15 Six finger Palm</p> | <p>Here we have shown six finger palm as an input and our model recognize the index finger correctly and volume is adjusted accordingly.</p> | <p>PASSED</p> |
| 8 |  <p>Fig No: 4.16 six finger with dark mode</p> | <p>Here we have shown the image of six finger palm in dark/night mode to our model but still it recognizes and tracks the index figure and volume is adjusted accordingly.</p> | <p>PASSED</p> |

CHAPTER-5

CONCLUSION AND FUTURE SCOPE

5.1 Conclusion:

- VOLUME ADJUSTMENT is an easy way to control the system's volume through fingers without touching the mouse and keyboard. This application is helpful for those who are afraid to use the system's { Novice users }.
- For better human-computer interaction (HCI) we represent a novel approach, where we are using the hand gesture recognition using the real-time camera to overcome the issues faced by the novice users and to make the whole system low-cost and more user-friendly. By using a camera and computer vision technology, such as the image segmentation and gesture recognition method to control the system's volume.
- Here we observed a real-time problem faced by novice users while using computers and we resolved one of the issues i.e. volume controlling

5.2 Scope

- As in the project implemented only one issue i.e., VOLUME ADJUSTMENT. We tried to implement the controlling system's brightness.
- Furthermore, similar technologies could be applied to create various applications like full control of the mouse tasks such as left and right-clicking, double-clicking, and scrolling. In the future, we plan to add more features such as enlarging and shrinking windows, closing windows, etc. This technology has wide applications in the fields of augmented reality, computer graphics, computer gaming, etc.
- Additional hardware is required to most of the applications which is often very costly. Our motive was to create this technology in the cheapest possible way and also to create it under a standardized operating system. Various application programs can be written exclusively for this technology to create a wide range of applications with the minimum requirement of resources.

CHAPTER-6

APPENDIX I

Note: keep both python files in the same folder.

HandTrackingModule.py

```
import cv2
import mediapipe as mp
import time

class handDetector():
    def __init__(self, mode=False, maxHands= 1, 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), 15, (255, 0, 255), cv2.FILLED)

        return lmList

def main():
    pTime = 0
    cTime = 0
    cap = cv2.VideoCapture(1)
    detector = handDetector()
    while True:
        success, img = cap.read()
        img = detector.findHands(img)
        lmList = detector.findPosition(img)
        if len(lmList) != 0:
            print(lmList[4])

        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()

```

VolumeAdjustment.py

```

import cv2

```

```

import time
import HandTrackingModule as ht
import numpy as np
import math
from ctypes import cast, POINTER
from comtypes import CLSCTX_ALL
from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume
import screen_brightness_control as bc

#640,1920
#480,1080
Width = 1920
height = 1080

Cap = cv2.VideoCapture(0,cv2.CAP_DSHOW) #, cv2.CAP_DSHOW

cap.set(3, width ) #setting the width of the window
cap.set(4, height) #setting the Height of the window
cap.set(10, 100) #setting the image clarity

detector = ht.handDetector(detectionCon=0.7)
devices = AudioUtilities.GetSpeakers()
interface = devices.Activate(
    IAudioEndpointVolume._iid_, CLSCTX_ALL, None)
volume = cast(interface, POINTER(IAudioEndpointVolume))
# volume.GetMute()
v1 = volume.GetMasterVolumeLevel()
volRange = volume.GetVolumeRange()
minVol = volRange[0]
maxVol = volRange[1]
vol = 0
brightness = 0
while True:
    success, img = cap.read()
    img = detector.findHands(img)
    #####
    lmList = detector.findPosition(img, draw=False)
    if len(lmList) != 0:
        print(lmList[8])
        x2, y2 = lmList[8][1], lmList[8][2]

