Volume and Brightness Control Using Hand Gestures

[1]D Sravani, [2]Nidjintha Trisha Reddy, [3]Kasturi Swamy, [4]Rishabh Singh

[1]Assistant Professor, Dept.of CSE, Teegala Krishna Reddy Engineering College, Meerpet, Hyderabad,

Damanapeta@gmail.com

[2]BTech student, Dept.of CSE, Teegala Krishna Reddy Engineering College, Meerpet, Hyderabad,
nidjinthatrisha@gmail.com

[3]BTech student, Dept.of CSE, Teegala Krishna Reddy Engineering College, Meerpet, Hyderabad,
kasturiswamy68@gmail.com

[4]BTech student, Dept.of CSE, Teegala Krishna Reddy Engineering College, Meerpet, Hyderabad,
er.singhrishabh@gmail.com

Abstract: *We are developing a volume and brightness controller in which we are using hand gestures as the input to control the system, Opencv module is basically used in this implementation to control the gesture. Hand gesture recognition system has developed excessively in the recent years, reason being its ability to cooperate with machine successfully. Gestures are considered as the most natural way for communication among human and PCs in virtual framework. We often use hand gestures to convey something as it is non-verbal communication which is free of expression. In our system, we used background subtraction to extract hand region. In this application, our PC's camera records a live video, from which a preview is taken with the assistance of its functionalities or activities. This system basically uses the web camera to record or capture the images /videos and accordingly on the basis of the input, the volume and brightness of the system is controlled by this application. The main function is to increase and decrease the volume and brightness of the system. The project is implemented using Python, OpenCV.*

Keywords: *Human Computer Interaction, Structuring Elements, Hand gesture, Region of Interest.*

I. INTRODUCTION

Hand gestures are unprompted and also robust transmission mode for Human Computer Interaction (HCI). Keyboard, mouse, joystick or touch screen are some input devices for connection with the computer but they don't provide appropriate interface whereas, the current system will contain either desktop or laptop interface in which hand gesture can be done by wearing data gloves or web camera used for snapping hand image. The first step towards this gesture recognition is hand capturing and analyzing. Sensors are used in Data-Glove methods for initializing fingers movement and other sensor will program hand movements. In comparison the vision-based method only needs a camera and hence identifying the actual interaction between human and computer without using any other devices. The challenges of this system are constant background, sometimes person and lighting also. Different procedure and algorithms which are used in this system are elaborated here along with the recognition techniques. The method of searching a connecting region in the picture with particular specification, being it color or intensity, where a pattern and algorithm is adjustable is known as segmentation. Vision Based method requires a web camera, so that one can realize natural interaction between humans and computer without using any other devices. The challenging part in these systems is background images or videos which is recorded or captured during taking the inputs i.e. hand gesture by the user, also sometime lightning effect the quality of the input taken which creates the problem in recognizing the gestures. Process to find a connected region within the image with some of the property such as color ,intensity and a relationship between pixels i.e. pattern is termed as segmentation. And have used some important packages which have OpenCv-python, tensorflow, numpy, mediapipe, imutils, scipy, numpy.

Detecting and locating the hand in real-time footage from the webcam is the initial stage in any hand processing system. Because of the variety in position, orientation, placement and scale, detecting a hand might be difficult. Variability is also aided by the varying levels of light in the room. Hand gesture recognition often requires numerous levels of processing, including image acquisition, pre-processing, feature extraction, and gesture identification. Using a webcam, image acquisition entails recording an image in a video frame by frame. The collected images are subjected to colour filtering, smoothing as part of the image pre-processing. Feature extraction is a technique for extracting features from a hand image, such as hand outlines, whereas gesture recognition is a technique for extracting features and recognizing hand gestures. Designing a hand gesture recognition system is a difficult task that comprises two major issues. The first is the detection of a person's hand. A webcam is used to detect the user's hand in real-time video in this acquisition some issue arises such as inconsistencies in brightness, noise, resolution, and contrast.

