A PROJECT REPORT ON

SIGN LANGUAGE DETECTION TO TEXT CONVERSION

SUBMITTED BY:

SANIKA BHATYE (16)

RAVIKUMAR PANDEY (26)

SIDDHESH BANGAR (28)

NIREEKSHA POOJARI (32)

PROJECT GUIDE :

MRS. S.K. SHUKLA

SUBMITTED IN THE PARTIAL FULFILMENT OF REQUIREMENT FOR THE AWARD OF

DIPLOMA IN

COMPUTER ENGINEERING

2019 – 2020

VIDYA PRASARAK MANDAL'S POLYTECHNIC

CHENDANI BUNDER ROAD, THANE WEST – 400 601.

VPM’S POLYTECHNIC, THANE

DEPARTMENT OF COMPUTER ENGINEERING

VISION

- Provide quality diploma education in Computer Engineering and train students to effectively apply this education to solve real world problems, thus enhancing their potential for lifelong career and entitle them a competitive advantage in the ever changing global work environment.

MISSION

- Provide sound technical foundation in Computer Engineering through outcome based education and application oriented learning.
- Provide ambiance for professional growth and lifelong learning for adapting to challenges in rapidly changing technology.
- Inculcate social and ethical values.
- Nurture the interpersonal and entrepreneurial skills so as to develop leadership qualities in information industry’s diverse culture.

ACKNOWLEDGEMENT

Motivation and correct guidance are the keys towards success.

The completion of a project is a milestone in a student’s life and its’ execution is inevitable in the hands of guide. This project work has been the most practical and exciting part of our learning experience, which would be an asset for the future career.

We would like to extend our sincere thanks to all sources of motivation. We are highly indebted to our project guide and our HOD, **Mrs. S.K Shukla** for her encouragement, invaluable support, timely help, lucid suggestions and excellent guidance which helped us to understand and achieve our goals. We also give her our appreciation for giving form and substance to this report. It is due to her enduring efforts, patience and enthusiasm which has given a sense of direction and purposefulness to this project and ultimately made it a success.

We would like to express our gratefulness towards all the staff members of the Computer Engineering department for their noticeable co-operation.

We would also like to express our deep regards and gratitude to the Principal, **Dr. D.K Nayak.**

We further extend our appreciation to all professors and other non-teaching staff for their valuable tips during the designing of the project. Their contributions have been valuable in so many ways that we actually find it difficult to acknowledge and praise them individually.
We are also thankful to all those who helped us directly or indirectly in completion of this project work.

PREFACE

We have made sincere attempts and taken every care to present this matter in precise and compact form, the language being as simple as possible.

We are sure that the information contained in this volume would certainly prove useful for better insight in the scope and dimension of this project in its true perspective. Due acknowledgement has been made in the text to all other material used.

The task of completion of the project, though being difficult, was made quite simple, interesting and successful due to deep involvement and complete dedication of our group members:

SANIKA BHATYE

RAVIKUMAR PANDEY

SIDDHESH BANGAR

NIREEKSHA POOJARI

ABSTRACT

Sign Language Recognition is one of the most growing fields of research area. Many new techniques have been developed recently in this area. The Sign Language is mainly used for the communication of deaf - dumb people.

Our project, Sign Language Detection to Text Conversion mainly addresses to facilitate deaf and dumb person’s life style. Dumb and deaf people communicate with common people throughout the world using hand gestures. But common people face difficulty in understanding the gesture language. And hence, there exists a communication barrier between normal and hearing-impaired people. To overcome this barrier, Sign Language Detection System (SLDS) is developed. This is a user friendly, cost effective system which reduces communication gap between dumb and deaf individuals with ordinary people.

A camera attached to the computer will capture images of hand in the two detection areas and the contour feature extraction is used to recognize the hand gesture combinations of the person. Based on these recognized gestures, the relevant text will be displayed on the computer screen.

If this project is developed further, Text to Speech Conversion can also be made possible by including the **gTTS** (Google Text To Speech) module of Python in the SLDS.

Keywords: Sign Language Detection System (SLDS), Hearing impaired (People who can’t hear properly), Gesture to Text, Text to Speech conversion, Communication Barrier, gTTS.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER ONE -

INTRODUCTION

1.1 SIGN LANGUAGE DETECTION TO TEXT CONVERSION:

In our society, we have people with many disabilities. The technology is developing day by day, but no significant developments are taking place for the betterment of these impaired people. About a billion people in this world are deaf and dumb. Communication between a hearing disabled and a normal person has always been a challenging task. Sign language helps disabled people to converse with normal human beings.

Sign language to Text conversion is mainly focused on communication between ordinary people and deaf, mute people. Sign Language paves the way for the hearing/speaking disabled people to communicate. It's a visual language that is used by deaf and dumb individuals as their mother tongue. This language visually transmits sign patterns using hand shapes, movements of hands, arms or body to convey word meanings. It also uses different medium such as hands, face or eyes rather than the vocal track or even ears for the communication purpose.

Our main aim for the project is to allow people who are deaf or dumb to have normal conversations with other people through a program that inputs sign language in text form to allow the person to understand. The entire process will go vice versa. Our goal is to provide a basic program that translates numbers and alphabets into a Sign Language. If the program is elaborated after our Major Project, it can be a program that allows people to have free conversations irrespective of the person being disabled or not.

Our initial approach was to start off small and progress gradually. We first found an algorithm that could do the translation in both the ways. We then learnt the general Sign Language for numbers and then learnt the letters of the alphabets, so that we could finally translate certain words.

For instance, imagine you want to have a conversation with a deaf person then sign language has various gestures and body languages to convey messages as opposite to that of a verbal speech pattern. This initially may seem a tedious task, if you have no idea on how to communicate with fellow people using sign language. Such is the problem faced by millions of deaf individuals who are unable to communicate and interact with other normal people. For example, Gesture and Text Recognition System. Recognition of a Sign Language is very important for the engineering field and also for the whole society. The developments in technology hold the promise of providing solution for the hearing impaired to communicate with the society.

This project mainly analyses the visual data from the camera. A processing platform of Python Programming along with Open CV (Open Source Computer Vision) is used for recognizing the signs or gestures. These recognized gestures are further converted into text output. Image processing is the basic technique implemented in our project. Image Processing involves basic techniques such as blurring, masking along with the program logic.

Sign Language is further categorized according to regions like Indian, American, Chinese, Arabic and so on. Researches on hand gesture recognition, pattern recognition, image processing have been carried out by countries to improve the applications and bring them on to the next level.

Our project majorly describes an SLDS – Sign Language Detection System.

Any Hand Gesture Recognition System provides intelligent, natural and a convenient way of the human computing interaction. Developing a Sign Language System is generally based on the gesture control system. The gesture based system has two major applications – one : Sign Language Recognition (SLR) and two : which aims to interpret Sign Language (SL) automatically by a computer in order to help the deaf community to converse efficiently and conveniently.

1.2 EXISTING SYSTEM:

The system identifies your hand gestures and later converts them into text form. It consists of two detecting areas which take the user input of the combinations of hand gestures.

1.3 OBJECTIVES:

- Design, integrate, program, interface and test an SLDS – Sign Language Detection System.
- Integrate between the engineering knowledge and the needs of day-to-day life.
- To build SLDS for hearing impaired people.
- Apply the major engineering skills and use the engineering knowledge acquired from the study of Computer Engineering.
- Appreciate the importance of co-ordinated team work.
- Know that any technology gains its' importance from the value it adds to our lives, and from its' role in solving a problem or satisfying a need.

1.4 FEASIBILITY OF SIGN LANGUAGE DETECTION TO TEXT CONVERSION :

The feasibility of the project is analyzed in this phase and the business proposal is put forth with a very general plan for the project and some cost estimates. During the system analysis, the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, understanding of the major requirements of the system is essential.

Three key considerations involved in the feasibility analysis are :

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY:

It means whether a business or a project is feasible cost wise and logistically. Economists calculate economic feasibility by analyzing the costs and revenues a business would incur by undertaking a certain project. Our project is economically feasible. The hardware components used aren't much expensive, plus the software used while making this project are free and easy to download.

TECHNICAL FEASIBILITY:

The study is carried out to check the technical feasibility i.e. the technical requirements of the system. Our project system, Sign Language Detection to Text Conversion will take the input combinations of hand gestures from the user and will later convert them into text form. The corresponding output will later be displayed on the screen.

SOCIAL FEASIBILITY:

The aspect of this study is to check the level of acceptance of the system by the user. The SLDS will help the hearing and speaking impaired people to communicate with normal individuals properly, thus removing the communication barrier. Our main aim for the project is to allow people who are deaf or dumb to have normal conversation with other people through a program that inputs sign language in text form to allow the person to understand.

CHAPTER TWO - LITERATURE SURVEY

The Hand Gesture Recognition System provides intelligent, natural and convenient way of human computing interaction. Developing a Sing Language System is generally based on the gesture control system. This Sign Language system will only work on a Sign Language (SL) and will only give output based on the gestures. The Gesture based control system has two major applications - one is that Sign Language Recognition (SLR) and the other which aims to interpret the Sign Language (SL) automatically by a computer in order to help the deaf, dumb community conveniently. Sign Language being the highly structured and largely symbolic human gesture set, serves a very good basic development on the basis of general gestures. In this project report, we will be discussing about the Signs Detection System and how to analyse the hand gesture motions.

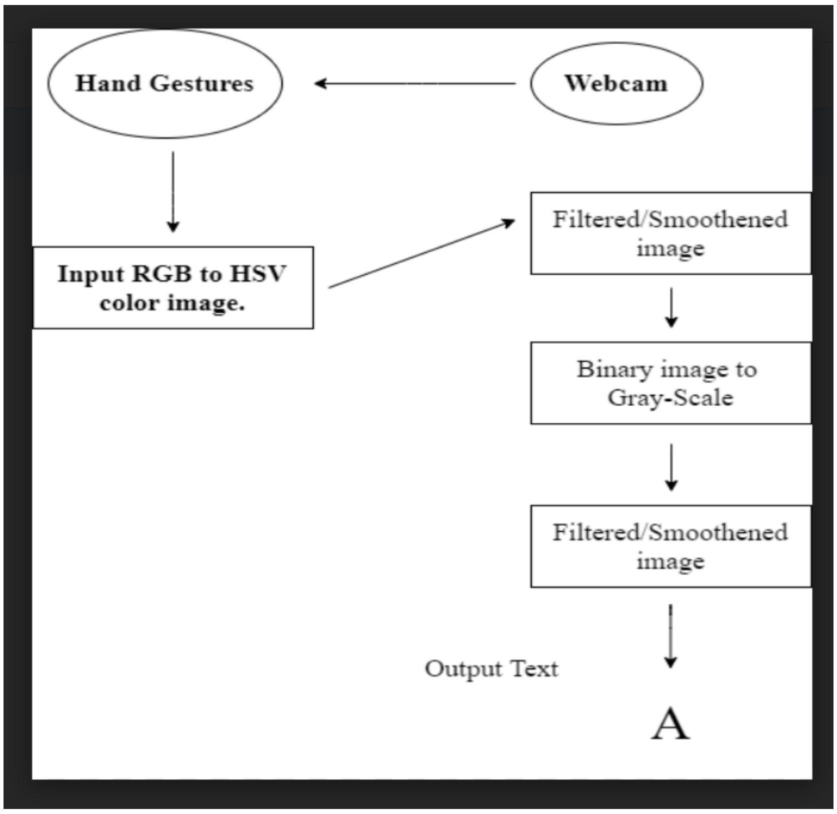

Hand Gestures
Webcam
Input RGB to HSV color image.
Filtered/Smoothened image
Binary image to Gray-Scale
Filtered/Smoothened image
Output Text
A

Figure 1. Working flow of SLDS

2.1 PROBLEM DEFINITION :

Communication between deaf-mute and a normal person has always been a challenging task. Sign language helps disabled people to converse with normal beings.