```

```

cv2.circle(img, (x2, y2), 15, (255, 0, 0), cv2.FILLED)
vol = np.interp(x2, [200, 1200], [maxVol, minVol])
volume. SetMaster VolumeLevel(vol, None)
brightness = np.interp(y2, [35, 350], [0, 100])
bc.set_brightness(int(brightness))
###
cv2.imshow("Image", img)
if cv2.waitKey(1) == ord('e'):
    break

```

Output:

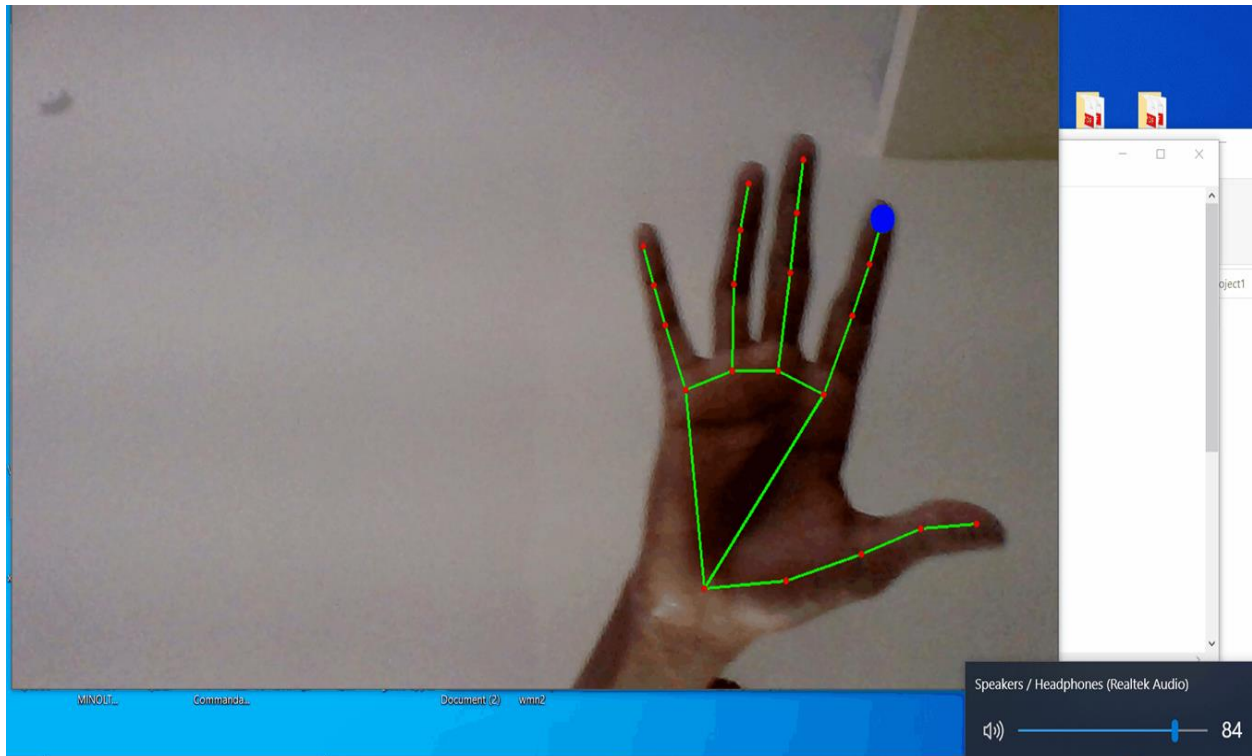


Fig no: 6.1 Volume Controlled

APPENDIX II

EXECUTION SCREENSHOTS

Module1:

