A CAL Project Report

on

HAND GESTURE RECOGNITION

to be submitted in partial fulfilling of the requirements for the course on

DIGITAL IMAGE PROCESSING (SWE1010)

by

S. BLESSY IDA GLADYS

16MIS0073

to

SRINIVASA PERUMAL R

Summer Semester 1 – 2018

TABLE OF CONTENTS

ABSTRACT

Gesture recognition is “the entire procedure of tracking gesture to their representation and converting them to some purposeful command.” Gesture recognition turns up to be important field in the recent years. Communication through gestures has been used for many other applications.

This hand gesture method is used for various purposes like human computer interactions, robotics, sign language recognition, controlling a music player, playing game, air browsing etc. Hand gestures replace peripheral devices like keyboard, mouse with hand gestures for human-computer interaction (HCI). Hand gesture recognition basically involves four steps –image acquisition, hand segmentation, feature extraction and classification. Mostly hand gesture techniques have been divided into two categories- sensor based and vision-based techniques. In short, the hand gestures are captured through a camera in the devices. The captured image of hand gesture is then processed to perform the specific control functions. The processing is done by the pre-written program code. In this, each gesture sign is assumed for a function.

1. INTRODUCTION

Gestures are the movement of any body part used to convey the meaningful information. Communication through gestures has been widely used by humans to express their thoughts and feelings. Human gesture is a mode of non - verbal interaction. Gestures recognition is the process of identifying gestures performed by human. Gestures has been classified in two categories static and dynamic. Static gestures refer to still body posture and dynamic refers to movement of body part. Gestures can be performed with any body part like head, face, arms, hands, etc.

The hand gestures are mainly used for the purpose of controlling the devices from a distance without the help of any intermediates like the remote. This is a direct interaction between the devices and the users. In general, the framework for hand gesture recognition includes taking in the input i.e. hand image, hand detection, preprocessing i.e. removing unwanted features, segmentation, feature extraction, recognition (gesture dictionary) and execution (commands).

Hand Gestures Recognition techniques have been divided into two categories- Sensor based and Vision Based recognition. Sensor based recognition collects the gesture data by using one or more different types of sensors. These sensors are attached to hand which record to get the position of the hand and then collected data is analyzed for gesture recognition. Eg data glove, Wii controller, EMG sensors, accelerometer sensors, etc. Sensor based recognition has certain limitations

a) it requires a proper hardware setup which is very expensive.
b) it hinders the natural movement of the hand.

So to overcome the limitation of sensor based recognition vision based techniques came into existence. Vision based techniques make use of camera to capture the image for hand gesture. Vision based recognition make use of many image processing algorithms to get hand posture information and movement of hand. This approach recognizes gesture from shapes, orientations, contours, and color or motions features of a hand. Colored markers are an example of vision-based recognition. But the vision based recognition also has some limitations that it is affected by illumination changes and cluttered backgrounds.

PLATFORM	Python3.5
LIBRARIES	OpenCV, numpy
HARDWARE REQUIREMENTS	Camera/Webcam
ALGORITHM	Contour detection and convex hull of palm region

2. LITERATURE REVIEW

Review on work done by various researchers on hand gesture recognition:

- ★ Chang-Yi Kao. et. Al [3] : hand gesture recognition technique based on path of hand motion by using HMM as a classifier. Eight different kinds of gestures have been developed using either single hand or both hands.

- ★ Amit Gupta, et. Al [4]: hand gesture recognition technique which utilizes an FPGA based smart camera for gesture analysis. The experiment is performed using shape-based features. For performing the experiment images of 25 different persons is captured and is found that system is able to recognize 10 different gestures with accuracy 94.40%.

- ★ Jing Lin, et. Al [5]: based on histrograms of oriented gradients (HOG) in order to remove the hindrance caused by cluttered background during hand localization and modeling.The technique is tested on 6 standard gestures and average accuracy of 91.7% is obtained but this system is able to work for complicated gestures.

