A

MAJOR PROJECT REPORT

ON

GESTURE RECOGNITION BASED ON COMPUTER VISION AND MEDIAPIPE

Submitted in partial fulfillment of the requirements

For the award of Degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

SIDDARTHA PULLAKHANDAM : 2453-18-733-107

Under the guidance of

G. SANTOSH KUMAR
Assistant Professor

Department of Information Technology

NEIL GOGTE INSTITUTE OF TECHNOLOGY

Kachavanisingaram Village, Hyderabad, Telangana 500058.

June 2022

NEIL GOGTE INSTITUTE OF TECHNOLOGY

A Unit of Keshav Memorial Technical Education (KMTES)
Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad

CERTIFICATE

This is to certify that the project work entitled "**GESTURE RECOGNITION BASED ON COMPUTER VISION AND MEDIAPIPE"** *is a bonafide work carried out by* **C. Abhishek Choudhary (2453-18-733-121) Gangishetty Mounik Teja (2453-18-733-080) Siddartha Pullakhandam (2453-18-733-107)** of IV-year VIII semester **Bachelor of Engineering** *in* **COMPUTER SCIENCE AND ENGINEERING** *by Osmania University, Hyderabad during the academic year* **2018-2022** *is a record of bonafide work carried out by them. The results embodied in this report have not been submitted to any other University or Institution for the award of any degree*

Internal Guide

Mr. G. Santosh kumar

Head of Department

Dr.K.V. Ranga Rao

External

NEIL GOGTE INSTITUTE OF TECHNOLOGY

A Unit of Keshav Memorial Technical Education (KMTES)
Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad

DECLARATION

We hereby declare that the Major Project Report entitled, "**GESTURE RECOGNITION USING COMPUTER VISION AND MEDIAPIPE** "submitted for the B.E degree is entirely our work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree.

Date:

Abhishek Choudhary	**Gangishetty Mounik Teja**	**Siddartha Pullakhandam**
(2453-18-733-121)	**(2453-18-733-080)**	**(2453-18-733-107)**

ACKNOWLEDGEMENT

We are happy to express our deep sense of gratitude to the principal of the college **Dr. D Jaya Prakash, Professor**, Neil Gogte Institute of Technology, for having provided us with adequate facilities to pursue our project.

We would like to thank**, Dr. K. V. Ranga Rao**, **Head of the Department,** Computer science and engineering, Neil Gogte Institute of Technology, for having provided the freedom to use all the facilities available in the department, especially the laboratories and the library.

We would also like to thank my internal guide **Mr. G Santosh Kumar, Assistant Professor** for his Technical guidance & constant encouragement.

We sincerely thank our seniors and all the teaching and non-teaching staff of the Department of Computer Science & Engineering and Information Technology for their timely suggestions, healthy criticism and motivation during the course of this work.

Finally, we express our immense gratitude with pleasure to the other individuals who have either directly or indirectly contributed to our need at the right time for the development and success of this work.

ABSTRACT

Computer vision and Machine Learning are used to understand images or video and extract information from them. In this project, we discuss how computer vision and machine learning are used to recognize gestures made with hand and these gestures are the general communication that are used on a day-to-day basis. This project aims to show various methods on how to use computer vision and machine learning to recognize gestures and how to implement these using OpenCV in python. Using these we can recognize gestures made by the hand the in air. The output is displayed on the output screen i.e., the feed of the computer webcam.

This can be done using hand movement detection where we detect the movement of the hand and plot landmarking on hand which helps to recognize the position of fingers of hand and shape made by the hand. After landmark detection. We now load the gesture recognizer model via the **pb files** of **keras** and the assets of the variables, in order to name the motions shown, we have a predefined list of gesture names which we have given like - peace, like, dislike, and so on. As we move our hand the gestures are recognized and these are differentiated using the landmarks given on the palm, then to differentiate the gestures from the trained list in real time.

TABLE OF CONTENTS

List of Figures

List of Tables

INTRODUCTION

CHAPTER - 1

INTRODUCTION

1.1 PROBLEM STATEMENT

In this project, our problem statement is to build a hands gesture detection model by plotting landmarks with the usage of a profound library Media pipe, and for other computer vision preprocessing we adapt the CV2 libraries. There happen to be many use cases in the market for this problem statement whether it acts as a communication medium for the people who are speech-deficient or in virtual reality for the real-time experience.

1.2 MOTIVATION

The motivation for this gesture recognition project is to develop a system which can detect the gestures and these gestures are widely used for conveying the information or to control the devices. Camera based solutions for gesture recognition has been widely used in numerous applications and capability to communicate through Human Computer Interaction. An idea of real time hand gesture recognition process through this device with the use of media pipe is explained along with insight of existing machine learning models. Finally, an attempt is made to explain the complexities with the device and the models along with its features.

1.3 SCOPE

This project deals with the problem of developing a vision-based real time hand gesture recognition system with the adaptation of the media pipe library to recognize the general gestures that can be made on a general usage such as (okay, not okay, stop, resume, call me, etc..). The proposed system consists of mainly three phases: the first phase (i.e., pre-processing), the next phase (i.e., feature extraction) and the final phase (i.e., classification). The first phase includes

hand segmentation that aims to isolate hand gesture from the background and removing the noises using special filters. This phase includes also landmark detection to find plot the points on the shown palm. The next phase, which constitutes the main part of this project, is devoted to the feature extraction problem where two feature extraction methods, namely, hand contour and complex moments are employed. These two extraction methods were applied in this study because they used different approaches to extract the features, namely, a boundary-based for hand contour and region-based for complex moments. The feature extraction algorithms deal with problems associated with hand gesture recognition such as scaling, translation and rotation. Then the hand gesture is recognized and shown as an output.

1.4 OUTLINE

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 handshakes) and lack high contrast patterns.

Media Pipe 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.

LITERATURE SURVEY

CHAPTER - 2

LITERATURE SURVEY

2.1 EXISITING SYSTEM

There have been a few developments on gesture language recognition with the usage of computer vision and image processing techniques. The overall purpose of these developments is to provide an alternative method to aid the disabled people in terms of communicating with each other to deliver a message or to depict an emotion without the usage of human voice or emotions. A model of Hand Gesture recognition made use of an Artificial Neural Network in order to train and recognize a static hand gesture and Jiahui Wu had equipped an acceleration-based gesture recognition approach, called FDSVM (Frame- based Descriptor and multi-class SVM), which needs only a wearable 3-dimensional accelerometer from which the gesture is collected and represented in the form of a frame-based descriptor".

Many approaches used position markers or colored bands to make the problem of hand recognition easier. However, they cannot be considered as a natural interface for the robot control due to their inconvenience. The motion recognition problem can be solved by combining basic image processing problems, namely, object detection, recognition, and tracking. There are many image processing algorithms that have been developed in target detection and recognition.ML techniques are general terms commonly used with basic feature extraction methods from original data and then combining, for example, SVM, decision tree, and nearest-neighbor, to train identity models. Since the detection and recognition algorithms require a large amount of computation and the accuracy cannot reach 100%, the object tracking techniques in gesture recognition are also widely applied to ensure the continuous real-time recording of subject location and avoid interference in multisubject environments.

There are many targeting algorithms for image processing such as BOOSTING, MIL, KCF , TLD, MEDIANFLOW , GOTURN , MOSSE, and CSRT. Depending on each problem towards the accuracy or processing speed, we can select the right algorithm. Besides, the problem requires processing image recognition in real time. Using CPU, GPU, and FPGA has their own advantages depending on the specific application for image processing algorithms. Image processing algorithms usually consume a lot of computing resources. In many cases, the continuously growing performance of CPUs is sufficient to handle such tasks within a specified time. However, GPU

and FPGA processors are widely used to replace CPUs for image processing applications. Besides, CNN (cellular neural network) technology is an analog parallel computing paradigm defined in space and found by the locality of connections between processing elements (cells or neurons). It has been introduced as a special high-speed parallel neural structure for image processing and recognition.

Another hand gesture recognition system was proposed to recognize the numbers from 0 to 10 where each number was represented by a specific hand gesture. This system has three main steps, namely, image capture, threshold application, and number recognition. It achieved a recognition rate of 89 The second approach, vision-based analysis, is based on how humans perceive information about their surroundings. In this approach, several feature extraction techniques have been used to extract the features of the gesture images. There has also been an approach where they collected the EMG corresponding to hand gestures using the Myo armband by Thalmic Labs Inc. The Myo armband is a wearable device provided with eight equally spaced non-invasive EMG electrodes and a Bluetooth transmission module. The EMG electrodes detect signals from the forearm muscles activity and afterwards the acquired data is sent to an external electronic device. The sampling rates for Myo data are fixed at 200 Hz and the data is returned as a unitless 8-bit unsigned integer for each sensor representing “activation” and does not translate to millivolts (mV). Another approach was via simple image processing algorithms such as the proposed algorithm.

