"Gesture based Web-Browser"

A

Project Report

submitted in partial fulfillment of the

requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

by

Name	Roll No.
PAWAN CHATURVEDI	R164217036
PRADEEP YADAV	R164217038
PRANJAL RAI	R164217039
RAGHWANDRA SINGH	R164217046

under the supervision of

Ms. Roohi Sille

Assistant Professor
in
Department of Systemics

SCHOOL OF COMPUTER SCIENCE
UNIVERSITY OF PETROLEUM & ENERGY STUDIES
Bidholi, Via Prem Nagar, Dehradun, UK

November – 2020

UPES

UNIVERSITY WITH A PURPOSE

CANDIDATE'S DECLARATION

We hereby certify that the project work entitled **" Gesture based Web-Browser"** in partial fulfilment of the requirements for the award of the Degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING with specialization in Internet of Things and Smart Cities and submitted to the Department of Systemics at School of Computer Science, University of Petroleum & Energy Studies, Dehradun, is an authentic record of our work carried out during a period from **August**, **2020** to **November**, **2020** under the supervision of **Ms. Roohi Sille, Assistant Professor in Department of Systemics**.

The matter presented in this project has not been submitted by us for the award of any other degree of this or any other University.

Pawan Chaturvedi, 36
Pradeep Yadav, 38
Pranjal Rai, 39
Raghwandra Singh, 46

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date: 07-11-2020

Dr. Neelu Jyoti Ahuja
Department of Systemics

Ms. Roohi Sille
Project Guide

ACKNOWLEDGEMENT

We wish to express our deep gratitude to our guide **Ms. Roohi Sille**, for all advice, encouragement and constant support she has given us throughout our project work. This work would not have been possible without her support and valuable suggestions.

We sincerely thank to our respected Program Head of the Department, **Dr. Neelu Jyoti Ahuja**, for her great support in doing our project in **Area (like IoT, Electronics etc.)** at **SoCS**.

We are also grateful to **Dr. Manish Prateek Dean SoCS**, UPES for giving us the necessary facilities to carry out our project work successfully.

We would like to thank all our **friends** for their help and constructive criticism during our project work. Finally, we have no words to express our sincere gratitude to our **parents** who have shown us this world and for every support they have given us.

Name	**Pawan Chaturvedi**	**Pradeep Yadav**	**Pranjal Rai**	**Raghwandra Singh**
Roll No.	**R164217036**	**R164217038**	**R164217039**	**R164217046**

ABSTRACT

These days scientists around the globe are effectively occupied with improvement of vigorous and productive sign language framework, all the more extraordinarily, hand gesture based human-machine interface for different applications. No matter how solid and incredible, we should be near machines and working alongside them. Also, that is the prime explanation hand gesture has become vital piece of non-verbal correspondence for individuals.

Internet Browsers is an exceptionally regular programming application which we utilize each day, consistently. Utilization of innovation can utilize internet browsers more and simpler.

The Internet is developing at an exceptionally quick pace. The utilization of internet browser is likewise developing. The Internet is an answer for everything. Presently, everybody has at any rate 2-3 most often visited site. Presently, imagine a scenario in which visits on those destinations can be made more straightforward. Gesture recognition helps in the environment known to be HMI (humans communication with the machine and interact) and that does not include touch of mechanical devices. Our project aims to address the current situation and tried to make the use of a web browser easier.

Keywords: Gestural interaction, Hand gesture recognition, Vehicular automation's, HCI (Human Computer Interaction), FEMD, gesture vocabulary, HMI(Human-machine Interface).

UPES

UNIVERSITY WITH A PURPOSE

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1. Hand gesture

Hand gesture detection systems are one of the hottest fields of research since it is of great significance in designing artificially intelligent Human-computer interfaces (HCI) [1] for vehicular applications. Gestures help to communicate a great deal. Seals, Regulators, Adapters, Metamorphic are particular sorts of gestures which are into training. A hand gesture driven system could bring in more naturalness and innovation for the increasingly popular gaming industry as demon-started by Nintendo's 'Wii' controller.[8] Such a hand gesture driven interface can free the users from using remote controllers for simple tasks such as issuing a command to a home entertainment system which may need start and play a music track or perhaps skip the current track.[9] Emerging technologies now make it possible to create gestural-interfaces for vehicles that make it easier for the driver to carry the tasks effectively.

1.2. Objectives

The Objective of our project is to design a robust and efficient system which will be able to control the browser via hand gestures i.e. different hand gestures will be assigned to different sites. So, if we show a particular hand gesture, a particular site will open up in a new tab.

1.3. Motivation

In today's world where technology is evolving so fast, Human can easily reduce its additional extra work using some interesting technology. One of the interesting application of this type of technology is 'Hand Gesture' . Think about your day to day life where you have to always opens up some common websites whenever you open your computer like Email, Facebook, Whats App, etc and for this you type URL in your web browser what is we can make this process a little bit easy like whenever you open your computer you just have to show different hand gestures in front of your camera and it opens up all regular websites for you like you can show one finger for Email, two finger for Facebook and three finger for Whats App it works. Think about a robot where you give a common task to a robot and as per your command it do its work same like this you give some command with your hands to your web browser and it do thinks according to that command.

This process does not requires any keyboard input and and makes quick access to all those common websites which takes extra time when you open them manually with your keyboard.

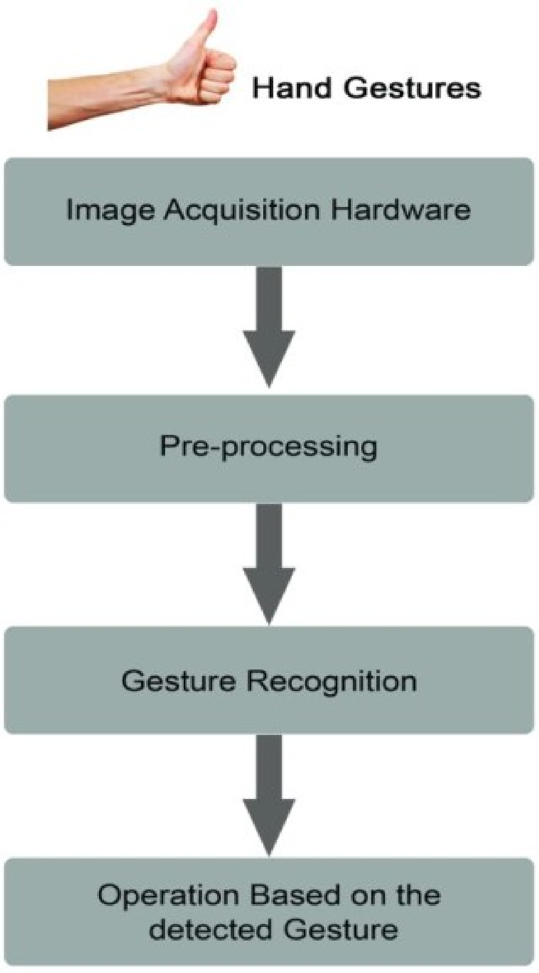

Hand Gestures
Image Acquisition Hardware
Pre-processing
Gesture Recognition
Operation Based on the detected Gesture

