EC 5512 SUMMER PROJECT

SIGN LANGUAGE DETECTION BY IMAGE PROCESSING

A SUMMER PROJECT REPORT

Submitted by

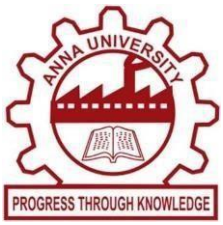
SARAVANAPERUMAL R 2020504579

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING



MADRAS INSTITUTE OF TECHNOLOGY

DEPARTMENT OF ELECTRONICS ENGINEERING

ANNA UNIVERSITY: CHENNAI 600 044

DECEMBER 2022

MADRAS INSTITUTE OF TECHNOLOGY
ANNA UNIVERSITY : CHENNAI 600 044

BONAFIDE CERTIFICATE

Certified that this project **"SIGN LANGUAGE DETECTION BY IMAGE PROCESSING"** is the bonafide work of

SARAVANAPERUMAL R 2020504579

who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other project report on the basis of which a degree was conferred on an earlier occasion on this or any other candidate

SIGNATURE

Dr. P INDUMATHI
HEAD OF THE DEPARTMENT
Department of Electronics Engineering
Madras Institute of Technology
Anna University
Chennai 600 044

SIGNATURE

Mrs. A DIVYA
FACULTY IN CHARGE
Assistant Professor
Department of Electronics Engineering
Madras Institute of Technology
Anna University, Chennai 600 044

ACKNOWLEDGEMENT

I consider it as our privilege and our primary duty to express our gratitude and respect to all those who guided and inspired us in the successful completion of the project.

I owe solemn gratitude to **Dr. J.Prakash**, Dean, Madras Institute of Technology, for having given consent to carry out the project work at MIT Campus, Anna University.

I wish to express our sincere appreciation and gratitude to **Dr. P.Indumathi** Professor and Head of the Department of Electronics Engineering, who has encouraged and motivated us in our endeavours.

I are extremely grateful to our faculty in charge **Dr. P.Indumathi** Professor and **Mrs. A.Divya** Assistant Professor, Department of Electronics Engineering for their encouragement for the completion of the project.

I also thank all the teaching and non-teaching staff members of the Departmentof Electronics Engineering for their support in all aspects.

Place: Chennai

Date : 23 - 12 - 2022

ABSTRACT

Speech impairment is a disability which affects one's ability to speak and hear. Such individuals use sign language to communicate with other people. Although it is an effective form of communication, there remains a challenge for people who do not understand sign language to communicate with speech impaired people. The aim of this paper is to develop an application which will translate sign language to English in the form of text and audio, thus aiding communication with sign language. The application acquires image data using the webcam of the computer, then it is preprocessed using a combinational algorithm and recognition is done using template matching. The translation is in the form of text. The database used for this system includes 300 images for each of English alphabets. The system produces 88% accuracy.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

Conversing with people with a hearing disability is a major challenge. Deaf and Mute people use a hand gestured sign language to communicate, which leads to others, outside their community, often facing difficulty in recognizing their language by the signs they make. Hence, there is a need for systems that recognize the different signs and conveys the information to common people.

1.2 PROBLEM STATEMENT

The most frequent sensory deficiency in people today is hearing loss. According to WHO estimates, there are roughly 63 million persons in India who suffer from Significant Auditory Impairment, putting the prevalence at 6.3 percent of the population. According to the NSSO study, there are 291 people with severe to profound hearing loss for every 100,000 people. A substantial number of them are youngsters between the ages of 0 - 14. With such a huge population of hearing-impaired young Indians, there is a significant loss of physical and economic output. The main problem is that people who are hard of hearing, such as the deaf and dumb, find it difficult to interact with normal people since people who are not impaired do not learn how to communicate with each other using sign language.

The solution is to develop a translator that can detect sign language used by a disabled person, and then feed that sign into a machine-learning algorithm called transfer learning, which is then detected by the neural network and translated on the screen so that a normal person can understand what the sign is saying.

1.3 SIGN LANGUAGE RECOGNITION

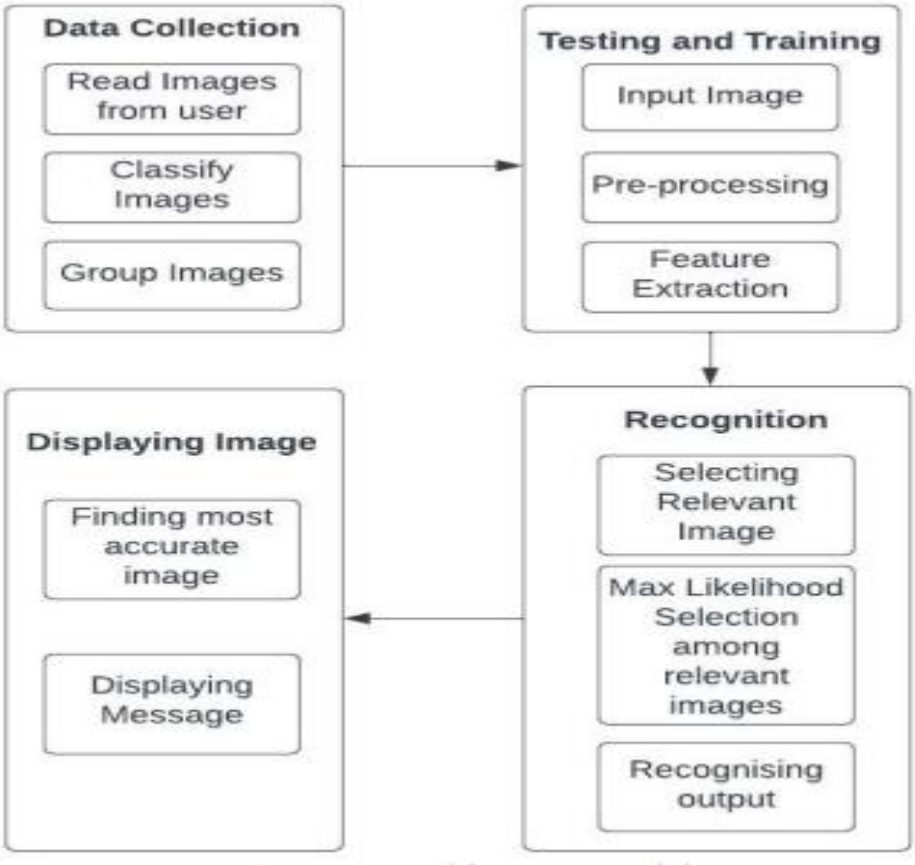
Sign language recognition is a technique of translating a user's gestures into text. It helps individuals, who are unable to interact, with others thereby reducing the visible gap between the special aided individuals to the general public. Raw images/videos are turned into text that can be read and comprehended using image processing techniques and neural networks which links the gesture to its corresponding text in the training data. Dumb people are frequently cut off from normal social interaction and it has been found that they

