**SMART SIGN LANGUAGE LIVE INTERPRETOR**

**FINAL REPORT**

Submitted for the course: Human Computer Interaction (CSE4015)

By

|  |  |
| --- | --- |
| SAUNAK SAHOO | 16BCE2167 |
| CORRINA MARIEANNE BARNABAS | 16BCE0754 |

Slot: C2+TC2

**Name of faculty: Dr. NAVAMANI T M**

**SCOOL OF COMPUTER SCIENCE AND ENGINNERING**



**Abstract**

This HCI application will be able to recognize hand gestures and translate them. This would give dumb and deaf people a new freedom because they would no longer need to rely on interpreters and people who know sign language to communicate. They can simply use this application to translate what they’re saying, live.We will be building our application to translate hand gestures into the letters of the English alphabet. English is one of the most universally spoken languages and hence this application can be used in most parts of the world. The application would use the OpenCV (Open source computer vision) platform to use the computers cameras to record the gestures. We will be using OpenCV’s object detection, contour detection and image processing modules. Machine Learning would be used to train the application for American Sign Language letters. We plan on using this platform on Python.

**Introduction**

The American Sign Language (ASL) is the most highly used sign language in the world. With over 250,000 to 500,000 users in the world, ASL transcends international borders for the dumb and deaf community and gives them an opportunity to interact with one another freely. The language relies on hand gestures as well as gestures of the face and torso.

Despite being widely used, deaf people continue to face problems in communicating with other non-deaf people. Only a handful of us take the effort to learn sign language to talk to or understand such people. These people often need interpreters to get their point across. In the 21st century, communication should no longer be a problem. Hence we want to build a software that would enable them to do just this.

This project looks to translate the American Sign Language (ASL) in real time so that there will be no dumb and deaf people who will have to rely on interpreters to communicate with the outside world.

It does this by using the computer’s webcam as the input gestures. Any person can make ASL gestures into the camera and get it’s English language translation back. ASL gestures are based on individual letters of the English alphabet (A-Z) and some other common phrases such as “hello”, “goodbye” etc. This application implements the letter by letter translation of ASL, which is the most basic technique required for comprehending what the gesture based language is trying to convey.

The main advantage of this application is that it is real time. There’s no need to record the video/capture an image of the gesture to be recognized later. Instead, processing takes place in real time and the camera keeps continuously capturing and matching it with existing images and patterns found in the dataset. There’s also no need to wear any gloves or special equipment.

**Motivation**

* The American Sign Language (ASL) is the most highly used sign language in the world with over 250,000 to 500,000 users in the world. So there is a large market for this application which can solve the problems of so many people across continents. Many different versions of this application may be made later with support for newer languages.
* Deaf people continue to face problems in communicating with other non-deaf people. Only a handful of us take the effort to learn sign language to talk to or understand such people.
* These people often need human interpreters who are not always available as well as expensive to hire

**Issues in existing systems**

* Many of the existing solutions in sign language translation do interpretation with some delay. They take time to process the input by either taking a video or an image and only then can they process it by matching it to an existing database. Real-time solutions are few and inaccurate on the most part [1][3][7]
* Most system utilize a complex system to identify gestures. These may include the use of multiple physical sensors embedded on the body or outside, or the use of specialized data gloves. While they may increase accuracy, they are expensive to manufacture and purchase. Also they require the users to wear it at all times which is an inconvenience. A user has to remember to carry it everywhere they may go. One can argue that basic sign language gesture recognition can incorporate some tiny percentage of errors with the tradeoff of increased ease of use [2][4]
* Some systems require gloves to be worn. While these aren’t data gloves which connect directly with the system, they are often coloured in a different bright colour to distinguish it from the surrounding environment. [5]
* Many of these applications require huge amounts of resources and processing power. Something that not all users may have access to. Our application can also be ported to mobile operating systems, and hence are not too resource hungry. [8] [9]
* Many gesture recognition systems have some certain level of inaccuracies. This application tries its best to remove them as much as it can. [3][4]

**Related Work**

1. Independent Bayesian classifier combination based sign language recognition using facial expression:  
     
   Automatic Sign Language Recognition (SLR) systems are usually designed by means of recognizing hand and finger gestures. However, **facial expressions** play an important role to represent the emotional states during sign language communication, has not yet been analyzed to its fullest potential in SLR systems. A SLR system is incomplete without the signer’s facial expressions corresponding to the sign gesture. This paper focused on presenting a novel multimodal framework for SLR system by incorporating facial expression with sign gesture using two different sensors, namely **Leap motion and Kinect.** Sign gestures are recorded using Leap motion and simultaneously a Kinect is used to capture the facial data of the signer. We have collected a dataset of 51 dynamic sign word gestures. The recognition is performed using Hidden Markov Model (HMM). Next, we have applied **Independent Bayesian Classification Combination (IBCC)** approach to combine the decision of different modalities for improving recognition performance. [1]
2. Early estimation model for 3D-discrete indian sign language recognition using graph matching  
     
   Machine translation of sign language is a critical task of computer vision. In this work, we propose to use **3D motion capture technology** for sign capture and graph matching for sign recognition. Two problems related to 3D sign matching are addressed in this work:   
   (1) how to identify same signs with different number of motion frames and   
   (2) sign extraction from a clutter of non-sign hand motions.   
   These two problems make the 2D or 3D sign language machine translation a challenging task. We propose **graph matching** with early estimation model to address these problems in two phases.   
   The first phase consists of intra graph matching for motion frame extraction, which retains motion intensive frames in database and query 3D videos.   
   The second phase applies inter graph matching with early estimation model on motion extracted query and dataset 3D videos.   
   The proposed model increases the speed of the graph matching algorithm in estimating a sign with fewer frames. To test the graph matching model, we recorded 350 words of **Indian sign language** with 3D motion capture technology. [2]
3. A signer-independent Arabic Sign Language recognition system using face detection, geometric features, and a Hidden Markov Model  
     