To identify the gestures, the technique involves segmentation and edge detection and with help of the openCV module we obtain the hand gesture and able to control the volume.

strong efforts have been carried out to develop intelligent and natural interfaces between users and computer-based systems based on human gestures. Gestures provide an intuitive interface to both humans and computers. Thus, such gesture-based interfaces can not only substitute the common interface devices but can also be exploited to extend their functionality. A robot is usually an electro-mechanical machine that can perform tasks automatically. Some robots require some degree of guidance, which may be done using a remote control or with a computer interface. Robots have evolved so much and are capable of mimicking humans that they seem to have a mind of their own. An important aspect of a successful robotic system is Human-Machine interaction. In the early years, the only way to communicate with a robot was to program which required extensive hard work. With the development in science and robotics, Gesture Recognition came into life

Hand gestures is the powerful communication medium for Human Computer Interaction (HCI). Several input devices are available for interaction with computer, such as keyboard, mouse, joystick and touch screen, but these devices do not provide easier way to communicate. Hand gestures will provide an easier and friendly way to communication with computer.

II. LITERATURE SURVEY

Hand gesture detection and utilizing them to control certain set of devices operations and allowing interaction with computer system without the aid of mouse and keyboard. In this paper we draw along the same line but we attributed the use of Haar-cascade classifier to identify hand gesture. Some of the related works in this field are described briefly as follows A non-local algorithm for hand gestures was proposed by A. Buades, B. Coll, and J. Morel [1]. At the moment, finding finger movement algorithms remains a valid task. Functional analysis and statistics collide. Despite the fact that most recently presented approaches have a high level of sophistication, Algorithms have not yet reached a satisfactory degree of performance applicability. All work admirably when the model matches the algorithm assumptions, but they all fail in general, producing defects in analysing the pixels through the camera. The primary goal of this study is to define a generic mathematical and experimental technique for comparing and classifying conventional hand movement recognition algorithms.

Moktader Daiyan et al.[2] suggested a high performance decision based median filter. This technique detects noise pixels iteratively over numerous phases before replacing them with the median value. Noise detection is accomplished by enlarging the field of view. Mask till 77% to keep the extraction of local data going. Furthermore, if the algorithm fails to find a noise-free pixel at 7 7, the processing pixel is replaced by the last processed pixel. If the noise-free median value isn't available in the 7th processing window, the last processed pixel is used to determine if it is noise-free. The method chooses a window size if the last processed pixel is noisy. Calculate the number of 0s and 255s in the processing window using the 1515 dimension. Then, in the selected window, replace the processed pixel with 0 or 255, whichever is higher in number. Rajeshwari Kumar Dewangan et.al [3] accurate object information and obtain a location using a deep learning object recognition technique.Object recognition algorithms are designed based on theSingle Shot MultiBox Detector (SSD) structure, an object recognition deep learning model, to detect objects using a camera.

H. Jabnoun et, al [4] suggested the system that restores a central function of the visual system which is the identification of surrounding objects which is based on the local features extraction concept. Using SFIT algorithm and keypoints matching showed good accuracy for detecting objects.

Košale U, et al. [5] suggested that Detection of obstacles is performed by Time of Flight (ToF) sensors, whose ranging data is then processed with an on-board microcontroller and send via Bluetooth connection to the belt. The belt is equipped with a second microcontroller, which interprets the data and presents it to the wearer with 15 vibration motors arranged in a square grid. The glasses are worn on the head, whereas the belt is attached around the stomach area. But the number of sensors detecting the obstacle decreased with the distance. Circle and square were detected better than triangle. This suggests that different shapes trigger different responses of sensors on glasses.

A. Jaiswal et al. [6] proposed an approach that used user generated picture denoising. The remaining task is broken down into four steps. The first image is denoised using a filtering process, and the second image is denoised using a different method. Wavelet-based approaches are used to denoise pictures, filtering, third hard thresholding, and thresholding. Finally, the approach was applied to a noisy image concurrently. PSNR output results are calculated by comparing all cases, the MSE (mean square error) is obtained. On the basis of PSNR, MSE, and image visual quality, experiments are conducted on 512 X 512 noisy images with noise variance of 0.04, output of median filter, Wiener filter, hard thresholding, and hard thresholding with median filtering. When the filtering and wavelet thresholding techniques are combined, they produce excellent results.