Our goal is to provide with a basic program that translates numbers and alphabets in a Sign Language. It inputs the gesture combinations through the detecting areas and correspondingly shows the text output. If our program is elaborated after our project, it can be a program that allows people to have a free conversation irrespective of the person being disabled or not.

The Sign Language Detection System (SLDS) would convert from standard SL into texts. This must be multi-user application for dumb and deaf, to include many forms of sign language and gesturing. If further developments are made in the project, this system can later convert text output in a speech format. One needs to correctly use the function/module of gTTS (Google Text To Speech) for that.

Sign language is a non-verbal language used by the hearing-impaired people for everyday communication among themselves. It is not just a random collection of gesture; it is a full-blown language in its' own right, complete with its own grammatical rules.

CHAPTER THREE -

SCOPE OF THE PROJECT

3.1 SCOPE:

The Sign Language Recognition System(s) are HCI (Human – Computer Interaction) based that are designed to enable engaging and effective interaction. It is a collaborative research area involving Pattern Matching, Computer Vision, Natural Language Processing, etc.

The SLDS has a wide range of applications and scope namely,

- Provides significant help for people with disabilities.
- Safest method to communicate with dumb and deaf people.
- Touch free interface is fun and provides powerful response.
- Non-Verbal Efficient Communication.
- Image Recognition and Motion Sensing.
- Image Retrieval, Image Sharpening and Restorations (To create a better image).
- Modeling of Sign Language Recognition System.

CHAPTER FOUR -

METHODOLOGY

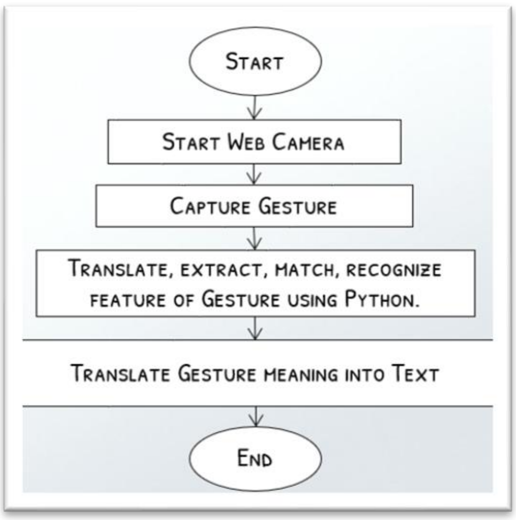

START
START WEB CAMERA
CAPTURE GESTURE
TRANSLATE, EXTRACT, MATCH, RECOGNIZE FEATURE OF GESTURE USING PYTHON.
TRANSLATE GESTURE MEANING INTO TEXT
END

Figure 2. Working flow of SLDS (2)

Sign language is a non-verbal language used by the hearing-impaired people for everyday communication among themselves. It is not just a random collection of gesture, but a full-blown language in its' own right, complete with its own grammatical rules.

Our SLDS is one such system consisting of the embedded combinations of the Sign Language, that further detects those gestures into a text format.

Using our SLDS is very easy. One first needs a good quality monitor for displaying the detecting areas, contours, and eventually the output. Keyboard and Mouse will also be required to operate this system. A Web camera will also be used for inputting the gestures.

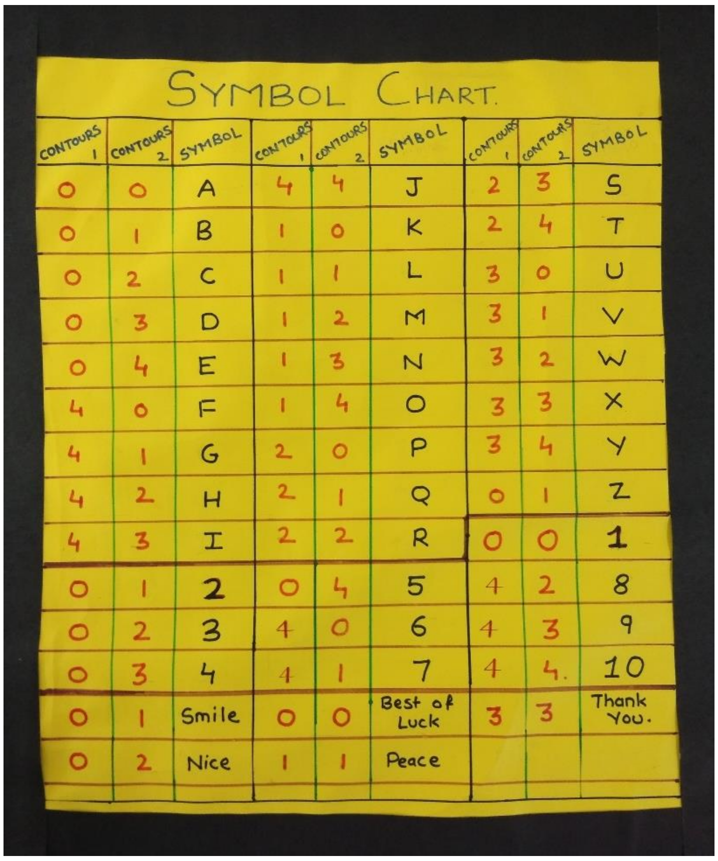
The main protagonist of the SLDS is the Raspberry Pi. Being an IOT device, it is a credit-card computer and does the work of a CPU in this project. The user simply needs to connect the Raspberry Pi to the monitor using an Adapter and HDMI cable.

Later, when the Raspberry Pi is successfully connected, one needs to insert an OS in order to access the device. In our case, we have used the Raspbian OS.

Once the connections are done, we can start with the actual working of the SLDS – Sign Language Detection System. When you run the program, the Webcam activates. Later, two detecting areas in the form of a square appear on the screen. These detecting areas contain 4 red dots, namely the Contours of the hand for the input of hand gestures.

Once the contours are detected, the focus is then shifted to the entire hand. This is known as the Convex Hull. Rest of the background is blurred and the focus remains only on one's hand. The algorithm of Gaussian Blur is used for making this possible.

Next is the part of the actual Sign Language. Below are the combinations used -



SYMBOL CHART

CONTOURS 1	CONTOURS 2	SYMBOL	CONTOURS 1	CONTOURS 2	SYMBOL	CONTOURS 1	CONTOURS 2	SYMBOL
0	0	A	4	4	J	2	3	S
0	1	B	1	0	K	2	4	T
0	2	C	1	1	L	3	0	U
0	3	D	1	2	M	3	1	V
0	4	E	1	3	N	3	2	W
4	0	F	1	4	O	3	3	X
4	1	G	2	0	P	3	4	Y
4	2	H	2	1	Q	0	1	Z
4	3	I	2	2	R	0	0	1
0	1	2	0	4	5	4	2	8
0	2	3	4	0	6	4	3	9
0	3	4	4	1	7	4	4.	10
0	1	Smile	0	0	Best of Luck	3	3	Thank You.
0	2	Nice	1	1	Peace			

Figure 3. Actual chart of the input combinations

When the user inputs the combination of 0 and 0 in first and second detecting area respectively, one needs to press the key ‘M’ on the keyboard in order to view the output of ‘A’ on the screen. If the user needs a numeric output, he needs to press the key ‘N’ on the keyboard. For viewing a full-text output, the key ‘W’ is to be pressed.

Similar is the case for all the other combinations ranging in between the numbers of 1 to 4 (since the human hand has 4 contours).

Try it out by yourself, it’s fun!

BLOCK DIAGRAM:

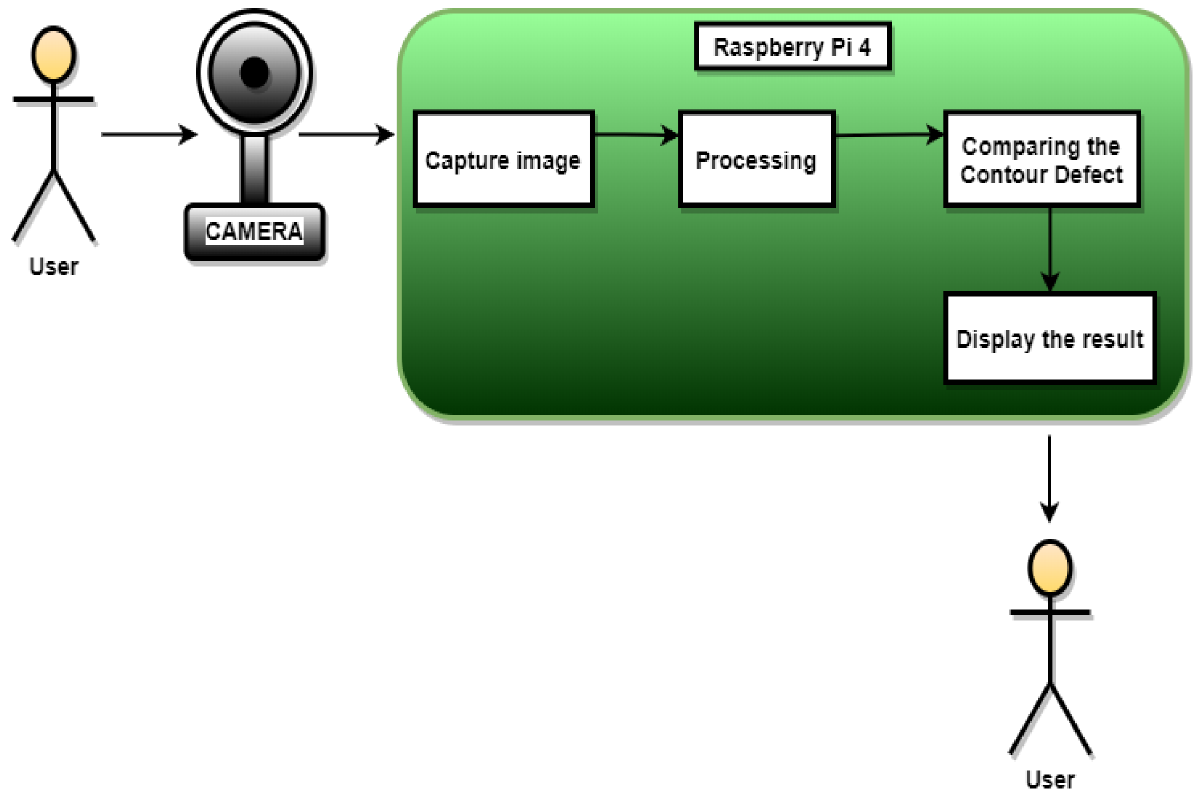


Figure 4. Block diagram of the proposed SLDS

CHAPTER FIVE -

DETAILS OF DESIGNS, WORKING AND PROCESSES

5.1 HARDWARE COMPONENTS USED:

(1) **Logitech C270 HD Webcam (Black)** - A Webcam is a video camera that streams an image or video in real time to or through a computer. We used the below camera to take the input of hand gestures.