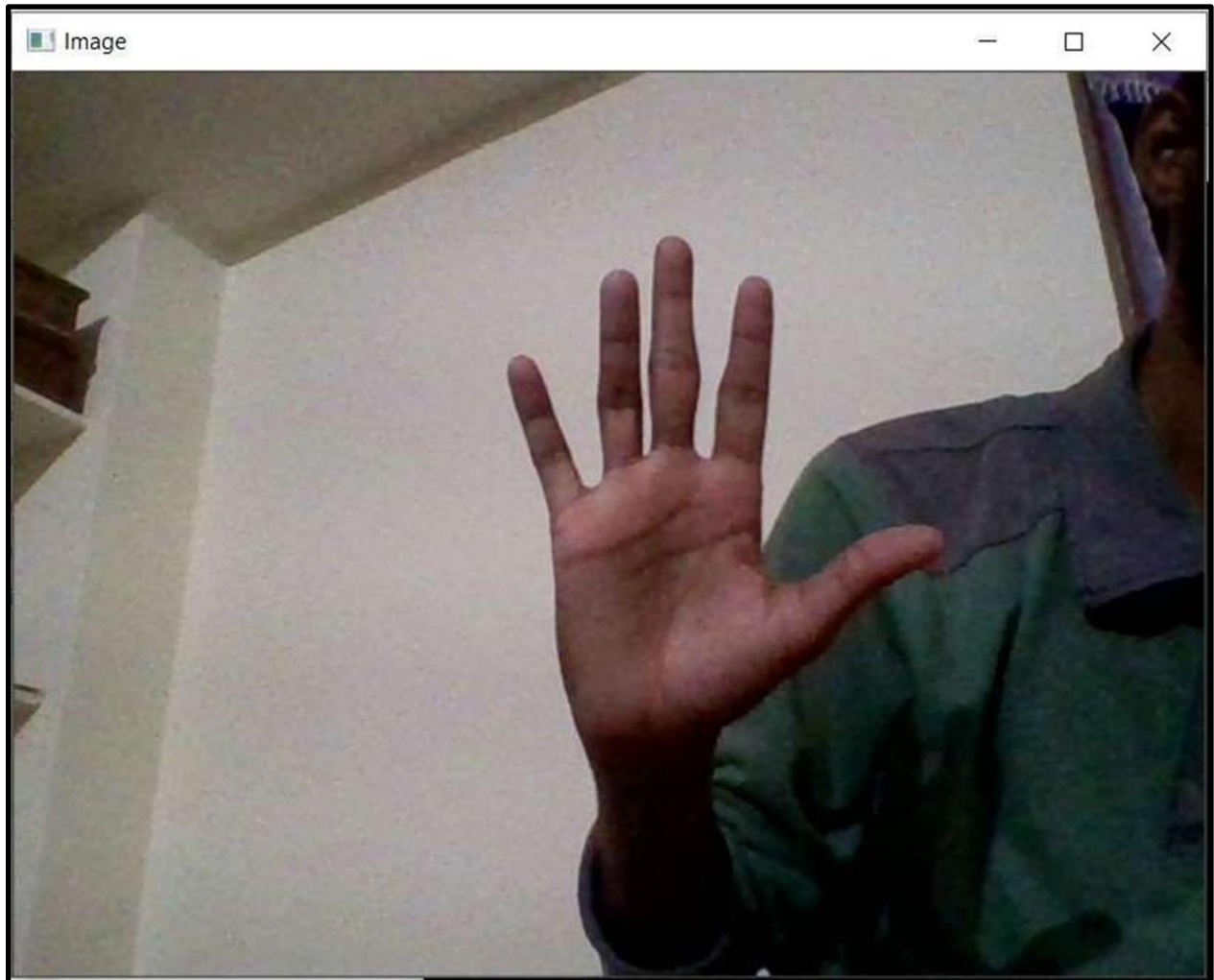


Fig no:6.2 Capturing objects from Webcam

- In module1, we are accessing System's webcam to capture the objects with help of the OpenCV library and send them to module2.