They also tried to find a simple and practical algorithm such as convex hull.[22,23] Woun Bo Shen and Tan Guat Yew used a convex hull in the feature extraction stage. This algorithm is simple and efficient, but the number of gestures recognized is small. In addition, based on Kinect’s method of detecting the angle of a hull,[24] and this algorithm also has the disadvantage of a low number of recognition gestures. The hardware equipment required for the above algorithms is expensive and therefore not conducive to popularization.

The optimal features are unknown in many cases, resulting in a significant variation of the performance of the classifier depending on the selected features. Some deep learning algorithms are large in scale and require high hardware performance[47–49] and require a large number of training samples. Some deep networks require both training and GPU support for online deployment, which pose high hardware demand and thus is not conducive to small, embedded artificial intelligence systems.

2.2 PROPOSED SYSTEM

We are able to detect and train the hand gestures using the media pipe API. Media pipe API is an ML pipeline that provides services of palm detection and drawing landmarks on the hand. The API first recognizes the palm in the frame and then draws landmarks in the hand. Following palm detection over the entire image, the hand landmark model in the media pipe API uses regression to accomplish exact key point localization of 21 3D hand-knuckle coordinates within the detected hand regions, i.e., direct coordinate prediction. The pipeline is implemented as a Media Pipe graph that uses a hand landmark tracking subgraph from the hand landmark module, and renders using a dedicated hand renderer subgraph. The hand landmark tracking subgraph internally uses a hand landmark subgraph from the same module and a palm detection subgraph from the palm detection module.

Our method addresses the above challenges using different strategies. First, we train a palm detector instead of a hand detector, since estimating bounding boxes of rigid objects like palms and fists is significantly simpler than detecting hands with articulated fingers. In addition, as palms are smaller objects, the non-maximum suppression algorithm works well even for two-hand self-occlusion cases, like handshakes. Moreover, palms can be modelled using square bounding boxes (anchors in ML terminology) ignoring other aspect ratios, and therefore reducing the number of anchors by a factor of 3-5. Second, an encoder-decoder feature extractor is used for bigger scene context awareness even for small objects. Lastly, we minimize the focal loss during training to support a large number of anchors resulting from the high scale variance.

Providing the accurately cropped hand image to the hand landmark model drastically reduces the need for data augmentation (e.g., rotations, translation and scale) and instead allows the network to dedicate most of its capacity towards coordinate prediction accuracy. In addition, in our pipeline the crops can also be generated based on the hand landmarks identified in the previous frame, and only when the landmark model could no longer identify hand presence is palm detection invoked to relocalize the hand. TensorFlow is an open-source software library. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research.

Our method adapts the k-NN algorithm. *k*-NN is a type of classification where the function is only approximated locally and all computation is deferred until function evaluation. Since this

algorithm relies on distance for classification, if the features represent different physical units or come in vastly different scales than normalizing the training data can improve its accuracy dramatically. Both for classification and regression, a useful technique can be to assign weights to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of 1/*d*, where *d* is the distance to the neighbor. The neighbors are taken from a set of objects for which the class (for *k*-NN classification) or the object property value (for *k*-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

A peculiarity of the *k*-NN algorithm is that it is sensitive to the local structure of the data. Even with partially visible hands and self-occlusions, the model develops a consistent internal hand posture representation. These landmarks are used for the prediction of the name of the gestures. First, we train a palm detector instead of a hand detector, since estimating bounding boxes of rigid objects like palms and fists is significantly simpler than detecting hands with articulated fingers. In addition, as palms are smaller objects, the non-maximum suppression algorithm works well even for two-hand self-occlusion cases, like handshakes. Moreover, palms can be modelled using square bounding boxes (anchors in ML terminology) ignoring other aspect ratios, and therefore reducing the number of anchors by a factor of 3-5.

Second, an encoder-decoder feature extractor is used for bigger scene context awareness even for small objects (similar to the RetinaNet approach). We chose the encoder-decoder feature extractor because the machine translation is a major problem domain for sequence transduction models, whose input and output are both variable-length sequences. To handle this type of inputs and outputs, we can design an architecture with two major components. The first component is an encoder: it takes a variable-length sequence as the input and transforms it into a state with a fixed shape. The second component is a decoder: it maps the encoded state of a fixed shape to a variable-length sequence. This is called an encoder-decoder architecture.

Lastly, we minimize the focal loss during training to support a large amount of anchors resulting from the high scale variance. The trained model has an accuracy of 95.7% in comparison to the existing models. This is a real-time predict, as we change our hand gestures the name appearing in the display also changes. The model continuously predicts the name of each gesture that is found in each of the frames.

SOFTWARE REQUIREMENTS SPECIFICATION

CHAPTER - 3

SOFTWARE REQUIREMENTS SPECIFICATION

3.1 OVERALL DESCRIPTION

In this project, the gestures posed by a hand are the main input sources that are used to give test cases to the trained Machine Learning model and this model uses features like landmarks to complete the project and give the output. Landmark is the identification portion in the model, landmarks are given at different points on palm of a hand using mediapipe API, we have a list of points that identify different parts of the palm, and then gestures are made. In this project, we try to use computer vision, machine learning and try to recognize hand gestures. Gestures are made using hands so they are distinguished using modules, and every gesture is stored to show on the output screen.

3.2 OPERATING ENVIRONMENT

Software Requirements

Operating System	:	Any updated Operating System
Language used	:	Python
Libraries	:	OpenCV, TensorFlow, NumPy, Media Pipe
Development Kit	:	Any python supported IDE

Hardware Requirements

Processor	:	Intel Pentium® Dual Core Processor (Min)
Speed	:	2.9 GHz (Min)
RAM	:	4 GB (Min)
Hard Disk	:	2 GB (Min)

3.3 FUNCTIONAL REQUIREMENTS

3.3.1. Capturing video:

Using the webcam of a computer we capture the video. From the video single frame is taken to process the image where in real-time the gesture recognition is done. Each frame is flipped horizontally before the landmarks are drawn to avoid mirror imaging of the frame and the standard BGR format of OpenCV is converted into the RGB while using and processing the frame.

3.3.2. Drawing landmarks on palm:

We are trying to detect the hand gestures made by our hand. We are able to detect hand gestures using the media pipe API. Media pipe API is an ML pipeline that provides services of palm detection and drawing landmarks on the hand. The API first recognizes the palm in the frame and then draws landmarks in the hand.

Following palm detection over the entire image, the hand landmark model in the media pipe API uses regression to accomplish exact key point localization of 21 3D hand-knuckle coordinates within the detected hand regions, i.e., direct coordinate prediction. Even with partially visible hands and self-occlusions, the model develops a consistent internal hand posture representation. These landmarks are used for the prediction of the name of the gestures.

3.3.3. Predicting the hand gesture using Trained ML model:

After drawing the landmarks on a hand then we need to predict what gesture it is, for which we use machine learning, we use TensorFlow to train a model which contains labelled data of landmark positions of a gesture and the gesture name. Using this data from the already drawn landmarks on the frame we are able to predict the gesture names. The trained model has an accuracy of 95.7%. This is a real-time prediction, as we change our hand gestures the name appearing in the display also changes. The model continuously predicts the name of each gesture that is found in each of the frames.

3.4 NON-FUNCTIONAL REQUIREMENTS

3.4.1. Performance Requirements

The performance requirements refer to static numerical requirements placed on the interaction between the users and the software.

3.4.1.1. *Response Time*

Average response time shall be less than 2 sec.

3.4.1.2. *Start-Up/Shutdown Time*

The system shall be operational within 20 seconds of starting up.

3.4.1.3. *Capacity*

The trained model can accommodate 1 hand at a time.

3.4.1.4. *Utilization of Resources*

We can add as many gestures we want and train the model, provided we have enough storage capacity or we have to extend the storage capacity.

3.4.2 Safety Requirements

-NA-

3.4.3 Security Requirements

-NA-

3.4.4 Software Quality Attributes

Scalability

The system will be designed in such a way that it will be extendable. We can train and add different gestures as we go.

Availability

The system will be available to all its users round the clock i.e., they can access the system at any time.

Usability

The interfaces of the system will be user friendly enough that every user will be able use it easily.