- ★ Zhou Ren, et. Al [6]: developed a robust part-based technique using kinect sensor keeping in view the limitations of glove based and vision-based techniques. kinect sensor is captures both color images and depth maps corresponding to that image. Using depth maps, hand can be easily detected even in cluttered background also by using depth thresholding. After detecting the hand, it is represented by its finger parts using time series curve. Then for gesture recognition a dissimilarity measure called Finger-Earth Mover's Distance (FEMD) is proposed which can recognize noisy hand contours as compared to other recognition methods and is robust to change in scale, orientation, local distortions and background conditions. For experiments dataset consisting of 10 person, 10 gestures and

10 cases/gestures id used and achieves accuracy of 93.2% and system is tested on two real time applications.

★ Jos´e Manuel Palacios[7]: developed hand gesture recognition technique using RGB-D sensors taking the advantage of depth information to remove the problems caused by lightning conditions and cluttered background. The proposed methodology includes four basic steps - Hand segmentation, Feature extraction, Static gesture classification and Dynamic gesture classification. For hand segmentation skin color segmentation and background subtraction is used. For static gesture recognition fingertip detection is used and fingertip is detected using maximum curvature and convexity defects. For dynamic, Euclidean distance and direction is used. The experiment is performed on dataset consisting of 90 images of 9 different persons with 10 dynamic and 6 static gestures and achieves Precision- 92.1% and Recall-83.3%.

★ Paulo Trigueiros, et. Al[8]: proposed a general human computer interaction system based on hand gestures. The system uses vision based and approach as they have advantage compared to traditional data glove approaches. Basically, the experiments consists of three steps- acquisition and preprocessing, feature extraction and recognition. For recognition machine learning classifier are used though they are not the only option but hereare used keeping in the view the extracted feature – centroid distance. For static geture recognition SVM is used and for dynamic HMM is used. The experiments obtains the 99.7% accuracy with SVM and 93.7 % with HMM with 11 predefined gestures.

★ Chetan Dhule, et. Al[9]: proposed a vision based hand gesture recognition technique for human computer interaction. They proposed the method keeping in view that most the earlier methods are based on gesture recognition algorithms that require ANN training which is very time consuming and is not much accurate. So by using color detection techniques they develop real time application to restrict the mouse's motion in windows by detecting change in pixel value of RGB colors and which is possible without ANN training.

3. Comparative study

TABLE 1: COMPARISON OF VARIOUS TECHNIQUES Year	Image acquisition Method	No. of Gestures	Segmentation Technique	Features Extracted	Recognitin Method	Accuracy
2011 [3]	Webcam	8	Bressenham's midpoint circle scan-conversion algorithm	Hand Orientation	HMM	96%
2012 [4]	CMOS image sensor	10	Illumination compensation of RGB color, Skin color segmentation	Shape based features- Area of hand , perimeter of hand, thumb detection, Radial profile	-	94.4%
2013 [5]	CCD cameras	6	Histogram oriented gradient	Velocity and angle	Mahalanobis distance	91.7%
2013 [6]	Kinect sensor	14	Depth thresholding	Finger's Earth mover's Distance	Template Matching	93.2%
2013 [7]	RGB Sensor	10- static 6- Dynamic	Skin color segmentation ,depth thresholding	Fingertips using curvature and convexity defects, Euclidean	Feature based classifier	Precision- 92.1% Recall - 83.3%

				distance, direction		
2014 [8]	Kinect Camera	11	_	Centroid distance	SVM	99.7%
					HMM for dynamic	93.7%
2014 [9]	Webcam	-	Skin color detection, Background detection	Pixel extraction , position of RGB color	-	-
2014 [10]	Kinect device	10	Depth Thresholding	Fourier Descriptors,	Nearest Neighbor classifier	99.5
				Apex structure Of Hand Contour (Finger Information)		96.3
2014 [11]	Depth Imaging sensor	4	Background subtraction	Hand silhouette, state of fingers.	Random forests	98.5%
2014 [12]	Kinect depth camera	9	Depth thresholding	Superpixel earth movers distance	Template Matching	99.2%

4. PROPOSED METHOD

Algorithm: Contour detection and convex hull of palm region algorithm

1.Capture frames and convert to grayscale

Our ROI i.e. region of interest, is the the hand region. Images of the hand are captured and converted to grayscale. We convert an image from RGB to grayscale and then to binary in order to find the ROI i.e. the portion of the image we are further interested for image processing. By doing this our decision becomes binary: "yes the pixel is of interest" or "no the pixel is not of interest".