III. PROPOSED WORK

As shown in figure 1, which indicates the proposed architecture of the system used in the volume controller. Here our input is hand gesture which is captured using web camera. Then the GUI (graphical user interface) helps to display the hand that conveys information and it processes the actions of the hand gestures of the user. By recognizing the gestures, the user is able to control the volume of the system which is final output

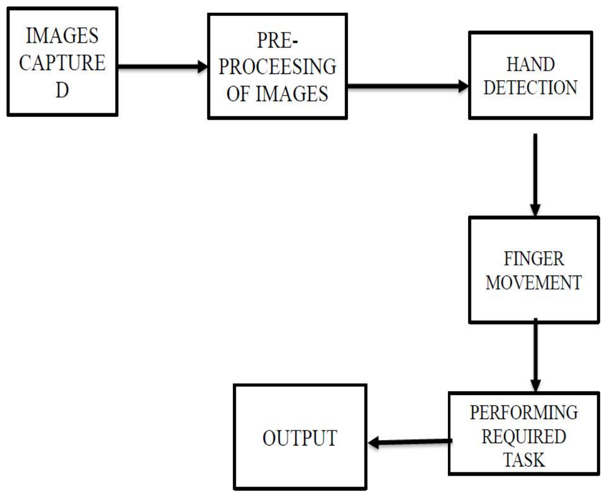


Fig.1 System architecture

Step 1: Start the Program

Step 2: Importing the Various Modules- Open Vision used for the AI recognition and various audio utilities which are of main concern.

Step 3: Capturing the Area of Interest by detecting the various contours and differentiating the white and balck region of the interest

Step 4: Executing the Loop to detect the various hand landmarks

Step 5: Getting the Hand Landmarks and verifying the distance between the index and thumb finger based on the algorithm given

Step 6: Display the frame giving the final values of the reading with complete decrease and increase in the volume using Artificial Intelligence. The Program is executed till the loop is iterated once it completes the iterations it comes out of the loop and program stops

Step 7: Stop

ALGORITHM

1.Conventional Neural Network

Step1: Convolutional operation

The first building block in our plan of attack is convolution operation. In this step, we will touch on feature detectors, which basically serve as the neural network's filters. We will also discuss feature maps, learning the parameters of such maps, how patterns are detected, the layers of detection, and how the findings are mapped out

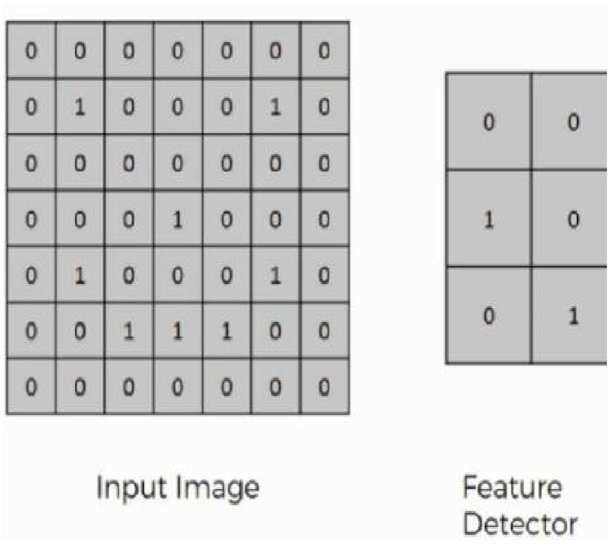
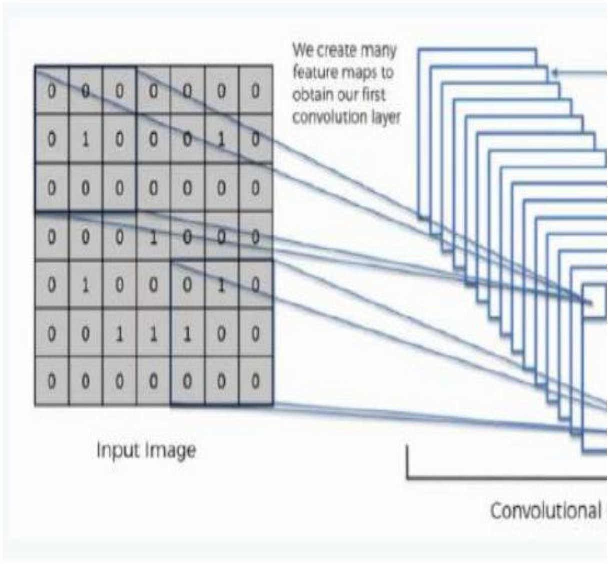


