REAL-TIME HAND GESTURE RECOGNITION USING FORE-ARM CONTOUR

CS 603: Computational Photography

Project Phase II

Nikhil Tank 13110070
Dept. of Electrical Engineering,
Indian Institute of Technology Gandhinagar

***Abstract* — This is a report of progress done in field of Real-time hand gesture recognition using fore-arm contour.**

Keywords—Thresholding, contours, convex hull, convex defects, mouse_gui

I. INTRODUCTION

The objective of the project is to detect hand gestures seen through webcam of a computer. The python script along with OpenCV lib is used to analyze the live-feed video form the webcam, and process the pre-programmed output.
Gesture recognition is becoming one of the most concerned topics, because of the ease of use. Processing live feed from camera and processing it to detect programmed output is widely used in modern TV, Laptops, PSP's, etc. The idea of working on a project based on hand gesture sensing came from gesture-detection software called flutterapp. https://flutterapp.com/.

II. RELATED WORK

There are many related works which somehow ties with the gesture recognition project. These methods can be implemented to find out features from different hand gestures.

1) Finger detection by special gloves with LED:

In this method, different fingers are equipped with LED of different colors. The position of each figure tip can be accurately detecting using color matching. But, the restrictions of the method are: must wear the glove and be under dark environment. [1]

2) Detecting Background (Image matting):

This is a process of obtaining a Foreground image from a Background image, using the equation.

$$C = (1\text{-alpha}) . B + \text{alpha} . F$$

Where C is captured image, α is matting coefficient, and B, F are foreground and Background images respectively. But, the restrictions of the method is that it uses recursive steps to find unknowns which make it a slow process.

3) Skin color detection using HSV color space

This is a method we convert RGB to HSV color space to extract human skin areas. The skin color areas will be represented in binary image. Hence we will perform morphology processing including erosion and dilation. Noises will be eliminated by erosion and the contour of skin color area will be smoothed by dilation. [1]
The other alternative process for fast feature detection is thresholding which will be used the project.

4) Optimum Image thresholding of Gaussian smooth image

This is a method in which Gauss smooth image is passed through a threshold function which remove unnecessary data from the image, by calculation the optimal threshold value based on the OSTU's algorithm for different image differently. [7]
This is the fastest method to remove unnecessary data from image. Hence this method is used in our approach.

III. Methodology:

The Gesture recognition is highly dependent on postures made by hand. Each gesture is unique in its way. But keeping in mind the broad similarity between 2 gestures, they can be categorized based on angles which figures make between with one another, along with the no. of figures involved.
Our focus to process the live feed data from webcam of a laptop and use that data to determine the no. of fingers observed in the image and run some operation based on them.
So, we have to identify the position of palm and no. of fingers.

The sequence of operation is:

1) Capture:

First, capturing the live video using webcam, using the OpenCV command “cv2.VideoCapture(0)”. This will capture a live video, frame by frame from webcam, which can be processed like an ordinary image.

2) BGR to Gray:

To make the processing fast the image information is converted from BGR to Gray, this will decrease the no. of variable to 1/3rd. To make the process even faster the region of interest is reduced by instructing the user to keep his/her hand in the top-right quarter of the recorded area. This area is now cropped.

By doing these operations the speed of operation is increased 12 times.

Since are comfortable of seeing ourself in mirror, so we have flipped the frames horizontally.

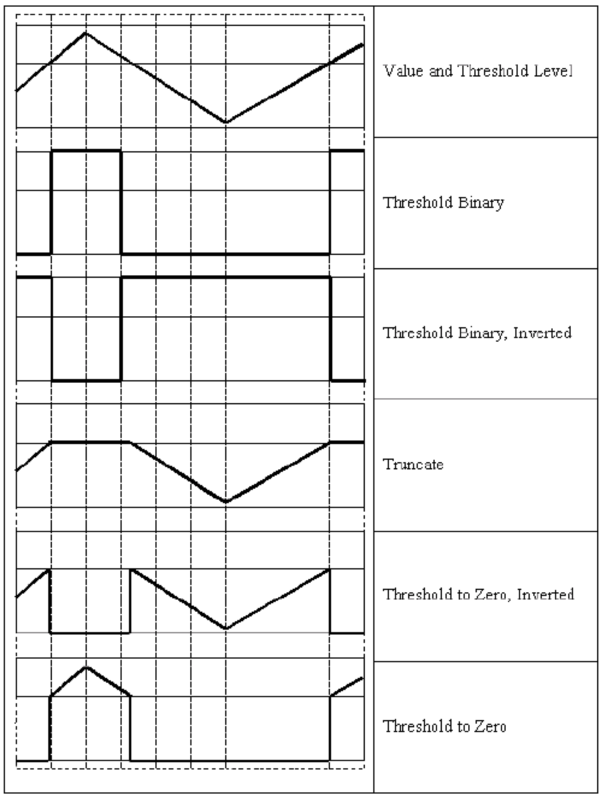
3) Blur image:

Then to remove unnecessary details and noise we have used Gaussian blurring, since we are not interested in sharp-details of the image. By blurring we will get smooth transition from contrast to another and this reduce the edge content.
The Gaussian blur command “cv2.GaussianBlur” is used with sigma (15,15).