2.Blur image

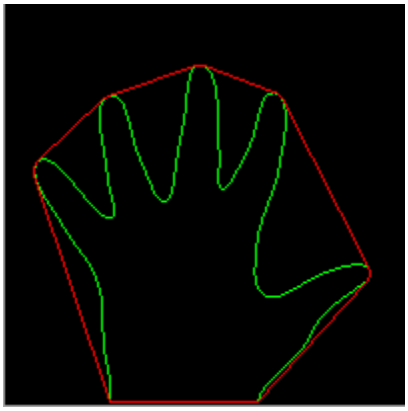
Gaussian Blurring is used on the original image. We blur the image for smoothing and to reduce noise and details from the image. We are not interested in the details of the image but in the shape of the object to track. By blurring, we create smooth transition from one color to another and reduce the edge content. We use thresholding for image segmentation, to create binary images from grayscale images.

3.Thresholding

In very basic terms, thresholding is like a Low Pass Filter by allowing only particular color ranges to be highlighted as white while the other colors are suppressed by showing them as black.

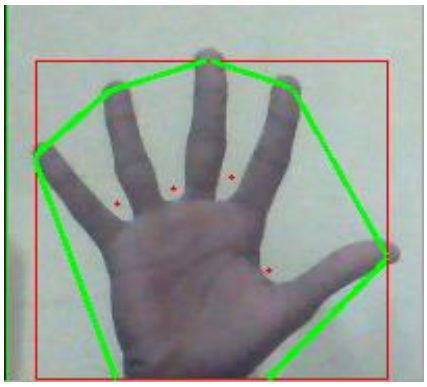
Otsu's Binarization method is used. In this method, OpenCV automatically calculates/approximates the threshold value of a bimodal image from its image histogram. But for optimal results, we may need a clear background in front of the webcam which sometimes may not be possible.

4. Draw contours



5. Find convex hull and convexity defects

We now find the convex points and the defect points. The convex points are generally, the tip of the fingers. But there are other convex point too. So, we find convexity defects, which is the deepest point of deviation on the contour. By this we can find the number of fingers extended and then we can perform different functions according to the number of fingers extended.



4.PYTHON CODE

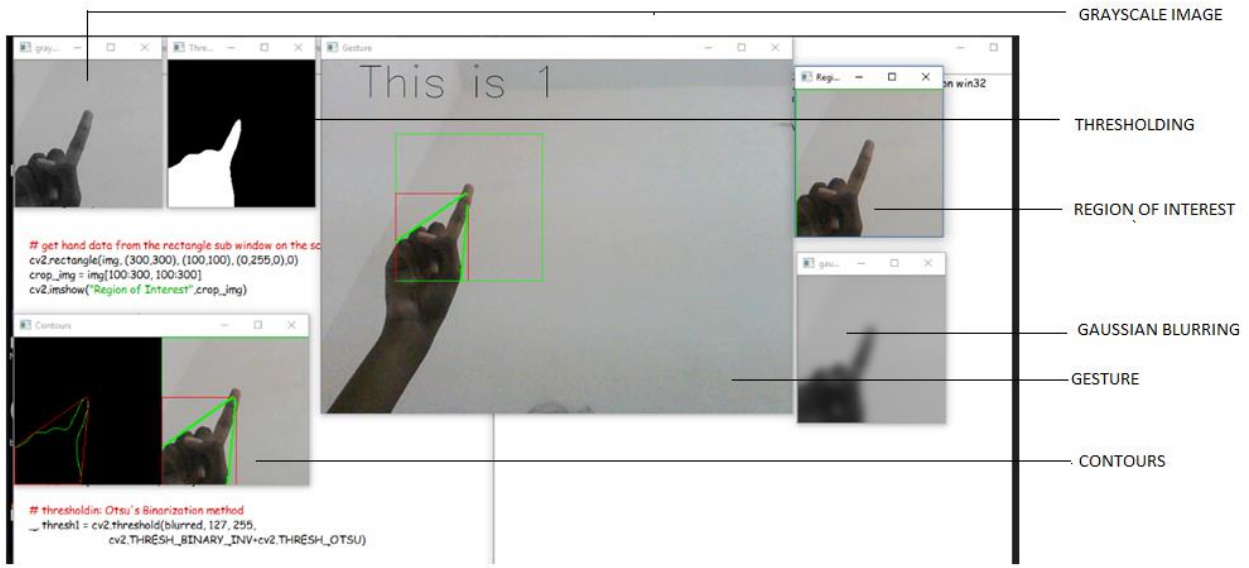
```
import cv2
import numpy as np
import math

cap = cv2.VideoCapture(0)
while(cap.isOpened()):

    # read image
    ret, img = cap.read()

    # get hand data from the rectangle sub window on the screen
    cv2.rectangle(img, (300,300), (100,100), (0,255,0),0)
    crop_img = img[100:300, 100:300]
    cv2.imshow("Region of Interest",crop_img)

    # convert to grayscale
    grey = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
    cv2.imshow('grayscale image',grey)

    # applying gaussian blur
    value = (35, 35)
    blurred = cv2.GaussianBlur(grey, value, 0)
    cv2.imshow('gaussianblur',blurred)

    # thresholdin: Otsu's Binarization method
    _, thresh1 = cv2.threshold(blurred, 127, 255,
                               cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
```

