Your name : Akshaya V Register no.: 312217205005

Batchmate Name: Anitha S Register no.: 312217205008

Batchmate Name:Pagadala Sreenivas Register no.: 312217205062

HAND GESTURE ANALYSIS FOR THE DEAF AND MUTE

Date: 13/05/2020

Signature(s): 1. Akshaya V

2. Anitha S

3. Pagadala Sreenivas

Mentor Signature

Dr. Joe Louis Paul I,

Associate Professor,

Department of Information Technology

ABSTRACT

Communication is important for every individual to convey whatever information they want to people and vice versa. This project presents a natural and intuitive form of communication i.e. via hand gestures. Hand gesture is one of the important methods of nonverbal communication for human beings. There are plenty of methods that are used to recognize hand gestures with different accuracies and precision, some have advantages and disadvantages. Hand gesture recognition is a technology that is becoming increasingly relevant, given the recent growth and popularity of Human Computer Interaction (HCI). The general objective of this project is to develop a hand gesture analysis system to recognize the gestures made by deaf and mute people. The gesture performed by the user is recognized by the laptop web camera and the text related to a specific gesture is displayed on the computer screen. It is coded in Python and uses the OpenCV library.

INTRODUCTION

Computer technology has come a long way in the past two decades. The things that can be done, and the time spent on electronic devices has increased tremendously as well. They have infiltrated every aspect of our lives; from how we learn, to how we share experiences with others. Although the devices have been advancing quickly, the methods of interacting with these devices have been largely neglected – until now. With so many aspects of our lives being affected by computers, people began to wonder if there was a better, more natural way to interact with devices, other than using the traditional keyboard and mouse. This led to the emergence of a new field: Human Computer Interaction (HCI). This new field called Human computer Interaction will help us to recognize the hand gestures and perform the required action.

Gesture is a form of nonverbal communication which involves movement of a part of the body, especially the hand usually to express an idea. In a human interaction, we make use of speech, gestures and body movements to convey information. The human-computer interaction which makes use of input devices such as keyboard or mouse for communication lacks natural communication between humans and machines. For this purpose, it is important to develop applications or devices that support intuitive communication.

As computer technology continues to grow, the need for natural communication between humans and machines also increases. Although our mobile devices make use of the touch screen technology, it is not cheap enough to be implemented in desktop systems.

The method proposed in this paper makes use of a webcam through which gestures provided by the user are captured, processed and the message related to that gesture is displayed. For example, a gesture “V” i.e. two fingers, could be predefined in the system to the number 2.

The system has four phases namely, image acquisition, image pre-processing, feature extraction and gesture recognition, as described by Babu et.al. Image acquisition involves capturing images frame by frame using a webcam.

The captured images go through the image pre-processing process which involves color filtering, smoothing and thresholding. Feature extraction involves extracting features of the hand

image such as hand contours. Gesture recognition involves recognising hand gestures with the help of extracted features.

LITERATURE SURVEY

There are many gesture recognition techniques developed for tracking and recognizing various gestures. Gesture recognition using a web camera is an appealing option for replacing human computer interaction.

Back in 2009, Bayazit et.al implemented a GPU-based system for gesture recognition which runs in real-time. While this system demonstrated the principle, it was too bulky and unwieldy for practical use. Advancements in development of optic modules have allowed for 3-D image sensors to emerge as a viable alternative to stereo and structured light imaging for capturing range information. Due to this gesture recognition has the potential to emerge as an alternative input device in the near future.

A system using pointing behaviours for a natural interface to classify the dynamic hand gesture has been proposed by Badgujar et.al. Though this system is suitable for controlling real-time computer systems, it is applicable only for the powerpoint presentations. To overcome the challenges of gesture recognition using color detection and their relative position with each other, gesture recognition using contour analysis is implemented.

The algorithm implemented in this paper detects the gesture based on the number of contours that are visible and thereby performs the necessary operation related to the gesture. This paper brings out an innovative idea to use the camera.