Fig.2 Convolution operation



We create many feature maps to obtain our first convolution layer
Input Image
Convolutional

Fig.3 Relu Layer

Step (1b): Relu Layer

The second part of this step will involve the Rectified Linear Unit or Relook. We will cover Relook layers and explore how linearity functions in the context of Convolutional Neural Networks. Not necessary for understanding CNNs, but there's no harm in a quick lesson to improve your skills.

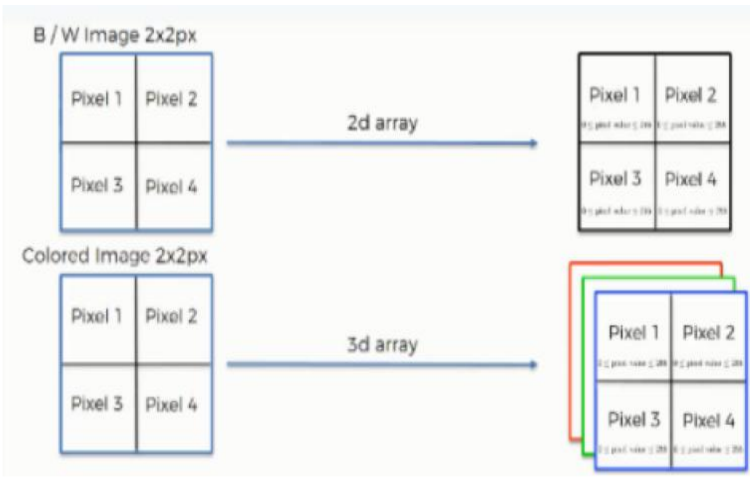


Fig.4 Pooling layer

In this part, we'll cover pooling and will get to understand exactly how it generally works. Our nexus here, however, will be a specific type of pooling; max pooling. We'll cover various approaches, though, including mean (or sum) pooling. This part will end with a demonstration made using a visual interactive tool that will definitely sort the whole concept out for you.

Step 3: Flattening

This will be a brief breakdown of the flattening process and how we move from

ISSN: 2366-1313

pooled to flattened layers when working with Convolutional Neural Networks.

Step 4: Full Connection

In this part, everything that we covered throughout the section will be merged together. By learning this, you'll get to envision a fuller picture of how Convolutional Neural Networks operate and how the "neurons" that are finally produced learn the classification of images.

Summary

In the end, we'll wrap everything up and give a quick recap of the concept covered in the section. If you feel like it will do you any benefit (and it probably will), you should check out the extra tutorial in which Softmax and Cross-Entropy are covered. It's not mandatory for the course, but you will likely come across these concepts when working with Convolutional Neural Networks and it will do you a lot of good to be familiar with them.