SYSTEM DESIGN

CHAPTER - 4

SYSTEM DESIGN

4.1 USE CASE DIAGRAMS

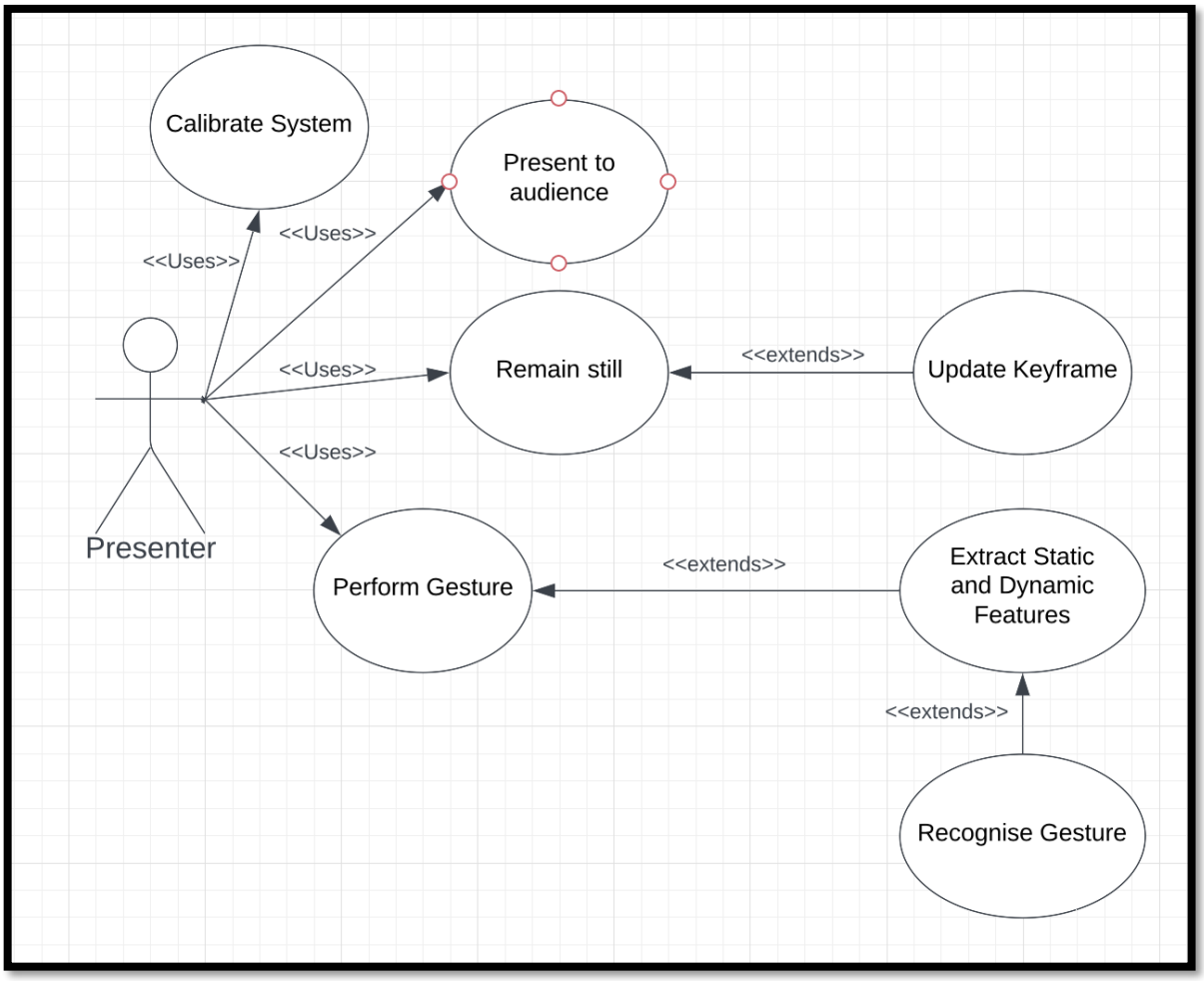


Fig 4.1: Use case diagram for Presenter

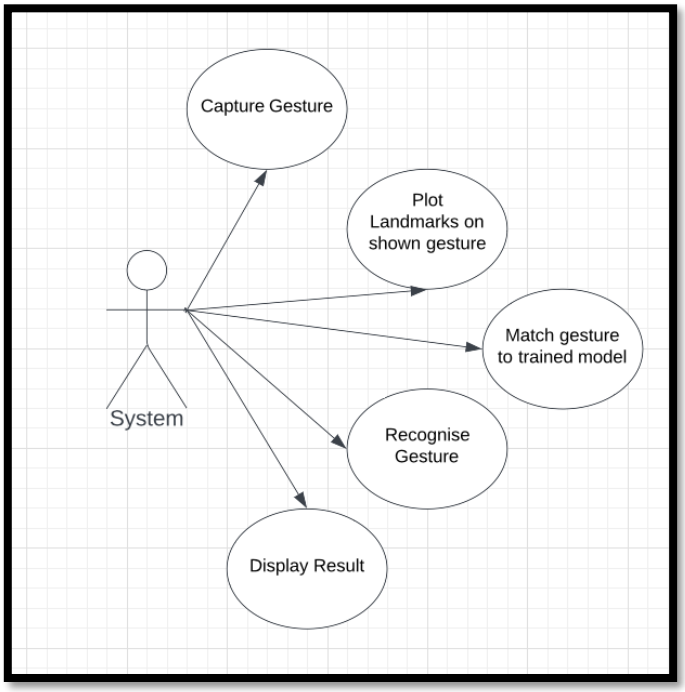


Fig 4.1.1: Use Case diagram for System

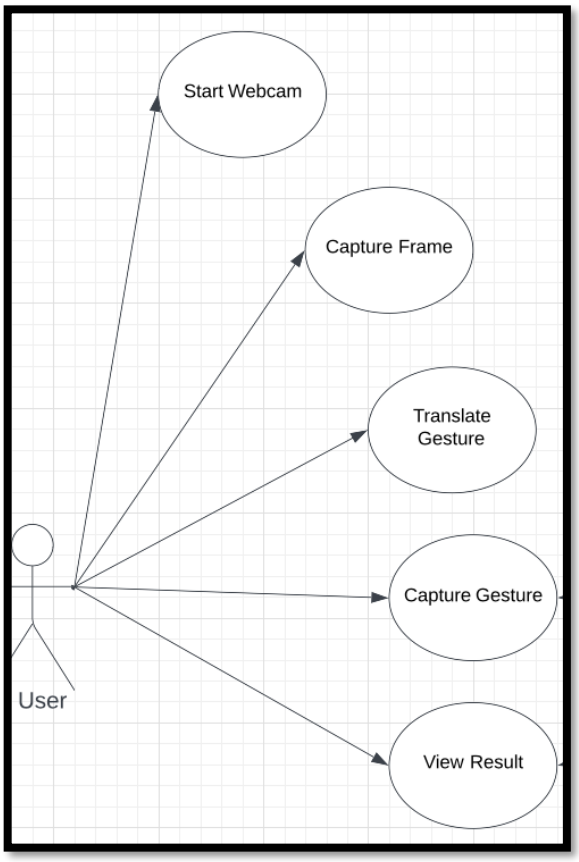


Fig 4.1.2: Use Case diagram for User

4.2 CLASS DIAGRAM

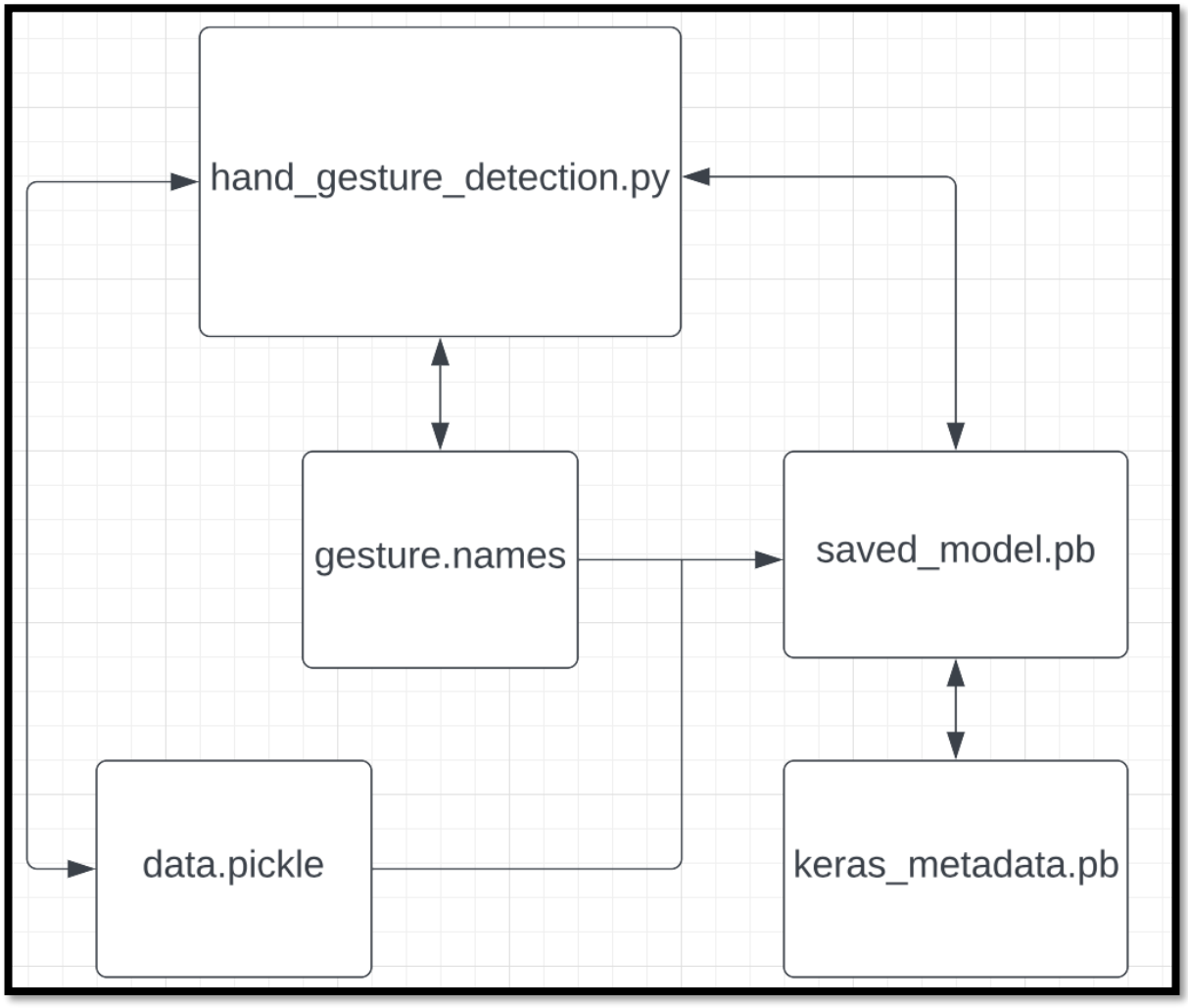
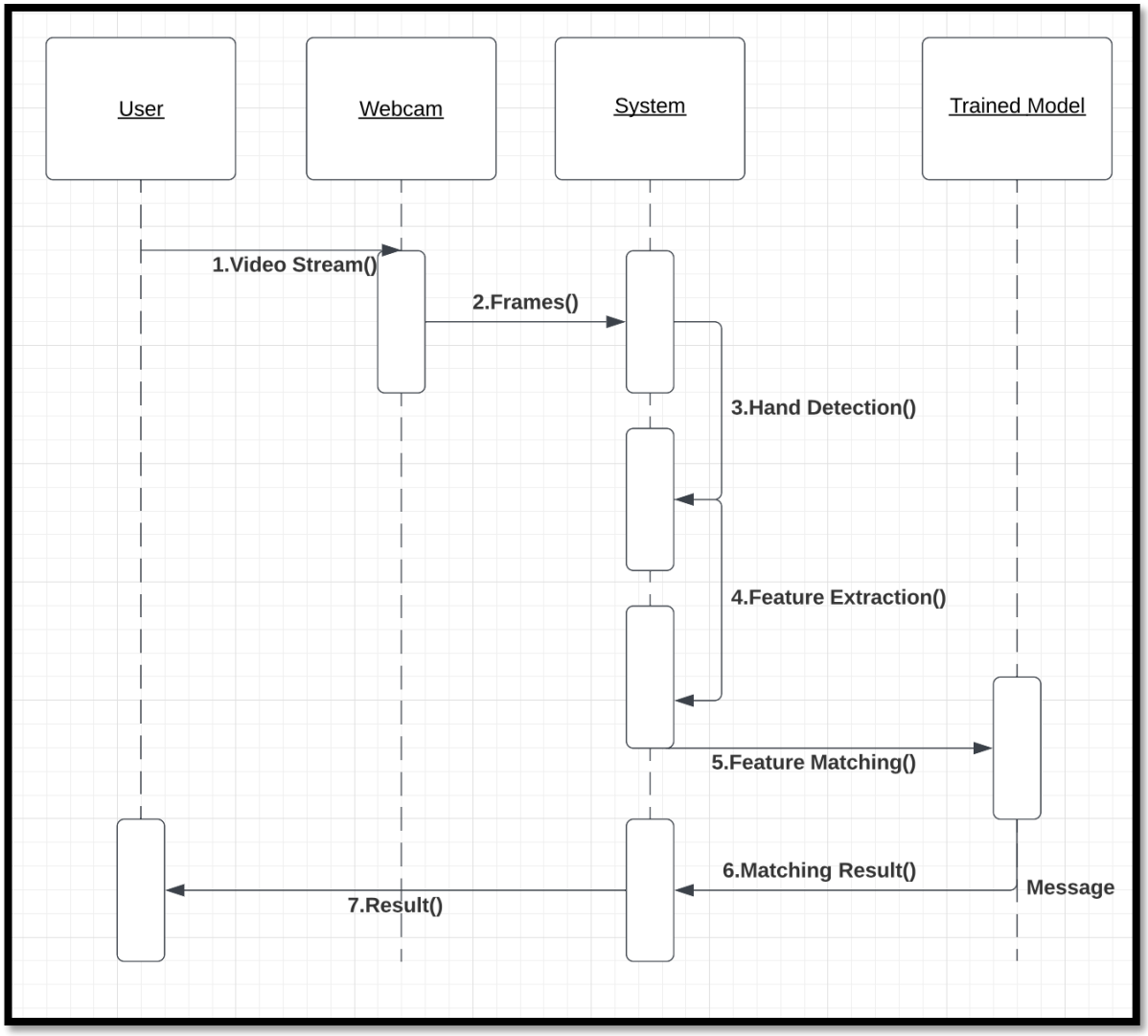


Fig 4.2: Class Diagram for Gesture Recognition based on Computer Vision and Media pipe

4.3 SEQUENCE DIAGRAM

Fig 4.3: Class Diagram for Gesture Recognition based on Computer Vision and Mediapipe



IMPLEMENTATION

CHAPTER - 5

IMPLEMENTATION

5.1 SAMPLE CODE

hand_gesture_detection.py

```
#importing libraries
import cv2
import numpy as np
import mediapipe as mp
import tensorflow as tf
from tensorflow.keras.models import load_model

# initialize mediapipe
recog_Hands = mp.solutions.hands
hands = recog_Hands.Hands(max_num_hands=1, min_detection_confidence=0.7)
mpDraw = mp.solutions.drawing_utils

# Load the gesture recognizer model
model = load_model('/Users/chaitanya/Desktop/Gesture-Recognition/Hand-Gesture-
Recognition/mp_hand_gesture')

# Load class names
f = open('/Users/chaitanya/Desktop/Gesture-Recognition/Hand-Gesture-
Recognition/gesture.names', 'r')
gesture_names = f.read().split('\n')
f.close()
print(gesture_names)

# Initialize the webcam
cap = cv2.VideoCapture(0)
```

```
while True:
    # Read each frame from the webcam
    _, frame = cap.read()

    x, y, c = frame.shape

    # Flip the frame vertically
    frame = cv2.flip(frame, 1)
    framergb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB) #we convert the base opencv bgr
to rgb

    # Get hand landmark prediction
    result = hands.process(framergb)

    # print(result)

    className = ''



 # post process the result
    if result.multi_hand_landmarks:
        landmarks = []
        for handslms in result.multi_hand_landmarks:
            for lm in handslms.landmark:
                # print(id, lm)
                lmx = int(lm.x * x)
                lmy = int(lm.y * y)

                landmarks.append([lmx, lmy])