MOTIVATION

According to the statistics of the World Federation of the Deaf and the World Health Organization, approximately 70 million people in the world are deaf–mute. A total of 360 million people are deaf, and 32 million of these individuals are children. The majority of speech- and hearing-impaired people cannot read or write in regular languages. Sign language (SL) is the native language used by the deaf and mute to communicate with others. SL relies primarily on

gestures rather than voice to convey meaning, and it combines the use of finger shapes, hand movements, and facial expressions. This language has the following main defects: a lot of hand movements, a limited vocabulary, and learning difficulties. Furthermore, SL is unfamiliar to those who are not deaf and mute, and disabled people face serious difficulties in communicating with able individuals. This communication barrier adversely affects the lives and social relationships of deaf people. Thus, dumb people need to use a translator device to communicate with able individuals. This can be achieved by developing a Hand Gesture Recognition System. It is a recognition system that can be employed for learning SL for both dumb and able people.

PROPOSED METHODOLOGY

HAND SEGMENTATION METHOD:

The basis for recognizing hand gestures is recognizing if there is a hand in the image. There are several methods of segmenting the hand from the background of the camera (or video) input. Efficient hand tracking and segmentation is the key of success towards any gesture recognition, due to challenges of vision based methods, such as varying lighting condition, complex background and skin color detection; variation in human skin color complexion required the robust development of algorithm for natural interface. Color is a very powerful descriptor for object detection. The method which is used for this project is **"Calibration and Thresholding'' [1]-[3]**. Figure 1 illustrates the system flow chart.

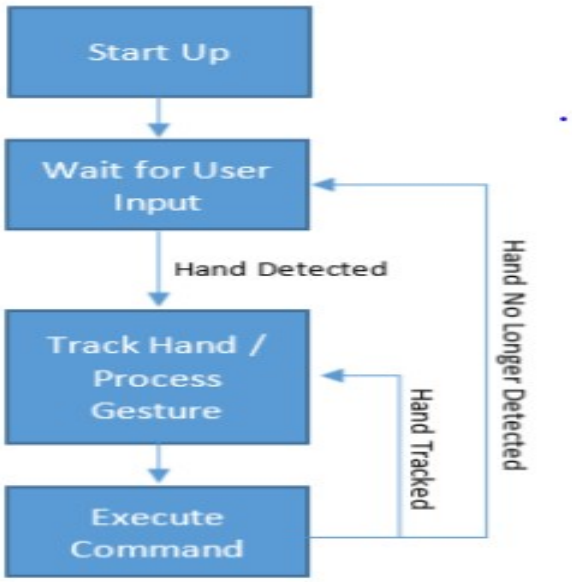


Fig. 1: System Flow Chart

Calibration and Thresholding:

This method involves having the user place his hand in front of the camera in a way that covers the calibration boxes displayed by the program. The program then gets the mean color values of each box, and saves these for future comparisons. When attempting to segment the hand, we can identify areas that differ from each of these color values by a given threshold. By compiling the segmentation given by each box's color value, we should be able to obtain a more complete segmentation of the hand; one that accounts for the variations in color across different regions of the hand [4]-[6]. Fig. 2 illustrates the Hand segmentation using calibration.