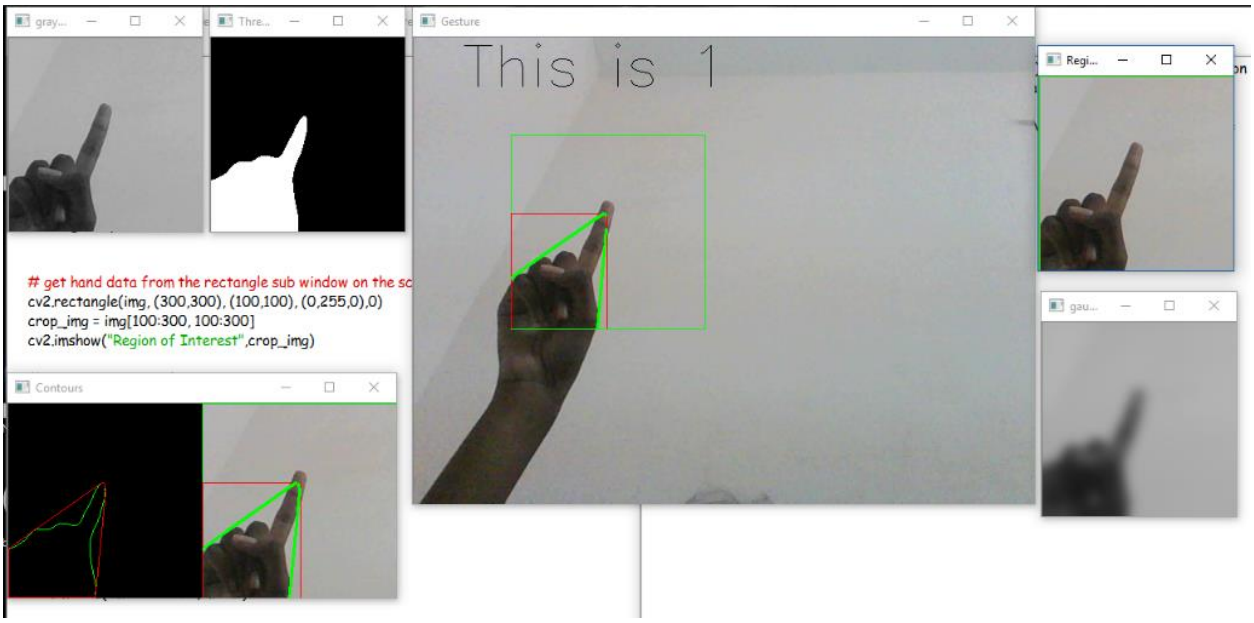
```
# show thresholded image
cv2.imshow('Thresholded', thresh1)

# check OpenCV version to avoid unpacking error
(version, _, _) = cv2.__version__.split('.')

if version == '3':
    image, contours, hierarchy = cv2.findContours(thresh1.copy(), \
        cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
elif version == '2':
    contours, hierarchy = cv2.findContours(thresh1.copy(),cv2.RETR_TREE, \
        cv2.CHAIN_APPROX_NONE)

# find contour with max area
cnt = max(contours, key = lambda x: cv2.contourArea(x))

# 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)

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

# 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)

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

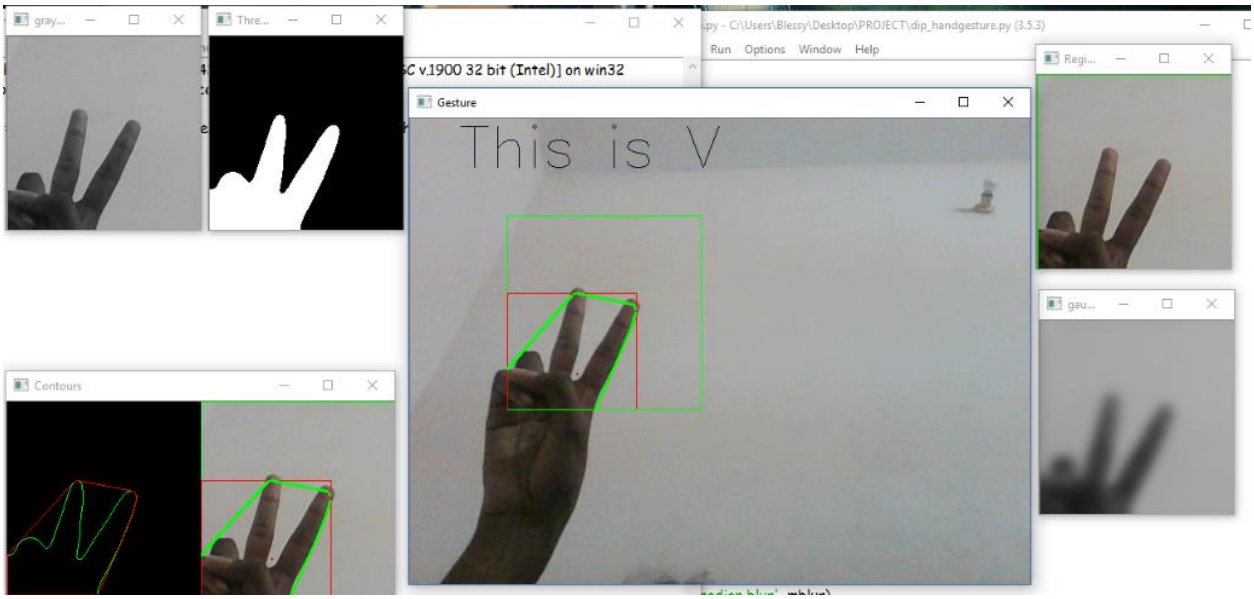
```
# finding convexity defects
defects = cv2.convexityDefects(cnt, hull)
count_defects = 0
cv2.drawContours(thresh1, contours, -1, (0, 255, 0), 3)