Module2 :

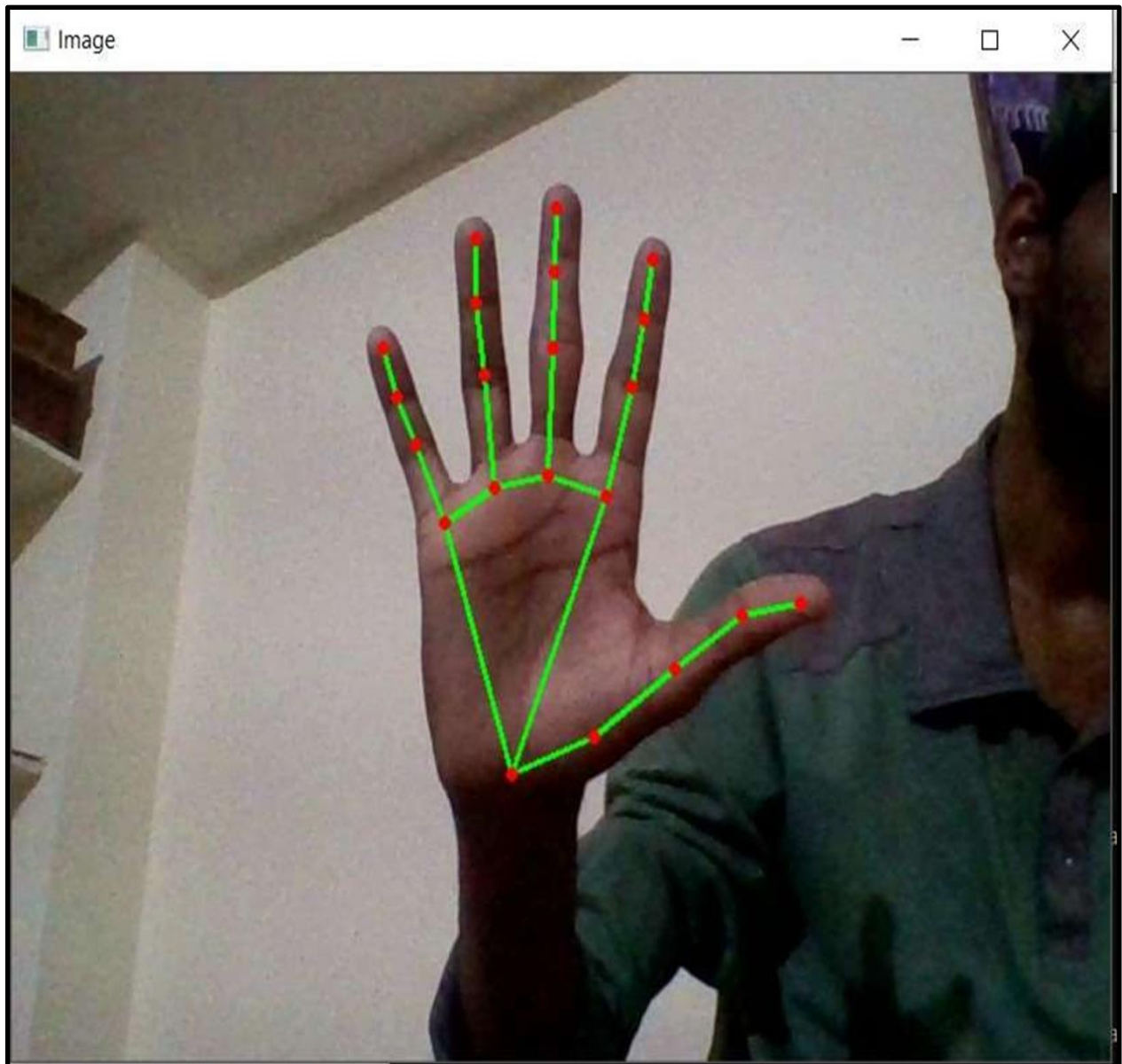