struggle to engage with regular people through their gestures often, as only handful of them are understood by the majority of people. This means, that people with hearing impairment or deaf people cannot talk like normal people, so they have to depend on some sort of visual communication most of the time.

Although impressive speech recognition systems were created earlier, there are yet any commercially available character recognition devices. The goal is to familiarize the computer to recognize human speech and create a user-friendly human-computer interface (HCI). A few steps in this technique include training your computer to recognize voices, expressions, and human gestures.

Gestures are nonverbal exchanges of messages, and a person can make a variety of gestures at once. Since human signals are perceived through vision, this is a fascinating subject for computer vision researchers. Complex programming strategies are required to code these gestures in machine language and help in creating an HCI that can detect human gestures. For better output creation, we are relying on Image Processing and Template Matching in our research.

Figure 1.1: Architecture of Sign Language Recognition System



In the sign language recognition model, the architecture provides a blueprint and ideal techniques to follow so as to developed a well-structured

application as per our requirement. This architecture mainly involves three phases:

Phase 1- Data Collection Phase: A model is being built and a sequence of images is being fed to the model. This phase is useful to further train the model based on the type of symbol.

Phase 2- Training and Testing Phase: This phase involves a set of inputs to the model and a particular output is being expected. Based on the outputs, the accuracy of the model is being identified.

Phase 3- Recognition of output: In this phase, an image is being given as input to the model. Based on the images being trained to the model, it matches the image with a particular output and generates a message. The sign being identified in the above phase has a particular meaning and a message is being assigned in the training phase. The meaning is displayed to the user.

1.4 SYSTEM CONFIGURATION

1.4.1 SOFTWARE REQUIREMENTS

The software requirements for this design are listed below:

- Windows/Mac OS
- Python (3.7.4)
- IDE (Pycharm)
- Numpy (version 1.16.5)
- cv2 (openCV) (version 3.4.2)
- Keras (version 2.3.1)
- Tensorflow (version 2.0.0)

1.4.2 HARDWARE REQUIREMENTS

The hardware requirements for this design are listed below:

- Processor: Intel Core i3
- RAM: 4GB
- Hard Disk: 10 GB Minimum
- Webcam: 3 Mega Pixels Minimum
- Monitor
- Keyboard
- Mouse

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

A review of the literature for the proposed framework indicates that numerous methods and algorithms have been used to tackle sign detection in videos and photos. Understanding neural networks was a major part of the domain analysis which was conducted for the project.

2.1.1 EXISTING SYSTEMS

In Literature survey we have gone through other similar works that are implemented in the domain of sign language recognition. The summaries of certain project works are mentioned below.

Glove Based System

In this method the signer has to wear a hardware glove, while the hand movements are getting captured. Based on the movement of the glove a message will be generated.

Vision Based System

Vision based system is classified into static and dynamic recognition. Statics is concerned with the diagnosis of static gestures (2d-images), whereas dynamic is concerned with the live record of gestures on a real-time basis. This entails the employment of a camera to record movement.

2.1.2 DISADVANTAGE OF EXISTING SYSTEMS

- Gloves are typically expensive, and not everyone can afford them.
- It's a hassle to trace a defect in a glove when isn't functioning.
- The parts that are being used to make these gloves are pricy. These materials can only be obtained in specific locations.

CHAPTER 3

PROPOSED WORK

3.1 INTRODUCTION

Our proposed system is sign language recognition system using convolution neural networks which recognizes various hand gestures by capturing video and converting it into frames. Then the hand pixels are segmented and the image it obtained and sent for comparison to the trained model. Thus our system is more robust in getting exact text labels of letters.

3.2 METHODOLOGY

3.2.1 TRAINING MODULE:

Supervised machine learning is one of the ways of machine learning where the model is trained by input data and expected output data. To create such model, it is necessary to go through the following phases:

1. model construction
2. model training
3. model testing
4. model evaluation

1. Model construction- It depends on machine learning algorithms. In this projectscase, it was neural networks. Such an algorithm looks like:

1. Begin with its object

2. Then consist of layers with their types

3. After adding a sufficient number of layers the model is compiled. At this moment Keras communicates with TensorFlow for construction of the model. During model compilation it is important to write a loss function and an optimizer algorithm. The loss function shows the accuracy of each prediction made by the model. Before model training it is important to scale data for their further use.

2. Model training- After model construction it is time for model training. In this phase, the model is trained using training data and expected output for this data. Progress is visible on the console when the script runs. At the end it will report the final accuracy of the model.

3. Model Testing- During this phase a second set of data is loaded. This data set has never been seen by the model and therefore it's true accuracy will be verified. After the model training is complete, and it is understood that the model shows the right result.

4. Model Evaluation- Finally, the saved model can be used in the real world. The name of this phase is model evaluation. This means that the model can be used to evaluate new data.

3.2.2 PREPROCESSING:

Uniform Aspect Ratio- An aspect ratio is a proportional relationship between an image's width and height. Essentially, it describes an image's shape. Aspect ratios are written as a formula of width to height.For example, a square image has an aspect ratio of 1:1, since the height and width are the same. The image could be 500px × 500px, or 1500px × 1500px, and the aspect ratio would still be 1:1. As another example, a portrait-style image might have a ratio of 2:3. With this aspect ratio, the height is 1.5 times longer than the width. So, the image could be 500px × 750px, 1500px × 2250px, etc. Cropping to an aspect ratio Aside from using built in site style options, you may want to manually crop an image to a certain aspect ratio. For example, if you use product images that have same aspect ratio, they'll all crop the same way on your site.