Figure 5. Logitech C270 HD Webcam

(2) **Raspberry Pi 4** (Model B, 2GB RAM) – It is an IOT based component, which is low cost, credit-card sized computer having a dual-display that plugs into a Computer or TV screen, and further uses a Keyboard and Mouse. Our project, SLDS is entirely embedded into the Raspberry Pi. This is a base foundation of the whole system.

Figure 6. Raspberry Pi 4

(3) **Monitor, Mouse and Keyboard -** To display and operate all the recordings, images, motion captures of the mini Camera.

(4) **5V Power Supply Adapter -** To connect devices to each other.

Figure 7. Power Supply Adapter

(5) **HDMI Cable**

Figure 8. HDMI Cable

5.2 SOFTWARE USED:

(1) **Raspbian Jessie OS -** A Debian-based, 32 bit, Computer Operating System for Raspberry Pi.

Figure 9 Actual screen of the Raspbian OS

(2) **Open CV -** The Open Source Computer Vision is a library of programming functions. It was originally developed by Intel, and mainly aims at real-time computer vision. We have used this for Image Processing.

OpenCV

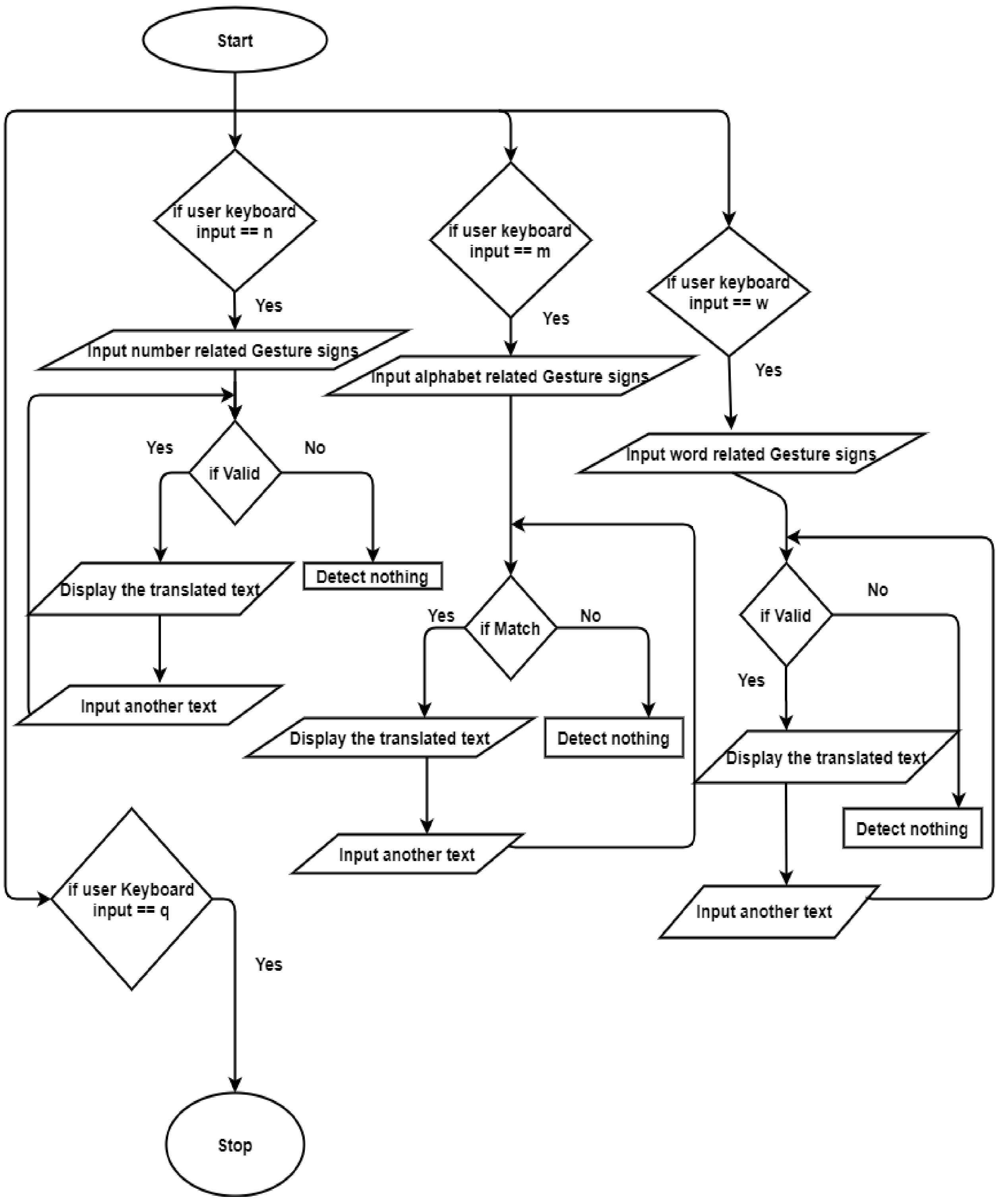
Figure 10. Logo of Open CV

(3) **Python** – The base programming language for our SLDS. It is a high-level, general-purpose, interactive, interpreted, object-oriented, dynamic programming language.

python™

Figure 11. Logo of Python

FLOW CHART:



Start
if user keyboard input == n
Yes
Input number related Gesture signs
if user keyboard input == m
Yes
Input alphabet related Gesture signs
if user keyboard input == w
Yes
Input word related Gesture signs
Yes
if Valid
No
Display the translated text
Detect nothing
Input another text
Yes
if Match
No
Display the translated text
Detect nothing
Input another text
if Valid
No
Yes
Display the translated text
Detect nothing
Input another text
if user Keyboard input == q
Yes
Stop

Figure 12. Flowchart of the entire SLDS

USE CASE DIAGRAM:

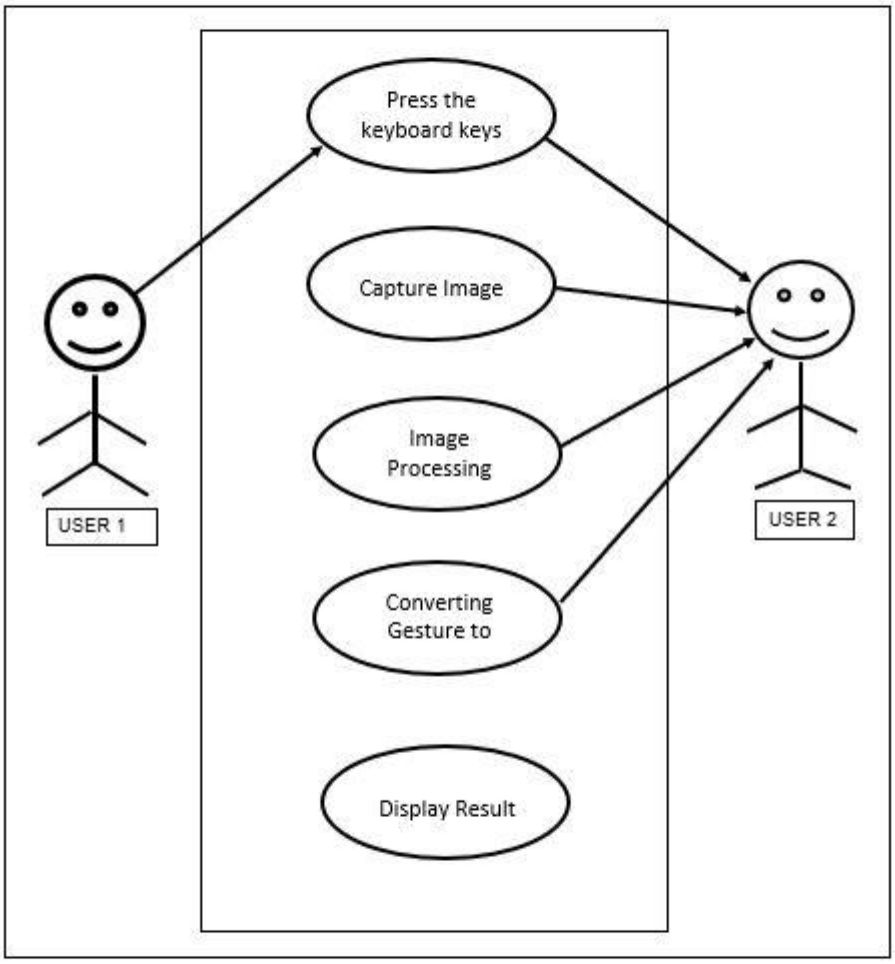


Figure 13. Use case diagram

DATA FLOW DIAGRAM:

Level 0:

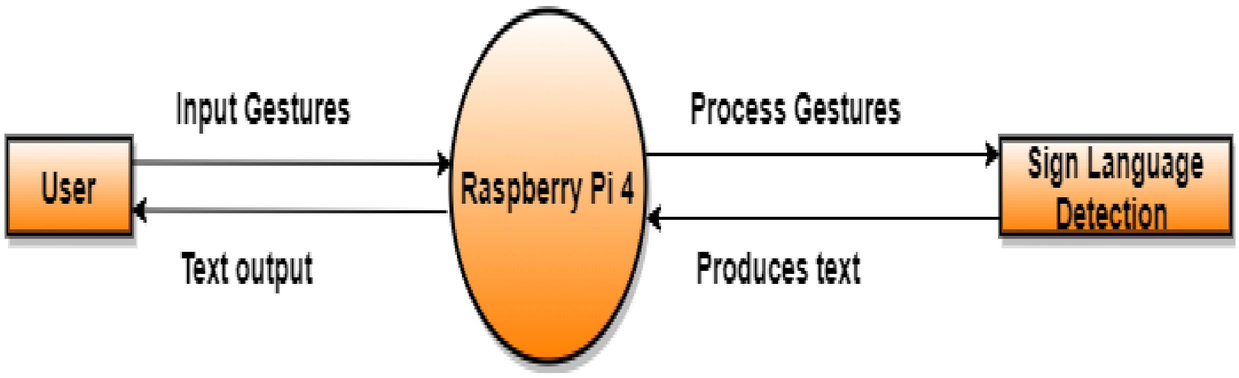


Figure 14. DFD Level 0

Level 1:

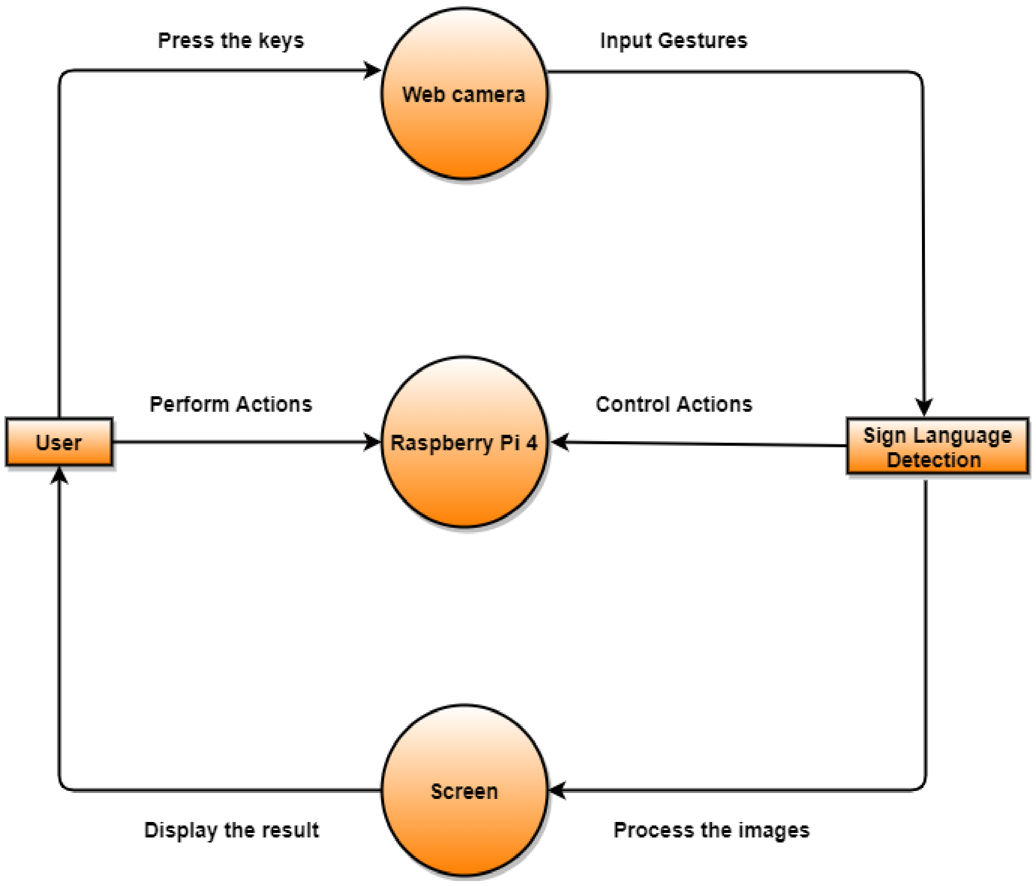


Figure 15. DFD Level 1

SEQUENCE DIAGRAM:

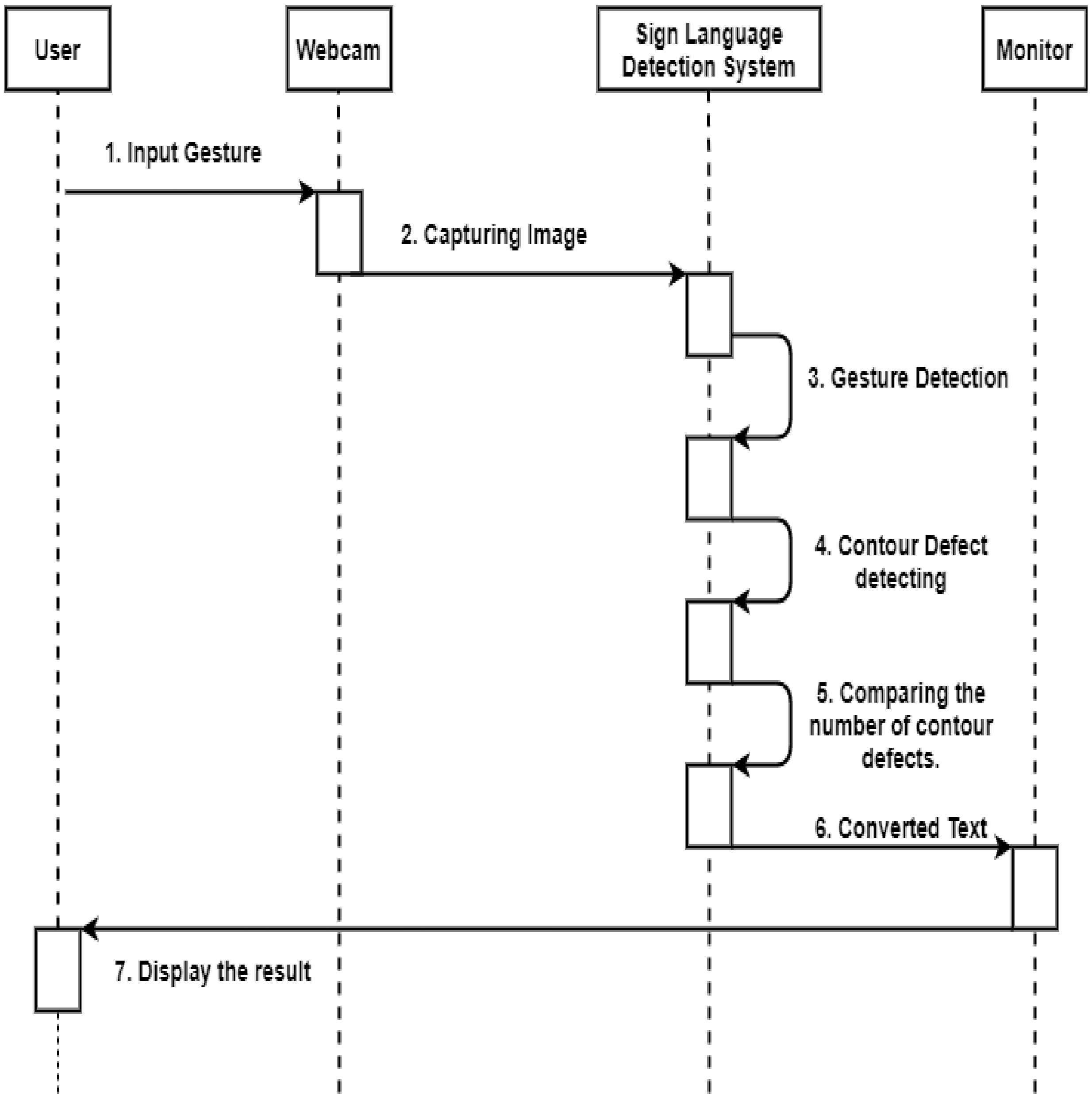


Figure 16. Sequence diagram of SLDS

5.3 ALGORITHM:

```
1. Start the program.
2. If the keyboard input == n, then go to step 2.1
   2.1.  Input Number related gestures.
   2.2.  If the input is matched then go to step 2.3.
   2.3.  Display the translated text.
   2.4.  Input another text if required and go to step 2.2.
3. If the keyboard input == m, then go to step 3.1
   3.1.  Input Number related gestures.
   3.2.  If the input is matched then go to step 3.3.
   3.3.  Display the translated text.
   3.4.  Input another text if required and go to step 3.2.
4. If the keyboard input == w, then go to step 4.1
   4.1.  Input Number related gestures.
   4.2.  If the input is matched then go to step 4.3.
   4.3.  Display the translated text.
   4.4.  Input another text if required and go to step 4.2.
5. If the keyboard input == q, then go to step 6.
6. Stop the program.
```