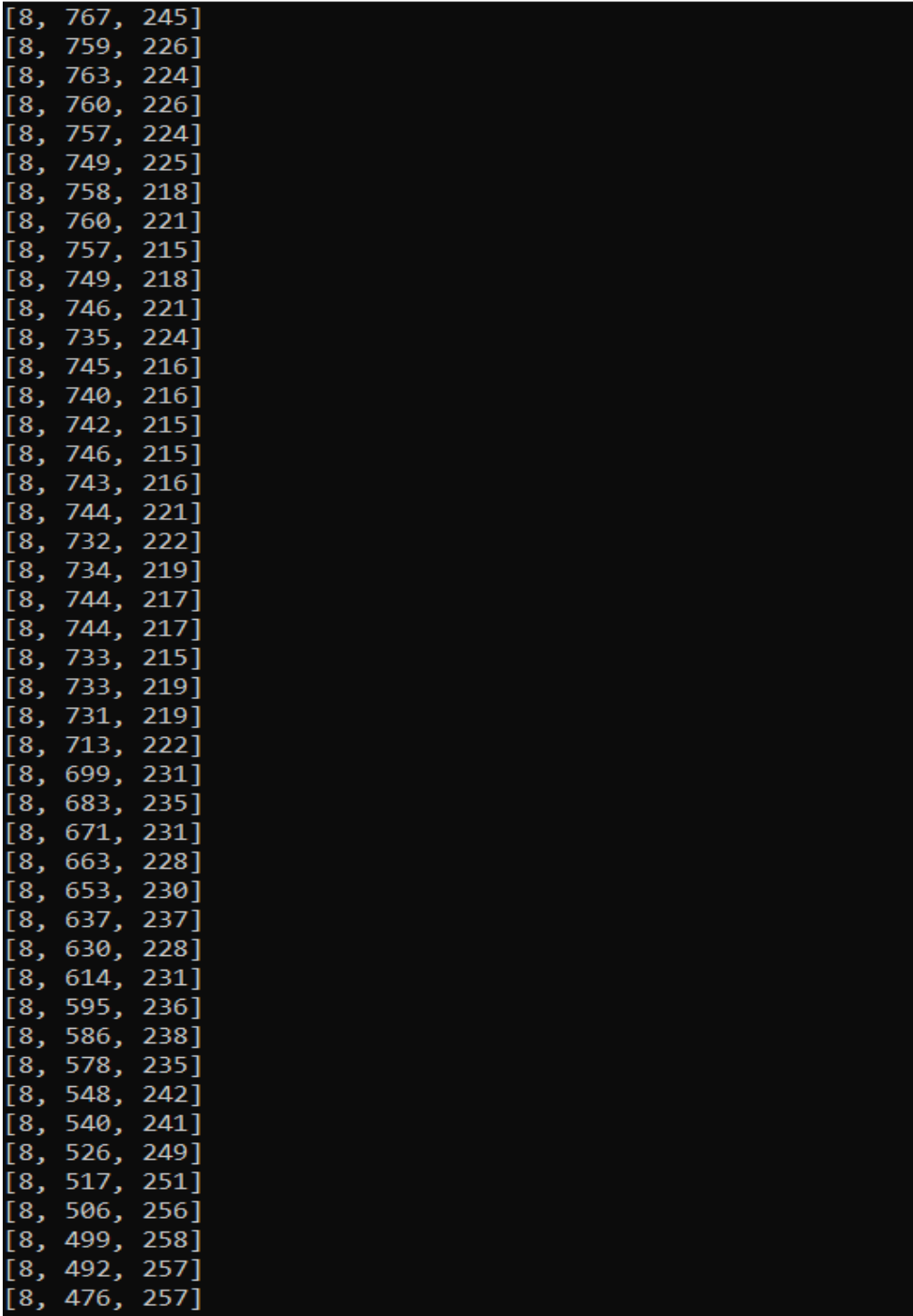