   In this paper, we propose an image-based system for Arabic Sign Language (ArSL) recognition. The algorithm starts by detecting the face of the signer using a **Gaussian skin color model**. The centroid of the detected face is then used as a reference point for tracking the hands’ movements. The hands regions are segmented using a region growing algorithm assuming the signer wears a yellow and an orange colored gloves. From the segmented hands regions, an optimal set of features is extracted. To represent the time varying feature patterns, a **Hidden Markov Model (HMM)** is then used. Before using HMM in testing, the number of states and the number of Gaussian mixtures are optimized. The proposed system was implemented for both signer dependent and signer independent conditions. The experimental results show that an accuracy of more than 95% can be achieved with a large database of 300 signs. The results outperform previous work on ArSL mainly restricted to small vocabulary size. [3]
4. A multimodal framework for sensor based sign language recognition  
     
   In this paper, we propose a novel multimodal framework for isolated Sign Language Recognition (SLR) using **sensor devices**. Microsoft Kinect and Leap motion sensors are used in our framework to capture finger and palm positions from two different views during gesture. One sensor (Leap Motion) is kept below the hand(s) while the other (Kinect) is placed in front of the signer for capturing horizontal and vertical movement of fingers during sign gestures. A set of features is next extracted from the raw data captured with both sensors. Recognition is performed separately by Hidden Markov Model (HMM) and **Bidirectional Long Short-Term Memory Neural Network (BLSTM-NN)** based sequential classifiers. In the next phase, results are combined to boost-up the recognition performance. The framework has been tested on a dataset of 7500 Indian Sign Language (ISL) gestures comprised with 50 different sign-words. [4]
5. Recognition of Sign Language Alphabets and Numbers based on Hand Kinematics using A Data Glove  
     
   This paper reports real-time recognition of Indian and American sign language alphabets and numbers based on hand kinematics assessment. The finger and wrist joint angles were acquired using an indigenously developed **data glove**. The data set was for single handed Indian sign language alphabets (C, I, J, L, O, U, Y, W), American sign language alphabets (A to Z) and sign numbers (0 to 9). The data were pre-processed through a moving average filter and standardized feature scaling methods. The glove was able to measure the finger joint angles with an accuracy±standard deviation for metacarpophalangeal (MCP) joint±2.14°, proximal inter phalangeal (PIP) joint± 1.73° and distal inter phalangeal (DIP) joint± 1.49°, during flexion/extension and abduction/adduction movements. [5]
6. Persian sign language (PSL) recognition using wavelet transform and neural networks  
     
   This paper presents a system for recognizing static gestures of alphabets in Persian sign language (PSL) using **Wavelet transform and neural networks (NN).** The required images for the selected alphabets are obtained using a digital camera. The color images are cropped, resized, and converted to grayscale images. Then, the discrete wavelet transform (DWT) is applied on the gray scale images, and some features are extracted. Finally, the extracted features are used to train a Multi-Layered Perceptron (MLP) NN. Our recognition system does not use any gloves or visual marking systems. This system only requires the images of the bare hand for the recognition. The system is implemented and tested using a data set of 640 samples of Persian sign images; 20 images for each sign [6]
7. Sign language recognition using machine learning  
     
   Before using HMM in testing, the number of states and the number of Gaussian mixtures are optimized. The proposed system was implemented for both signer dependent and signer independent conditions. The experimental results show that an accuracy of more than 95% can be achieved with a large database of 300 signs. The results outperform previous work on ArSL mainly restricted to small vocabulary size. However, **facial expressions** play an important role to represent the emotional states during sign language communication, has not yet been analyzed to its fullest potential in SLR systems. A SLR system is incomplete without the signer’s facial expressions corresponding to the sign gesture. This paper focused on presenting a novel multimodal framework for SLR system by incorporating facial expression with sign gesture using two different sensors [7]
8. German sign language (GSL) recognition using image processing and neural networks  
     
   Uses 3D modelling techniques to train the application on sign language and recognize it. They use graph matching. The main advantages are that the model is able to distinguish similar looking sign languages because it makes a complete 3D model of each word.  
   The data were pre-processed through a moving average filter and standardized feature scaling methods. Microsoft Kinect and Leap motion sensors are used in our framework to capture finger and palm positions from two different views during gesture. One sensor (Leap Motion) is kept below the hand(s) while the other (Kinect) is placed in front of the signer for capturing horizontal and vertical movement of fingers during sign gestures. A set of features is next extracted from the raw data captured with both sensors [8]
9. A sensor based sign language recognition  
     
   In this paper, we propose a novel multimodal framework for isolated Sign Language Recognition (SLR) using **sensor devices**. Microsoft Kinect and Leap motion sensors are used in our framework to capture finger and palm positions from two different views during gesture. One sensor (Leap Motion) is kept below the hand(s) while the other (Kinect) is placed in front of the signer for capturing horizontal and vertical movement of fingers during sign gestures. A set of features is next extracted from the raw data captured with both sensors. Recognition is performed separately by Hidden Markov Model (HMM) and **Bidirectional Long Short-Term Memory Neural Network (BLSTM-NN)** based sequential classifiers. In the next phase, results are combined to boost-up the recognition performance. The framework has been tested on a dataset of 7500 Indian Sign Language (ISL) gestures comprised with 50 different sign-words. [9]