# 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)
```

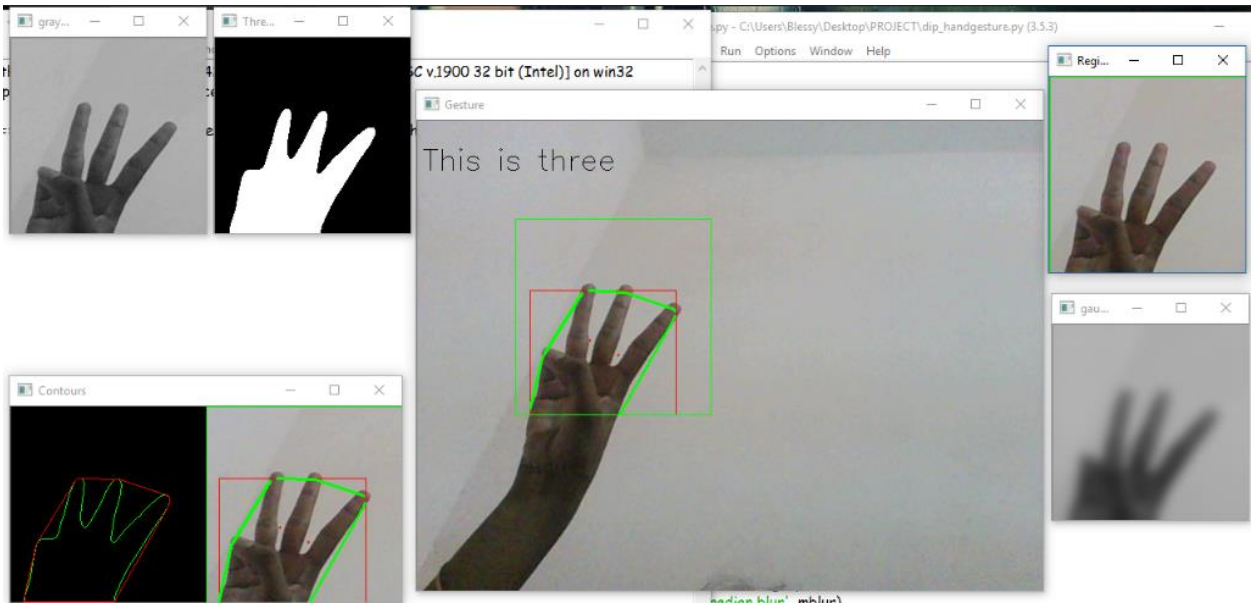
```
            cv2.line(crop_img,start, end, [0,255,0], 2)
            #cv2.circle(crop_img,far,5,[0,0,255],-1)