Fig no: 6.3 Palm and Hand Landmark detectors

- Module2, after capturing the objects, has to remove other objects except for the palm and by mideapipe framework helps to detect the palm and also points the 21 key points of the palm.



[8, 767, 245]
[8, 759, 226]
[8, 763, 224]
[8, 760, 226]
[8, 757, 224]
[8, 749, 225]
[8, 758, 218]
[8, 760, 221]
[8, 757, 215]
[8, 749, 218]
[8, 746, 221]
[8, 735, 224]
[8, 745, 216]
[8, 740, 216]
[8, 742, 215]
[8, 746, 215]
[8, 743, 216]
[8, 744, 221]
[8, 732, 222]
[8, 734, 219]
[8, 744, 217]
[8, 744, 217]
[8, 733, 215]
[8, 733, 219]
[8, 731, 219]
[8, 713, 222]
[8, 699, 231]
[8, 683, 235]
[8, 671, 231]
[8, 663, 228]
[8, 653, 230]
[8, 637, 237]
[8, 630, 228]
[8, 614, 231]
[8, 595, 236]
[8, 586, 238]
[8, 578, 235]
[8, 548, 242]
[8, 540, 241]
[8, 526, 249]
[8, 517, 251]
[8, 506, 256]
[8, 499, 258]
[8, 492, 257]
[8, 476, 257]

Fig No: 6.4 INDEX_FINGER_TIP values

Module3 :