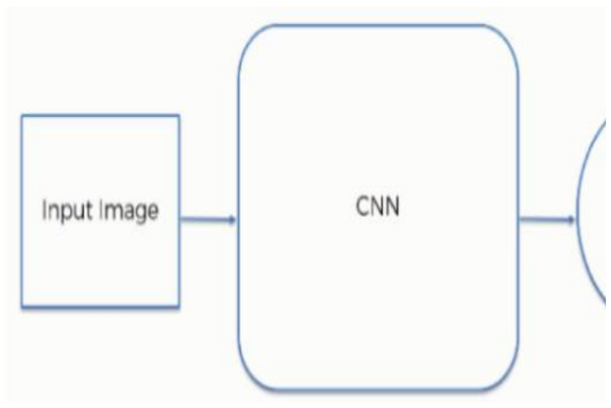


Fig.5 Convolutional Neural Networks results

IV. IMPLEMENTATION

Its implemented using the IDE software interpreter Pycharm it can also be implemented using the command prompt as well. Import the open CV Library to python project which is used as a computer vision tool and to read the image which is nothing but hand in this context. Then we have to use MediaPipe which is a cross-platform framework for building multimodal applied machine learning pipelines. It is used for detection purpose. hypot() method returns the Euclidean norm. The Euclidean norm is the distance from the origin to the coordinates given. Then to Get default audio device using PyCAW we have used comtypes which is a basic library and audio utilities. The video capture object to capture the information if the video cam is open. Then Media Pipe Hands is a high-fidelity hand and finger tracking solution. If the hands are detected then we have drawn the following outline of the hand using the audio utility function.

Obtain the default audio device using PyCAW.After which we have interfaced to the required volume and then found the range which is from 0 to 100, read the frames from the webcam and convert the image to RGB. If Fit List of lm or glm

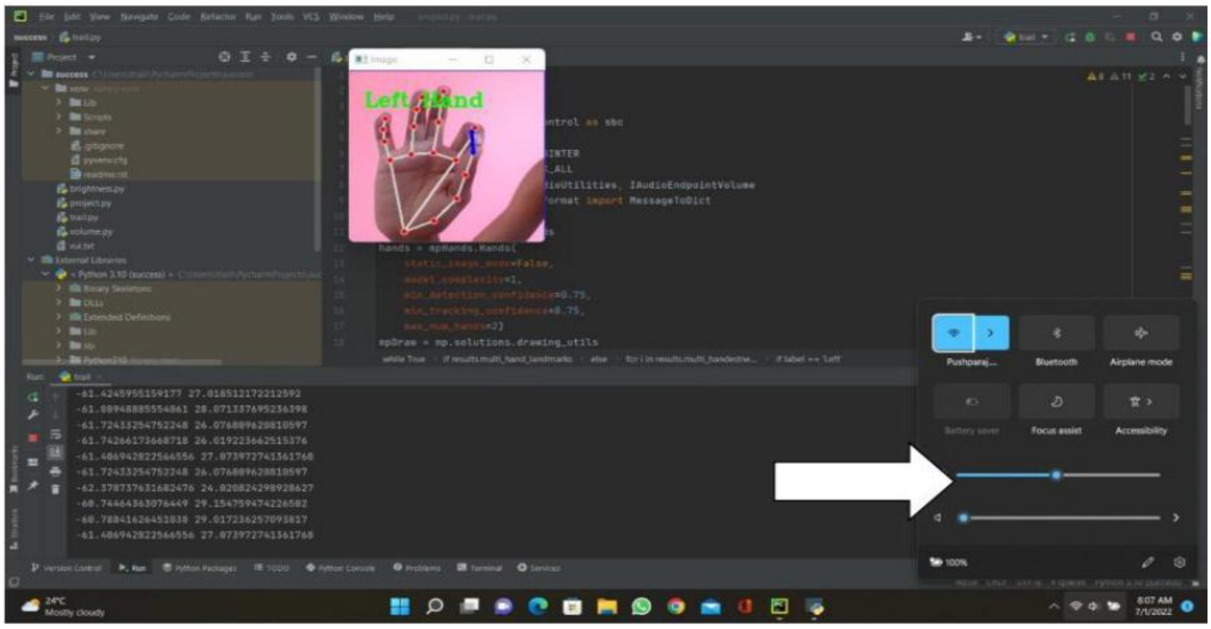
Objects with a Common Model is null then we detect hands in the frame with the help of "hands.process()" function. Once the hands get detected we will locate the key points and then we highlight the dots in the key points using cv2.circle, and connect the key points using mpDraw.draw_landmarks. The tip of the index and middle finger then we print (x1, y1, x2, y2). Then we check which fingers are up and we print the fingers. Convert Coordinates and then smoothen the values. Both Index and middle fingers are close then we reduce the volume and if the index and middle finger are away then we increase the volume. We then find the length of the line through the coordinates. Map the distance of thumb tip and index finger tip with volume range. For our case, the distance between thumb tip and index finger tip was within the range of 15 – 220 and the volume range was from -63.5 – 0.0. Suppose in this case we have taken -8.0 to 194.83366 as maximum volume and similarly the reading for various volumes is taken.