5.4 PROJECT CODE:

```
# -*- coding: utf-8 -*-
"""
Created on Fri Jan 17 11:36:37 2020

@author: User
"""

# -*- coding: utf-8 -*-
"""
Created on Fri Jan 10 11:35:17 2020

@author: User
"""
import cv2
import numpy as np
import math

cap = cv2.VideoCapture(0)
while 1:
    # read image
    ret, img = cap.read()
    ret1, img1 = cap.read()

    val=(300,300)
    val1=(100,100)

    # get hand data from the rectangle sub window on the screen
    cv2.rectangle(img, val, val1, (0,255,0),0)
    crop_img = img[100:300, 100:300]

    cv2.rectangle(img, (500,300), (300,100), (0,255,0),0)
    crop_img1 = img[100:300, 300:500]

    # convert to grayscale
    grey = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
    grey1= cv2.cvtColor(crop_img1, cv2.COLOR_BGR2GRAY)

    # applying gaussian blur
    value = (35, 35)
    blurred = cv2.GaussianBlur(grey, value, 0)
    blurred1 = cv2.GaussianBlur(grey1, value, 0)
    # thresholdin: Otsu's Binarization method
    ret, thresh1 = cv2.threshold(blurred, 127,
255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
    ret1, thresh2 = cv2.threshold(blurred1, 127,
255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)

    # show thresholded image
    cv2.imshow('Thresholded', thresh1)
    cv2.imshow('Thresholded2',thresh2)

    # check OpenCV version to avoid unpacking error
    image, contours, hierarchy = cv2.findContours(thresh1.copy(),cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)
    max_area = -1
    for i in range(len(contours)):
      cnt=contours[i]
      area = cv2.contourArea(cnt)
      if(area>max_area):
          max_area=area
          cl=i

    image1, contours1, hierarchy1 = cv2.findContours(thresh2.copy(),cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)
    max_area1 = -1
    for l in range(len(contours1)):
       cnt1=contours1[l]
       area1 = cv2.contourArea(cnt1)
       if(area1>max_area1):
          max_area1=area1
          cl1=l

    # find contour with max area
    cnt = contours[cl]
    cnt1 = contours1[cl1]

    # create bounding rectangle around the contour (can skip below two lines)
    x, y, w, h = cv2.boundingRect(cnt)
    cv2.rectangle(crop_img, (x, y), (x+w, y+h), (0, 0, 255), 0)
    cv2.rectangle(crop_img, (x, y), (x+w, y+h), (0, 0, 255), 0)

    x1, y1, w1, h1 = cv2.boundingRect(cnt1)
    cv2.rectangle(crop_img1, (x1, y1), (x1+w1, y1+h1), (0, 0, 255), 0)
```

```
cv2.rectangle(crop_img1, (x1, y1), (x1+w1, y1+h1), (0, 0, 255), 0)

# finding convex hull
hull = cv2.convexHull(cnt)
hull1 = cv2.convexHull(cnt1)

# drawing contours
drawing = np.zeros(crop_img.shape,np.uint8)
cv2.drawContours(drawing, [cnt], 0, (0, 255, 0), 0)
cv2.drawContours(drawing, [hull], 0,(0, 0, 255), 0)

drawing1 = np.zeros(crop_img1.shape,np.uint8)
cv2.drawContours(drawing1, [cnt1], 0, (0, 255, 0), 0)
cv2.drawContours(drawing1, [hull1], 0,(0, 0, 255), 0)

 #finding convex hull
hull = cv2.convexHull(cnt, returnPoints=False)
hull1 = cv2.convexHull(cnt1, returnPoints=False)

# finding convexity defects
defects = cv2.convexityDefects(cnt, hull)
count_defects = 0
cv2.drawContours(thresh1, contours, -1, (0, 255, 0), 3)

defects1 = cv2.convexityDefects(cnt1, hull1)
count_defects1 = 0
cv2.drawContours(thresh2, contours1, -1, (0, 255, 0), 3)

# applying Cosine Rule to find angle for all defects (between fingers)
# with angle > 90 degrees and ignore defects
for i in range(defects.shape[0]):
    s,e,f,d = defects[i,0]

    start = tuple(cnt[s][0])
    end = tuple(cnt[e][0])
    far = tuple(cnt[f][0])

    # find length of all sides of triangle
    a = math.sqrt((end[0] - start[0])**2 + (end[1] - start[1])**2)
    b = math.sqrt((far[0] - start[0])**2 + (far[1] - start[1])**2)
    c = math.sqrt((end[0] - far[0])**2 + (end[1] - far[1])**2)

    # apply cosine rule here
```

```
        angle = math.acos((b**2 + c**2 - a**2)/(2*b*c)) * 57

          #ignore angles > 90 and highlight rest with red dots
        if angle <= 90:
            count_defects += 1
            cv2.circle(crop_img, far, 1, [0,0,255], -1)
        #dist = cv2.pointPolygonTest(cnt,far,True)

        # draw a line from start to end i.e. the convex points (finger tips)
        # (can skip this part)
        cv2.line(crop_img,start, end, [0,255,0], 2)
        cv2.circle(crop_img,far,5,[0,0,255],-1)

    for l in range(defects1.shape[0]):
        s1,e1,f1,d1 = defects1[l,0]

        start1 = tuple(cnt1[s1][0])
        end1 = tuple(cnt1[e1][0])
        far1 = tuple(cnt1[f1][0])

        # find length of all sides of triangle
        a1 = math.sqrt((end1[0] - start1[0])**2 + (end1[1] - start1[1])**2)
        b1 = math.sqrt((far1[0] - start1[0])**2 + (far1[1] - start1[1])**2)
        c1 = math.sqrt((end1[0] - far1[0])**2 + (end1[1] - far1[1])**2)

        # apply cosine rule here
        angle1 = math.acos((b1**2 + c1**2 - a1**2)/(2*b1*c1)) * 57

        # ignore angles > 90 and highlight rest with red dots
        if angle1 <= 90:
            count_defects1 += 1
            cv2.circle(crop_img1, far1, 1, [0,0,255], -1)
        #dist = cv2.pointPolygonTest(cnt,far,True)

        # draw a line from start to end i.e. the convex points (finger tips)
        # (can skip this part)
        cv2.line(crop_img1,start1, end1, [0,255,0], 2)
        cv2.circle(crop_img,far,5,[0,0,255],-1)

    if cv2.waitKey(1)== ord('n'):
        if (count_defects1 == 0):
            cv2.putText(img,"1", (50, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
        elif (count_defects1 == 1):
            cv2.putText(img,"2", (50, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
        elif (count_defects1 == 2):
            cv2.putText(img,"3", (50, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
        elif (count_defects1 == 3):
            cv2.putText(img,"4", (50, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
        elif (count_defects1 == 4):
            cv2.putText(img,"5", (50, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)

        if (count_defects == 0 and count_defects1 == 4):
            cv2.putText(img,"6", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
        elif (count_defects == 1 and count_defects1 == 4):
            cv2.putText(img,"7", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
        elif (count_defects == 2 and count_defects1 == 4):
            cv2.putText(img,"8", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
        elif (count_defects == 3 and count_defects1 == 4):
            cv2.putText(img,"9", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
        elif (count_defects == 4 and count_defects1 == 4):
            cv2.putText(img,"10", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)

    if cv2.waitKey(1) == ord('w'):
        if (count_defects == 1 and count_defects1 == 1):
            cv2.putText(img,"Peace", (200, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
        elif (count_defects == 0 and count_defects1 == 0):
            cv2.putText(img,"Best of luck", (200, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
        elif (count_defects == 1):
            cv2.putText(img,"Smile", (200, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
        elif (count_defects == 2):
            cv2.putText(img,"Nice", (200, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
        elif (count_defects ==3 and count_defects1 == 3):
            cv2.putText(img,"Thank you", (200, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)

    if cv2.waitKey(1) == ord('m'):
        if (count_defects1 == 0 or count_defects == -1):
            cv2.putText(img,"A", (50, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
        elif (count_defects1 == 1):
            cv2.putText(img,"B", (50, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
        elif (count_defects1 == 2):
            cv2.putText(img,"C", (50, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
        elif (count_defects1 == 3):
            cv2.putText(img,"D", (50, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
        elif (count_defects1 == 4):
            cv2.putText(img,"E", (50, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
```

```
if (count_defects == 4 and count_defects1 == 0):
    cv2.putText(img,"F", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
elif (count_defects == 4  and count_defects1 == 1):
    cv2.putText(img,"G", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
elif (count_defects == 4 and count_defects1 == 2):
    cv2.putText(img,"H", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
elif (count_defects == 4 and count_defects1 == 3):
    cv2.putText(img,"I", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
elif (count_defects == 4 and count_defects1 == 4):
    cv2.putText(img,"J", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)

if (count_defects == 1 and count_defects1 == 0):
    cv2.putText(img,"K", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
elif (count_defects == 1 and count_defects1 == 1):
    cv2.putText(img,"L", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
elif (count_defects == 1 and count_defects1 == 2):
    cv2.putText(img,"M", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
elif (count_defects == 1 and count_defects1 == 3):
    cv2.putText(img,"N", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
elif (count_defects == 1 and count_defects1 == 4):
    cv2.putText(img,"O", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)

if (count_defects == 2 and count_defects1 == 0):
    cv2.putText(img,"P", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
elif (count_defects == 2 and count_defects1 == 1):
    cv2.putText(img,"Q", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
elif (count_defects == 2 and count_defects1 == 2):
    cv2.putText(img,"R", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
elif (count_defects == 2 and count_defects1 == 3):
    cv2.putText(img,"S", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
elif (count_defects == 2 and count_defects1 == 4):
    cv2.putText(img,"T", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)

if (count_defects == 3 and count_defects1 == 0):
    cv2.putText(img,"U", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
elif (count_defects == 3 and count_defects1 == 1):
    cv2.putText(img,"V", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
elif (count_defects == 3 and count_defects1 == 2):
    cv2.putText(img,"W", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
elif (count_defects == 3 and count_defects1 == 3):
    cv2.putText(img,"X", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
elif (count_defects == 3 and count_defects1 == 4):
    cv2.putText(img,"Y", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)
```

```
        elif (count_defects == 0 and count_defects1 == 1):
            cv2.putText(img,"Z", (500, 50), cv2.FONT_HERSHEY_COMPLEX, 2, 2)

    # show appropriate images in windows
    cv2.imshow('Gesture', img)
    all_img = np.hstack((drawing, crop_img))
    cv2.imshow('Contours', all_img)
    cv2.imshow('Gesture', img)
    all_img1 = np.hstack((drawing1, crop_img1))
    cv2.imshow('Contours1', all_img1)

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

cap.release()
cap1.release()
cv2.destroyAllWindows()
```

IMPLEMENTATION OF THE PROJECT:

Phase	Task	Months							
		Jun	Jul	Aug	Oct	Nov	Dec	Jan	Feb
1	Problem Definition	▨	▨						
	Rigorous study & Analysis		▨	▨	▨	▨			
2	Project Planning				▨	▨			
	Designing				▨	▨	▨		
3	Implementation					▨	▨	▨	
	Testing					▨	▨	▨	▨
	Modification					▨	▨	▨	▨

Figure 17. Project Implementation

GANTT CHART: Depicts the schedule of completion of Project.

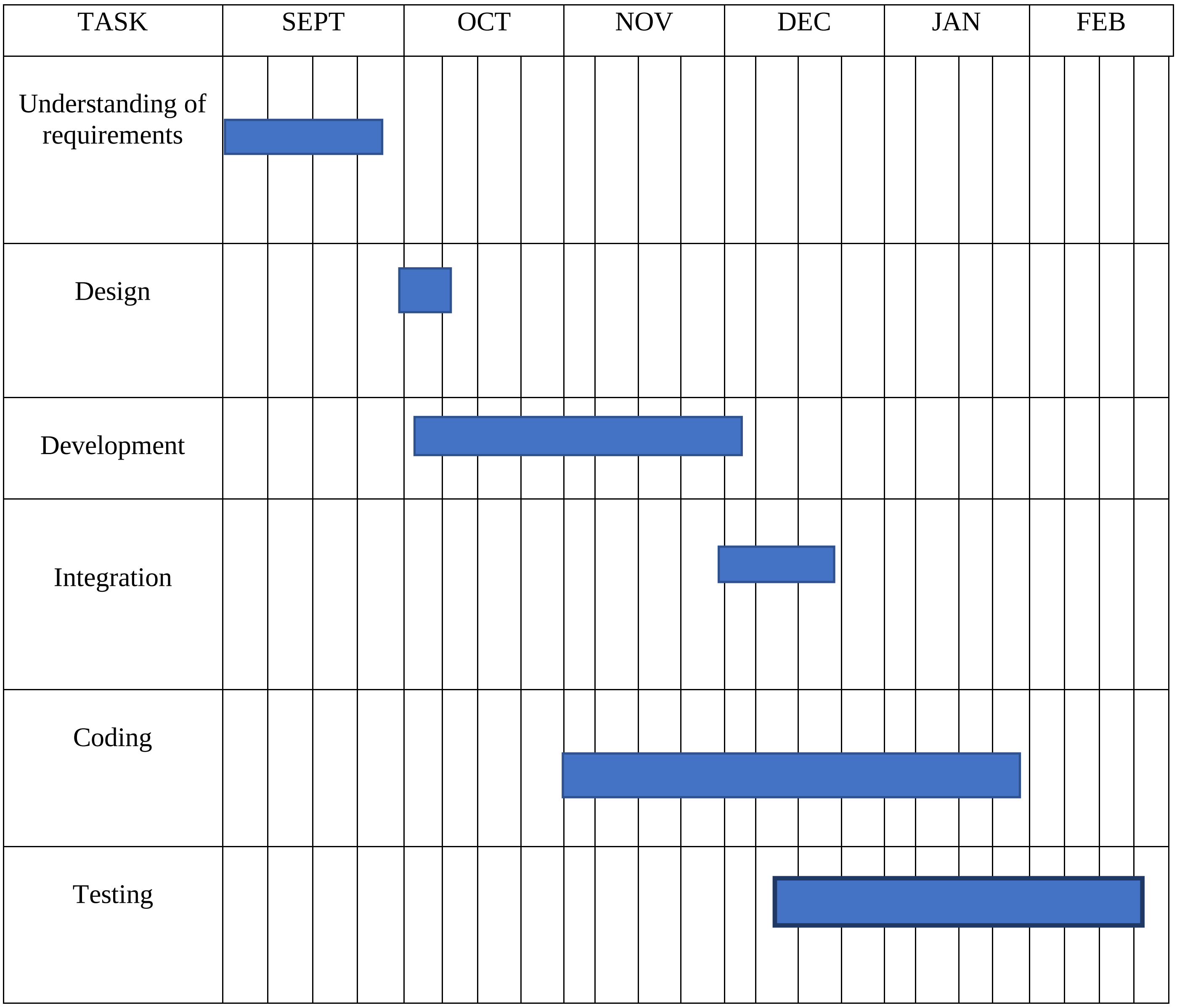


Figure 18. Gantt Chart

5.5 TEST CASES:

(TEST CASES BASED ON SOFTWARE MODULE)

Sr. No	Test Case ID	Test Case Name	Pre - Requisites	Steps	Input Data	Expected Result	Actual Result	Status
1.	TC-01	To Check whether the detecting area is available on the screen.	Raspbian OS, Raspberry Pi connected to monitor keyboard, Webcam and mouse.	Run the program on Raspberry pi's inbuilt IDE for python.	Python Program.	Detecting area should be displayed on the screen.	Detecting area is successfully displayed on the screen.	Pass.
2.	TC-02	To Check whether Hand Gestures are detected or not.	Raspbian OS, Raspberry Pi connected to monitor keyboard, Webcam and mouse.	Run the program on Raspberry pi's inbuilt IDE for python.	Python Program.	Hand Gestures should be detected.	Hand Gestures are successfully detected.	Pass.
3.	TC-03	To Check whether the Contours are detected and displayed on the hands or not.	Raspbian OS, Raspberry Pi connected to monitor keyboard, Webcam and mouse.	Run the program on Raspberry pi's inbuilt IDE for python.	Python Program.	Contours should be detected and displayed on the hands on the Output Screen.	Contours are detected and displayed on the hands on the Output Screen.	Pass.