Fig 1.1 Hand Gesture Detection System

1.4. Pert Chart Legend

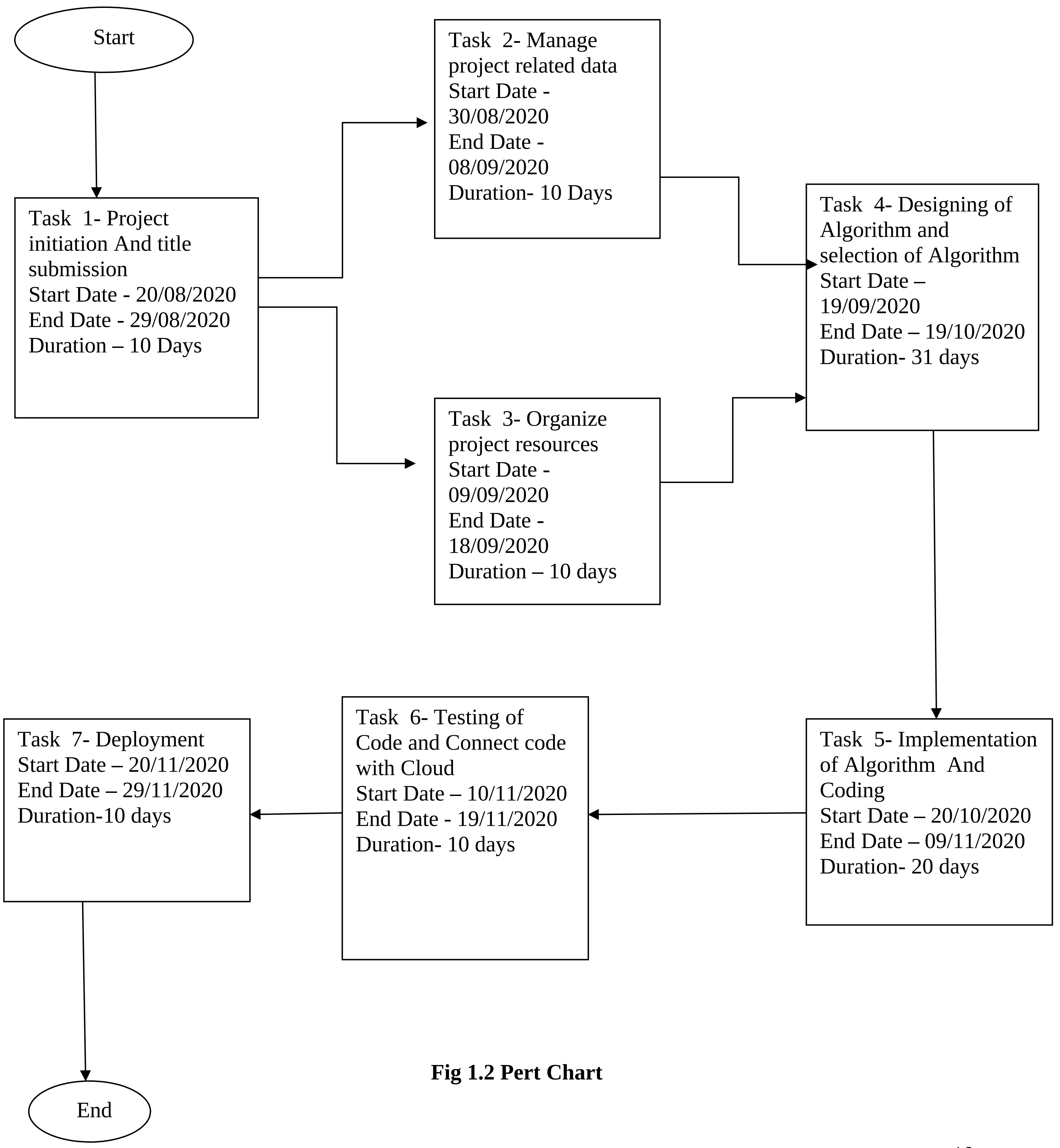


Fig 1.2 Pert Chart

2. SYSTEM ANALYSIS

2.1. Existing System

- Almost all the hand gesture detection systems mostly follow the mentioning steps in detecting and recognizing the gestures. Image acquisition, pre-processing and hand gesture recognition. [5] Image acquisition involves fetching the movements of the hands by any image acquisition hardware and the captured gestures are then moved to the second pre-processing segment. Pre-processing methods [4, 5] are mainly based on the combinations of several processing operations. Pre-processing is a process that receives the data obtained from image acquisition steps and then prepares the data for the following procedures. This is done in a particular way as that the data can be more easily and effectively processed by the next detection steps.
- Different pre-processing steps involve filtering, edge detection, histogram equalization, thresholding, etc.[5] Pre-processing is the step which determines the accuracy of the system. Image acquisition modules are different related to different detection systems. The modules can be an accelerometer sensor or a data glove or a in vision based approach a camera replaces the above mentioned modules. The data fed from the image acquisition section may be mostly error affected .Therefore pre-processing is the step which enhance the quality of the information from acquisition step and ultimately improves the accuracy of the detection systems.
- Recognition step identifies the processed hand movements as a particular gesture and then determines what the operation should be executed for a particular hand gesture [6]. Recognition step involves different algorithms for individual detection systems. In this paper it is described some of the gesture detection systems and the related algorithms used, results obtained, challenges faced etc. Rest of the paper is organized as follows. Section 2 describes the motivation and related work and section 3 presents some of the hand gesture detection systems, their recognition methods, applications and design challenges etc. Section 4 discusses a performance evaluative study and a hand gesture vocabulary is proposed and section 5 concludes the paper. This paper actually makes a

comparative analysis between the different existing hand gesture detection systems and finally on comparing these systems pointing out the features of Kinect sensor based vision gesture detection systems in detail in the conclusion section of the paper.

- Many gesture detection systems have been applied successfully to different fields with some good results. The reason for choosing hand as the gesture detection input in vehicular automation is, the hand is more flexible than any of the body part and more no gestures that a hand can be generated than that can be generated by the head or face [1]. So gesture ‘vocabulary’ will be more if hand movements as the gesture. In many of the areas gestures replaces some of the conventional input devices as they are found more feasible than the conventional types in some particular areas or applications.
- In 1987 a commercially available data glove was introduced, equipped with 15 or more sensors with different gesture tracking abilities. Power Glove was developed in 1989 which was commercialized by Mattel Intellivision as control device for the video game console. The P5 glove an updated version of the power glove in 2002 provides users intuitive interaction with 3D and virtual environments. Robert Wang and Papovic [7] proposed a colored glove based hand tracking system with the custom pattern imprinted on it. The proposed system was in expensive and which simplified the pose estimation problem. Feng Wang and Chong Wah [5] proposed a real time gesture detection system for lecture videos. Daeho Lee and Park [6] presented a universal remote control system based on computer vision. The motion and skin color of the hands are utilized to detect the waving hands and requesting control commands.

2.2. Proposed System

Recognition of human gestures comes within the more general framework of pattern recognition. In this framework, systems consist of two processes: the representation and the decision processes. The representation process converts the raw numerical data into a form adapted to the decision process which then classifies the data (see Fig 2.1).

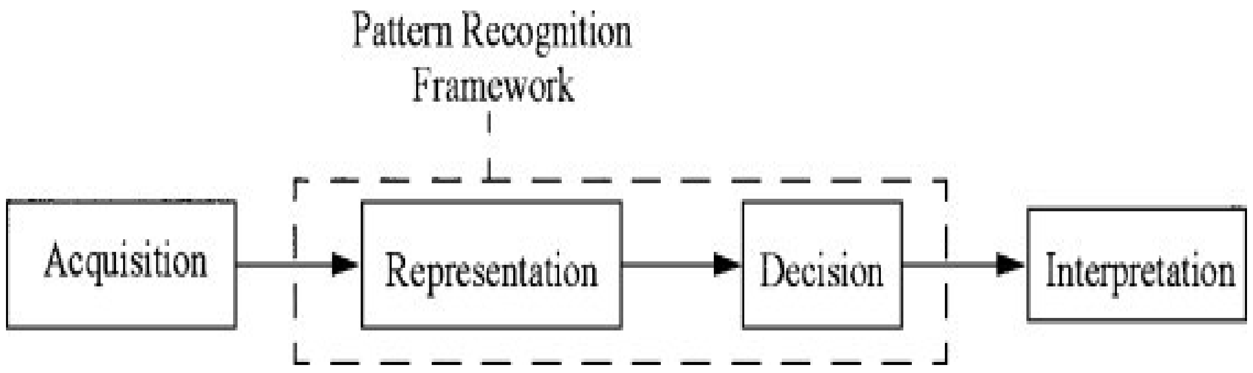


Fig 2.1 Pattern Recognition

We plan to design a system to solve above problem using image processing techniques and controlling the browser using hand gestures. At, first the user needs to enter the sites which he/she frequently visits and needs to control those sites using hand gestures. A user needs to enter the site names for the particular hand gestures. That's all, the user can now control the browser, open sites in new tabs using hand gesture and can even close the browser using the hand gesture. Thus, a hand gesture controlled browser is made. In this way a user can easily open the sites which he/she wants in the new tab using the hand gesture.