OpenCV:

OpenCV is a library of programming capacities mostly focused on ongoing PC vision. Initially created by Intel, it was later bolstered by Willow Garage then Itseez. The library is cross-stage and free for use under the open-source BSD permit. OpenCV underpins a few models from profound learning structures like TensorFlow, Torch, PyTorch (in the wake of changing over to an ONNX model) and Caffe as indicated by a characterized rundown of upheld layers. It advances Open Vision Capsules, which is a versatile configuration, perfect with every other organization. Authoritatively propelled in 1999 the OpenCV venture was at first an Intel Research activity to propel CPU-concentrated applications, some portion of a progression of undertakings including constant beam following and 3D show dividers. The principal supporters of the undertaking remembered various improvement specialists for Intel Russia, just as Intel's Performance Library Team. OpenCV is written in C++ and its essential interface is in C++, yet it despite everything holds a less far reaching however broad more seasoned C interface. There are ties in Python, Java and MATLAB/OCTAVE. Since variant 3.4, OpenCV.js is a JavaScript official for chose subset of OpenCV capacities for the web stage. The entirety of the new turns of events and calculations in OpenCV are presently evolved in the C++ interface.

OpenCV runs on the accompanying work area working frameworks: Windows, Linux, macOS, FreeBSD, NetBSD, OpenBSD. OpenCV runs on the accompanying portable working frameworks: Android, iOS, Maemo, BlackBerry 10. The client can get official discharges from SourceForge or take the most recent sources from GitHub. OpenCV utilizes CMake

V. RESULTS

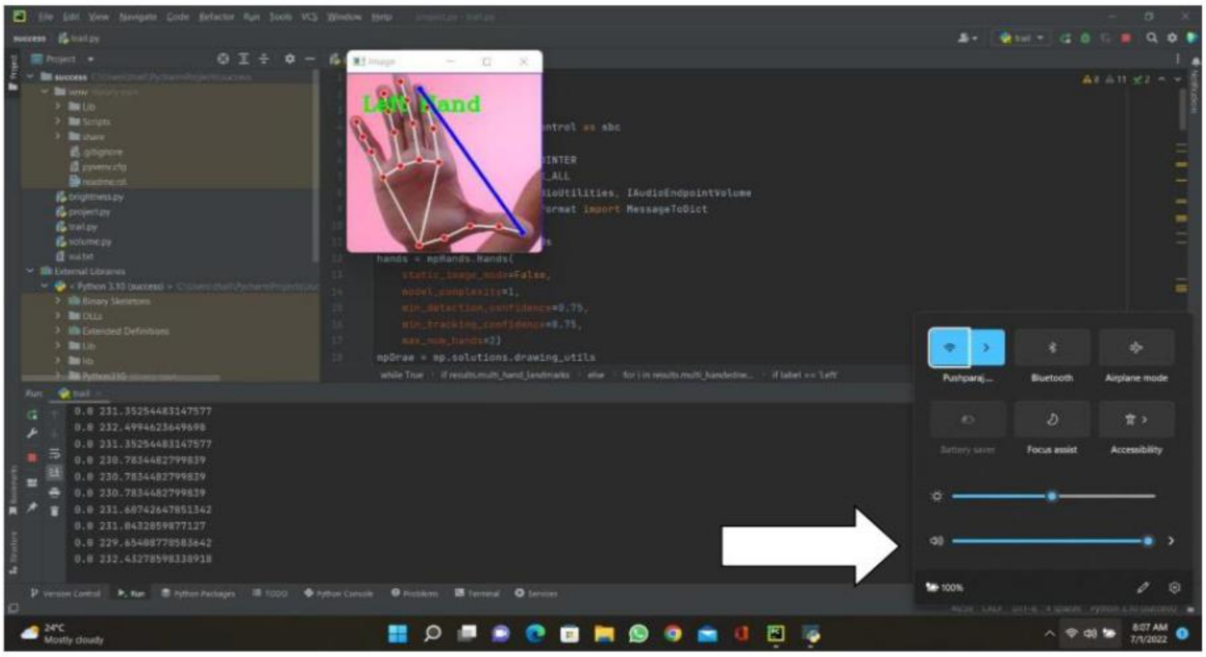


```
ntrol as sbc
INTER
_ALL
ioUtilities, IAudioEndpointVolume
ormat import MessageToDict
s
hands = mpHands.Hands(
    static_image_mode=False,
    model_complexity=1,
    min_detection_confidence=0.75,
    min_tracking_confidence=0.75,
    max_num_hands=2)
mpDraw = mp.solutions.drawing_utils
```

```
-61.4245955159177 27.018512172212592
-61.089488855548861 28.071337695236398
-61.72433254752248 26.076889628810597
-61.742661736687l8 26.019223662515376
-61.406942822566556 27.073972741361768
-61.72433254752248 26.076889628810597
-62.378737631682476 24.020824298928627
-60.74464363076449 29.154759474226582
-60.78841626451838 29.017236257093817
-61.406942822566556 27.073972741361768
```