Sr. No	Test Case ID	Test Case Name	Pre - Requisites	Steps	Input Data	Expected Result	Actual Result	Status
4.	TC-04	To Check whether Thresholding is generated on a different window or not.	Raspbian OS, Raspberry Pi connected to monitor keyboard, Webcam and mouse.	Run the program on Raspberry pi's inbuilt IDE for Python.	Python Program.	Thresholding should be generated on a different window.	Thresholding is generated on a different window.	Pass.
5.	TC-05	To Check whether Convexity algorithm is working or not i.e. whether the program detects the gestures even if the hands are in motion inside the detecting area.	Raspbian OS, Raspberry Pi connected to monitor keyboard, Webcam and mouse.	Run the program on Raspberry pi's inbuilt IDE for python.	Python Program.	The Convexity algorithm should work correctly in the program i.e. Program should detect the gestures even when user is moving his hands in motion.	The Convexity algorithm works correctly i.e. the program detects the gestures even when the user is moving his hands in motion.	Pass.
6..	TC-06	To Check whether Gesture to Text Conversion is displayed or not.	Raspbian OS, Raspberry Pi connected to monitor keyboard, Webcam and mouse.	Run the program on Raspberry pi's inbuilt IDE for python.	Python Program.	Gesture to Text Conversion should be displayed on the top left or right side of the output screen.	Gesture to Text Conversion is successfully done.	Pass.

Sr. No	Test Case ID	Test Case Name	Pre - Requisites	Steps	Input Data	Expected Result	Actual Result	Status
7.	TC-07	To Check whether the combination of 5 & 1 shows 6 while pressing the button 'N'.	Raspbian OS, Raspberry Pi connected to monitor keyboard, Webcam, Open CV should get installed and mouse.	Run the program on Raspberry pi's inbuilt IDE for python.	Python Program.	Number Output 6 should be displayed on the Screen.	Number Output 6 is displayed on the Screen.	Pass.
8.	TC-08	To Check whether the combination of 0 & 0 shows 'A' while pressing button 'M'.	Raspbian OS, Raspberry Pi connected to monitor keyboard, Webcam, Open CV should get installed and mouse.	Run the program on Raspberry pi's inbuilt IDE for python.	Python Program.	Text Output 'A' should be displayed on the screen.	'A' is displayed on the screen.	Pass.
9.	TC-09	To Check whether the Combination of 1 & amp; 1 show the output 'Peace'.	Raspbian OS, Raspberry Pi connected to monitor keyboard, Webcam, Open CV should get installed and mouse.	Run the program on Raspberry Pi's inbuilt IDE for python.	Python Program.	Text Output 'Peace' should be displayed on the screen.	Text Output 'Peace' is displayed on the screen.	Pass.

Sr. No	Test Case ID	Test Case Name	Pre - Requisites	Steps	Input Data	Expected Result	Actual Result	Status
10.	TC-10	To Check whether the program terminates while pressing ‘Q’.	Raspbian OS, Raspberry Pi connected to monitor keyboard, Webcam and mouse.	Run the program on Raspberry pi’s inbuilt IDE for python.	Python Program.	The program should be terminated.	The program doesn’t terminate.	Fail.

(TEST CASES BASED ON HARDWARE MODULE)

Sr. No	Test Case ID	Test Case Name	Pre - Requisites	Steps	Input Data	Expected Result	Actual Result	Status
1.	TC_ 01	To check whether Raspberry pi starts when Raspbian OS get loaded in Raspberry pi.	Raspbian OS, Raspberry Pi connected to Monitor Keyboard, Webcam, Open CV installed and mouse.	1.Download Raspbian OS in SD Card. 2.Attach SD Card to Raspberry Pie. 3. Connect Raspberry Pie to Power Supply and also connect it to Monitor and needed IO Devices.	Raspbian OS.	Raspberry should start and Raspbian OS installation process should get started.	Raspberry starts and Raspbian OS installation process should get started.	Pass.
2.	TC_ 02	To check whether Webcam get Access through Raspberry Pi after connecting to it Raspberry Pi.	Raspbian OS, Raspberry Pi connected to monitor keyboard, Webcam, Open CV installed, Mouse and Simple program to access webcam in Python.	1. Connect Raspberry Pie to Power Supply and also connect it to Monitor and needed IO Devices. 2.Now Connect Webcam to Raspberry Pi. 3. In Python IDE, type simple program to access webcam.	Raspbian OS, OpenCV, Simple Python Program to access Webcam.	Webcam should get accessed, A window should appear and should display the camera view.	Webcam get accessed, A window appeared and displayed the camera view.	Pass.

CHAPTER SIX -

RESULTS AND APPLICATIONS

6.1 OUTPUT SCREENS:

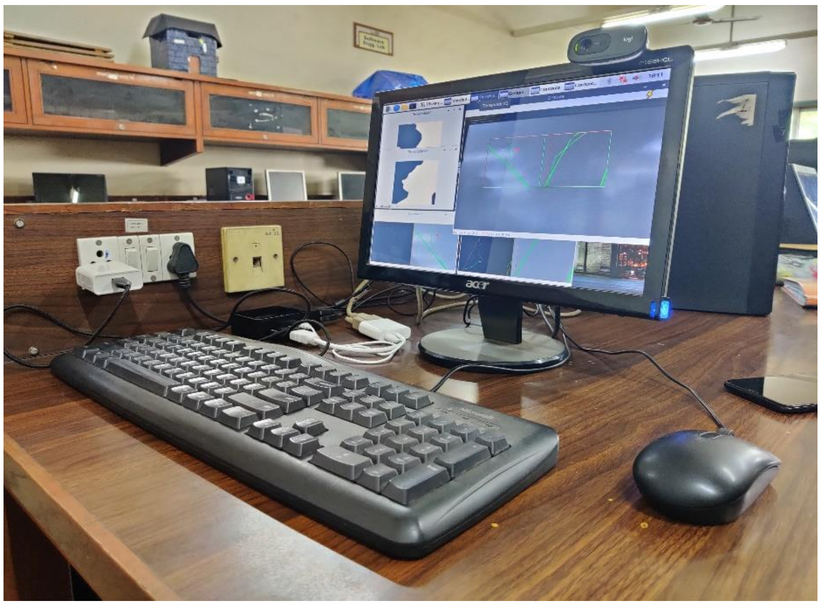


Figure 19. Setup of the SLDS, including the hardware and software parts

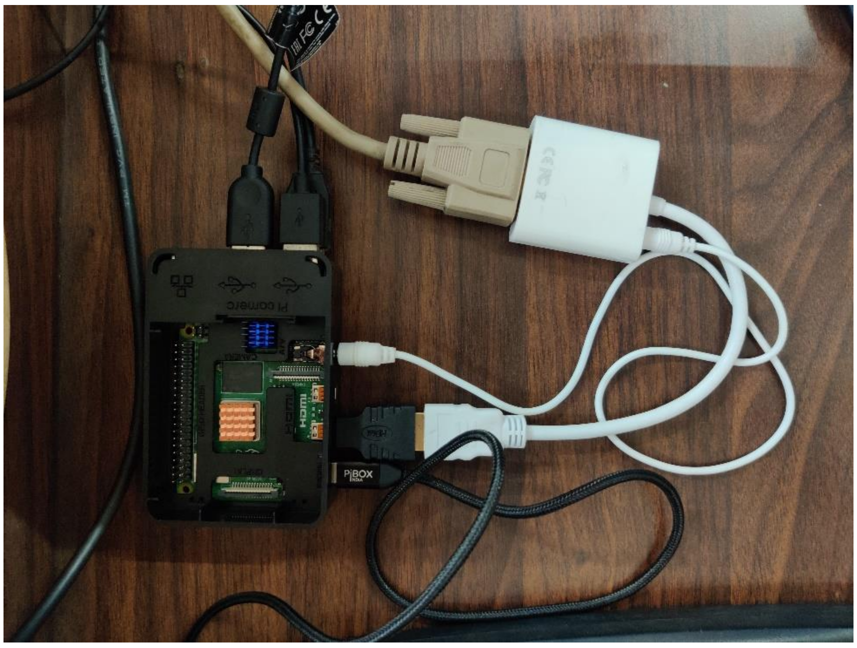


Figure 20. Connection of Raspberry Pi.