**Summary of Existing System**

1. Emphasizes that along with recognition of fingers and the wrists, sign language also depends heavily on the facial expression of the communicator. Hence, the paper argues that in addition to tracking of hands, facial expressions must also be mapped to make it more accurate. They used 2 different sensors to map the hand gestures and the face. They trained their model with 51 different words and used Independent Bayesian Classification Combination (IBCC) to recognize the various gestures. [1]
2. Uses 3D modelling techniques to train the application on sign language and recognize it. They use graph matching. The main advantages are that the model is able to distinguish similar looking sign languages because it makes a complete 3D model of each word. Also, when an invalid sign language is made, it will make no attempt to match it to something close to it, hence greatly reducing errors. [2]
3. Uses the entire features of the body to interpret what the speaker is trying to say. The model is trained to work with orange gloves and the camera keeps track of the entire outline of the body using the Gaussian skin color model. [3]
4. Instead of using cameras, this model uses sensors to understand sign language. It trained the model on over 7500 instances of sign language words to make the model work accurately. [4]
5. Instead of using cameras or sensors, this model uses gloves to detect the sign language. It measures the angles made by the user of the gloves along different axes and produces moderately accurate results [5]
6. Uses wavelet technology along with neural networks to recognize sign language. Instead of live tracking, the application takes an image of your hand and converts it into gray-scale to try and figure out the contours of the hand. This contour is then matched to outline images of sign languages to produce the output. [6]
7. Uses the entire features of the body to interpret what the speaker is trying to say. The model is trained to work with orange gloves and the camera keeps track. [7]

**Proposed System**

**Overview**

We will be building our application to translate hand gestures into the letters of the English alphabet. English is one of the most universally spoken languages and hence this application can be used in most parts of the world.

The application would use the OpenCV (Open source computer vision) platform to use the computers cameras to record the gestures. We will be using OpenCV’s object detection, contour detection and image processing modules. Machine Learning would be used to train the application for American Sign Language letters. We plan on using this platform on Python.

OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez.

The platform supports a variety of operating systems such as Windows, Linux, macOS, FreeBSD, NetBSD, OpenBSD. It can also be used on mobile operating systems such as Android, iOS, Maemo and BlackBerry 10. Hence this application can effectively be used on effectively every desktop or mobile in the world.

**System Architecture**

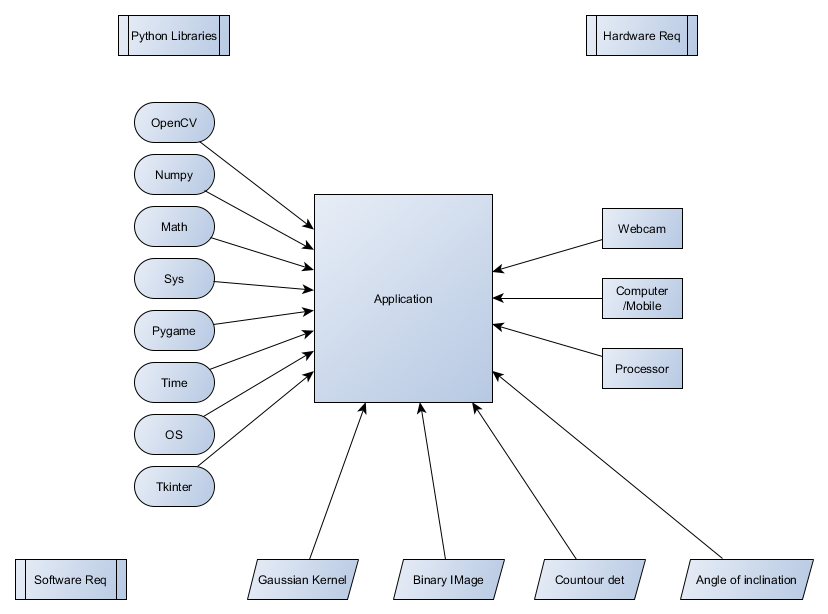


Fig 1: System Architecture

The architecture of this application mainly consist of the following 3 components:

1. **Python libraries**: The application is built on Python. It uses various libraries of Python to implement the project. Some of them are:

* OpenCV: This module is responsible for computer vision and machine learning software functions of the application. It is open-source too, hence it is easy to add and remove components from the library to be tailormade for our project
* Numpy: This is the scientific package of python which adds mathematical and computational ability to Python programs. Here it is used to calculate various dense matrices such as calculating threshold of pixels, calculating angles of inclination etc
* Math: Standard math library for python used for calculating square roots etc
* Sys: System specific functions and parameters. This is used to access specific core functionality associated with the system such as opening windows, operating the webcam etc
* Pygame: Usually used for writing game programs, here it is used for things like drawing graphs and vectors to recognize hand gestures
* Time: Keeps time of how long a function runs. This is used to recognize some specific kinds of gestures
* OS: Used to access core operating system functionality such as windows, webcam etc
* Tkinter: This is used to define the Graphical User Interface of the application.