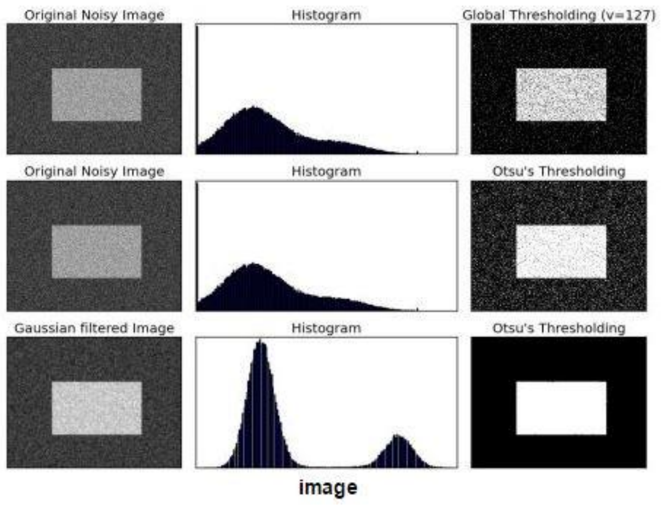
4) Threshold:

The thresholding will act as a low pass filter by allowing a particular color range to be highlighted as white (region of hand), while the other colors becomes black.
(Note: we may need a clear background behind the hand to separate the hand clearing from other objects).
To make the thresholding independent of different surrounding we are using OSTU threshold along with Binary_Inv Thresholding, which will calculation the optimal threshold value based on the OSTU’s algorithm for different image differently.
Description of thresholding tried:

```
#    ret,thresh1 = cv2.threshold(blur,150,255,cv2.THRESH_BINARY)
#abovethres=1, belowthresh=0
#    ret,thresh2 =
cv2.threshold(blur,150,255,cv2.THRESH_BINARY_INV)
#abovethres=0, belowthresh=1
#    ret,thresh3 = cv2.threshold(blur,150,255,cv2.THRESH_TRUNC)
#works as a chopper/low pass, chops higgher value
#    ret,thresh4 =
cv2.threshold(blur,150,255,cv2.THRESH_TOZERO) #works as
highpass
#    ret,thresh5 =
cv2.threshold(blur,150,255,cv2.THRESH_TOZERO_INV) #works as
a chopper, turn chopped value zero
#    ret,thresh6 = cv2.threshold(blur,0,255,cv2.THRESH_OTSU)
```


Value and Threshold Level
Threshold Binary
Threshold Binary, Inverted
Truncate
Threshold to Zero, Inverted
Threshold to Zero

[8]



[7]

5) Draw contours:

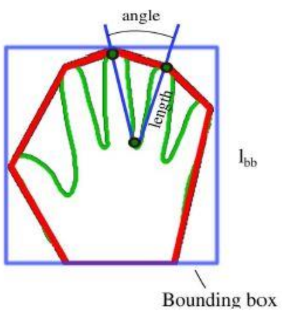
To obtain correct region of interest we have drawn contours along the threshold image. This will give outlines of the hand. [9][10]
We have used "cv2.findContours" command in OPENCV lib to calculate contours.
(This is could be done using canny edge filter, which is inbuilt feature of OpenCV lib.)
To find the contour that enclosed the max area of the image, we have tried to use *max* method to compare the contours. [9][10]

7) Convex Hull:

Keeping in mind the importance of fingers in gesture recognition we have to find the position of fingers (basically no.). This is done by finding the convex points which are generally, tip of the fingers.
Convex Hulls are the outlines of the contours, similar to first order reconstruction. This makes a line between 2 convex points outside the hand.
The "cv2.convexHull(source[])" command is used to give the hull paths, and cv2.convexHull(source[, returnPoints = False) was used to get the hull pints in the image.

6) Convex point and defect points:

We find convexity defects, which is the deepest point of deviation on the contour (valley region between fingers). By this we can find the number of fingers extended and then we can perform different functions according to the number of fingers extended.
Determining valid convex defect is the most important part of the work as. Only the defects which hare present between the figures are allowed. So using human anatomy we can say that angle between 2 figures cannot exceed than 80 degree, so we have used cosine formula to identify the angle between the start-far point and end-far point for a defect. [11]



For eg:

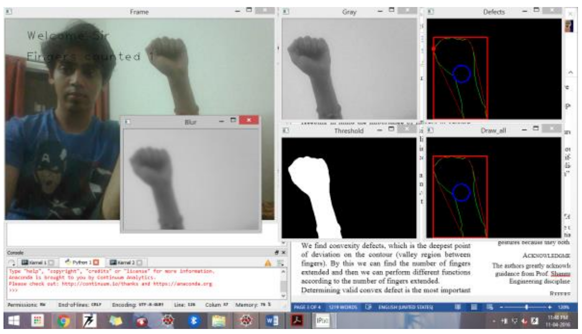
- Fist and one figure corresponds to 1 convex defect
- Two figure corresponds to 2 convex defect, and similarly.