Option 1- Crop to a pre-set shape Use the built-in Image Editor to crop images to a specific shape. After opening the editor, use the crop tool to choose from preset aspect ratios.

Option 2- To crop images to a custom aspect ratio not offered by our built-in Image Editor, use a third-party editor. Since images don't need to have the same dimensions to have the same aspect ratio, it's better to crop them to a specific ratio than to try to match their exact dimensions. For best results, crop the shorter side based on the longer side. For instance, if your image is 1500px × 1200px, and you want an aspect ratio of 3:1, crop the shorter side to make the image 1500px × 500px.

Image Scaling- In computer graphics and digital imaging, image scaling refers to the resizing of a digital image. In video technology, the magnification of digital material is known as upscaling or resolution enhancement. When scaling a vector graphic image, the graphic primitives that make up the image can be scaled using geometric transformations, with no loss of image quality. When scaling a raster graphics image, a new image with a higher or lower number of pixels must be generated. In the case of decreasing the pixel number (scaling down) this usually results in a visible quality loss. From the standpoint of digital signal processing, the scaling of raster graphics is a two-dimensional example of sample-rate conversion, the conversion of a discrete signal from a sampling rate (in this case the local sampling rate) to another.

3.2.3 DATASETS FOR TRAINING:

Figure 3.1: Dataset used for training

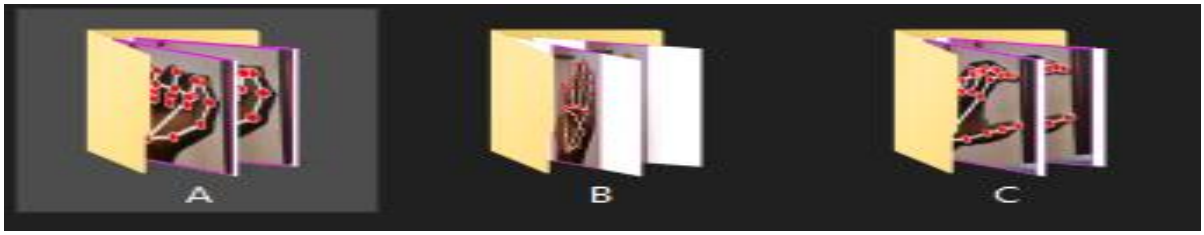


Figure 3.2: American sign language

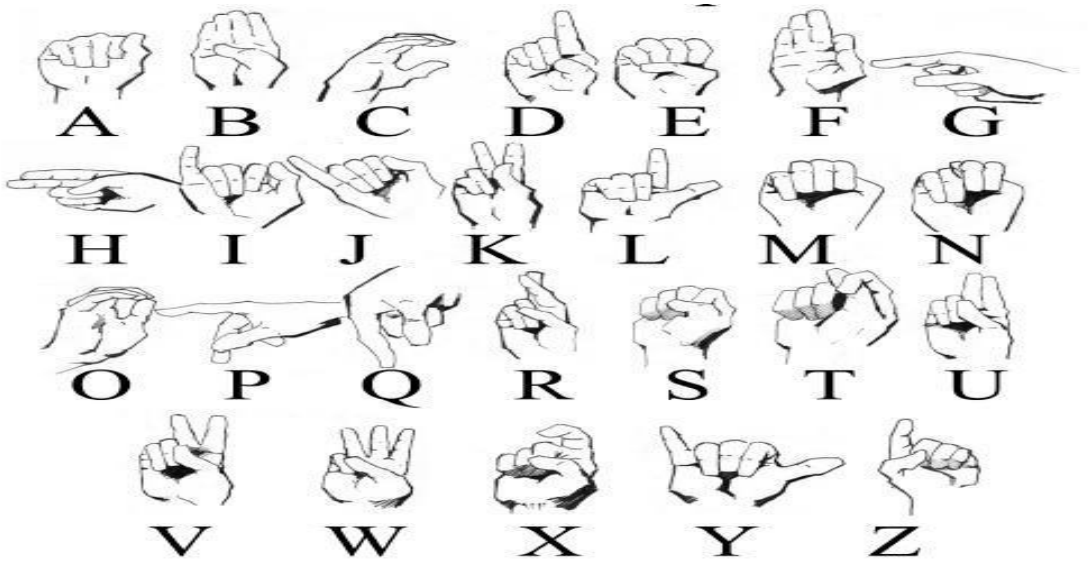
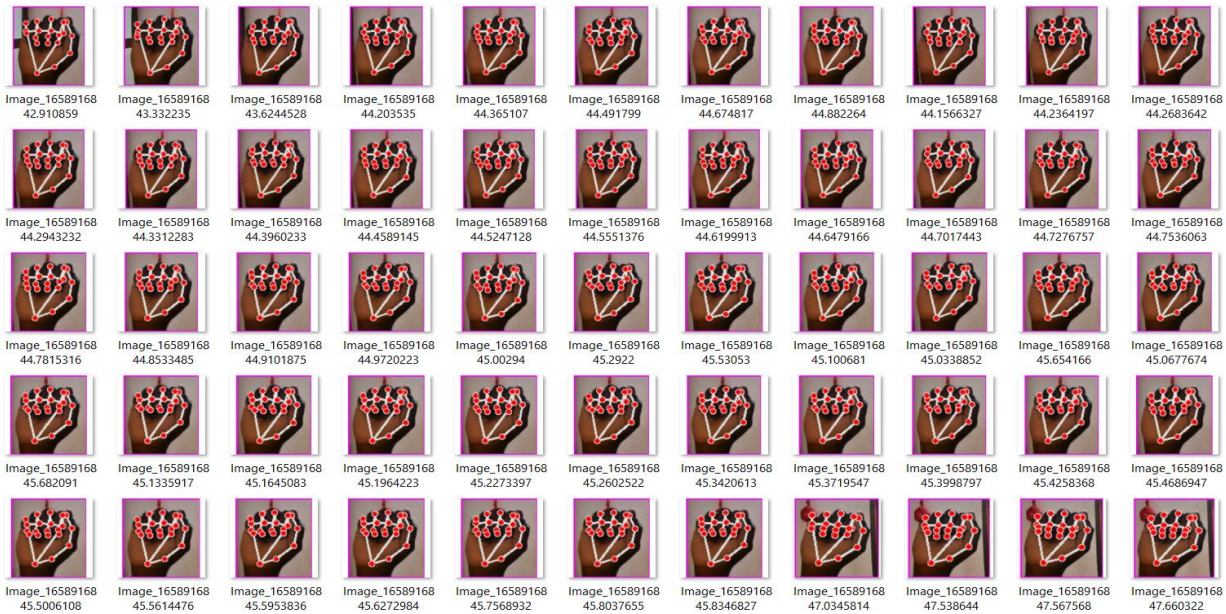