2.3. Procedure

1) First we import opencv library
2) Then we use cv2.VideoCapture() to capture a video image by image from Camera.
3) We use a loop till the frame is opened.
4) Then we read a image from the video frame, crop the image, convert this image to grayscale, apply gaussian blur onto the image, apply threshold technique onto the image.
5) After that we will check the version of the openCV library.
6) After that we will find the contours, max area , convex hull , find the convexity Defects.
7) After this we will come on the mathematical part of the code which is for the Counting the number of fingers we will apply Cosine rule and if the angle is less Then 90 then we will count it as finger.
8) After reading the number of finger we will open the tab in the default browser.

3. DESIGN

3.1. System Modelling

3.1.1. Flow Chart

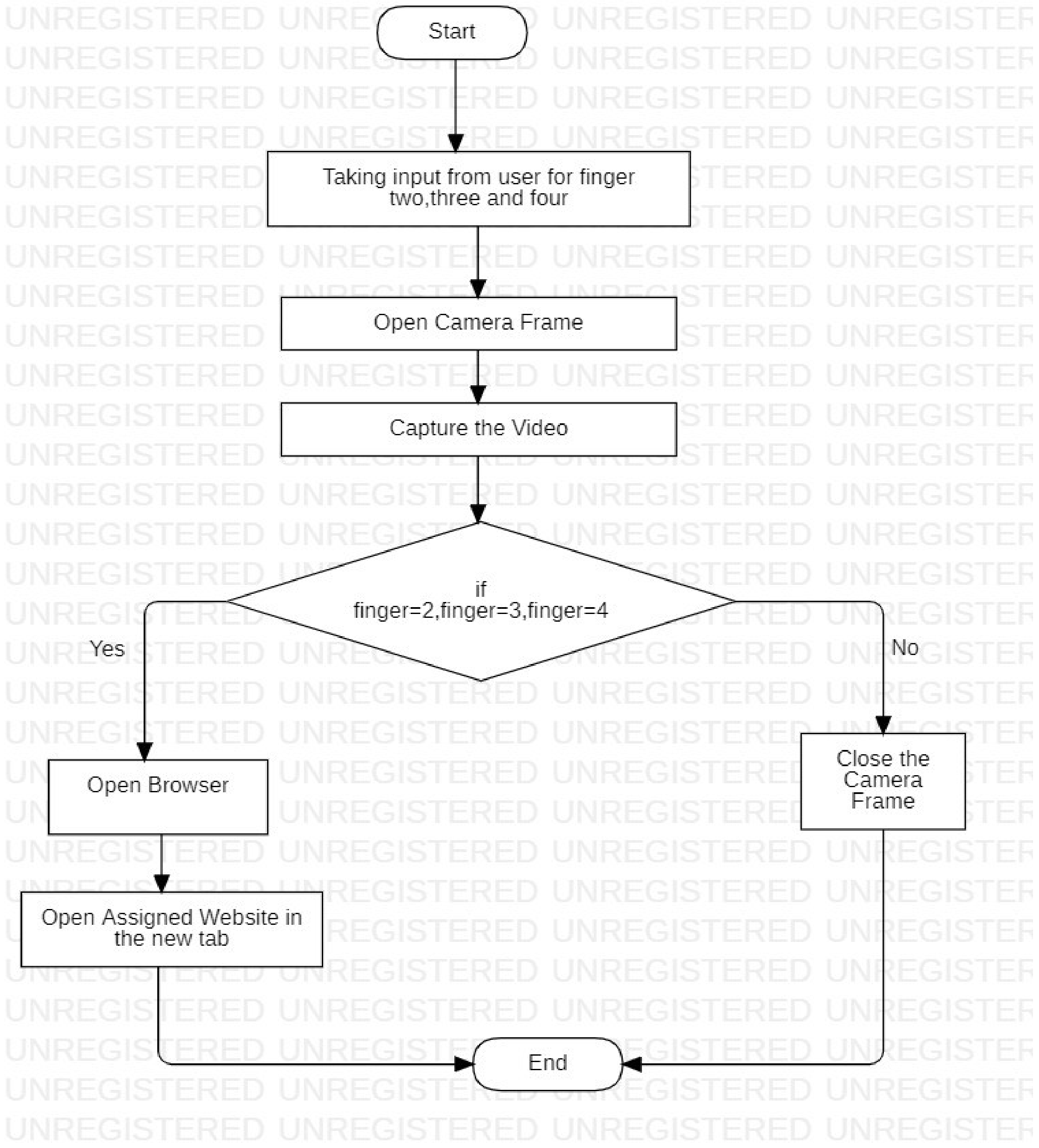

Start
Taking input from user for finger two,three and four
Open Camera Frame
Capture the Video
if finger=2,finger=3,finger=4
Yes
No
Open Browser
Open Assigned Website in the new tab
Close the Camera Frame
End

Fig. 3.1 Flow Chart

4. Hand Gesture based Web-Browser

4.1. Hand Gesture Web-Browser

This project is able to open your favourite websites using image processing techniques and controlling your browser using the hand gestures like two finger, three finger, four finger. First we are taking the variable and then user needs to enter the name of his f avourite websites for the three of the variable and with the help of opencv library a frame will opens up and user have to show the particular hand gesture for which he wants to open the website on showing the hand gesture browser will automatically opens the assigned website in the new tab.

5. IMPLEMENTATION

5.1. OpenCV | cv2.cvtColor() method

OpenCV-Python is a library of Python bindings designed to solve computer vision problems. `cv2.cvtColor()` method is used to convert an image from one color space to another.

Syntax: cv2.cvtColor(src, code[, dst[, dstCn]])

Parameters:
src: It is the image whose color space is to be changed.
code: It is the color space conversion code.
dst: It is the output image of the same size and depth as src image. It is an optional parameter.
dstCn: It is the number of channels in the destination image. If the parameter is 0 then the number of the channels is derived automatically from src and code. It is an optional parameter.

Return Value: It returns an image.

5.2. GaussianBlur() Method

It is basically blurring an image using open-cv.
You can use `GaussianBlur()` method of `cv2` library to blur an image. In order to use cv2 library, you need to import cv2 library using `import statement`.

$$f(x) = a \cdot \exp\left(-\frac{(x-b)^2}{2c^2}\right)$$

Gaussian functions are widely used in statistics to describe the normal distributions, in signal processing to define Gaussian filters, in image processing where two-dimensional Gaussians are used for Gaussian blurs.

Parameters

1. `src:` Source/Input of n-dimensional array.
2. `ksize:` Kernal is matrix of an (no. of rows)*(no. of columns) order .Its Size is given in the form of tuple (no. of rows, no. of columns). no. of rows and no. of columns should be odd .If ksize is given as (0 0), then ksize is computed from given sigma values i.e. sigmaX and sigmaY.
3. `sigmaX:` Standard deviation value of kernal along horizontal direction.
4. `sigmaY:` Standard deviation value of kernal along vertical direction.
5. `borderType:` This specify boundaries of an image while kernel is applied on borders of an image.

SigmaX − A **variable** of the type double representing the Gaussian kernel standard deviation in X direction.

Return Value

It returns Output blurred image of n-dimensional array.

a) In `GaussianBlur()` method, you need to pass `src` and `ksize` values everytime and either one, two, or all parameters value from remaining `sigmax`, `sigmaY` and `borderType` parameter should be passed.

b) Both sigmaX and sigmaY parameters become optional if you mention the `ksize(kernal size)` value other than `(0,0)`

cv2 GaussianBlur() Method example

Using `GaussianBlur()` method with `src`,`ksize` and `sigmaX` parameters.