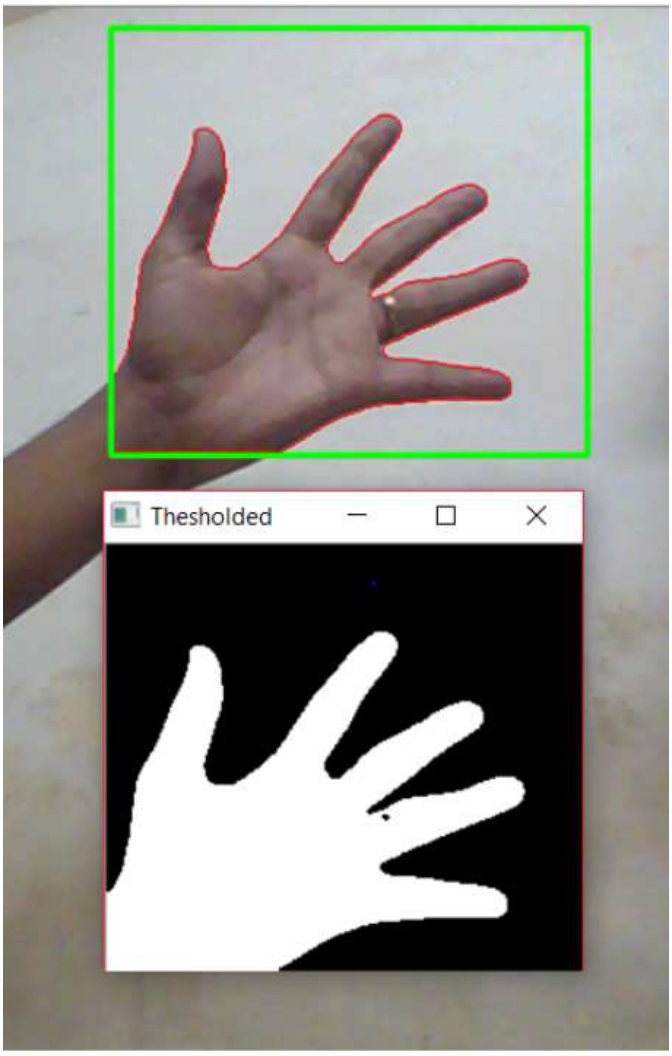


Fig. 2: Hand Segmentation using Calibration

```
 cv2.rectangle(frame,(100,100),(300,300),(0,255,0),0)
 hsv = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)

# define range of skin color in HSV
    lower_skin = np.array([0,20,70], dtype=np.uint8)
    upper_skin = np.array([20,255,255], dtype=np.uint8)


#extract skin colur imagw
    mask = cv2.inRange(hsv, lower_skin, upper_skin)
```

This method uses the HSV threshold values and OpenCV's inRange() function. The reasoning for this was that the main differentiating factor of the hand was its hue, and the variance in how a hand appears in different environments could be attributed to saturation and value.

GESTURE RECOGNITION:

Gesture Recognition comes after the segmentation of hand from the image. There are a lot of ways to gesture recognition and one of the effective methods is "**CONVEX HULL METHOD".**

Convex Hull Method:

The convex hull method takes the outline of a shape, and identifies the convex and concave (defect) points along the outline. These points provide us with a rough idea of the shape of the object. In the case of a hand, it should have 5 convex points (one for each finger) and 4 defects (one between two adjacent fingers). Using this method, we can identify the number of fingers the user is showing to us by the number of convex and defect points. For example, if there are 2 convex and 2 defect points, it is likely that the user is showing us 2 fingers [7]-[8].

In our implementation, we used OpenCV functions such as findContours(), convexHull() and convexityDefects() to obtain the convex hull and defect points.

```
#find contours
    _,contours,hierarchy= cv2.findContours(mask,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
 #make convex hull around hand
     hull = cv2.convexHull(cnt)
```

```
#define area of hull and area of hand
    areahull = cv2.contourArea(hull)
    areacnt = cv2.contourArea(cnt)

#find the defects in convex hull with respect to hand
   hull = cv2.convexHull(approx, returnPoints=False)
   defects = cv2.convexityDefects(approx, hull)


#code for finding no. of defects due to fingers
     for i in range(defects.shape[0]):
          s,e,f,d = defects[i,0]
          start = tuple(approx[s][0])
          end = tuple(approx[e][0])
          far = tuple(approx[f][0])
          pt= (100,180)
```

Convexity defects obtained, is a structure that returns four values, start point, end point, farthest point and approximate distance to farthest point, out of which three have been used. The figure below (i.e. Fig. 3), denotes for one of the contours the start, the end and the far point. C represents the start point, B represents the end point and A is the farthest point. The Figure 3 illustrates angle between two fingers.