Fig3.3: Training data for A



3.2.4 SEGMENTATION

Image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as image objects). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyse. Modern image segmentation techniques are powered by deep learning technology. If, we take an example of Autonomous Vehicles, they need sensory input devices like cameras, radar, and lasers to allow the car to perceive the world around it, creating a digital map. Autonomous driving is not even possible without object detection which itself involves image classification/segmentation.

Image Segmentation involves converting an image into a collection of regions of pixels that are represented by a mask or a labelled image. By dividing an image into segments, you can process only the important segments of the image instead of processing the entire image. A common technique is to look for abrupt discontinuities in pixel values, which typically indicate edges that define a region. Another common approach is to detect similarities in the regions of an image. Some techniques that follow this approach are region growing, clustering, and thresholding.

3.2.5 CONVOLUTION NEURAL NETWORK

Image classification is the process of taking an input(like a picture) and outputting its class or probability that the input is a particular class. Neural networks are applied in the following steps:

1. One-hot Encoding
2. Defining the model
3. Compiling the model
4. Training the model
5. Testing the model

1. One-hot Encoding- A one-hot encoding can be applied to the integer representation. This is where the integer encoded variable is removed and a new binary variable is added for each unique integer value.

2.Defining the model- A model said in a very simplified form is nothing but a function that is used to take in certain input, perform certain operation to its best on the given input (learning and then predicting/classifying) and produce the suitable output.

3.Compiling the model- The optimizer controls the learning rate. We will be using ‘adam’ as our optimizer. Adam is generally a good optimizer to use for many cases. The adam optimizer adjusts the learning rate throughout training.

The learning rate determines how fast the optimal weights for the model are calculated. A smaller learning rate may lead to more accurate weights (up to a certain point), but the time it takes to compute the weights will be longer.

4.Training the model- Training a model simply means learning (determining) good values for all the weights and the bias from labelled examples. In supervised learning, a machine learning algorithm builds a model by examining many examples and attempting to find a model that minimizes loss; this process is called empirical risk minimization.

5.Testing the model- A convolutional neural network convolves learned featured with input data and uses 2D convolution layers.

Convolution Operation- In purely mathematical terms, convolution is a function derived from two given functions by integration which expresses how the shape of one is modified by the other.

Convolution formula

$$(f * g)(t) = \int \int_{-\infty}^{\infty} f(\tau) g(t - \tau) \mathrm{d}\tau$$

Here are the three elements that enter into the convolution operation:

- Input image
- Feature detector
- Feature map

Steps to apply convolution layer

- You place it over the input image beginning from the top-left corner within the borders you see demarcated above, and then you count the number of cells in which the feature detector matches the input image.
- The number of matching cells is then inserted in the top-left cell of the feature map
- You then move the feature detector one cell to the right and do the same thing. This movement is called a and since we are moving the feature detector one cell at time, that would be called a stride of one pixel.
- What you will find in this example is that the feature detector's middle-left cell with the number 1 inside it matches the cell that it is standing over inside the input image. That's the only matching cell, and so you write “1” in the next cell in the feature map, and so on and so forth.
- After you have gone through the whole first row, you can then move it over to the next row and go through the same process. There are several uses that we gain from deriving a feature map.

3.2.6 TESTING

The purpose of testing is to discover errors. Testing is a process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. Software testing is an important element of the software quality assurance and represents the ultimate review of specification, design and coding.

3.2.7 ALGORITHM:

Histogram Calculations- Histograms are collected counts of data organized into a set of predefined bins When we say data we are not restricting it to be intensity value. The data collected can be whatever feature you find useful to describe your image.

Back Propagation- Back-propagation is the essence of neural net training. It is the method of fine-tuning the weights of a neural net based on the error rate obtained in the previous epoch (i.e., iteration). Proper tuning of the weights allows you to reduce error rates and to make the model reliable by increasing its generalization. It is a standard method of training artificial neural networks. This method helps to calculate the gradient of a loss function with respects to all the weights in the network.

Optimizer (Adam)- Optimizer can be looked at as a combination of RMSprop(Root Mean Square Propagation) and Stochastic Gradient Descent with momentum. It uses the squared gradients to scale the learning rate like RMSprop and it takes advantage of momentum by using moving average of the gradient instead of gradient itself like SGD with momentum. It is an adaptive learning rate method, which means, it computes individual learning rates for different parameters. Its name is derived from adaptive moment estimation.

Loss Function- Categorical cross entropy is a loss function that is used for single label categorization. This is when only one category is applicable for each data point. In other words, an example can belong to one class only.