            # Drawing landmarks on frames
```

```
            mpDraw.draw_landmarks(frame, handslms, recog_Hands.HAND_CONNECTIONS)

            # Predict gesture
            prediction = model.predict([landmarks])
            # print(prediction)
            classID = np.argmax(prediction)
            className = gesture_names[classID]

    # show the prediction on the frame
    cv2.putText(frame, className, (10, 50), cv2.FONT_HERSHEY_SIMPLEX,
                    1, (0,0,255), 2, cv2.LINE_AA)



    # Show the final output
    cv2.imshow("Output", frame)



    if cv2.waitKey(1) == ord('q'):
        break

# release the webcam and destroy all active windows
cap.release()

cv2.destroyAllWindows()
```

TESTING

CHAPTER - 6

TESTING

6.1 TEST CASES

Test Case to check whether the required Software is installed on the systems

Test Case ID:	1
Test Case Name:	Required Software Testing
Purpose:	To check whether the required Software is installed on the systems
Input:	Enter python command
Expected Result:	Should Display the version number for the python
Actual Result:	Displays python version
Failure	If the python environment is not installed then the Deployment fails

Table 6.1.1 python Installation verification

Test Case to check Program Integration Testing

Test Case ID:	2
Test Case Name:	Programs Integration Testing
Purpose:	To ensure that all the modules work together
Input:	All the modules should be accessed.
Expected Result:	All the modules should be functioning properly.
Actual Result:	All the modules should be functioning properly.
Failure	If any module fails to function properly, the implementation fails.

Table 6.1.2 python Programs Integration Testing

Test Case to Collect Dataset and Load the Dataset

Test Case ID:	3
Test Case Name:	Collect Dataset and Load the Dataset
Purpose:	Check Dataset is collected and the data is stored
Input:	Provide Dataset as input
Expected Result:	Dataset is collected and view the Dataset and store the Dataset
Actual Result:	Load the Dataset and view the Dataset and store
Failure	If the dataset is not loaded, it will throw an error.