        # define actions required
        if count_defects == 1:
            cv2.putText(img,"This is V  ", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
        elif count_defects == 2:
            str = "This is three"
            cv2.putText(img, str, (5, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, 2)
        elif count_defects == 3:
            cv2.putText(img,"This is  4", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
        elif count_defects == 4:
            cv2.putText(img,"Hi!!", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
        else:
            cv2.putText(img,"This is 1", (50, 50),\
                        cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
    # show appropriate images in windows
        cv2.imshow('Gesture', img)
        all_img = np.hstack((drawing, crop_img))
        cv2.imshow('Contours', all_img)
        k = cv2.waitKey(10)
        if k == 27:
            break
```

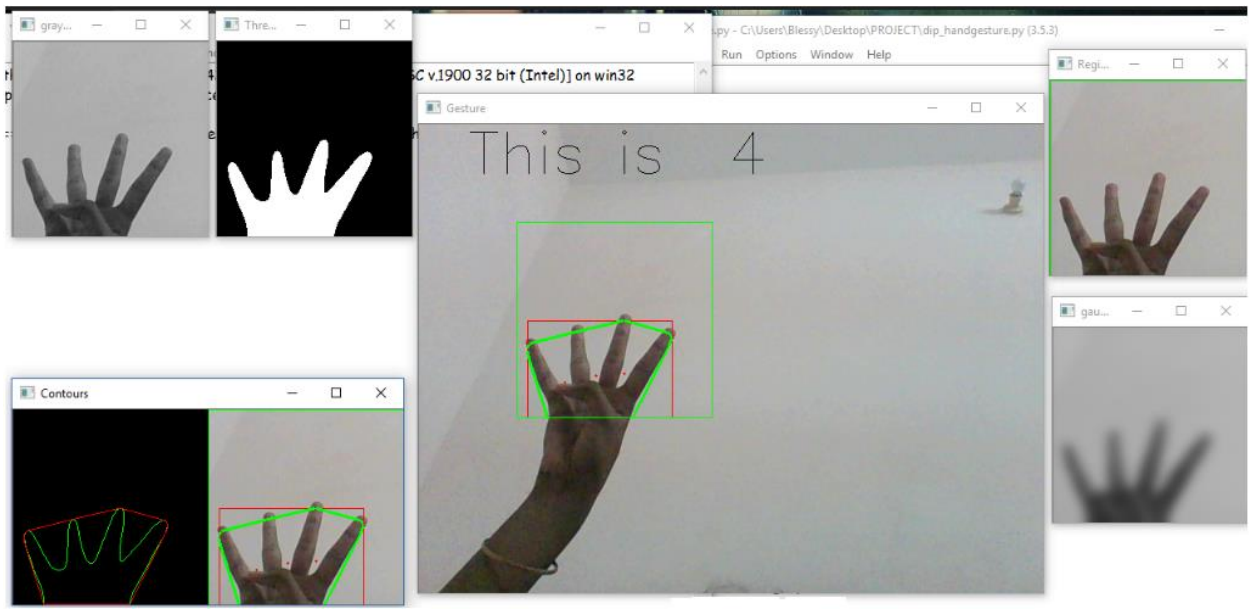
5. RESULTS AND DISCUSSION

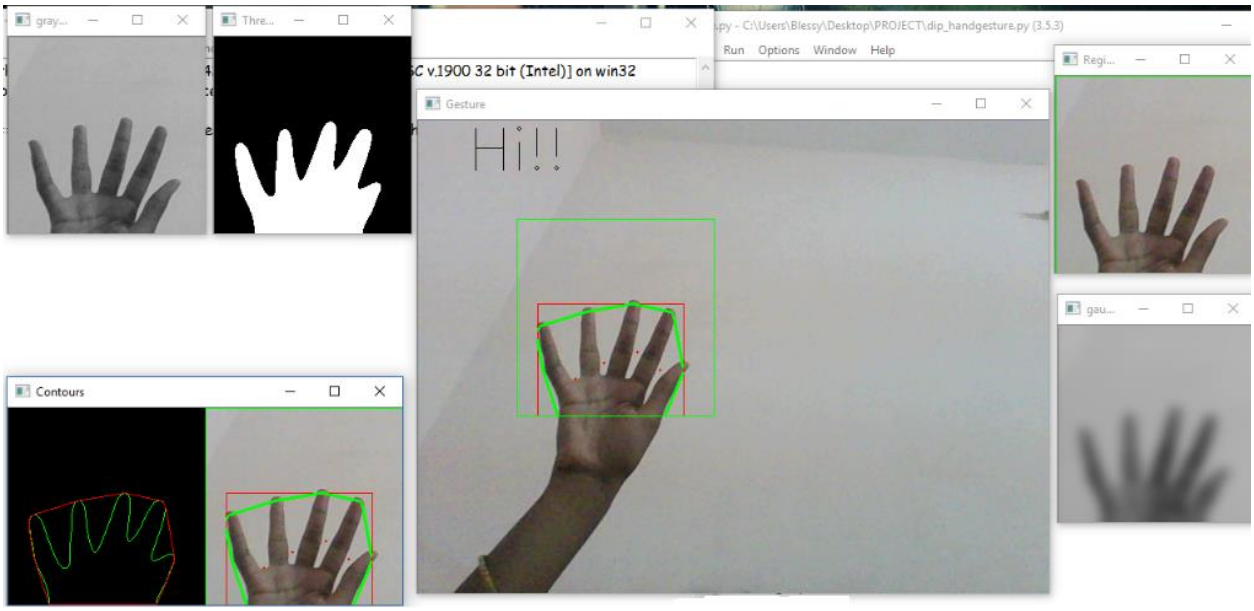





gray...
Thre...
C v.1900 32 bit (Intel)] on win32
Run Options Window Help
Regi...
Gesture
This is V
gau...
Contours


gray...
Thre...
C v.1900 32 bit (Intel)] on win32
Run Options Window Help
Regi...
Gesture
This is three
gau...
Contours


gray...
Thre...
Run Options Window Help
C v.1900 32 bit (Intel)] on win32
Gesture
This is 4
Regi...
gau...
Contours


gray...
Thre...
Run Options Window Help
C v.1900 32 bit (Intel)] on win32
Gesture
Hi!!
Regi...
gau...
Contours

5. CONCLUSION

Hand Gesture Recognition has number of applications like human computer interactions, robotics, sign language recognition, etc. Lots of work has been done hand gesture recognition field. We have discussed the basic methodology and various techniques of hand gesture recognition. In the future, hand gesture can be combined with other gestures like body posture, face recognition, etc. for better results.