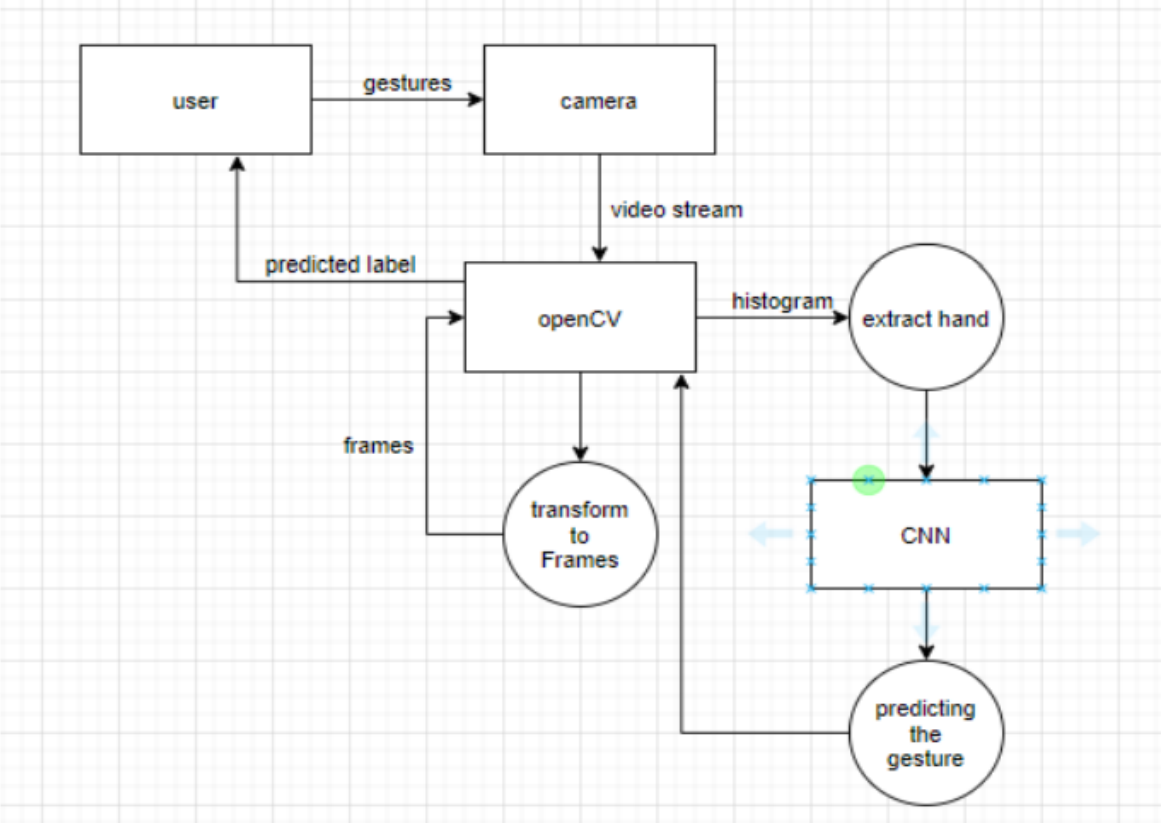
3.3 DATAFLOW DIAGRAM

The Data Flow Diagram (DFD) is also known as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generated by the system. It maps out the flow of information for

any process or system, how data is processed in terms of inputs and outputs. It uses defined symbols like rectangles, circles and arrows to show data inputs, outputs, storage points and the routes between each destination. They can be used to analyse an existing system or model of a new one. A DFD can often visually "say" things that would be hard to explain in words and they work for both technical and non- technical. There are four components in DFD:

1. External Entity
2. Process
3. Data Flow
4. data Store

Figure 3.4: Dataflow Diagram for Sign Language Recognition



3.4 USE CAE DIAGRAM

Use Case during requirement elicitation and analysis to represent the functionality of the system. Use case describes a function by the system that yields a visible result for an actor. The identification of actors and use cases result in the definitions of the boundary of the system i.e., differentiating the tasks

accomplished by the system and the tasks accomplished by its environment. The actors are outside the boundary of the system, whereas the use cases are inside the boundary of the system. Use case describes the behaviour of the system as seen from the actor's point of view. It describes the function provided by the system as a set of events that yield a visible result for the actor.

Purpose of Use Case Diagrams- The purpose of use case diagram is to capture the dynamic aspect of a system. However, this definition is too generic to describe the purpose, as other four diagrams (activity, sequence, collaboration, and state chart) also have the same purpose. We will look into some specific purpose, which will distinguish it from other four diagrams. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analysed to gather its functionalities, use cases are prepared and actors are identified. When the initial task is complete, use case diagrams are modelled to present the outside view. In brief, the purposes of use case diagrams can be said to be as follows

- Used to gather the requirements of a system.
- Used to get an outside view of a system.
- To identify the external and internal factors influencing the system.
- To show the interaction among the requirements are actors.

Use case diagrams are considered for high level requirement analysis of a system. When the requirements of a system are analyzed, the functionalities are captured in use cases. We can say that use cases are nothing but the system functionalities written in an organized manner. The second thing which is relevant to use cases are the actors. Actors can be defined as something that interacts with the system. Actors can be a human user, some internal applications, or may be some external applications. When we are planning to draw a use case diagram, we should have the following items identified. Functionalities to be represented as use case Actors Relationships among the use cases and actors. Use case diagrams are drawn to capture the functional requirements of a system. After identifying the above items, we have to use the following guidelines to draw an efficient use case diagram The name of a use case is very important. The name should be chosen in such a way so that it can identify the functionalities performed. Give a suitable name for actors. Show relationships and dependencies clearly in the diagram. Do not try to include all types of relationships, as the main purpose of the diagram is to identify the requirements. Use notes whenever required to clarify some important points.