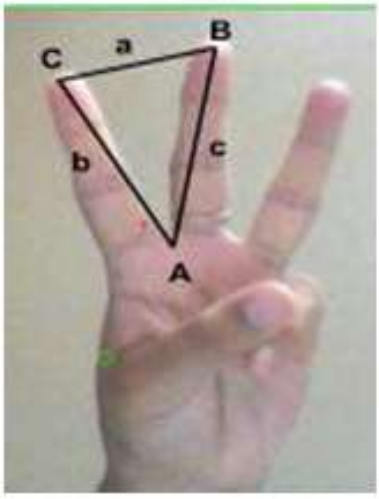


Fig. 3: Angle between the fingers

The angle made by the two fingers must be found to correctly determine if a finger is held up. This is done using the triangle formed by A, B and C. The length of each line is found using the distance formula as

```
# 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)
s = (a+b+c)/2
ar = math.sqrt(s*(s-a)*(s-b)*(s-c))
```

Once the length of each side of the triangle has been found, using the Cosine rule, as given in (1),

$$a^2 = b^2 + c^2 - 2bc\cos A \qquad (1)$$

the angle is found using

```
# apply cosine rule here
angle = math.acos((b**2 + c**2 - a**2)/(2*b*c)) * 57
```

If the angle A is less than 70°, it is considered that two fingers are held up. The same technique along with the area ratio is applied on all the defects to find the number of fingers held up.

```
#find the percentage of area not covered by hand in convex hull
    arearatio=((areahull-areacnt)/areacnt)*100
```

```
# ignore angles > 90 and ignore points very close to convex hull(they generally come due to noise)
if angle <= 90 and d>30:
    l += 1
    cv2.circle(roi, far, 3, [255,0,0], -1)
```

```
#print corresponding gestures which are in their ranges
font = cv2.FONT_HERSHEY_SIMPLEX
if l==1:
    if areacnt<2000:
        cv2.putText(frame,'Put hand in the box',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
    else:
        if arearatio<12:
            cv2.putText(frame,'0',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
        elif arearatio<17.5:
            cv2.putText(frame,'Best of luck',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

        else:
            cv2.putText(frame,'1',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
```

```
elif l==2:
    cv2.putText(frame,'2',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

elif l==3:

      if arearatio<27:
            cv2.putText(frame,'3',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
      else:
            cv2.putText(frame,'ok',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

elif l==4:
    cv2.putText(frame,'4',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

elif l==5:
    cv2.putText(frame,'5',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

elif l==6:
    cv2.putText(frame,'reposition',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

else :
    cv2.putText(frame,'reposition',(10,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
```

Two issues can be seen in this implementation. The first is that the number of defects is higher than expected, and is likely to be due the shape of the hand not being smooth. The presence of noise in the hand's contour led to additional defects.

EXPERIMENTAL RESULTS AND DISCUSSION

The experimental setup requires a web camera in order to capture the gestures performed by the user, preferably with a plain background. The web camera should be at a fixed position to help in capturing images more efficiently. Figure 4a) - 4h) illustrates the detection of various hand gestures.