Table 6.1.3 Collect Dataset and Load the Dataset

Test Case to check whether the landmarks are getting plotted

Test Case ID:	4
Test Case Name:	Plotting Landmarks
Purpose:	Plotting the 21 landmarks on the palm
Input:	Provide dataset and run the mediapipe module
Expected Result:	After loading the dataset, the landmarks will be plotted
Actual Result:	Data is extracted and landmarks are plotted on the palm
Failure	If the data is not extracted it display the alert message

Table 6.1.4 Plotting Landmarks

Test Case to check whether the gesture is recognized

Test Case ID:	5
Test Case Name:	Gesture Recognition
Purpose:	Gesture Recognition using MediaPipe
Input:	Provide dataset and pose a gesture
Expected Result:	After Evaluation we get the gesture name
Actual Result:	After Evaluation we get the gesture name
Failure	If the data is not Evaluated it does not display the gesture

Table 6.1.5 Gesture Recognition

Test Case to check whether the newly trained gesture is recognized

Test Case ID:	6
Test Case Name:	Gesture Recognition #2
Purpose:	Gesture Recognition using MediaPipe
Input:	Provide dataset and pose a gesture
Expected Result:	After Evaluation we get the gesture name
Actual Result:	After Evaluation we get the gesture name
Failure	If the data is not Evaluated it does not display the gesture

Table 6.1.6 Newly trained gesture recognition

SCREENSHOTS

CHAPTER - 7

SCREENSHOTS

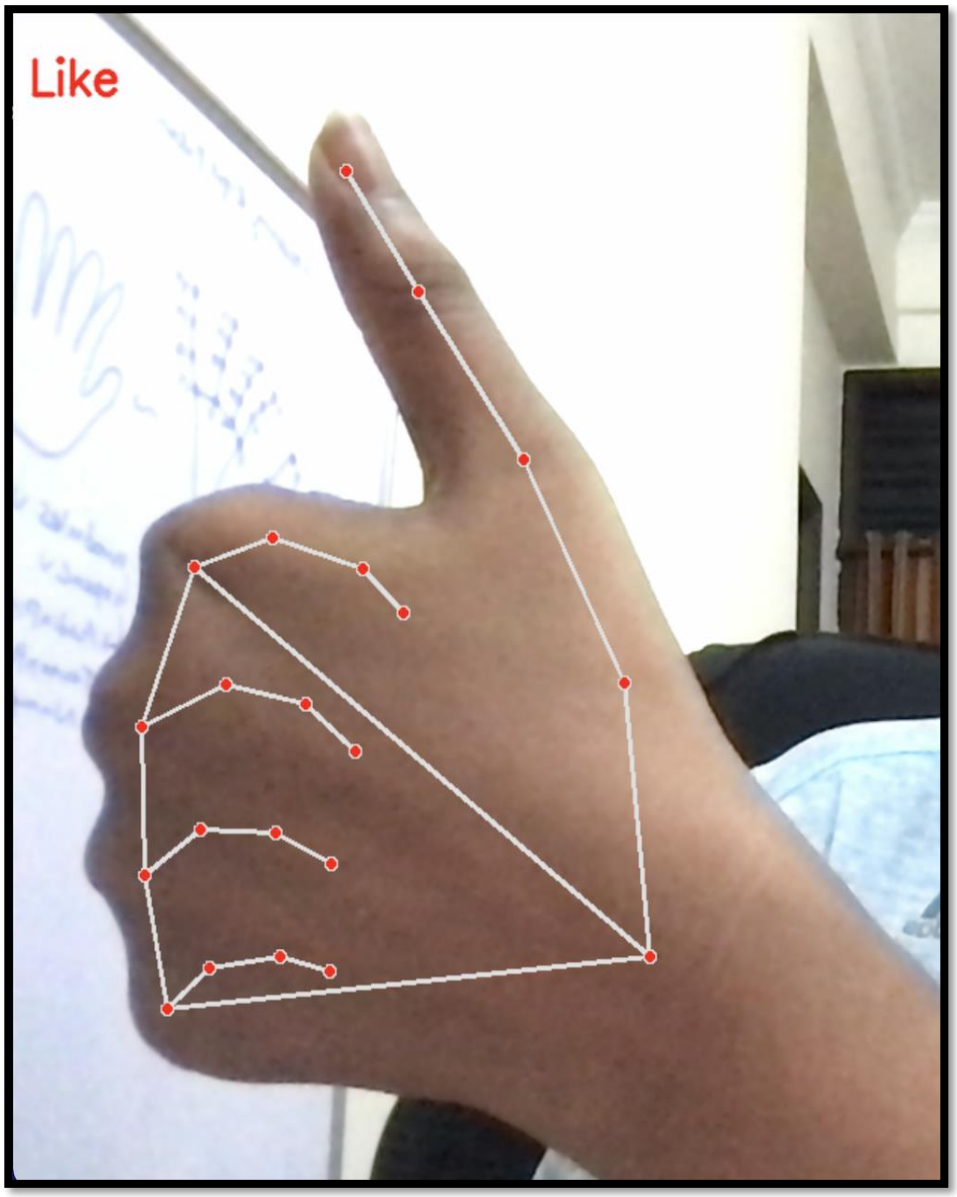


Fig 7.1: Gesture showing the output as “Like”

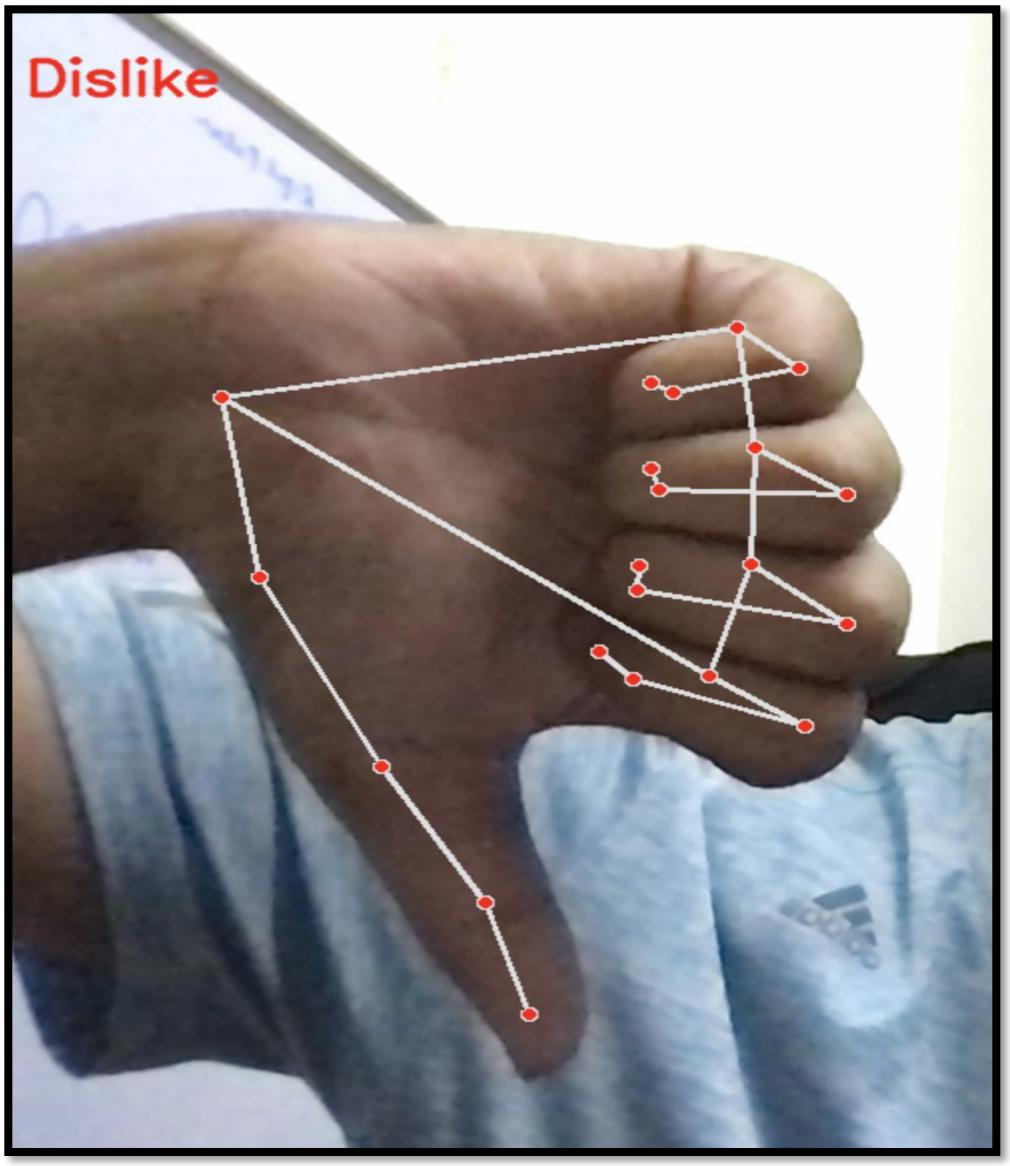


Fig 7.2: Gesture showing the output as “Dislike”