Figure 3.5: Use case diagram of sign language recognition System

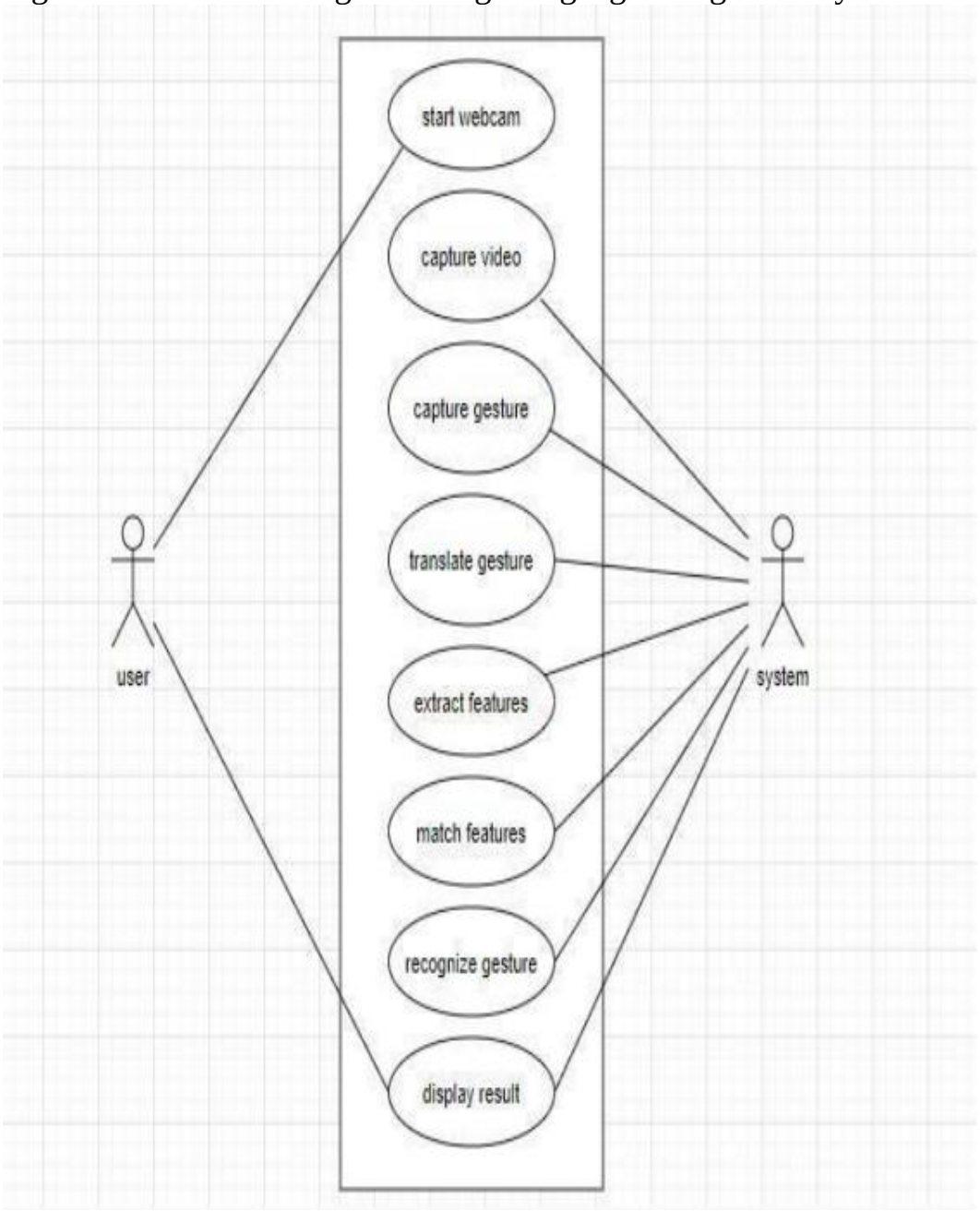


Table 3.1: Use case Scenario for sign language recognition system

Use case name	**Sign language recognition**
Participating actors	User, System
Flow of events	Start the system (user) Capturing video (system) Capture gestures (system) Translate gestures (system) Extract features (system) Match features (system) Recognizing gesture (system) Display result
Entry condition	Run the code
Exit condition	Displaying the label
Quality requirements	Cam pixels clarity, good light combination

3.5 SEQUENCE DIAGRAM

Sequence diagram displays the time sequence of the objects participating in the interaction. This consists of the vertical dimension(time) and horizontal dimension (different objects

Object can be viewed as an entity at a particular point in time with specific value and as a holder of identity. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

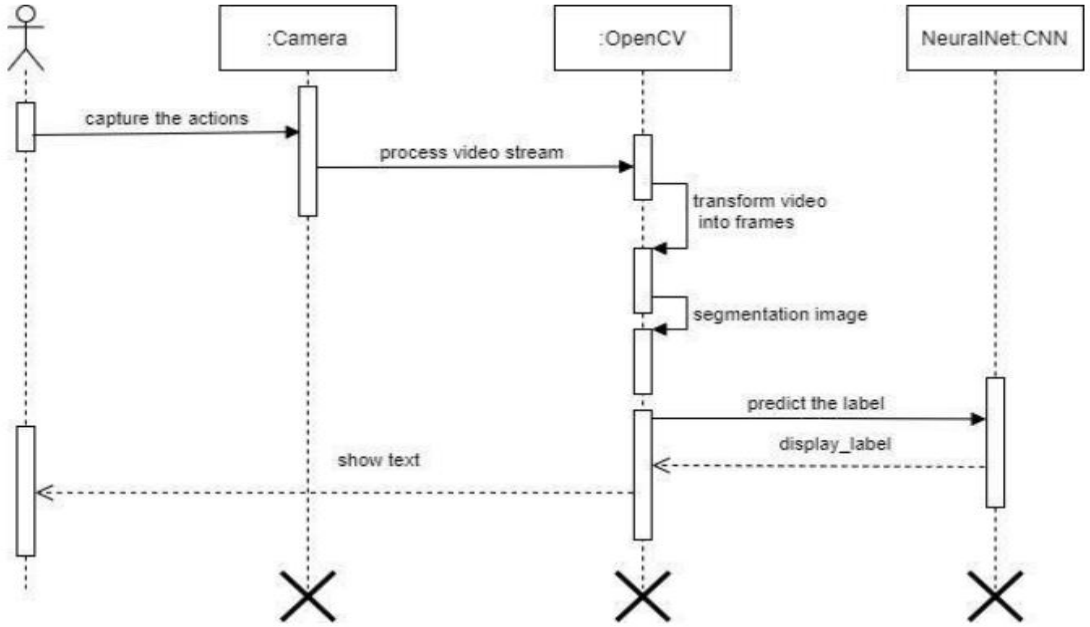
Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner. If the lifeline is that of an object, it demonstrates a role. Leaving the instance name blank can represent anonymous and unnamed instances. Messages, written with horizontal arrows with the message name written above them, display interaction.