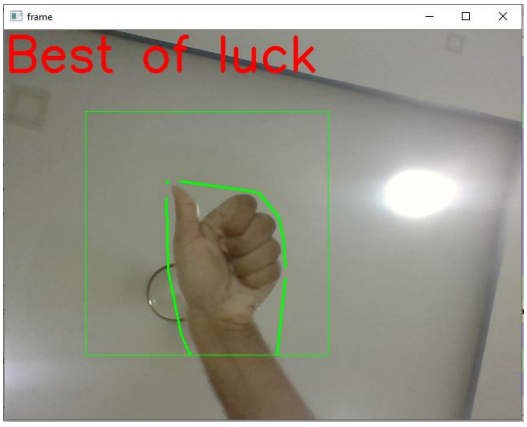


Fig. 4a): Detecting best of luck

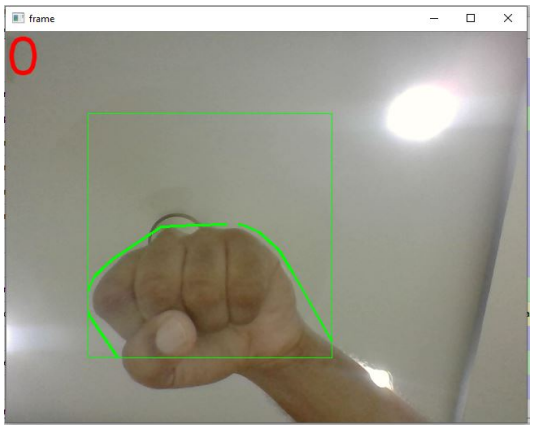


Fig. 4b): Detecting 0

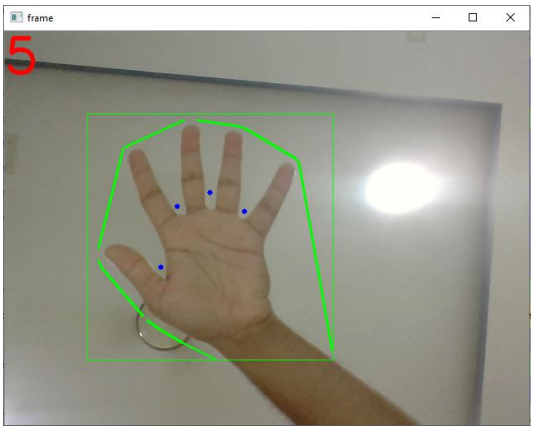


Fig. 4c): Detecting 5

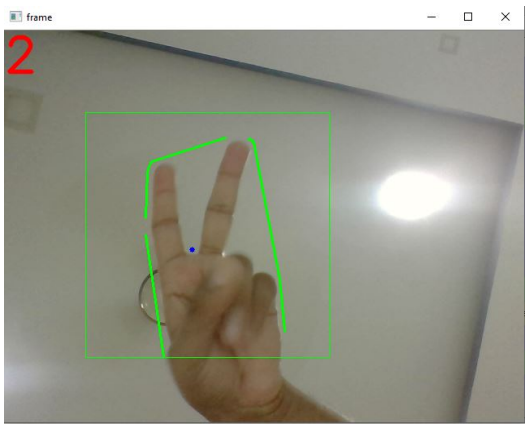


Fig. 4d): Detecting 2

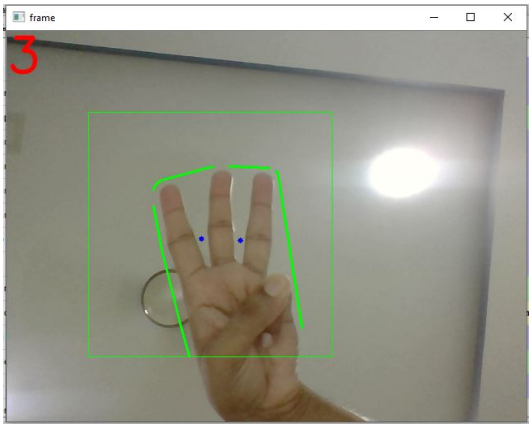


Fig. 4e): Detecting 3

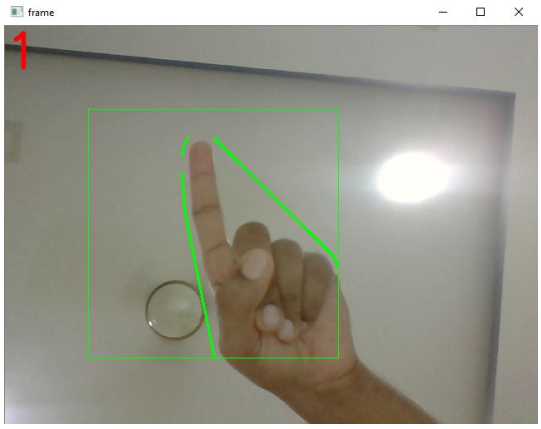
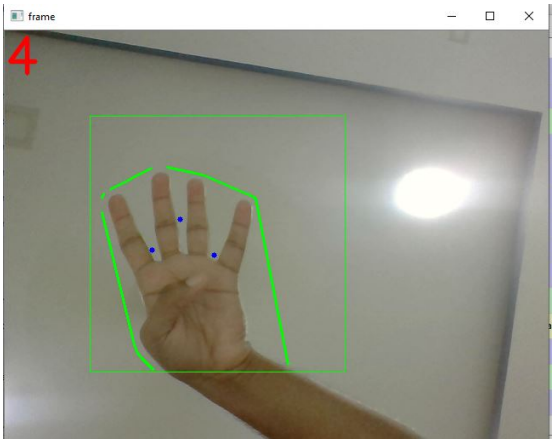