6. REFERENCES

- Gesture Recognition: A Survey
- Sushmita Mitra, Senior Member, IEEE, and Tinku Acharya, Senior Member, IEEE
- Vision based hand gesture recognition for human computer interaction: a survey Siddharth S, Rautaaray , Anupam Agrawal
- Eleventh International Multi-Conference on Information Processing-2015 (IMCIP-2015) Human Computer Interaction using Hand Gesture
- Ram Pratap Sharma and Gyanendra K. Verma
- National Institute of Technology Kurukshetra, Kurukshetra, Haryana 136 119, India
- https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=5&cad=rja&uact=8&ved=0ahUKEwiZudWS2_7WAhUGpo8KHUCTDzcQFgg7MAQ&url=http%3A%2F%2Fwww.ijarcsms.com%2Fdocs%2Fpaper%2Fvolume3%2Fissue5%2FV3I5-0034.pdf&usg=AOvVaw3K33nZyx190bqK0Qq0ETse
- www.Wikipedia.com
- https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwj3oezt6f7WAhUHTo8KHRzFBP8QFggqMAA&url=https%3A%2F%2Fpdfs.semanticscholar.org%2F7785%2F7eef494b30663e0c17270486f35851246e6d.pdf&usg=AOvVaw2lS76xPV7QXEEOa6ni5THM
- https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwipuNWL6v7WAhXIqI8KHVZBy0QFgglMAA&url=https%3A%2F%2Fpdfs.semanticscholar.org%2F7785%2F7eef494b30663e0c17270486f35851246e6d.pdf&usg=AOvVaw2lS76xPV7QXEEOa6ni5THM
- Haitham Hasan , Sameem Abdul-Kareem, "Human–computer interaction using vision-based hand gesture recognition systems: a survey", Neural Comput & Applic,2013.
- Fahn, C.S,Sun, H, "Development of a fingertip glove equipped with magnetic tracking sensors", Sensors 2010,vol. 10 ,pp. 1119–1140.,2010
- Chang-Yi Kao,Chin-Shyurng Fahn, "A Human-Machine Interaction Technique: Hand Gesture Recognition Based on Hidden Markov Models with Trajectory of Hand Motion", Procedia Engineering 15, pp. 3739 – 3743,2011.