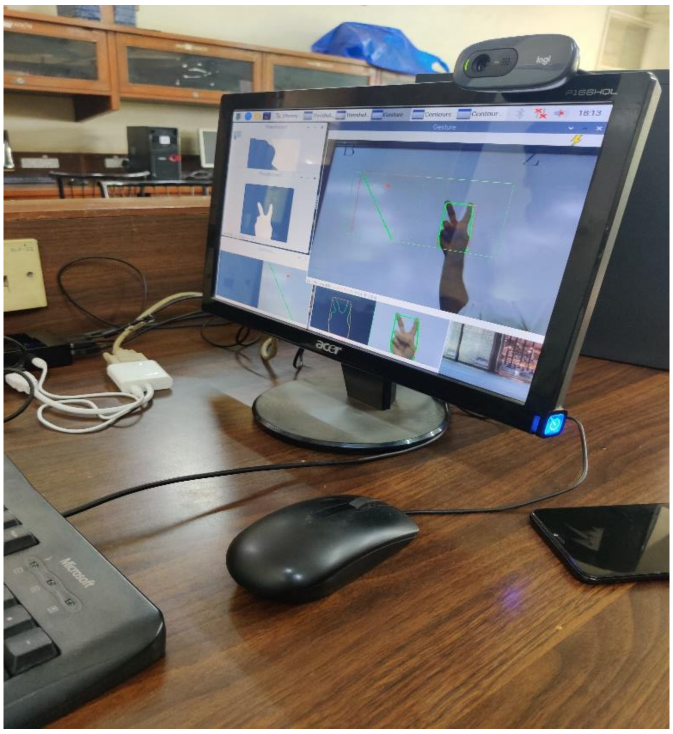


Figure 21. Detection Demonstration

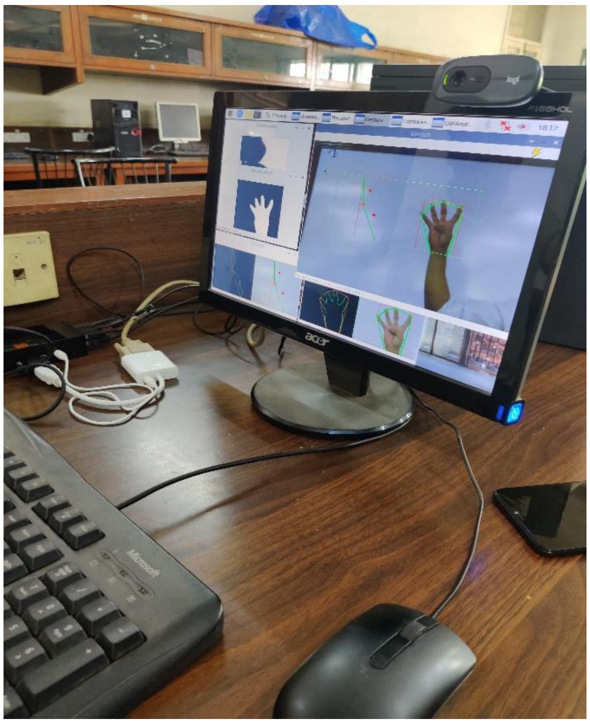
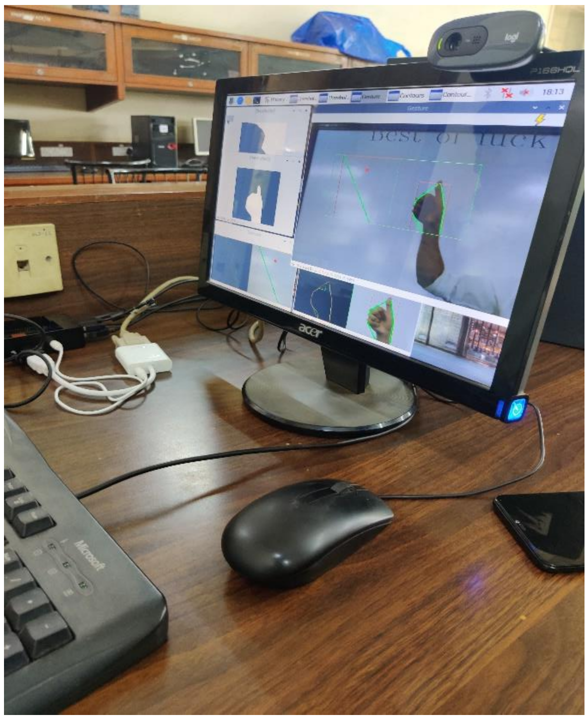



Figure 22. Detection Demonstration of words

The above pictures/Output screens depict the outputs of all Alphabetic, Numeric and Text format, when the corresponding hand gesture is inputted to the system.

6.2 APPLICATIONS OF SLDS:

1. Helpful Product for Hearing Disabled people –

   The Project we developed, i.e. ‘Signs Detection System’ coverts the Hand Gestures into simple symbols like alphabets, numbers and emotions (Best of luck, Smile, etc), which is really helpful for people who have hearing problems as we provide them the perfect GUI software project through which they can convert the signs used by them to understand the basics of Language visually.

2. Can Also be used as a Calculator –

   If modifications are done in the project, we can also use this project system as a calculator to manipulate and do some basic calculations with numbers like addition, subtraction, division and multiplication which can also be used in the scope of education to children.

3. Custom Gesture System –

   An Individual who is good in Python programming and Open CV field, can further modify the gestures used in the project and can make use of that gestures for his own purpose, whether it be commercial or personal use.

4. Hand Gestures recognition for Sign Language System –

   We are providing a static hand gesture recognition system using digital image processing as well as motion sensing which only detects Hand gestures and converts it into Sign Language symbols rather than using complex hand gestures. For this, we have provided fingers combination for the conversion.

5. Deaf mute Community interpreter –

   This project is build and designed to create a better mode of communication with and for the deaf community. Two major communication methods used by the deaf people are - wearable communication devices and another is the Online Learning system. Out of which, we have developed an easy-to-use SLDS for the betterment of these impaired people.

CHAPTER SEVEN -

CONCLUSIONS AND FUTURE SCOPE

Figure 23. A person using a recognition system

Our Sign Language Detection / Recognition System is developed and limited only to the classification of basic static numbers and alphabets, but if advancements are made in this, it can successfully recognize dynamic movements that come in continuous sequences of images.

This particular system only displays the Text output on the computer screen by inputting the gesture combinations onto the detecting areas. But, text to speech conversion can also be made possible by using the module of gTTS (Google Text To Speech). It powers applications to read the text on the screen aloud, with support for many languages.

Nowadays, the Researchers and Developers are paying more attention to make a large vocabulary for SDLS. Many of them are made using small vocabulary and self-made database. Talking in a global view, the SLDS is famous in many parts around the world, in developed and developing countries. The only variation is the sign language embedded. The SL varies from one country to another. All these advancements lead to only one thing : betterment in the life of deaf, mute people!

APPENDIX

- Modules used:

I. **NumPy** - Fundamental package for scientific computing with Python. It contains a powerful N-dimensional array object, tools for integrating codes of C, C++, Fortran languages, and many other sophisticated functions. Also useful for linear algebra.

II. **Math** – Standard module in Python. Always used for mathematical functions.

III. **VideoCapture (0)** – Used to capture or grab the video from the webcam.

IV. **Morphology** – It means a particular form, shape or structure. This function is used for image dilution and erosion.

- **Gaussian Blur** – Specially designed to remove high frequency of noise for image processing.

- **Contours -** An outline representing or bounding a shape or form of something. A way in which something varies.

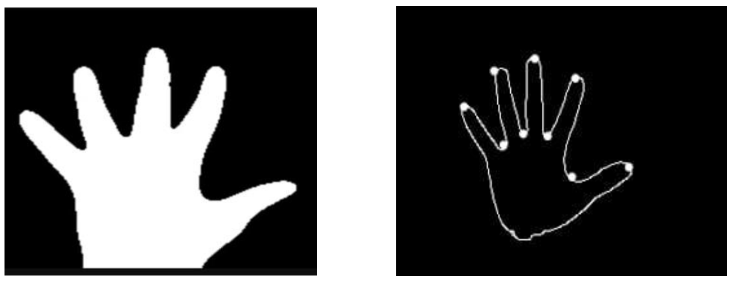


Figure 24. Contours of the human hand

- **Threshold** – A point beyond which there is a change in the manner of program execution, or the minimum input that produces a corrective action in a system.

- **Convexity –** This is one of the concepts used in the project. It is a measure of the curvature or the degree of the curve.

- **Convex Hull -** It is a mathematical form, the convex hull or convex envelope or convex closure of a set x of point (nothing but the contours). this convex hull bounds each and every point x to another one and detects the shape and outline of any object, that is nothing but our hand gestures.

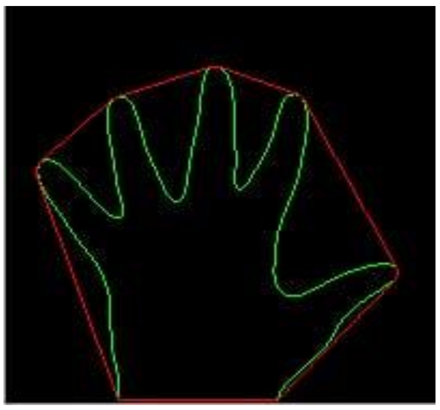


Figure 25. Depiction of the Convex Hull

- **RGB to HSV** – HSV stands for Hue, Saturation, Value. RGB stands Red, Green, Blue.

  RGB defines colours in terms of a combination of primary colours, on the other hand, HSV describes colours using more familiar comparisons like vibrancy, brightness, etc.

USER MANUAL

About SLDS -

The Sign Language Detection System (SLDS) is a product that is developed in Python Programming Language with the help of Open CV and NumPy libraries of Python.

It is made for the community of people who cannot listen or cannot see properly, and due to their disabilities, we developed this project for a better communication with these people. This product presents you output on the display screen as per the gestures and their detection.

Hardware Requirements -

- Keyboard and Mouse (Input devices)
- Display Monitor (Output device)
- HDMI Cable
- Web Camera
- Raspberry Pi (3 or 4)
- 5V Battery Charger (C Type)
- Micro SD Card (16 GB Recommended)

System Requirements -

To make this product flexible and remote, we developed it on Raspberry Pi, so make sure your Raspberry Pi meets these requirements:

RAM	256 MB (2 GB Recommended)
Available Disk Space	200 MB
Operating System	Raspbian OS or Noobs
Other Software/Libraries	Python and Open CV

Connect all input as well as output devices to the Raspberry Pi and insert the SD Card with Raspbian Operating System and the project program available in the card and once done with all the connections, provide the power supply to Raspberry Pi using 5V Battery Charger.

To Run the Product: Run the main program file using any Python IDE that is available in the Raspberry Pi OS (mainly Python IDE).

Getting Started with the Sign Language Detection System -

➢ Exploring the Output Screen:

Start the main program and once it gets compiled and executed, the output screens will be shown up in the monitor.

There will be almost 5 popup windows on the display monitor. They are:

1. **Main Output Window** – This is where all the detection of gestures, conversion and text output will be displayed, this window consists of two detecting areas (marked red in rectangle).
2. **Contours Window 1** – This window will show the Contours that are detected on the one detecting area from the main output window screen.
3. **Contours Window 2** – This window will show the Contours that are detected on the second detecting area from the main output window screen.
4. **Threshold Window 1** – This window will show the Thresholding which is developed from Contours that are detected on the one detecting area from the main output window screen.
5. **Threshold Window 2** – This window will show the Thresholding which is developed from Contours that are detected on the second detecting area from the main output window screen.

➢ Working with Sign Detection System:

Once you understand the user interface and all about the displaying content, you are ready to use the product on your own.

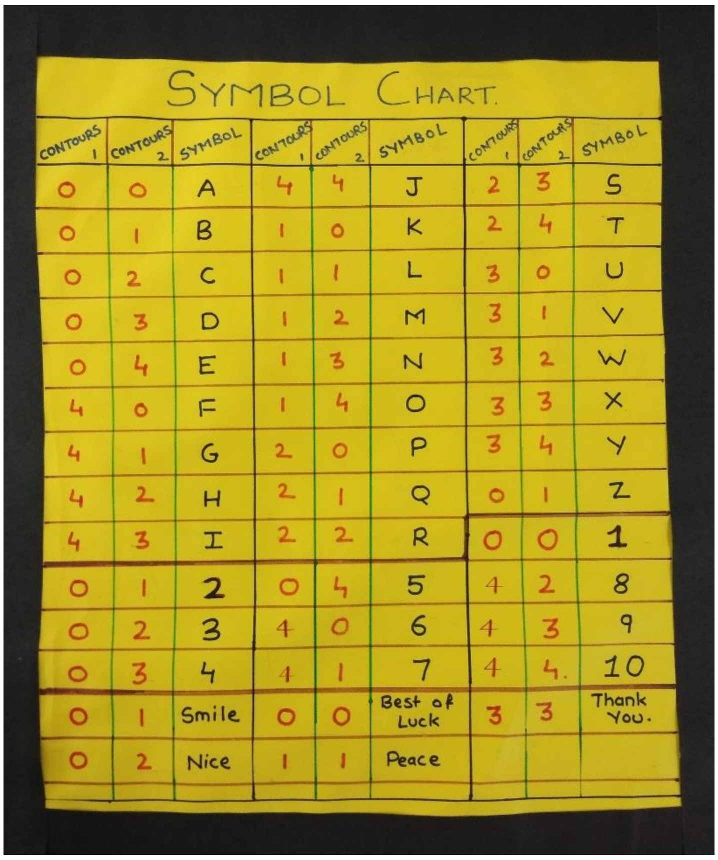
Procedure:

- This Product is very easy to use, just the thing you have to show some gestures, (finger gestures from one to five) one hand on one detecting area and another hand on the other detecting area.
- To ensure that your gestures are detected, you just have to look on the Contours windows as well as Threshold windows, to check whether your hand is successfully detected by the camera or not.
- Afterwards, you have to activate any one mode that we have provided in this product just by keep pressing the particular button as per the mode instructions explained below:

  Basically, we have provided three modes in this product for user experience:

  1. **Alphabetical Mode** – In this, the gestures will be converted and will only display letters on the Top left and Top right corner on the Main Output Window. To activate it, you have to keep pressing the Button 'M' on your keyboard.
  2. **Numbers Mode** – In this, the gestures will be converted and will only display numbers on the Top left and Top right corner on the Main Output Window. To activate it, you have to keep pressing the Button 'N' on your keyboard.
  3. **Words Mode** – In this, the gestures will be converted and will only display some words and phrases on the Top left and Top right corner on the Main Output Window. To activate it, you have to keep pressing the Button 'W' on your keyboard.