Solid arrow heads represent synchronous calls, open arrow heads represent asynchronous messages, and dashed lines represent reply messages. If a caller

sends a synchronous message, it must wait until the message is done, such as invoking a subroutine. If a caller sends an asynchronous message, it can continue processing and doesn't have to wait for a response.

Figure 3.6: Sequence diagram of sign language recognition system



3.6 STATE CHART

A state chart diagram describes a state machine which shows the behaviour of classes. It shows the actual changes in state not processes or commands that create those changes and is the dynamic behaviour of objects over time by modelling the life cycle of objects of each class. It describes how an object is changing from one state to another state. There are mainly two states in State Chart Diagram:

1. Initial State

2. Final-State.

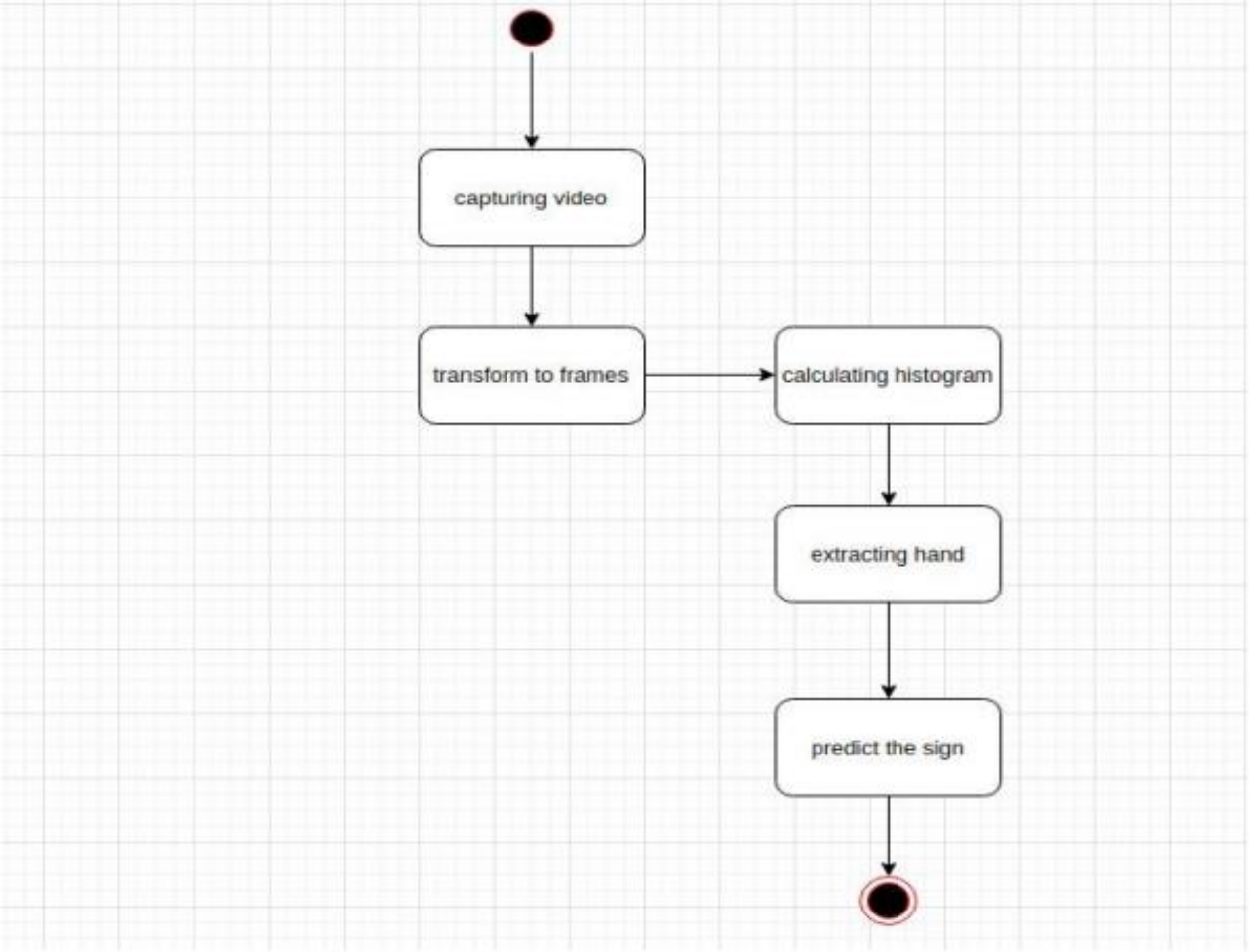
Some of the components of State Chart Diagram are:

1. State- It is a condition or situation in life cycle of an object during which it's satisfies same condition or performs some activity or waits for some event.

2. Transition- It is a relationship between two states indicating that object in first state performs some actions and enters into the next state or event.

3. Event- An event is specification of significant occurrence that has a location in time and space.

Figure 3.7: State Chart diagram of sign language recognition system



3.7 SOURCE CODE

3.7.1 CODE FOR DATA COLLECTION

```
import cv2
from cvzone.HandTrackingModule import HandDetector
import numpy as np
import math
import time
cap = cv2.VideoCapture(0)
detector = HandDetector(maxHands=1)
offset = 20
imgSize = 300
folder = "Data/C"
counter = 0
while True:
    success, img = cap.read()
    hands, img = detector.findHands(img)
    if hands:
        hand = hands[0]
        x, y, w, h = hand['bbox']
        imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) * 255
        imgCrop = img[y - offset:y + h + offset, x - offset:x + w + offset]
        imgCropShape = imgCrop.shape
        aspectRatio = h / w
        if aspectRatio > 1:
            k = imgSize / h
            wCal = math.ceil(k * w)
            imgResize = cv2.resize(imgCrop, (wCal, imgSize))
            imgResizeShape = imgResize.shape
            wGap = math.ceil((imgSize - wCal) / 2)
```

```
            imgWhite[:, wGap:wCal + wGap] = imgResize
        else:
            k = imgSize / w
            hCal = math.ceil(k * h)
            imgResize = cv2.resize(imgCrop, (imgSize, hCal))
            imgResizeShape = imgResize.shape
            hGap = math.ceil((imgSize - hCal) / 2)
            imgWhite[hGap:hCal + hGap, :] = imgResize
        cv2.imshow("ImageCrop", imgCrop)
        cv2.imshow("ImageWhite", imgWhite)
    cv2.imshow("Image", img)
    key = cv2.waitKey(1)
    if key == ord("s"):
        counter += 1
        cv2.imwrite(f'{folder}/Image_{time.time()}.jpg', imgWhite)
        print(counter)
```