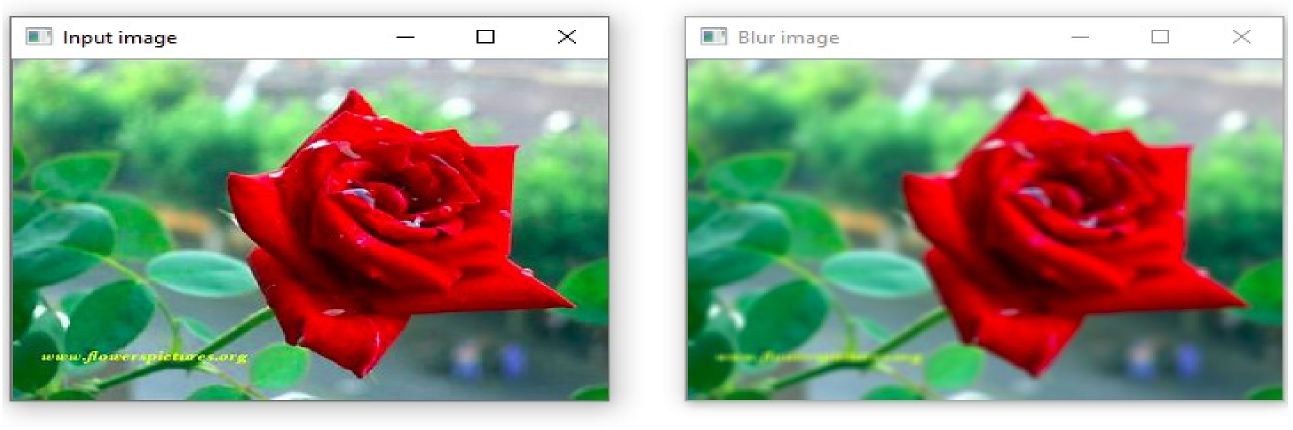


Fig. 5.2. Gaussian Blurring Example

5.3. Image Thresholding

(a) For every pixel, the same threshold value is applied. If the pixel value is smaller than the threshold, it is set to 0, otherwise it is set to a maximum value [10].
(b) The first argument is the source image, which **should be a grayscale image**.
(c) The second argument is the threshold value which is used to classify the pixel values.
(d) The third argument is the maximum value which is assigned to pixel values exceeding the thresholded
(e) The method returns two outputs. The first is the threshold that was used and the second output is the **thresholded image**

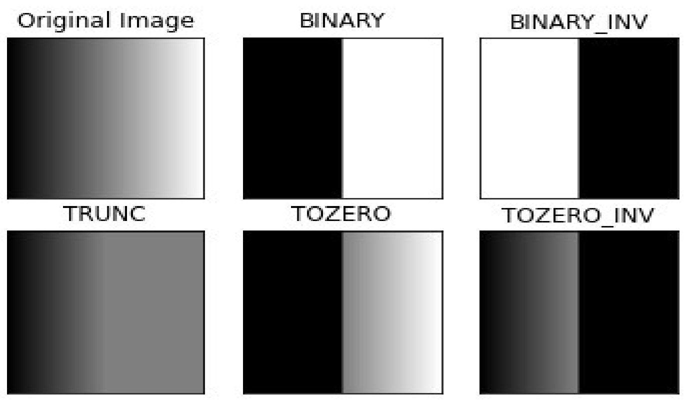


Fig. 5.3. Example of Thresholding Image

5.4. Contours

5.4.1. Finding Contours()

(a) It finds an outline representing or bounding the shape or form of something in a binary image.
(b) This is useful tool for shape analysis and object detection and recognition [11].

An overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts

Parameters

(a) **image :** We take 8-bit single-channel image. Non-zero pixels are treated as 1's. Zero pixels remains 0's, so the image is treated as binary.We can use **compare(**Performs the per- element comparison of two arrays or an array and scalar value**), inRange(**Checks if array elements lie between the elements of two other arrays.**), threshold(**The function applies fixed-level thresholding to a multiple-channel array. The function is typically used to get a bi-level (binary) image out of a grayscale image **), adaptiveThreshold(**The function transforms a grayscale image to a binary image **), Canny(**Used to find edges in the image**),** and others to create a binary image out of a grayscale or color one. If mode equals to (RETR_CCOMP or RETR_FLOODFILL) an algorithm to retrieve all of the contours in two level hierachy, in first level there boundaries and second level boundaries of the holes, If there is another contour inside a hole of a connected component, it is still put at the top level he input can also be a 32-bit integer image of labels.
(b) **Contours :** Each contour is stored as a vector of points
(c) **hierarchy :** containing information about the image topology.
(d) **Mode :** retrieval mode.
(e) **Method :** Approximation method.
(f) **Offset :** Useful in extraction of contours and analyzing it as a whole image.

```
void cv:: findContours(    InputArray                 image,
                           OutputArrayOfArrays        contours,
                           OutputArray                hierarchy,
                           int                        mode,
                           int                        method,
                           Point                      offset=Point()
                       )
```

Python:
contours, hierarchy = cv.findContours(image, mode,method[contours[, hierarchy[, offset]]])

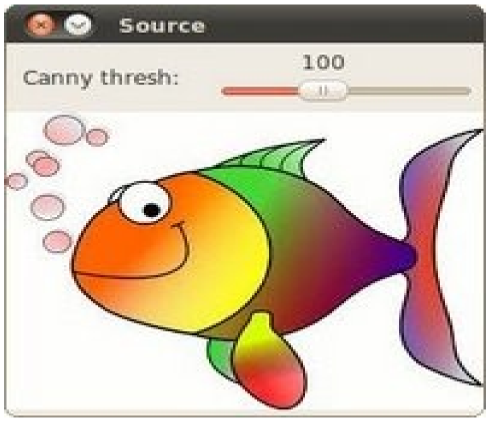
5.4.2. Draw Contours()

(a) It draws contours outlines.
(b) draws contour outlines in the image if thickness≥0 or fills the area bounded by the contours if thickness<0 .

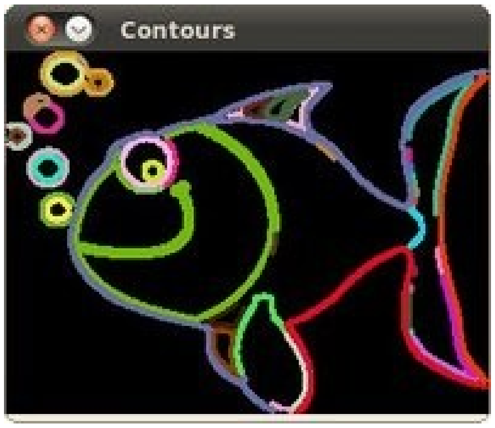
Parameters

(a) Image : Destination Image.
(b) Contours : All the input contours.
(c) ContourIdx : Parameter indicating a contour to draw.
(d) Color : of the Contours
(e) Thickness : of lines the contours are drawn with. If it is negative (for example, thickness=FILLED), the contour interiors are drawn.
(f) LineType : Connectivity
(g) hierarchy ; only needed if you want to draw only some of the contours.
(h) MaxLevel : If it is 0, only the specified contour is drawn. If it is 1, the function draws the contour(s) and all the nested contours. If it is 2, the function draws the contours, all the nested contours, all the nested-to-nested contours, and so on. This parameter is only taken into account when there is hierarchy available.
(i) Offset : Shift all the drawn contours by the specified offset=(*dx*,*dy*)

```
void cv:: findContours(   InputOutputArray          image,
                          InputArrayOfArrays        contours,
                          int                       contourIdx,
                          const Scalar &            color,
                          int                       thickness = 1,,
                          int                       linetype =Line_8,
                          InputArray                hierarchy = noArray(),
                          int                       maxLevel = INT_MAX ,
                          Point                     offset=Point()
                        )
```