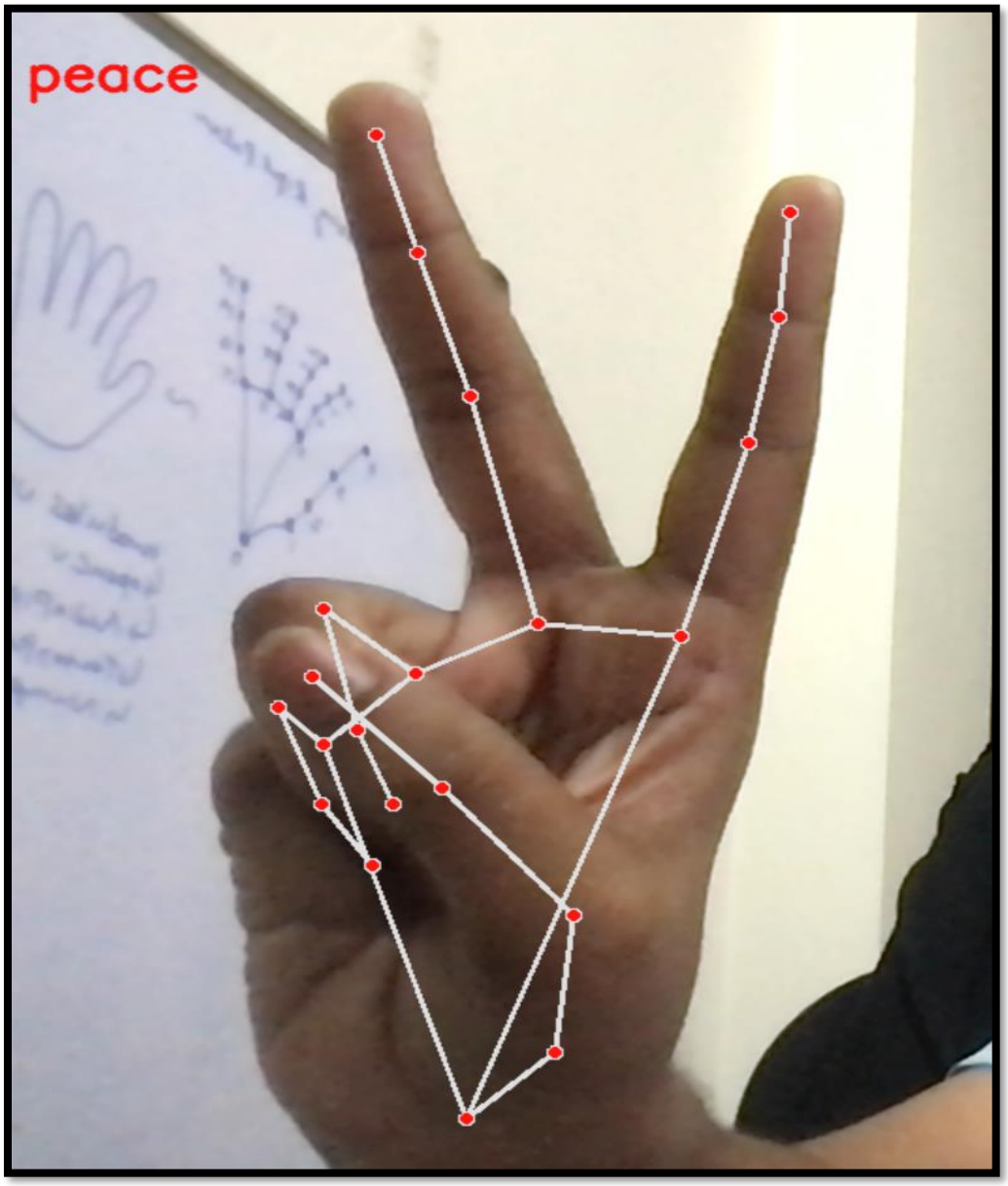


Fig 7.3: Gesture showing the output as “Peace”

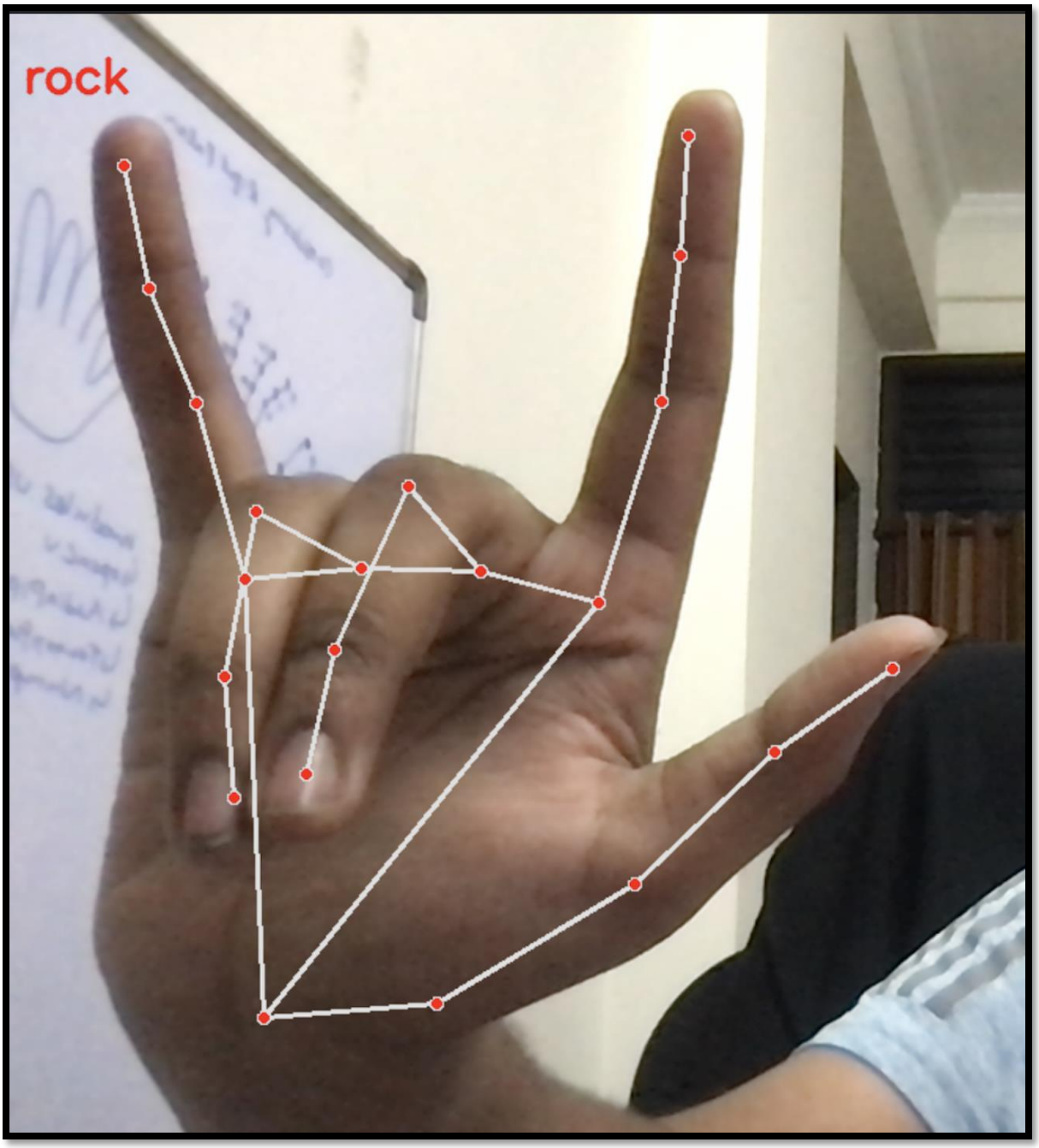


Fig 7.4: Gesture showing the output as “Rock”