7) Implementation of gestures:

These counted fingers could be used to run different operation, like triggering if-else command, make the mouse pointer move and click, etc. This is done using "win32api" and "win32con" library.

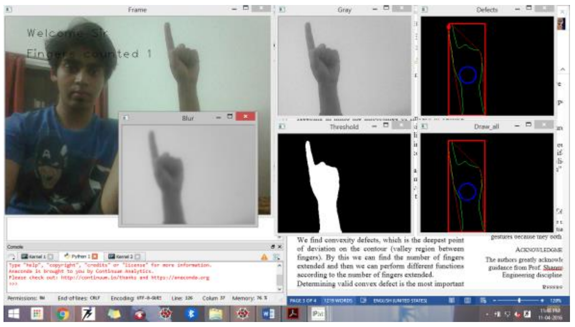
Gallery:

Fist:



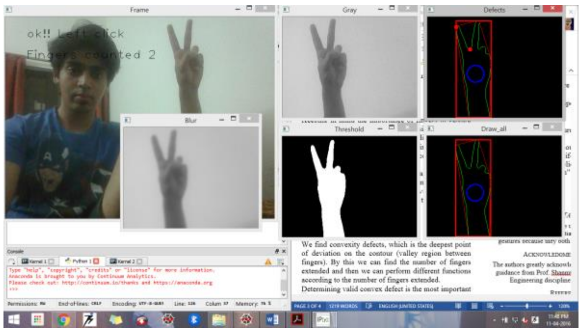
```
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
>>>
```

One:

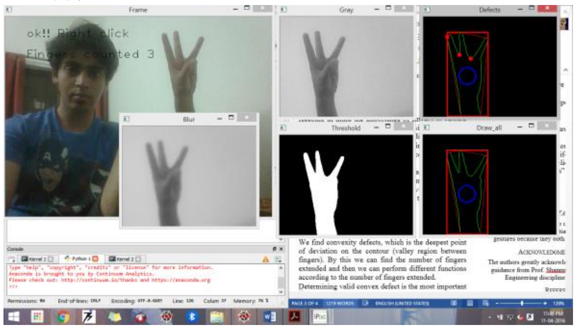


```
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
>>>
```

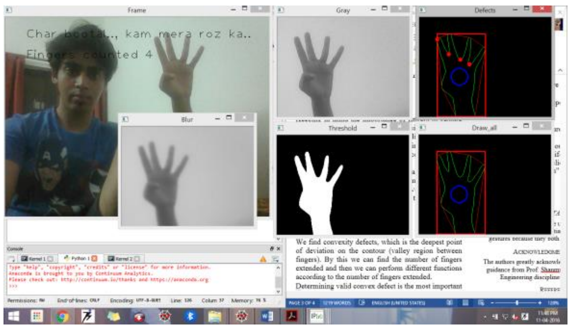
Two:



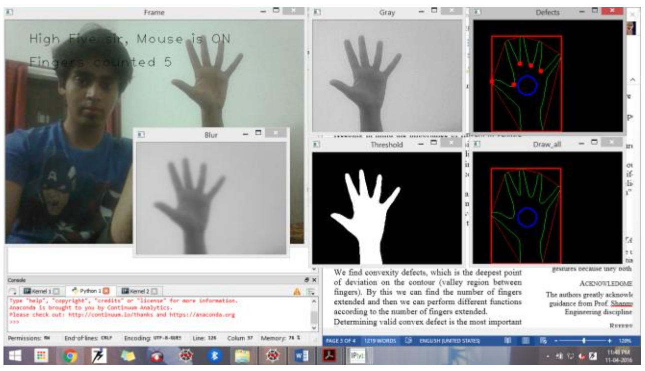
Three:



Four:



Five:



LIMITATION

The background should be uniform behind the image.

ACKNOWLEDGMENT *(Heading 5)*

The authors greatly acknowledge continues support and guidance from Prof. Shanmuga R, from the Electrical Engineering discipline at IIT Gandhinagar.

REFERENCES

[1] "Real-Time Palm Tracking and Hand Gesture Estimation, Based on Fore-Arm Contour"

[2] Wen-Yuan Chen, Yu-Ming Kuo, Chin-Ho Chung

[3] http://creat-tabu.blogspot.in/2013/08/opencv-python-hand-gesture-recognition.html

[4] http://vipulsharma20.blogspot.in/2015/03/gestur e-recognition-using-opencv-python.html

[5] https://www.youtube.com/watch?v=QYiypuWZ

[6] http://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html

[7] http:// docs.opencv.org/3.1.0/d7/d4d/tutorial_py_thresholding.html#gsc.tab=0

[8] http://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html

[9] http://docs.opencv.org/3.1.0/d4/d73/tutorial_py_contours_begin.html#gsc.tab=0

[10] http://docs.opencv.org/3.1.0/dd/d49/tutorial_py_contour_features.html#gsc.tab=0

[11] http://simena86.github.io/blog/2013/08/12/hand-tracking-and-recognition-with-opencv/

[12] http://www.allaboutcircuits.com/textbook/reference/chpt-5/non-right-triangle-trigonometry/