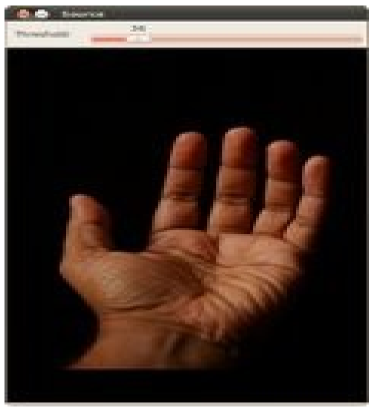
Normal Image



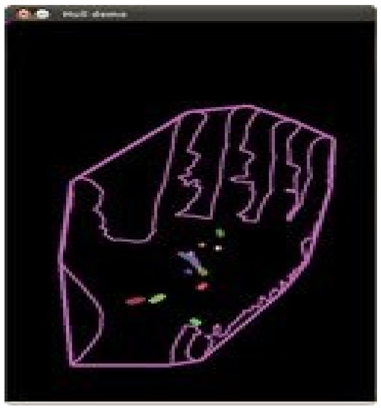
Contour Image

Fig 5.4. Example of Contoured Image

5.5. Convex Hull



Original Image



Convex Hulling

Fig 5.5.1. Example of Convex Hulling

(a) The convex hull of a binary image is the set of pixels included in the smallest convex polygon that surround all white pixels in the input.
(b) Finds the convex hull of a 2D point set using the Sklansky's algorithm [12] that has *O(N logN)* complexity in current implication.
(c) `points` and `hull` should be different arrays, inplace processing isn't supported.

Parameters

(a) **points :** Input 2-D point set, stored in std::vector
(b) **hull :** Output either an integer vector or vector of points. irst case, the hull elements are 0-based indices of the convex hull points in the original array (since the set of convex hull points is a subset of the original point set). In the second case, hull elements are the convex hull points themselves.
(c) **Clockwise :** Orientation flag. If true, Output is oriented clockwise otherwise counter-clockwise.
(d) **ReturnPoints :** Operation flag. In matrix if flag is true, returns convex hull points. Output depends on the type of the vector:std::vector<int> implies returnPoints=false, std::vector<Point> implies returnPoints=true.

```
void cv:: findContours(     InputArray          Points,
                            OutputArray         hull,
                            bool                clockwise = false,
                            bool                returnPoints = true,
                      )
```

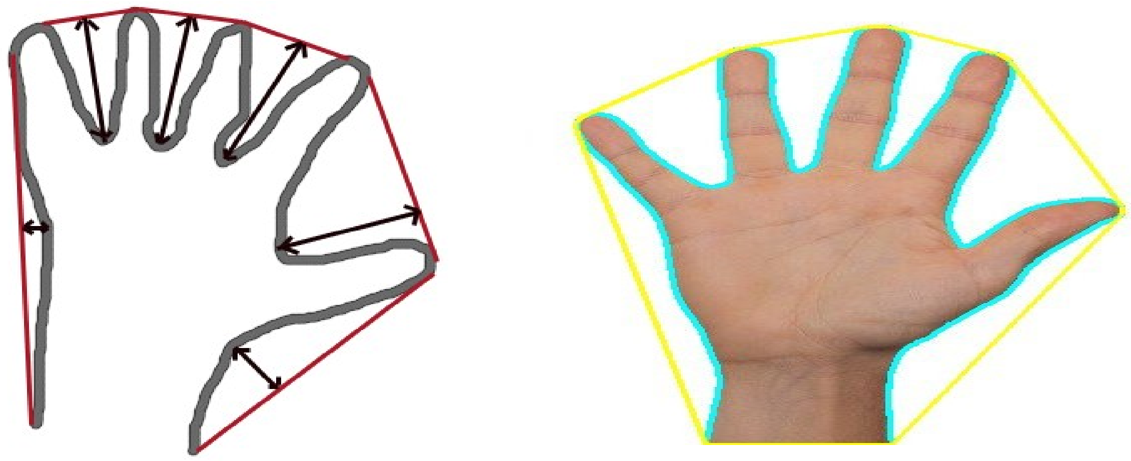


Fig 5.5.2. Representation of Convex Hulling

5.6. Creating Bounding boxes and circles for contours

boundingRect()

(a) Calculates the up-right bounding rectangle of a point set.
(b) The function calculates and returns the minimal up-right bounding rectangle for the specified point set or non-zero pixels of gray-scale image.[13]

minEnclosingCircle()

```
void cv::minEnclosingCircle  (    Input Array   points,
                                  Point2f &     center,
                                  float & radius
                             )
```

(a) Finds a circle of the minimum area enclosing a 2D point set.

(b) The function finds the minimal enclosing circle of a 2D point set using an iterative algorithm.

Parameters

points	Input vector of 2D points, stored in std::vector<> or Mat
center	Output center of the circle.
radius	Output radius of the circle.

Explanation

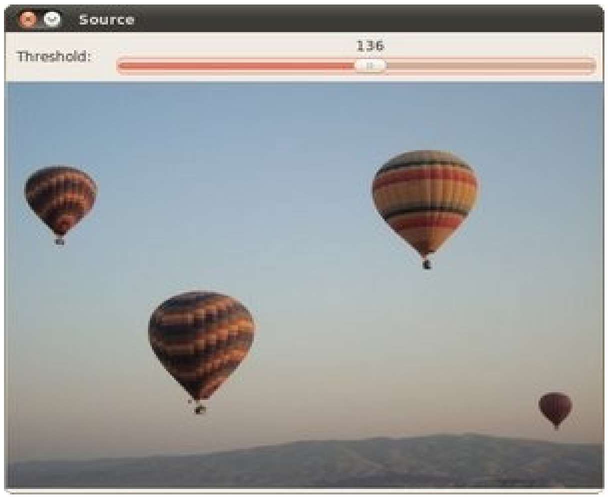
- Open the image, convert it into grayscale and blur it to get rid of the noise.
- Create a window with header "Source" and display the source file in it.
- Create a trackbar on the `source_window` and assign a callback function to it. In general callback functions are used to react to some kind of signal, in our case it's trackbar's state change. Explicit one-time call of `thresh_callback` is necessary to display the "Contours" window simultaniously with the "Source" window.
- Use cv::Canny to detect edges in the images
- Finds contours and saves them to the vectors `contour` and `hierarchy`.
- For every found contour we now apply approximation to polygons with accuracy +-3 and stating that the curve must be closed. After that we find a bounding rect for every polygon and save it to `boundRect`. At last we find a minimum enclosing circle for every polygon and save it to `center` and `radius` vectors.

We found everything we need, all we have to do is to draw.

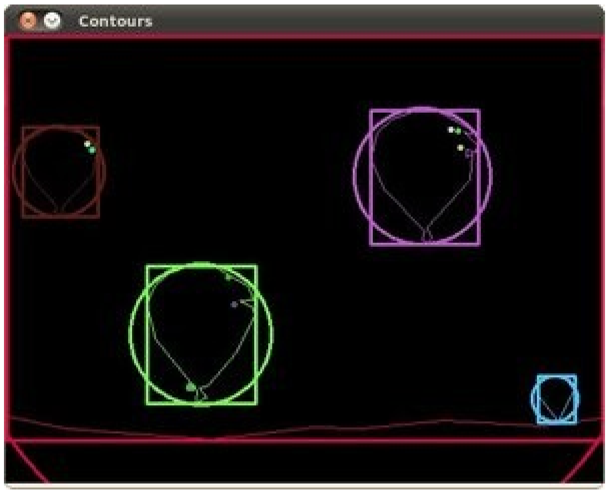
Now,

- Create new Mat of unsigned 8-bit chars, filled with zeros. It will contain all the drawings we are going to make (rects and circles).
- For every contour: pick a random color, draw the contour, the bounding rectangle and the minimal enclosing circle with it.
- Display the results: create a new window "Contours" and show everything we added to drawings on it.