1. **Hardware Requirements:**
   * **Webcam:** For computer vision to see sign language gestures
   * **Computer/mobile:** This application has been demonstrated on a computer but due to OpenCV being supported on almost every major operating system in the world it can very easily be ported
   * **Processor:** Processing power for the application
2. **Software Requirments:**
   * **Gaussian Kernel:** This is used to smoothen out the rough edges and make the image easier for the computer to understand
   * **Binary Image:** An image divided into only black and white components
   * **Kernel Detection:** Used to measure various distances, such as from the centre of the image to the borders, distance between fingers etc.
   * **Angle of Inclination:** Angle between fingers (contours) to detect different letters.

**Functional Architecture**

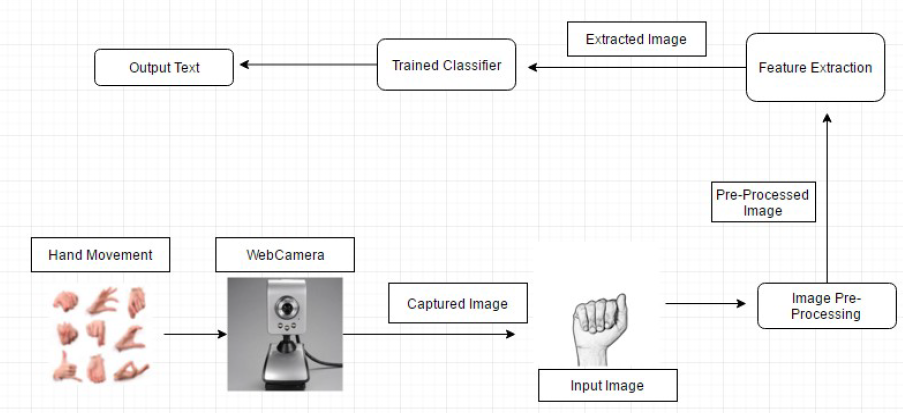
****

Fig 2: Functional Architecture

The above figure shows the functional architecture of this application

Hand movement is captured by a web camera which then processes the captured image. This input image is sent for pre-processing. This involves smoothening rough edges, converting into binary images and calculating distances etc.

Next phase extracts the features from the image so formed. It sends all this information to the trained classifier which then interprets the image and produces the output text.

**Innovation**

* Real time identification of sign language. The camera continuously processes what it sees and matches it with the database as well as calculate various other things such as surface area, angles of inclination, number of defects using which it recognizes the letter being depicted
* Simple system relying on primary camera instead of complex system of sensors or data gloves. So it can work readily on any computer or smartphones without the requirement of any additional hardware
* No brightly coloured gloves need to be worn to use object detection
* Portable system (can be used on a mobile phone as OpenCV is supported on many platforms including Andoird and iOS)
* Due to the wide number of parameters being used for letters detection, accuracy is increased and the chances of an erroneous letter being displayed is greatly reduced.

**Implementation**

**Module 1: Image recognition and interpretation**

**Algorithm:**

Step 1: User takes a real time image of the hand to be tested through the webcam

Step 2: The image is converted into gray scale and smoothed using a Gaussian kernel.

Step 3: Convert the gray scale image into a binary image. Set a threshold so that the pixels that

are above a certain intensity are set to white and those below are set to black

Step 4: Find contours, then remove noise and smooth the edges to smooth big contours and melt

numerous small contours.

Step 5: The largest contour is selected as a target.

Step 6: The angles of inclination of the contours and also the location of the center of the contour

with respect to the center of the image are obtained through the bounding box

information around the contour.

Step 7: The hand contours inside the bounding boxes are extracted and rotated in such a way that

the bounding boxes are made upright (inclination angle is 0) so that matching becomes

easy.

Step 8: Both the images are scaled so that their widths are set to the greater of the two widths and

their heights are set to the greater of the two heights. This is done so that the images are

the same size.

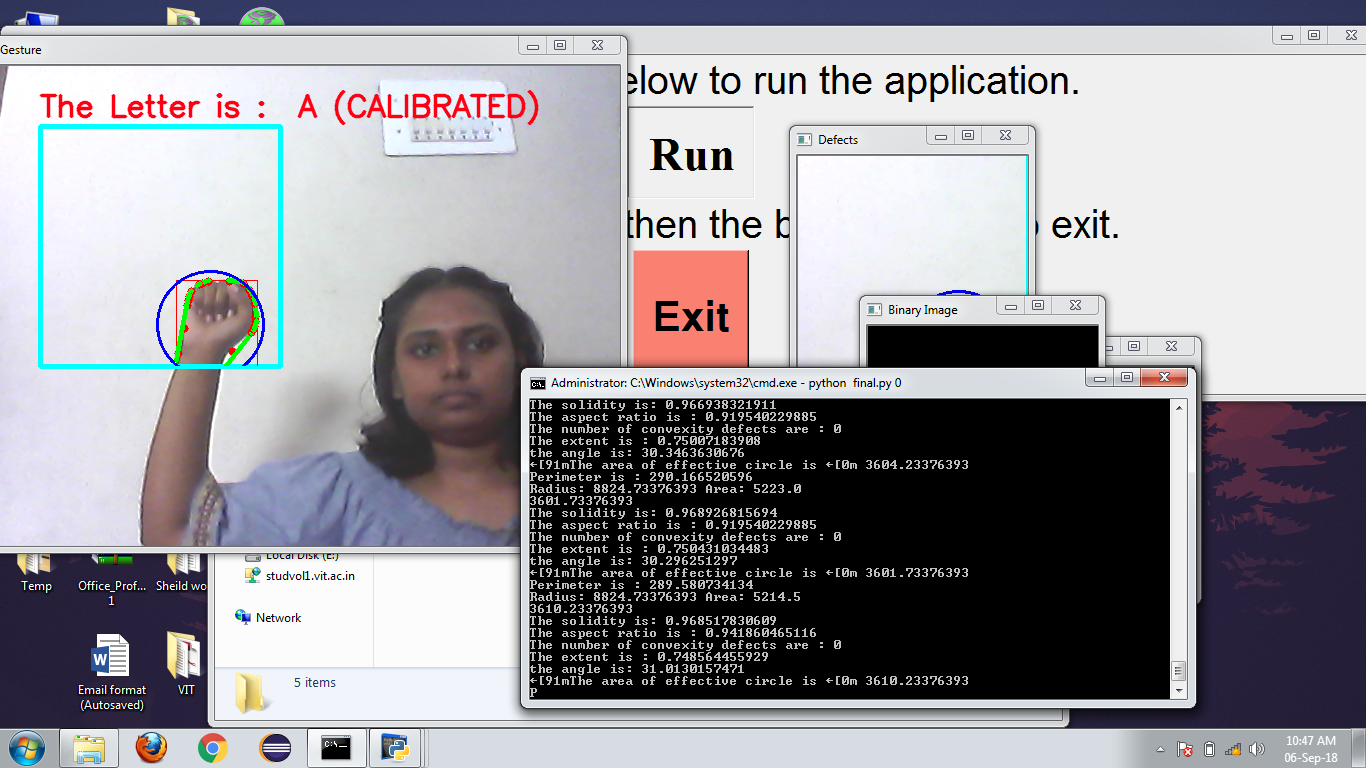
Step 9: The distance transform of both the query image and the candidate images are computed