- Amit Gupta, Vijay Kumar Sehrawat, Mamta Khosla, "FPGA Based Real Time Human Hand Gesture Recognition System", Procedia Technology 6 ,pp. 98 – 107, 2012.

- Jing Lin, Yingchun Ding, "A temporal hand gesture recognition system based on hog and motion trajectory", Optik 124 pp.6795– 6798, 2013.
- Z. Ren, J. Yuan, J. Meng, and Z. Zhang, "Robust part-based hand gesture recognition using Kinect sensor," IEEE Trans. Multimedia, vol. 15, no. 5, pp. 1110–1120, Aug. 2013.
- Jos´e Manuel Palacios, Carlos Sagues , Eduardo Montijano and Sergio Llorente , "Human-Computer Interaction Based on Hand Gestures Using RGB-D Sensors" , Sensors 2013, vol.13, pp. 11842-11860, 2013.
- Paulo Trigueiros,Fernando Ribeiro, Luis Paulo Reis "Generic System for Human-Computer Gesture Interaction" ,IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), Espinho, Portugal ,pp. 175-180, May 2015.
- Chetan Dhule ,Trupti Nagrare , "Computer Vision Based Human-Computer Interaction Using Color Detection Techniques" , Fourth International Conference on Communication Systems and Network Technologies, pp.934-938, 2014.
- Stergios Poularakis and Ioannis Katsavounidis, "Finger Detection And Hand Posture Recognition Based On Depth Information", IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP), pp. 4329-4333, 2014.
- Dong-Luong Dinha, Jeong Tai Kimb, and Tae-Seong Kimc, "Hand Gesture Recognition and Interface via a Depth Imaging Sensor for Smart Home Appliances", 6th International Conference on Sustainability in Energy and Buildings, SEB-14, pp. 576 – 582 ,2014.
- Chong Wang,Zhong Liu, and Shing-Chow Chan, "Superpixel-Based Hand Gesture Recognition With Kinect Depth Camera", IEEE Transactions On multimedia, vol. 17, no. 1, pp. 29-39, January 2015.