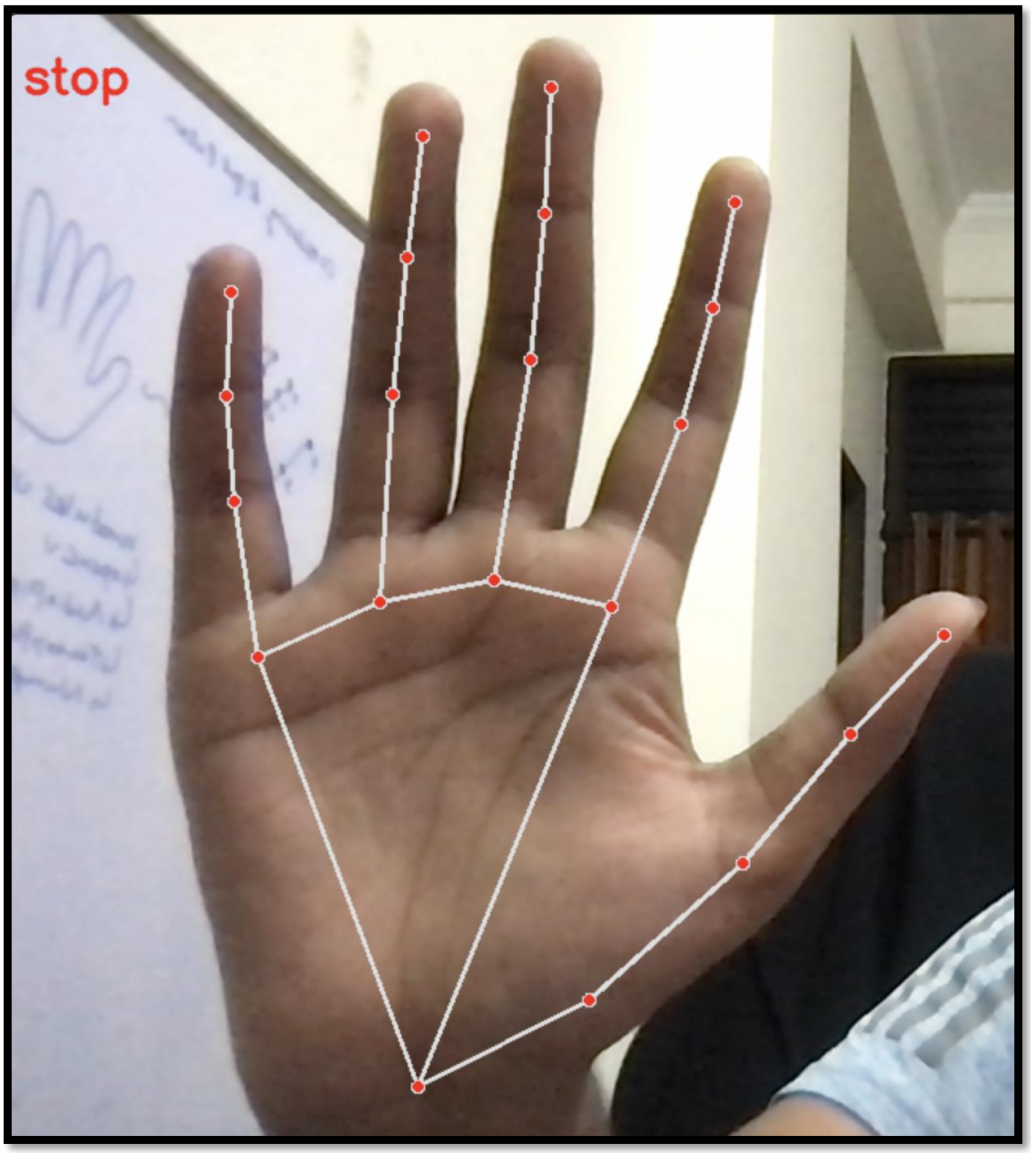


Fig 7.5: Gesture showing the output as “Stop”

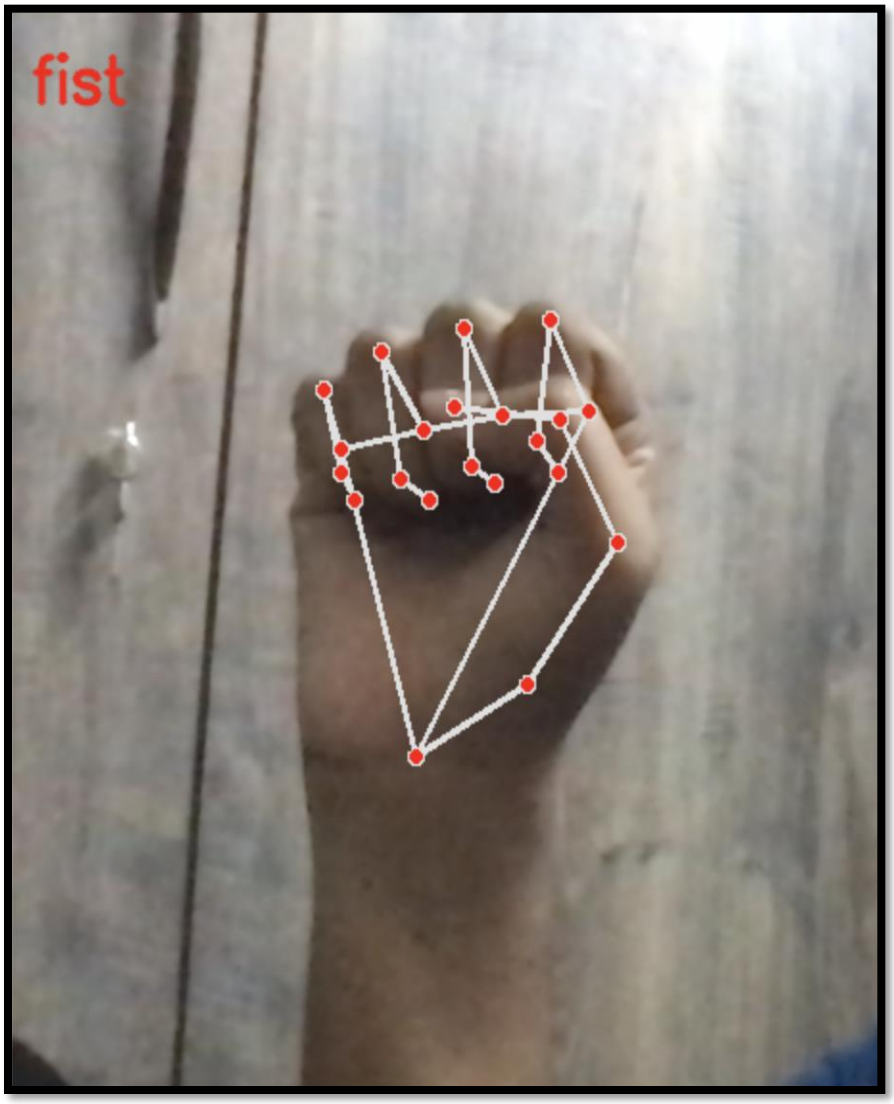


Fig 7.6: Gesture showing the output as “Fist”

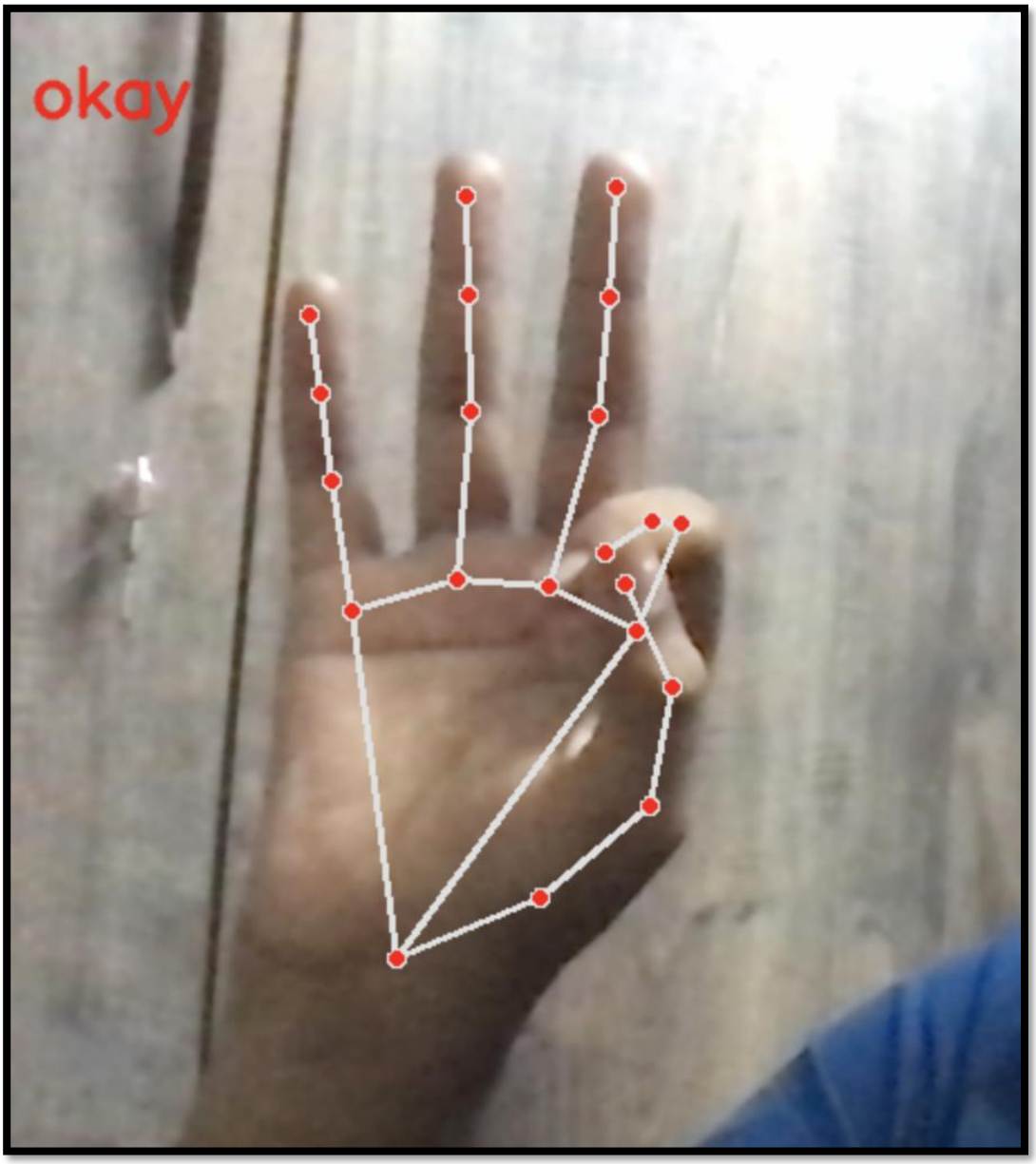


Fig 7.7: Gesture showing the output as “Okay”