and the best match is returned.

**Test cases:**

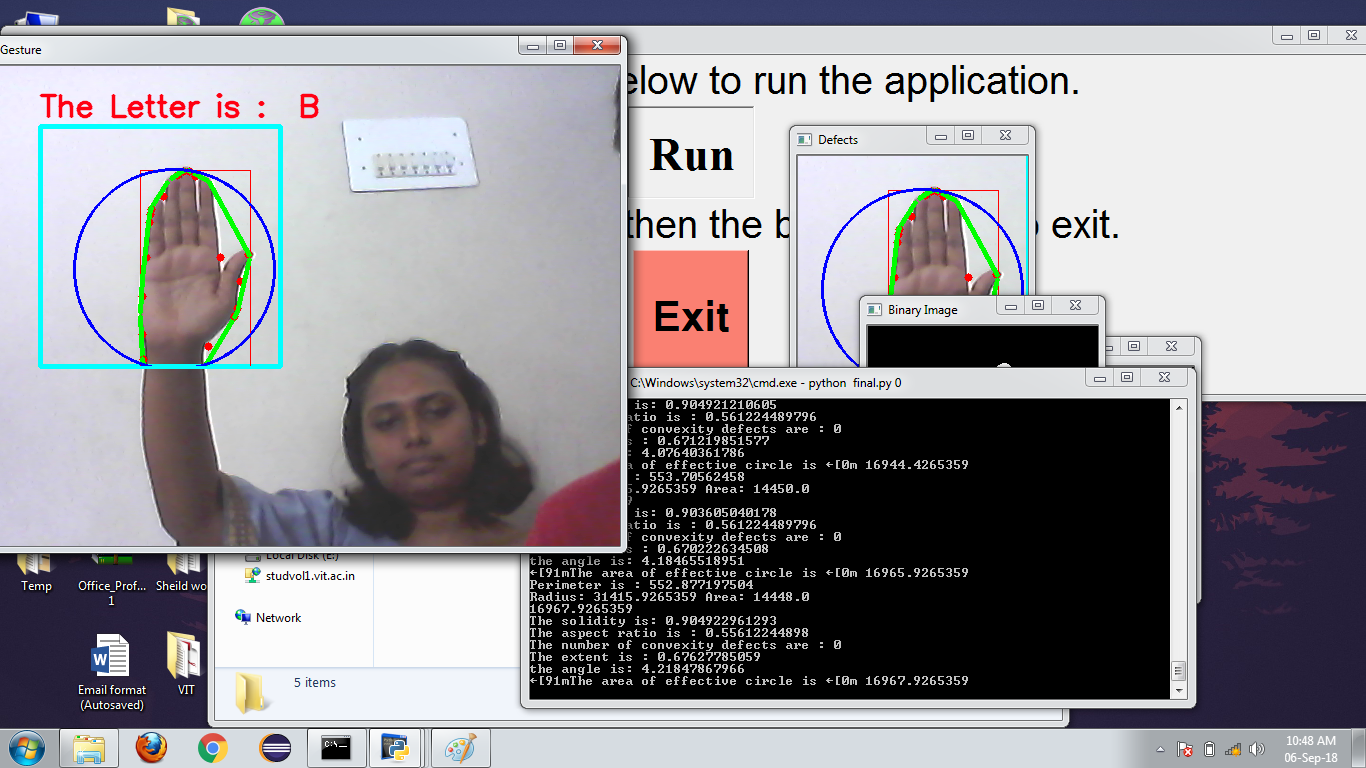
The following test cases look at the proper implementation of the project and if the application gives the correct output