- To understand which gesture pattern, show which kind of Output. You can take the help of the chart below:



SYMBOL CHART

CONTOURS 1	CONTOURS 2	SYMBOL	CONTOURS 1	CONTOURS 2	SYMBOL	CONTOURS 1	CONTOURS 2	SYMBOL
0	0	A	4	4	J	2	3	S
0	1	B	1	0	K	2	4	T
0	2	C	1	1	L	3	0	U
0	3	D	1	2	M	3	1	V
0	4	E	1	3	N	3	2	W
4	0	F	1	4	O	3	3	X
4	1	G	2	0	P	3	4	Y
4	2	H	2	1	Q	0	1	Z
4	3	I	2	2	R	0	0	1
0	1	2	0	4	5	4	2	8
0	2	3	4	0	6	4	3	9
0	3	4	4	1	7	4	4.	10
0	1	Smile	0	0	Best of Luck	3	3	Thank You.
0	2	Nice	1	1	Peace			

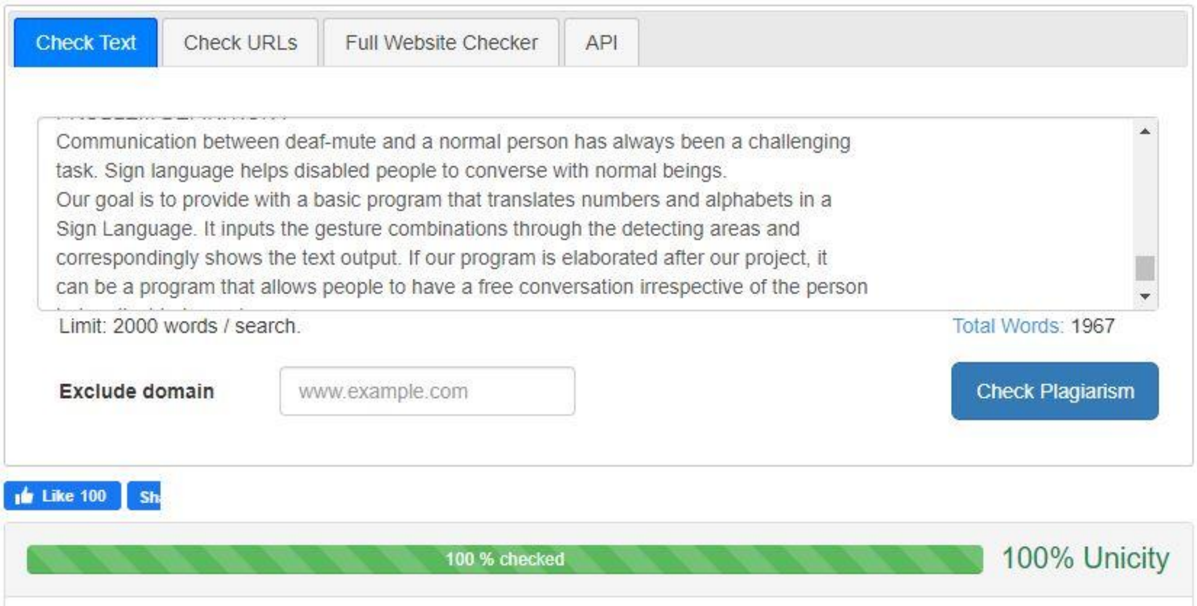
- Once you're done with the program, you can close it by pressing 'Q' on your Keyboard.

REFERENCES AND BIBLIOGRAPHY

Here’s the list of sources used as references:

- **www.sciencedirect.com**
- **www.ijser.org**
- **www.researchgate.net**
- **www.speechbuddy.com**
- **149_Real_IEEE.pdf**
- **Ext_01694.pdf**
- **www.csee.usf.edu.com**
- **www.towardsdatascience.com**
- **github.com**
- **Signing made easy: A complete program for learning and using sign language in everyday life** - By Rod R. Butterworth.
- **The Everything Sign Language Book** – By Irene Duke.

PLAGIARISM CHECK

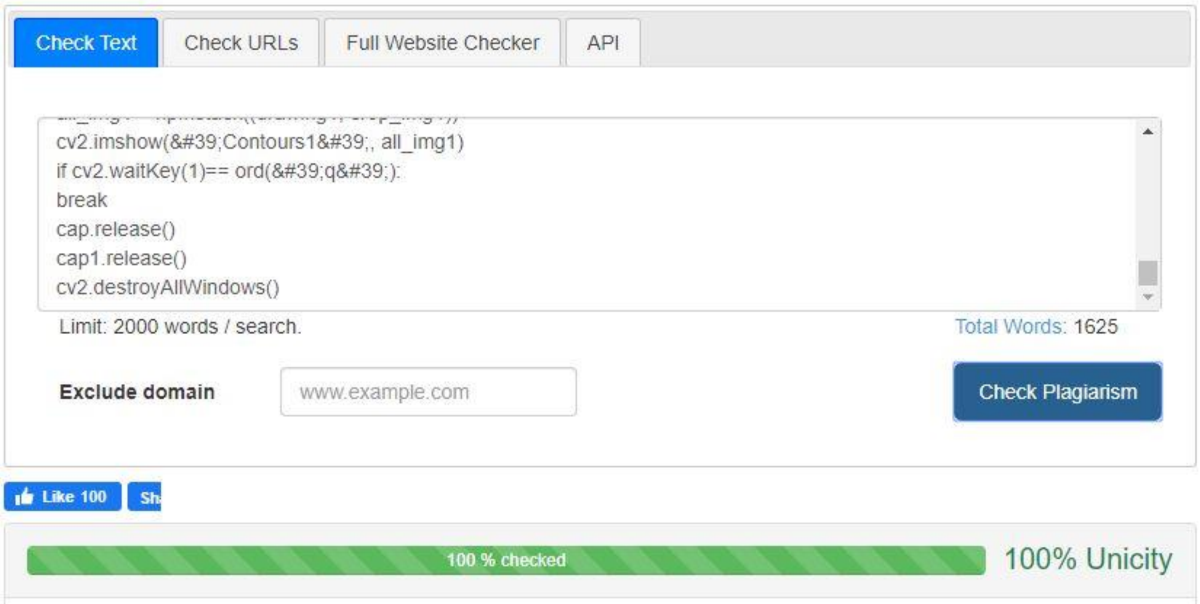


Communication between deaf-mute and a normal person has always been a challenging task. Sign language helps disabled people to converse with normal beings.
Our goal is to provide with a basic program that translates numbers and alphabets in a Sign Language. It inputs the gesture combinations through the detecting areas and correspondingly shows the text output. If our program is elaborated after our project, it can be a program that allows people to have a free conversation irrespective of the person

Limit: 2000 words / search. Total Words: 1967

Exclude domain: www.example.com

100 % checked — 100% Unicity

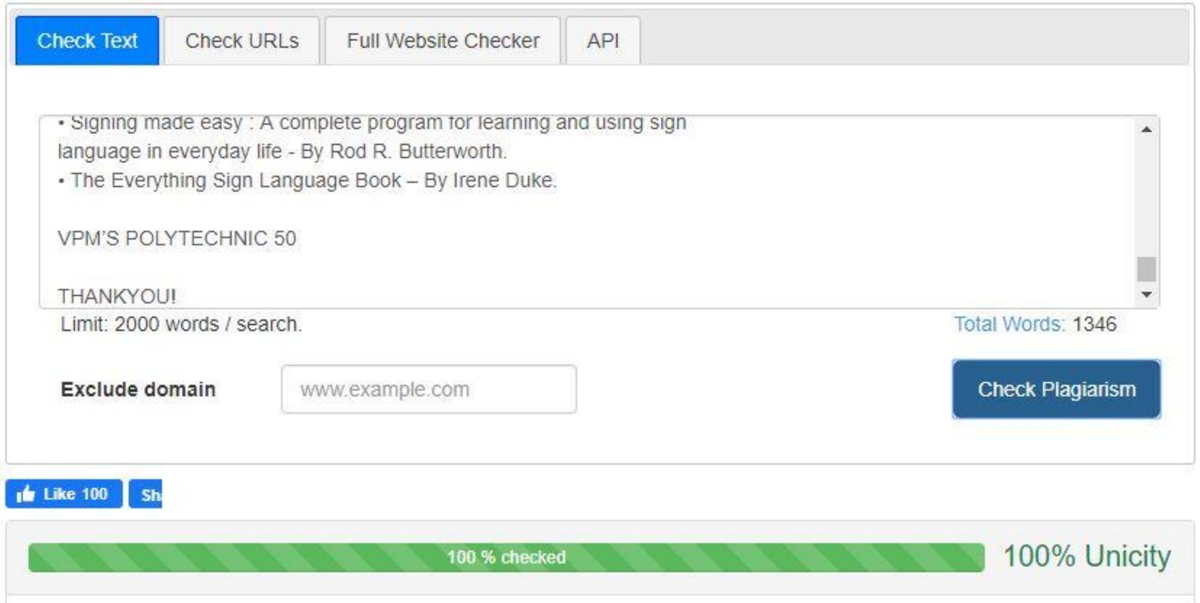


```
cv2.imshow(&#39;Contours1&#39;, all_img1)
if cv2.waitKey(1)== ord(&#39;q&#39;):
break
cap.release()
cap1.release()
cv2.destroyAllWindows()
```

Limit: 2000 words / search. Total Words: 1625

Exclude domain: www.example.com

100 % checked — 100% Unicity



• Signing made easy : A complete program for learning and using sign language in everyday life - By Rod R. Butterworth.
• The Everything Sign Language Book – By Irene Duke.

VPM'S POLYTECHNIC 50

THANKYOU!

Limit: 2000 words / search. Total Words: 1346

Exclude domain: www.example.com

100 % checked — 100% Unicity