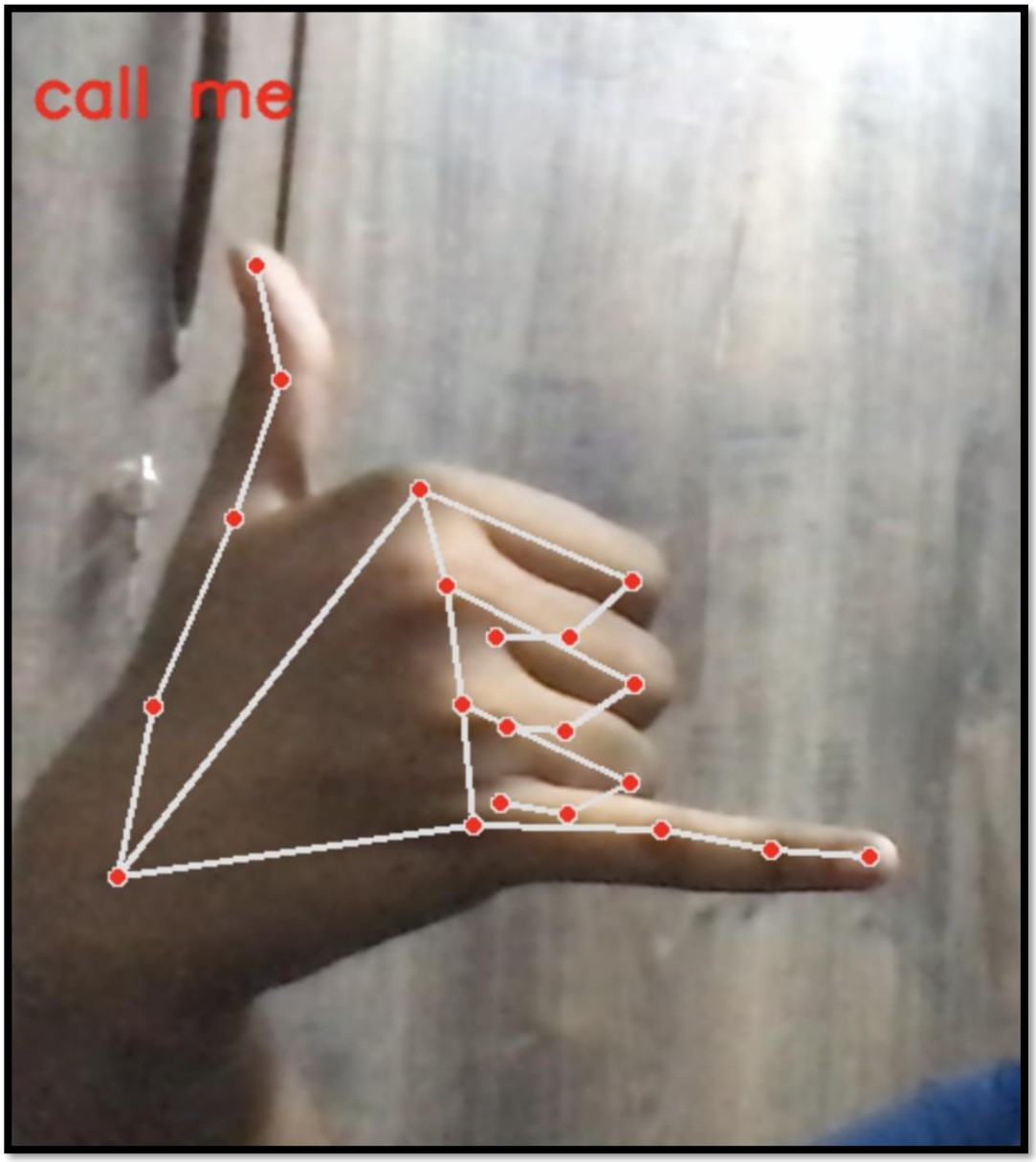


Fig 7.8: Gesture showing the output as “Call Me”

CONCLUSION AND FUTURE SCOPE

CHAPTER - 8
CONCLUSION AND FUTURE SCOPE

By using computer vision and media pipe we are able to draw landmarks on the hand and by using the KNN model we are able to predict the hand gesture with 95.7% accuracy and fast output. The future scope of this project is to break the barrier and act as a medium between the speechless and the people who could communicate. And to also communicate gesture recognition will be able to convey the messages in a hassle freeway. Integration of this technology into different devices for wider use is a future aim.

Our aim is to decrease the response time to predict the gesture in the future iterations of this project and to add an extensive set of gestures for wider usage which can be updated anytime such that our model is up to date. The Hand Gesture recognition is moving at tremendous speed for the futuristic products and services and major companies are developing technology based on the hand gesture system and that includes companies like Microsoft, Samsung, Sony. The verticals include where the Gesture technology is and will be evident are Entertainment, Artificial Intelligence, Education and Medical and Automation fields.

It’as a brilliant feature turning data into features with mix of technology and Human Gesture. Smart phones have been experiencing 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. Google Glass has been also in the same cadre. This ttechnology can be embedded into smart televisions nowadays as well, which can be easily controlled by hand gestures. In the medical fields Hand Gesture may also be experienced in terms of Robotic Nurse and medical assistance.

BIBLIOGRAPHY

[1] Trong-Nguyen Nguyen, Huu-Hung Huynh, Jean Meunier, (2013), “Static gesture recogniser by training an artificial neural network”. In proceedings of Journal of Image and Graph

[2]Jiahui Wu, Gang Pan, Daqing Zhang, Guande Qi, Shijian Li, (2009), “Gesture Recognition with the use of 3-D Accelerometer”, In proceedings of International Conference on Ubiquitous Intelligence and Computing UIC 2009: Ubiquitous Intelligence and Computing pp 25-38

[3] Mäntylä, V.-M.: “Hidden Markov Models implementation to Isolated User-Specefic Hand Gesture Recognition”. In proceedings of VTT publications (2001).

[4] Siddharth S. Rautaray, Anupam Agrawal, (2012), “Hand Gesture Recognition System for Real - Time Applications”. In proceedings of International Journal of UbiComp (IJU), Vol.3, No.1, January 2012

[5] Pavlovic. V., Sharma R., & Huang, T.S. (1997), “Visual elucidation of hand gestures for HCI”: A review.” IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), 7(19): pp. 677–695.

[6] Munir Oudah, Ali Al-Naji, Javaan Chahl, (2020), “A Review on the implementation of computer vision for gesture recognition”. In proceedings of the Journal of Imaging.

[7] Fakhreddine Karray, Milad Alemzadeh, Jamil Abou Saleh, Mo Nours Arab, (2008). “Human-Computer Interaction: An overview on state of gesture recognition”, International Journal on Smart Sensing and Intelligent Systems, Vol. 1(1).

[8] E. Ueda, “A hand pose estimation for vision-based human interfaces,” (2003), IEEE Transactions on Industrial Electronics, vol. 50, No. 4, pp. 676–684

[9] A. Utsumi and J. Ohya, “Multi hand gesture tracking with the usage multiple cameras”, in Proc. Int. Conf. on Computer Vision and Pattern Recognition, pp. 473–478, 1999.

[10] F. Chen, C. Fu, & C. Huang, 2003 , “Hand gesture recognition with the use of a real-time tracking method and hidden Markov models” Image and Vision Computing, pp. 745-758.

[11] T. Starner, A. Pentland, Visual recognition of American Sign Language using hidden Markov models, Proceedings of International. Workshop on Automatic Face- and Gesture-Recognition, Zurich, Switzerland, 1995.

[12] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg and Matthias Grundmann.”MediaPipe: A Framework for Perceiving and Processing Reality”. In proceedings of Google Research at Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR) 2019.

APPENDIX A: TOOLS AND TECHNOLOGIES

- PYTHON V3: The Python language comes with many libraries and frameworks that make coding easy. This also saves a significant amount of time.
- JUPYTER NOTEBOOK: The Jupyter Notebook is an open-source web application that you can use to create and share documents that contain live code, equations, visualizations, and text.
- TENSORFLOW: TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.
- NUMPY: NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices.
- WINDOWS 10: Windows 10 was used as the operating system.
- MACOS: MACOS was used as the operating system
- MEDIAPIPE: Media Pipe is a Framework for building machine learning pipelines for processing time-series data like video, audio, etc. This cross-platform Framework works in Desktop/Server, Android, iOS, and embedded devices like Raspberry Pi and Jetson Nano.
- KERAS: Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.
- COMPUTER VISION: Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information.