1. **Case ‘A’ (Note that A is also used to calibrate the system):**

****

**Fig A: Letter A**

1. **Case ‘B’:**

****

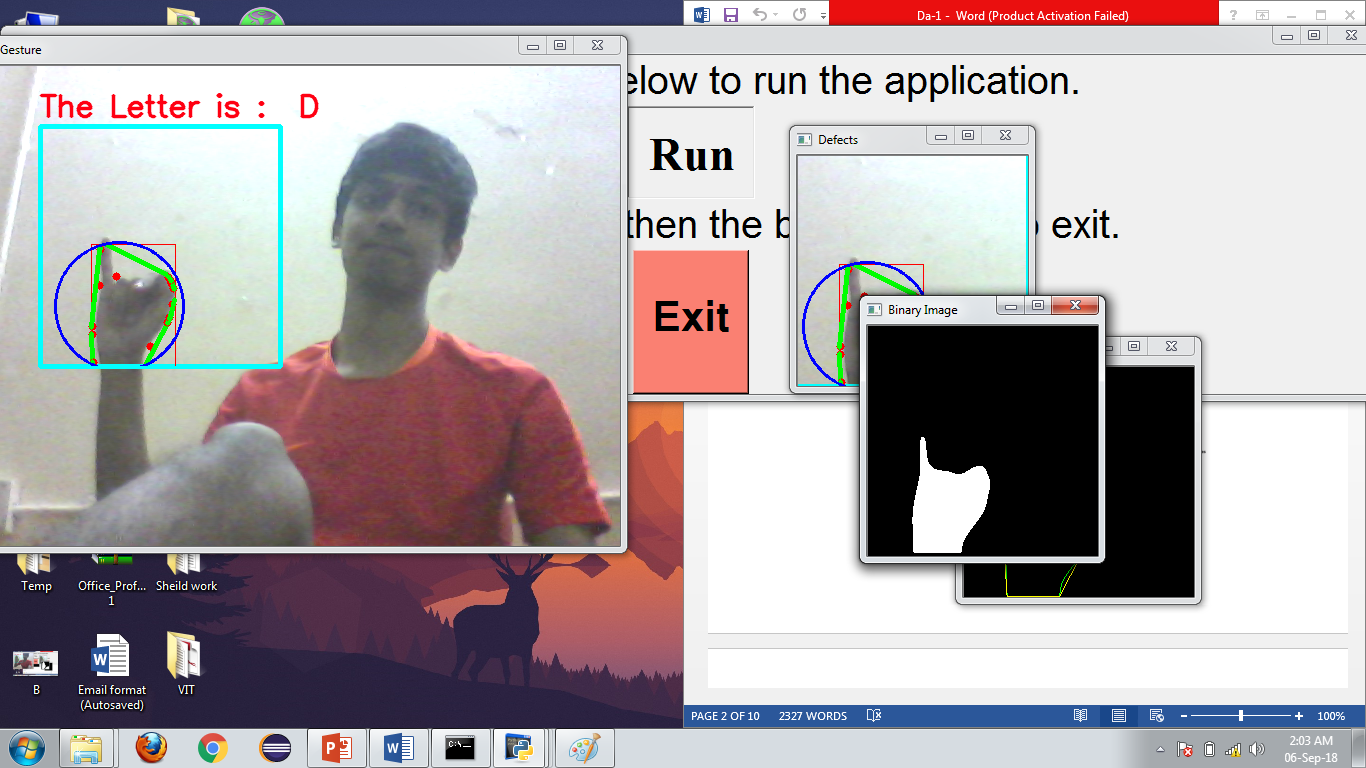
**Fig. B: Letter B**

1. **Case ‘C’:**

****

**Fig C: Letter C**

1. **Case ‘D’:**

****

**Fig D: Letter D**

**Sample Code**

**Python Code:**

if area\_of\_circle - area < 5000:

letter\_correspond = "A.txt"

destination ='Letters\_stash\_for\_sounds/'

createFile(destination)

#raw\_input("done!!!")

output = "A"

outFile = open('Letters\_stash\_for\_sounds/A.txt', 'w')

outFile.write(output)

cv2.putText(img, "The Letter is : A (CALIBRATED)", put\_text\_pos, font, 1 , put\_text\_color, 2, cv2.LINE\_AA)

#cv2.putText(img, "The letter is :", (60,50), font, 1 , put\_text\_color, 2, cv2.LINE\_AA)

#cv2.putText(img, "A", (320,55), font, 2 , (50,100,190), 3, cv2.LINE\_AA)

elif count\_defects ==1:

if angle\_t < 10:

letter\_correspond = "V.txt"

destination ='Letters\_stash\_for\_sounds/'

createFile(destination)

#raw\_input("done!!!")

output = "V"

outFile = open('Letters\_stash\_for\_sounds/V.txt', 'w')

outFile.write(output)

#letter\_correspond = "V.txt"

cv2.putText(img, "The Letter is : V", put\_text\_pos, font, 1, put\_text\_color, 2, cv2.LINE\_AA)

elif 40 < angle\_t < 66:

letter\_correspond = "C.txt"

destination ='Letters\_stash\_for\_sounds/'

createFile(destination)

#raw\_input("done!!!")

output = "C"

outFile = open('Letters\_stash\_for\_sounds/C.txt', 'w')

outFile.write(output)

cv2.putText(img, "The Letter is : C", put\_text\_pos, font, 1, put\_text\_color, 2, cv2.LINE\_AA)

elif 20 < angle\_t < 35:

letter\_correspond = "L.txt"

destination ='Letters\_stash\_for\_sounds/'

createFile(destination)

#raw\_input("done!!!")