Fig.6 RIGHT HAND

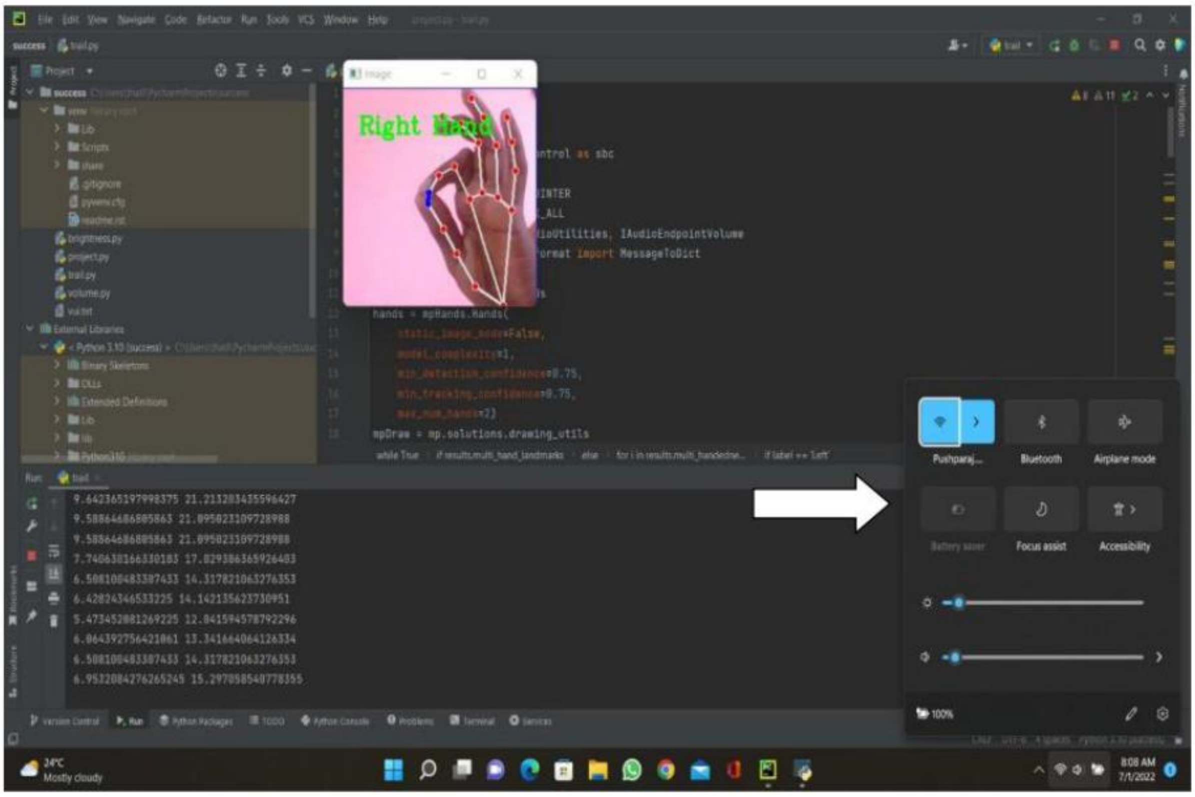
When the thumb and index fingers distance is minimum the volume is minimum



```
ntrol as sbc
INTER
_ALL
ioUtilities, IAudioEndpointVolume
ormat import MessageToDict
s
hands = mpHands.Hands(
    static_image_mode=False,
    model_complexity=1,
    min_detection_confidence=0.75,
    min_tracking_confidence=0.75,
    max_num_hands=2)
mpDraw = mp.solutions.drawing_utils
```

```
0.0 231.35254483147577
0.0 232.4994623649698
0.0 231.35254483147577
0.0 230.7834482799839
0.0 230.7834482799839
0.0 230.7834482799839
0.0 231.60742647851342
0.0 231.0432859877127
0.0 229.65408778583642
0.0 232.43278598338918
```