Result



Original Image



Boundary boxes

Fig. 5.6. Example of Creating bounding boxes and circle over the image

UPES
UNIVERSITY WITH A PURPOSE

5.7. Creating Bounding boxes and Ellipses for contours

minAreaRect()

(a) Finds a rotated rectangle of the minimum area enclosing the input 2D point set.

(b) The function calculates and returns the minimum-area bounding rectangle (possibly rotated) for a specified point set. Developer should keep in mind that the returned RotatedRect can contain negative indices when data is close to the containing Mat element boundary.

Parameters

points	Input vector of 2D points, stored in std::vector<> or Mat

fitEllipse()

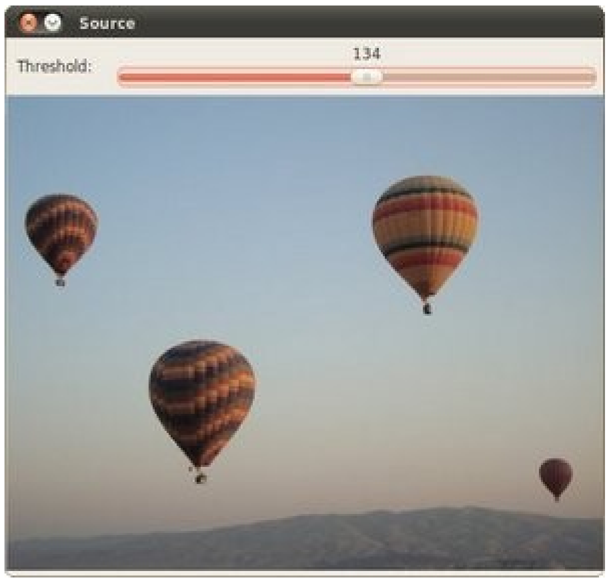
(a) Fits an ellipse around a set of 2D points.

(b) The function calculates the ellipse that fits (in a least-squares sense) a set of 2D points best of all. It returns the rotated rectangle in which the ellipse is inscribed. The first algorithm described by [14] is used. Developer should keep in mind that it is possible that the returned ellipse/rotatedRect data contains negative indices, due to the data points being close to the border of the containing Mat element.

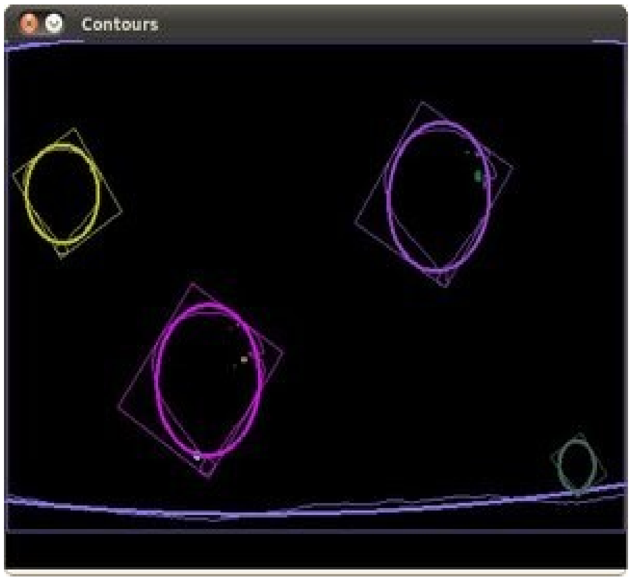
Parameters

points	Input 2D point set, stored in std::vector<> or Mat

Result:



Original Image



Rotated Boundary axis

Fig. 5.7. Example of Creating bounding boxes and ellipse over the image

5.8. Convexity Defects

It is basically the distortion of the object.

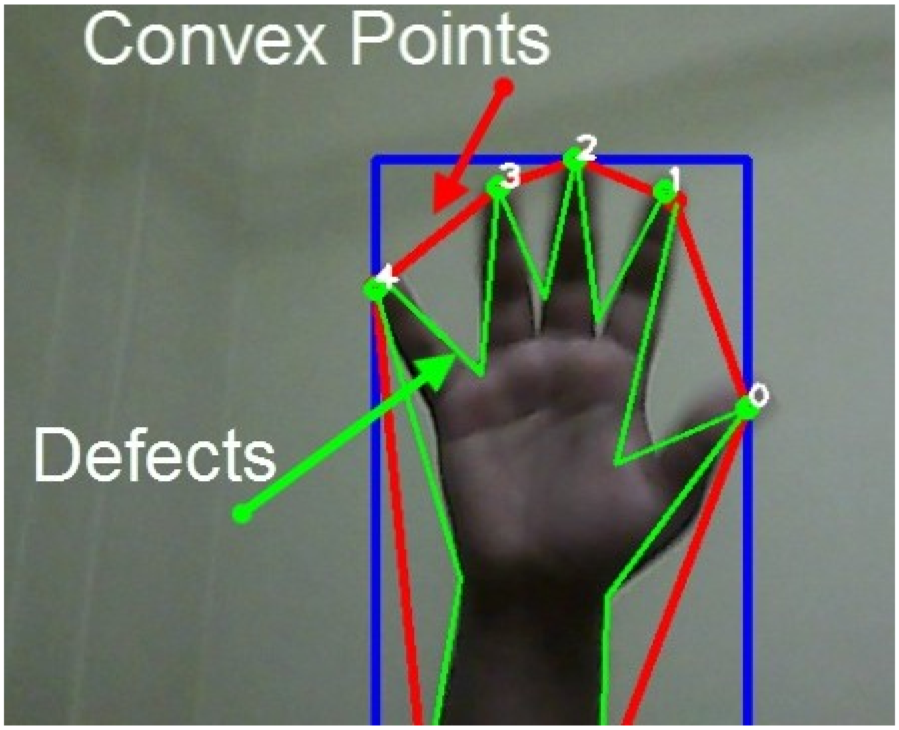
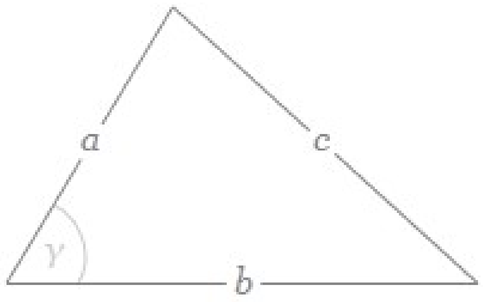


Fig. 5.8.1. Convexity Defects Example

Cosine Theorem

In trigonometry, the law of cosines relates the lengths of the sides of a triangle to the cosine of one of its angles. Using notation as in shown figure below, the law of cosines states where γ denotes the angle contained between sides of lengths a and b and opposite the side of length c.



Formula

$$c = \sqrt{a^2 + b^2 - 2ab\cos\gamma}$$

By seeing this formula now we understand that if we have; *a*,*b* and *gama* then we also find *c* as well as if we have; *a*,*b*,*c* then we also find *gamma (vice-versa)*

For finding gamma this formula is used:

$$\gamma = \cos^{-1}\left(\frac{a^2 + b^2 - c^2}{2ab}\right)$$

Using Cosine theorem to recognize fingers

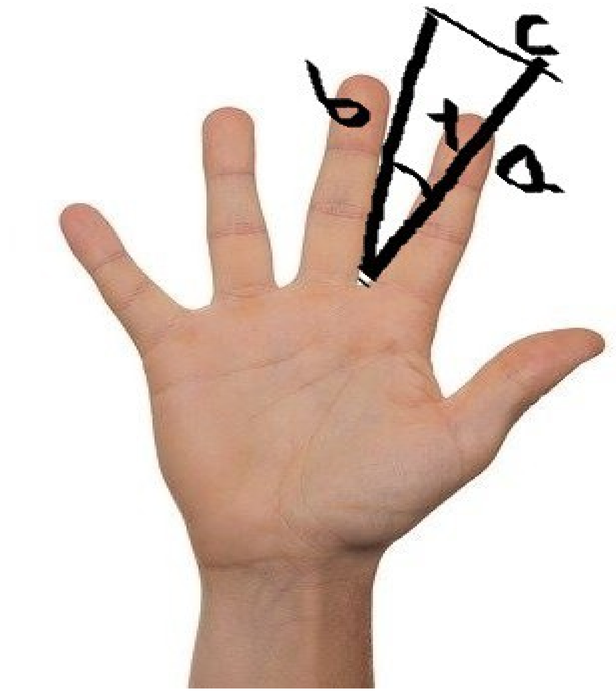


Fig. 5.8.2. Cosine theorem implementation over finger

In above figure Side: *a,b,c* and angle: *gamma*. Now this gamma is always less than 90 degree, So we can say: If *gamma* is less than 90 degree or *pi/2* we consider it as a finger.