output = "L"

outFile = open('Letters\_stash\_for\_sounds/L.txt', 'w')

outFile.write(output)

cv2.putText(img, "The Letter is : L", put\_text\_pos, font, 1, put\_text\_color, 2, cv2.LINE\_AA)

else:

letter\_correspond = "Y.txt"

destination ='Letters\_stash\_for\_sounds/'

createFile(destination)

#raw\_input("done!!!")

output = "Y"

outFile = open('Letters\_stash\_for\_sounds/Y.txt', 'w')

outFile.write(output)

cv2.putText(img,"The Letter is : Y", put\_text\_pos, font, 1, put\_text\_color, 2, cv2.LINE\_AA)

#print "Its 2"

elif count\_defects == 2: # Its either W or F

if angle\_t > 100:

letter\_correspond = "F.txt"

destination ='Letters\_stash\_for\_sounds/'

createFile(destination)

#raw\_input("done!!!")

output = "F"

outFile = open('Letters\_stash\_for\_sounds/F.txt', 'w')

outFile.write(output)

cv2.putText(img, "The Letter is : F", put\_text\_pos, font, 1, put\_text\_color, 2, cv2.LINE\_AA)

else:

letter\_correspond = "W.txt"

destination ='Letters\_stash\_for\_sounds/'

createFile(destination)

#raw\_input("done!!!")

output = "W"

outFile = open('Letters\_stash\_for\_sounds/W.txt', 'w')

outFile.write(output)

cv2.putText(img, "The Letter is : W", put\_text\_pos, font, 1, put\_text\_color, 2, cv2.LINE\_AA)

elif count\_defects == 4:

letter\_correspond = "CALIBRATE.txt"

destination ='Letters\_stash\_for\_sounds/'

createFile(destination)

#raw\_input("done!!!")

output = "CALIBRATE"

outFile = open('Letters\_stash\_for\_sounds/CALIBRATE.txt', 'w')

outFile.write(output)

cv2.putText(img,"Hello There ! Callibrate by letter A", put\_text\_pos, font,1, put\_text\_color, 2, cv2.LINE\_AA)

else:

if area > 12000:

letter\_correspond = " B.txt"

destination ='Letters\_stash\_for\_sounds/'

createFile(destination)

#raw\_input("done!!!")

output = "B"

outFile = open('Letters\_stash\_for\_sounds/B.txt', 'w')

outFile.write(output)

cv2.putText(img,"The Letter is : B", put\_text\_pos, font, 1, put\_text\_color, 2, cv2.LINE\_AA)

else:

if solidity < 0.85:

if aspect\_ratio < 1:

if angle\_t < 20:

letter\_correspond = " D.txt"

destination ='Letters\_stash\_for\_sounds/'

createFile(destination)

#raw\_input("done!!!")

output = "D"

outFile = open('Letters\_stash\_for\_sounds/D.txt', 'w')

outFile.write(output)

cv2.putText(img,"The Letter is : D", put\_text\_pos, font, 1, put\_text\_color, 2, cv2.LINE\_AA)

elif 169<angle\_t <180:

letter\_correspond = " I.txt"

destination ='Letters\_stash\_for\_sounds/'

createFile(destination)

#raw\_input("done!!!")

output = "I"

outFile = open('Letters\_stash\_for\_sounds/I.txt', 'w')

outFile.write(output)

cv2.putText(img,"The Letter is : I", put\_text\_pos, font, 1, put\_text\_color, 2, cv2.LINE\_AA)

elif angle\_t < 168:

letter\_correspond = " J.txt"

destination ='Letters\_stash\_for\_sounds/'

createFile(destination)

#raw\_input("done!!!")

output = "J"

outFile = open('Letters\_stash\_for\_sounds/J.txt', 'w')

outFile.write(output)

cv2.putText(img,"The Letter is : J", put\_text\_pos, font, 1, put\_text\_color, 2, cv2.LINE\_AA)

elif aspect\_ratio > 1.01:

letter\_correspond = " Y.txt"

destination ='Letters\_stash\_for\_sounds/'

createFile(destination)

#raw\_input("done!!!")

output = "Y"

outFile = open('Letters\_stash\_for\_sounds/Y.txt', 'w')

outFile.write(output)

cv2.putText(img,"The Letter is : Y", put\_text\_pos, font, 1, put\_text\_color, 2, cv2.LINE\_AA)

else:

if angle\_t > 30 and angle\_t < 100:

letter\_correspond = " H.txt"

destination ='Letters\_stash\_for\_sounds/'

createFile(destination)

#raw\_input("done!!!")

output = "H"

outFile = open('Letters\_stash\_for\_sounds/H.txt', 'w')

outFile.write(output)

cv2.putText(img,"The Letter is : H", put\_text\_pos, font, 1, put\_text\_color, 2, cv2.LINE\_AA)

elif angle\_t > 120:

letter\_correspond = " I.txt"

destination ='Letters\_stash\_for\_sounds/'

createFile(destination)

#raw\_input("done!!!")

output = "I"

outFile = open('Letters\_stash\_for\_sounds/I.txt', 'w')

outFile.write(output)

cv2.putText(img,"The Letter is : I", put\_text\_pos, font, 1, put\_text\_color, 2, cv2.LINE\_AA)

else:

letter\_correspond = " U.txt"

destination ='Letters\_stash\_for\_sounds/'

createFile(destination)

#raw\_input("done!!!")

output = "U"

outFile = open('Letters\_stash\_for\_sounds/U.txt', 'w')

outFile.write(output)

cv2.putText(img,"The Letter is : U", put\_text\_pos, font, 1, put\_text\_color, 2, cv2.LINE\_AA) #

**Conclusion**

This sign recognizing application would be perfect for deaf and dumb people who don’t want to depend on an interpreter to communicate with others. They can take this application with them and use it anywhere and with any type of person.