Fig. 4f): Detecting 1



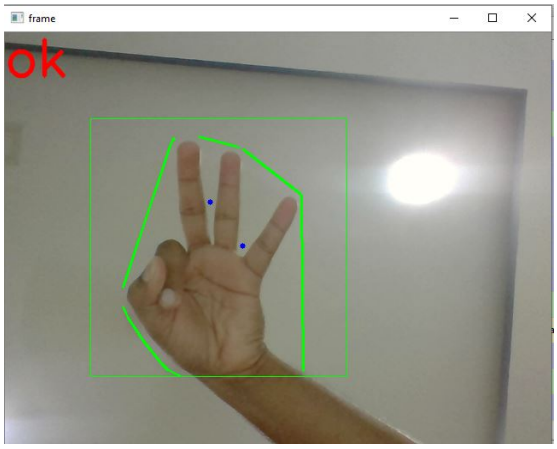


Fig. 4g): Detecting 4

Fig. 4h): Detecting ok

Fig. 4a)- 4h): Detection of various Hand Gestures

CONCLUSION

While the system implemented has much room for improvement, it can be said that the implementation process was a success to some extent. The method proposed here successfully created a hand gesture recognition system that is able to recognize which gesture is performed by the user and perform the functionality associated with it.

FUTURE ENHANCEMENT

The current system gives best results in a plain background and hence puts certain constraints on the user for successful working. The future work will include implementation of additional gestures which will enable the user to perform more functions with ease. Experiments need to be done on a larger scale so that results can be more accurate.

REFERENCES

1. Ann Abraham Babu, Satishkumar Varma, Rupali Nikhare, “Hand Gesture Recognition System for Human Computer Interaction Using Contour Analysis”, International Journal of Research in Engineering and Technology, eISSN: 2319-1163, pISSN: 2321-7308

2. L. Gu, X. Yuan and T. Ikenaga, “Hand gesture interface based on improved adaptive hand area detection and contour signature”, IEEE International Symposium on Intelligent Signal Processing and Communications Systems, pp. 463-468, 2012.

3. A. S. Ghotkar and G. K. Kharate, “Hand Segmentation Techniques to Hand Gesture Recognition for Natural Human Computer Interaction”, International Journal of Human Computer Interaction, vol. 3, no. 1, pp. 15-25, 2012.

4. A. Chonbodeechalermroong, "Dynamic Contour Matching for Hand Gesture Recognition from Monocular Image", IEEE International Joint Conference on Computer Science and Software Engineering, pp. 47-51, 2015.

5. Y. Fang, K. Wang, J. Cheng and H. Lu, "A Real-Time Hand Gesture Recognition Method", 2007 IEEE International Conference on Multimedia and Expo, pp. 995-998, 2007.

6. Itseez, Open Source Computer Vision Library - itseez2015opencv, 2015, accessed at https://github.com/itseez/opencv on 01/2/2017

7. Itseez, The OpenCV Reference Manual - itseez2014theopencv 2.4.9.0, April 2014, accessed at http://opencv.org/on 01/2/107

8. Numpy Developers, numpy 1.13 0rc2, accessed at https://pypi.python.org/pypi/numpy on 01/2/107 .