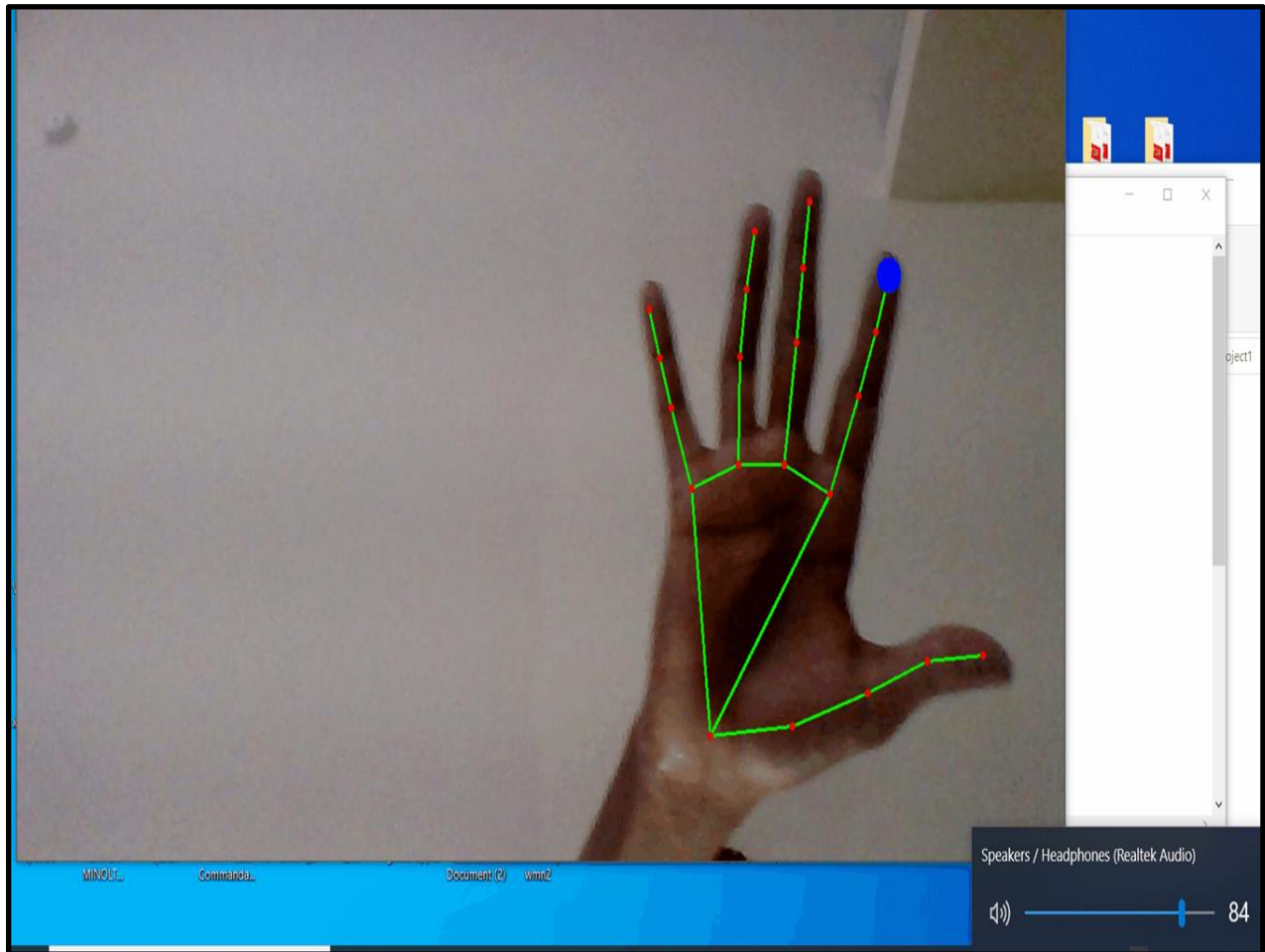


Fig no: 6.5 Volume controlled

- Module3 with the key points will control the system's volume with the pycaw library by moving the palm throughout the system's webcam.

References

- [1] Devyani Randive Parag Mane , Priti Khapre , Aniket Dange4 Mr.S.D. Kakde. Arduino Based Voice Controlled System. IJARIE, Vol-4 Issue-2 2018,Page No:2901-2904.
- [2] Keith walker,_A Remote Controlled Stereo Volume Control. PROJECT HUB.
- [3] Sheng-Yu Peng,Kanoksak Wattanachote,_Hwei-Jen Lin,_Kuan-Ching Li. A Real-Time Hand Gesture Recognition System for Daily Information Retrieval from Internet. IEEE, 5992061.
- [4] Zhou Ren, Junsong Yuan, Jingjing Meng, Zhengyou Zhang. Hand Gesture Recognition Using Kinect Sensor. Research Gate, IEEE. Page No: 1-11.
- [5] Fabio Dominio, Mauro Donadeo, Pietro Zanuttigh._Combining multiple depth-based descriptors for hand gesture recognition. University of Padova, Italy. Page No: 1-38.
- [6] GOOGLE LLC. MediaPipe Hands. GitHub,2020.
- [7] Valentin Bazarevsky and Fan Zhang. On-Device, Real-Time Hand Tracking with MediaPipe. Google AI blogs, Monday, August 19, 2019.
- [8]Sugandha Lahoti. Google open sources an on-device, real-time hand gesture recognition algorithm built with MediaPipe. Packt Hub, August 21, 2019.
- [9] Nuno Santos. Python: Hand landmark estimation with MediaPipe. techtutorialsx.
- [10] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, Matthias Grundmann. MediaPipe Hands: On-device Real-time Hand Tracking. Arxiv:2006.10214v1, 18 Jun 202.

[11] Wei Liu , Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu , Alexander C. Berg. SSD: Single Shot MultiBox Detector. Arxiv:1512.02325v5, 29 DEC 2016.

[12] Sk. Abdul Soniya, R.V. Harshitha, Y. Veera Reddy, Dr. Ratna Raju Mukiri, A Novel Approach: Hand Gesture Volume Control System, International Journal of Research, Page. No:200-206, Volume7,July/2018.

[13] A. Dhawan and V. Honrao, “Implementation of Hand Detection based Techniques for Human Computer Interaction,” Int. J. Comput. Appl., vol. 72, no. 17, pp. 6–13, 2013.