Counting Finger

Convexity Defects returns an array where each row contains these values :

- start point
- end point
- farthest point
- approximate distance to farthest point

By, this point we can easily derive Sides: a,b,c (see CODE) and from cosine theorem we can also derive gamma or angle between two finger. As you read earlier, if gamma is less than 90 degree we treated it as a finger. After knowing gamma we just draw circle with radius 4 in **approximate distance to farthest point.** And after we just simple put text in images we represent finger counts (cnt).

Final Result:

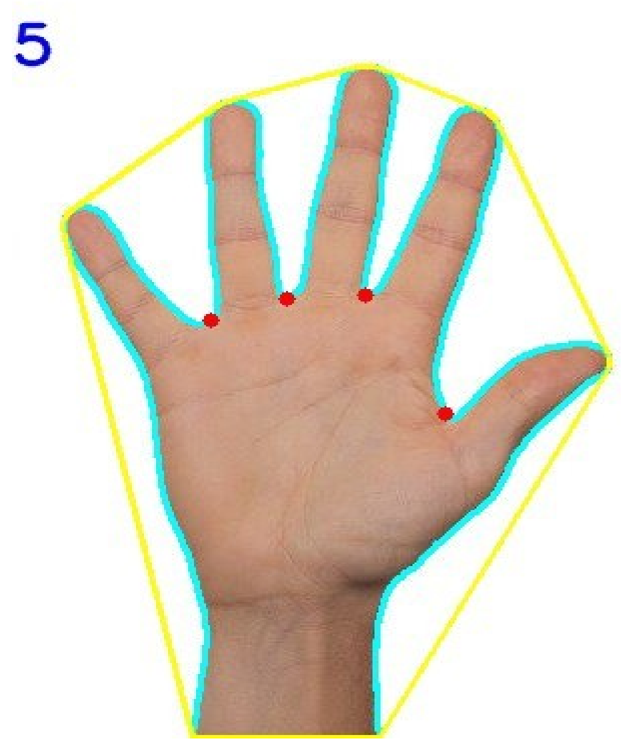
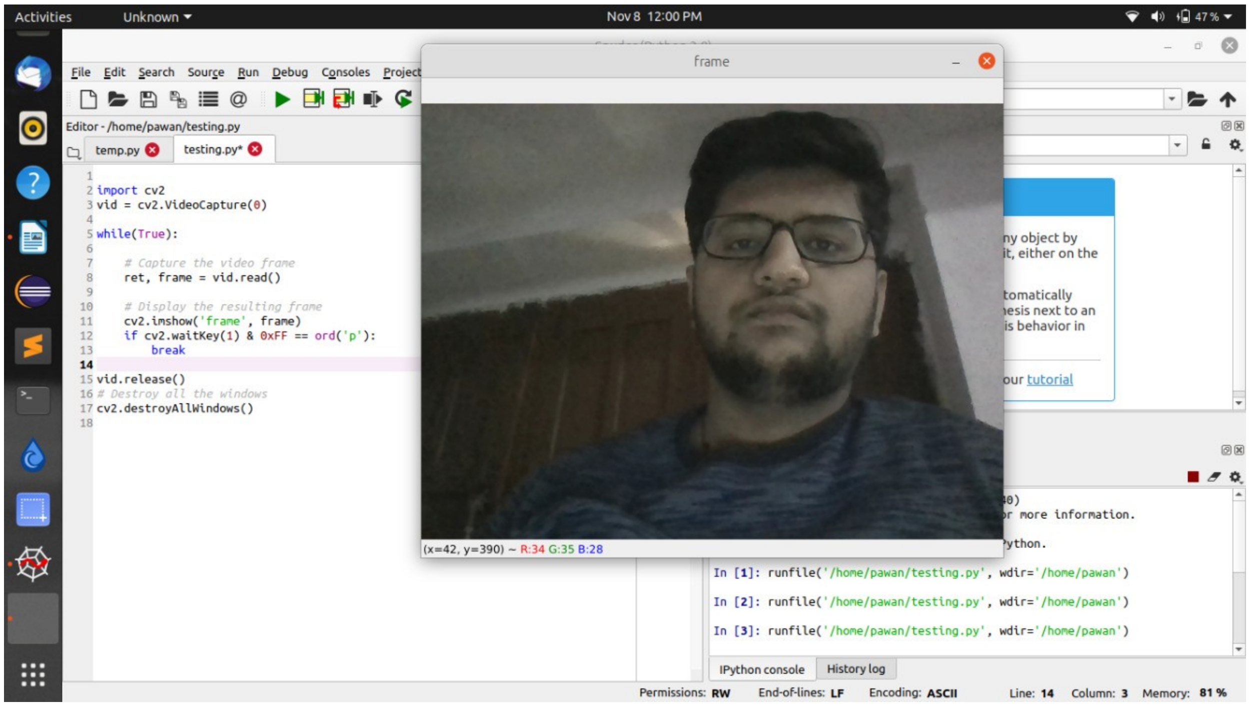


Fig. 5.8.3. Final Result

6. OUTPUT SCREENS



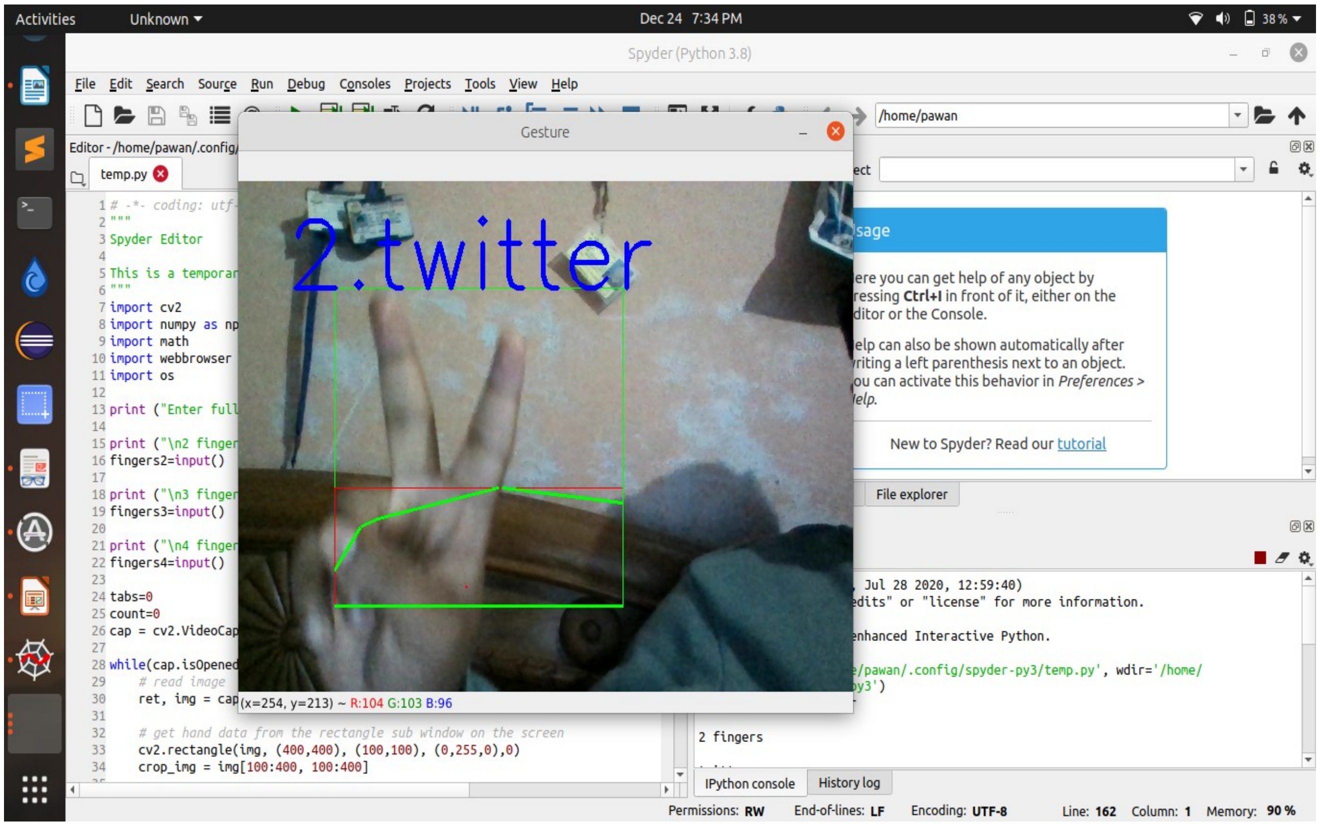
Activities
Unknown
Nov 8 12:00 PM
47 %
File Edit Search Source Run Debug Consoles Project
Editor - /home/pawan/testing.py
temp.py
testing.py*
frame
(x=42, y=390) ~ R:34 G:35 B:28
IPython console
History log

```
import cv2
vid = cv2.VideoCapture(0)

while(True):

    # Capture the video frame
    ret, frame = vid.read()

    # Display the resulting frame
    cv2.imshow('frame', frame)
    if cv2.waitKey(1) & 0xFF == ord('p'):
        break

vid.release()
# Destroy all the windows
cv2.destroyAllWindows()
```

```
In [1]: runfile('/home/pawan/testing.py', wdir='/home/pawan')

In [2]: runfile('/home/pawan/testing.py', wdir='/home/pawan')

In [3]: runfile('/home/pawan/testing.py', wdir='/home/pawan')
```