3.7.1 CODE FOR EXECUTION:

```
import cv2
from cvzone.HandTrackingModule import HandDetector
from cvzone.ClassificationModule import Classifier
import numpy as np
import math
cap = cv2.VideoCapture(0)
detector = HandDetector(maxHands=1)
classifier = Classifier("Model2/keras_model.h5", "Model2/labels.txt")
offset = 20
imgSize = 300
counter = 0
```

```
labels = ["A", "B", "C","NONE"]
while True:
    success, img = cap.read()
    imgOutput = img.copy()
    hands, img = detector.findHands(img)
    if hands:
        hand = hands[0]
        x, y, w, h = hand['bbox']
        imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) * 255
        imgCrop = img[y - offset:y + h + offset, x - offset:x + w + offset]
        imgCropShape = imgCrop.shape
        aspectRatio = h / w
        if aspectRatio > 1:
            k = imgSize / h
            wCal = math.ceil(k * w)
            imgResize = cv2.resize(imgCrop, (wCal, imgSize))
            imgResizeShape = imgResize.shape
            wGap = math.ceil((imgSize - wCal) / 2)
            imgWhite[:, wGap:wCal + wGap] = imgResize
            prediction, index = classifier.getPrediction(imgWhite, draw=False)
            print(prediction, index)
        else:
            k = imgSize / w
            hCal = math.ceil(k * h)
            imgResize = cv2.resize(imgCrop, (imgSize, hCal))
            imgResizeShape = imgResize.shape
            hGap = math.ceil((imgSize - hCal) / 2)
            imgWhite[hGap:hCal + hGap, :] = imgResize
            prediction, index = classifier.getPrediction(imgWhite, draw=False)
```

```
        cv2.rectangle(imgOutput, (x - offset, y - offset - 50),
                      (x - offset + 90, y - offset - 50 + 50), (255, 0, 255), cv2.FILLED)
        cv2.putText(imgOutput,labels[index],(x,y-
26),cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)
        cv2.rectangle(imgOutput, (x - offset, y - offset),
                      (x + w + offset, y + h + offset), (255, 0, 255), 4)
        cv2.imshow("ImageCrop", imgCrop)
        cv2.imshow("ImageWhite", imgWhite)
    cv2.imshow("Image", imgOutput)
    cv2.waitKey(1)
```

CHAPTER 4
RESULT AND DISCUSSION

4.1 RESULT

The proposed sign language recognition system is performed with the help of convolutional neural networks is performed and the screenshots of results are given below in figure 4.1.

Figure 4.1: (a) Screenshot of letter A (b) Screenshot of letter B
(c) Screenshot of letter C (d) Screenshot of unknown sign

(a)

(b)

(c)

(d)

4.2 DISCUSSION

The proposed sign language recognition system used to recognize sign language letters can be further extended to recognize gestures facial expressions. Instead of displaying letter labels it will be more appropriate to display sentences as more appropriate translation of language. This also increases readability. The scope of different sign languages can be increased. More training data can be added to detect the letter with more accuracy. This project can further be extended to convert the signs to speech.

CHAPTER 5

CONCLUSION

Nowadays, applications need several kinds of images as sources of information for elucidation and analysis. Several features are to be extracted so as to perform various applications. When an image is transformed from one form to another such as digitizing, scanning, and communicating, storing, etc. degradation occurs. Therefore, the output image has to undertake a process called image enhancement, which contains of a group of methods that seek to develop the visual presence of an image. Image enhancement is fundamentally enlightening the interpretability or awareness of information in images for human listeners and providing better input for other automatic image processing systems. Image then undergoes feature extraction using various methods to make the image more readable by the computer. Sign language recognition system is a powerful tool to prepare an expert knowledge, edge detect and the combination of inaccurate information from different sources. The intend of convolution neural network is to get the appropriate classification

REFERENCES

[1] A. Kumar, K. Thankachan and M. M. Dominic, "Sign language recognition," 2016 3rd International Conference on Recent Advances in Information Technology (RAIT), 2016, pp. 422-428, doi: 10.1109/RAIT.2016.7507939.

[2] A. S. Nikam and A. G. Ambekar, "Sign language recognition using image based hand gesture recognition techniques," 2016 Online International Conference on Green Engineering and Technologies (IC-GET), 2016, pp. 1-5, doi: 10.1109/GET.2016.7916786.

[3] Suharjito, Ricky Anderson, Fanny Wiryana, Meita Chandra Ariesta, Gede Putra Kusuma, Sign Language Recognition Application Systems for Deaf-Mute People: A Review Based on Input-Process-Output, Procedia Computer Science,Volume 116, 2017, Pages 441-448, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2017.10.028.

[4] Bowden, R., Zisserman, A., Kadir, T., & Brady, M. (2003, April). *Vision based interpretation of natural sign languages.* Paper presented at the Third International Conference on Computer Vision Systems, Graz, Austria.

[5] Cox, S., Lincoln, M., Tryggvason, J., Nakisa, M., Wells, M., Tutt, M., et al. (2002, July). *TESSA, a system to aid communication with deaf people.* Paper presented at the Fifth International ACM Conference on Assistive Technologies, Edinburgh, Scotland.

[6] Saiteja Juluru , Bhavya Empati, Deepak Kallepalli , Anitha , “Sign Language Recognition using Open CV ”, Paper ID: SR22503220851 doi: 10.21275/SR22503220851