Fig.7 LEFT HAND



```
ntrol as sbc
INTER
_ALL
ioUtilities, IAudioEndpointVolume
ormat import MessageToDict
hands = mpHands.Hands(
    static_image_mode=False,
    model_complexity=1,
    min_detection_confidence=0.75,
    min_tracking_confidence=0.75,
    max_num_hands=2)
mpDraw = mp.solutions.drawing_utils
```

```
9.642365197998375 21.213203435596427
9.588644686805863 21.095023109728988
9.588644686805863 21.095023109728988
7.740630166330183 17.029386365926403
6.508100483387433 14.317821063276353
6.428243465333225 14.142135623730951
5.473452081269225 12.041594578792296
6.064392756421061 13.341664064126334
6.508100483387433 14.317821063276353
6.952084276265245 15.297058540778355
```

Fig.8 DATA COLLECTION PAGE

VI. CONCLUSION

The project presented a program that allowed user to perform hand gestures for easy software control. A vision-based hand Gesture system that does not require any special markers or gloves and can operate in real-time on a commodity PC with low-cost cameras. Specifically, the system can track the tip positions of the counters and index finger for each hand. The motivation for this hand Gesture was a desktop-based Hand gesture analysis can be divided into two main approaches, namely, glove-based analysis, vision-based analysis. The glove-based approach employs sensors (mechanical or optical) attached to a glove that acts as transducer of finger flexion into electrical signals to determine hand posture.

REFERENCES

1. Coll, Bartomeu, Antoni Buades & Morel, Jean-Michel. (2005). A Review of Image Denoising Algorithms, with a New One. SIAM Journal on Multiscale Modeling and Simulation. 4. 10.1137/040616024.
2. Daiyan, Golam & Mottalib, M.A. (2012). High performance decision based median filter for salt and pepper

noise removal in images. 107-112. 10.1109/ICCITechn.2012.6509707.

3. Rajeshwari Kumar Dewangan, Siddharth Chaubey, "Object Detection System with Voice Output using Python," 2021 International Journal of Science & Engineering Development Research, Vol. 6, Issue 3, pp. 15-20, 2021.
4. H. Jabnoun, F. Benzarti, H. Amiri, "Object detection and identification for blind people in video scene," 2015 15th International Conference on Intelligent Systems De- sign and Applications (ISDA), pp. 363-367, 2015.
5. Košale U, Žnidaršic P, Stopar K, "Detection of different shapes and materials by glasses for blind and visually impaired," 2019 6th Student Computer Science Research Conference, Vol. 57, 2019.
6. Ayushi Jaiswal Ajay Somkuwar Jayprakash Upadhyay 'Image denoising and quality measurements by using filtering and wavelet-based techniques' AEU - International Journal of Electronics and Communications 68(8):699-705
7. https://itsourcecode.com/free-projects/python-projects/brightness-control-withhand-detection-opencv-python-source-code/
8. https://github.com/i5han2/Volume_and_Brightness_Control_Using_Hand_Gestures
9. https://www.hackster.io/as4527/volume-control-using-hand-gesture-using-pythonand-opencv-7aab9f
10. https://www.section.io/engineering-education/creating-a-hand-gesture-volumecontroller-using-python-and-pycharm/
11. https://sourcecodehero.com/volume-control-with-hand-detection-opencv-pythonwith-source-code/