Since this application is made for the American Sign Language (ASL) which uses English as its primary language, many other people all over the world would be able to recognize what the deaf and dumb people are trying to say.

**Future Work**

We will develop an easy to use computer version of the application. In future, the application can be ported to be used with a mobile phone. This would make a the application truly easy to use as every person in the world with a mobile phone will be able to use this application to understand sign language.

In future, the application can also include many other sign languages. Here in India itself there are a lot of sign languages for whom the application can be trained using Machine Learning modules of OpenCV to support that language. ASL was used in this application as it is the most widely used sign language in the world.

**References**

1. Pradeep Kumar, Partha Pratim Roy, Debi Prosad Dogra,

Independent Bayesian classifier combination based sign language recognition using facial expression,

Information Sciences,

Volume 428,

2018,

Pages 30-48,

ISSN 0020-0255,

https://doi.org/10.1016/j.ins.2017.10.046.

(http://www.sciencedirect.com/science/article/pii/S0020025516307897)

Keywords: Sign language recognition; Depth sensors; Hidden Markov model (HMM); Bayesian combination

2. E. Kiran Kumar, P.V.V. Kishore, D. Anil Kumar, M. Teja Kiran Kumar,

Early estimation model for 3D-discrete indian sign language recognition using graph matching,

Journal of King Saud University - Computer and Information Sciences,

2018,

,

ISSN 1319-1578,

https://doi.org/10.1016/j.jksuci.2018.06.008.

(http://www.sciencedirect.com/science/article/pii/S1319157818303719)

Keywords: 3D motion capture; 3D sign language; 3D graph matching; Pattern classification; Spatial graph matching; Temporal graph matching

3. M. Mohandes, M. Deriche, U. Johar, S. Ilyas,

A signer-independent Arabic Sign Language recognition system using face detection, geometric features, and a Hidden Markov Model,

Computers & Electrical Engineering,

Volume 38, Issue 2,

2012,

Pages 422-433,

ISSN 0045-7906,

https://doi.org/10.1016/j.compeleceng.2011.10.013.

(<http://www.sciencedirect.com/science/article/pii/S0045790611001637>)

4. Pradeep Kumar, Himaanshu Gauba, Partha Pratim Roy, Debi Prosad Dogra,

A multimodal framework for sensor based sign language recognition,

Neurocomputing,

Volume 259,

2017,

Pages 21-38,

ISSN 0925-2312,

https://doi.org/10.1016/j.neucom.2016.08.132.

(http://www.sciencedirect.com/science/article/pii/S092523121730262X)

Keywords: Sign language recognition; Gesture recognition; Multimodal framework; Sensor fusion

5. Nayan M. Kakoty, Manalee Dev Sharma,

Recognition of Sign Language Alphabets and Numbers based on Hand Kinematics using A Data Glove,

Procedia Computer Science,

Volume 133,

2018,

Pages 55-62,

ISSN 1877-0509,

https://doi.org/10.1016/j.procs.2018.07.008.

(http://www.sciencedirect.com/science/article/pii/S1877050918309529)

Keywords: Finger kinematics; Sign language recognition; Data glove

6. Ali Karami, Bahman Zanj, Azadeh Kiani Sarkaleh,

Persian sign language (PSL) recognition using wavelet transform and neural networks,

Expert Systems with Applications,

Volume 38, Issue 3,

2011,

Pages 2661-2667,

ISSN 0957-4174,

https://doi.org/10.1016/j.eswa.2010.08.056.

(http://www.sciencedirect.com/science/article/pii/S0957417410008523)

Keywords: Persian sign language; Hand gesture; Discrete wavelet transform (DWT); Neural network (NN)

7. Pradeep Kumar, Himaanshu Gauba, Partha Pratim Roy, Debi Prosad Dogra,

A multimodal framework for sensor based sign language recognition,

Neurocomputing,

Volume 259,

2017,

Pages 21-38,

ISSN 0925-2312,

https://doi.org/10.1016/j.neucom.2016.08.132.

(http://www.sciencedirect.com/science/article/pii/S092523121730262X)

Keywords: Sign language recognition; Gesture recognition; Multimodal framework; Sensor fusion

8. 2. E. Kiran Kumar, P.V.V. Kishore, D. Anil Kumar, M. Teja Kiran Kumar,

Early estimation model for 3D-discrete indian sign language recognition using graph matching,

Journal of King Saud University - Computer and Information Sciences,

2018,

,

ISSN 1319-1578,

https://doi.org/10.1016/j.jksuci.2018.06.008.

(http://www.sciencedirect.com/science/article/pii/S1319157818303719)

Keywords: 3D motion capture; 3D sign language; 3D graph matching; Pattern classification; Spatial graph matching; Temporal graph matching

9. M. Mohandes, M. Deriche, U. Johar, S. Ilyas,

A signer-independent Arabic Sign Language recognition system using face detection, geometric features, and a Hidden Markov Model,

Computers & Electrical Engineering,

Volume 38, Issue 2,

2012,

Pages 422-433,

ISSN 0045-7906,

https://doi.org/10.1016/j.compeleceng.2011.10.013.

1. OpenCV.org
2. https://en.wikipedia.org/wiki/American\_Sign\_Language