Permissions: RW End-of-lines: LF Encoding: ASCII Line: 14 Column: 3 Memory: 81 %

Fig. 6.1. Frame Snapshot


Activities
Unknown
Dec 24 7:34 PM
38 %
Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
/home/pawan
Editor - /home/pawan/.config/
temp.py
Gesture
2.twitter
(x=254, y=213) ~ R:104 G:103 B:96
File explorer
IPython console
History log

```
# -*- coding: utf-
"""
Spyder Editor

This is a temporar
"""
import cv2
import numpy as np
import math
import webbrowser
import os

print ("Enter full

print ("\n2 finger
fingers2=input()

print ("\n3 finger
fingers3=input()

print ("\n4 finger
fingers4=input()

tabs=0
count=0
cap = cv2.VideoCap

while(cap.isOpened
    # read image
    ret, img = cap

    # get hand data from the rectangle sub window on the screen
    cv2.rectangle(img, (400,400), (100,100), (0,255,0),0)
    crop_img = img[100:400, 100:400]
```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

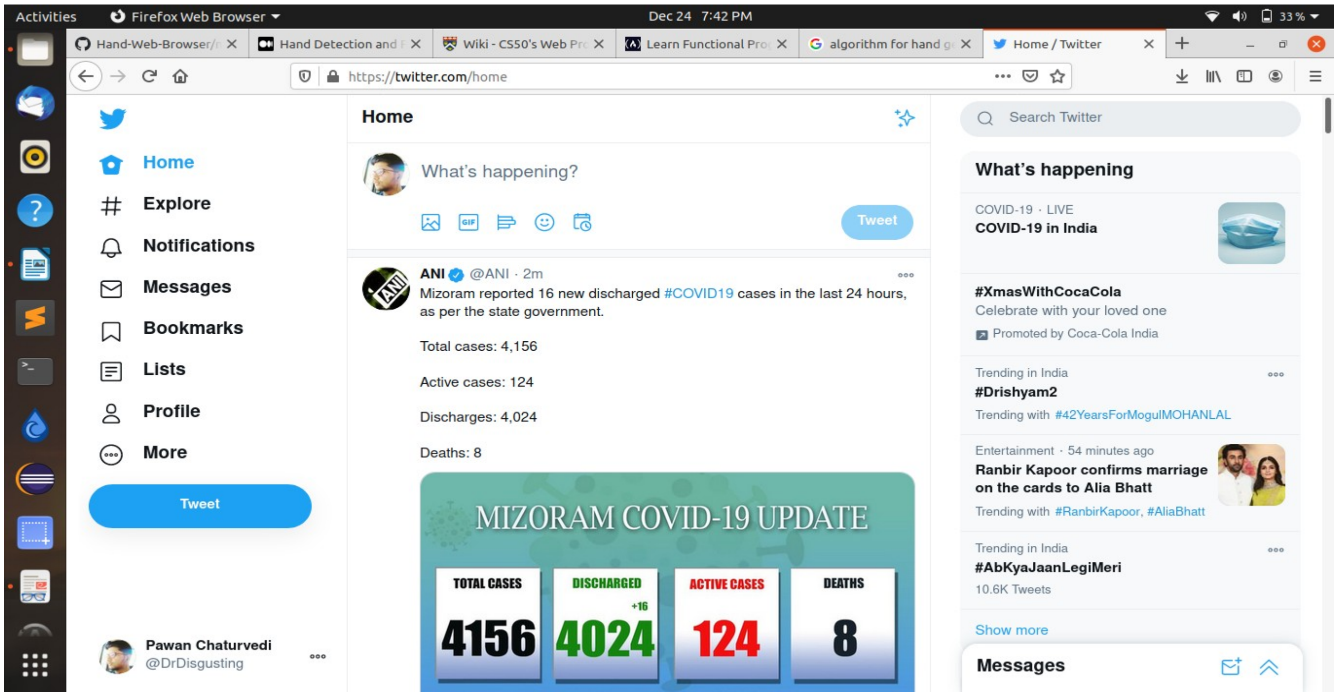
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help.*

New to Spyder? Read our tutorial

```
, Jul 28 2020, 12:59:40)
edits" or "license" for more information.

enhanced Interactive Python.

e/pawan/.config/spyder-py3/temp.py', wdir='/home/
py3')

2 fingers
```

Permissions: **RW** End-of-lines: **LF** Encoding: **UTF-8** Line: **162** Column: **1** Memory: **90 %**

Fig. 6.2. Data Acquisition by Fingers



ANI @ANI · 2m
Mizoram reported 16 new discharged #COVID19 cases in the last 24 hours, as per the state government.

Total cases: 4,156

Active cases: 124

Discharges: 4,024

Deaths: 8

MIZORAM COVID-19 UPDATE

TOTAL CASES	DISCHARGED	ACTIVE CASES	DEATHS
4156	+16 4024	124	8

COVID-19 · LIVE
COVID-19 in India

#XmasWithCocaCola
Celebrate with your loved one
Promoted by Coca-Cola India

Trending in India
#Drishyam2
Trending with #42YearsForMogulMOHANLAL

Entertainment · 54 minutes ago
Ranbir Kapoor confirms marriage on the cards to Alia Bhatt
Trending with #RanbirKapoor, #AliaBhatt

Trending in India
#AbKyaJaanLegiMeri
10.6K Tweets

Fig. 6.3. Opening of Web Page

Appendix A

Software Source Code

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
import cv2
import numpy as np
import math
import webbrowser as wb
import os

print ("Enter the name of websites")

print ("fingers for 2")
finger_2=input()

print (“fingers for 3")
finger_3=input()

print ("fingers for 4")
finger_4=input()

tabs=0
count=0
vid = cv2.VideoCapture(0)              // provides an interface to capture live stream with
camera

while(vid.isOpened()):                 // cap = vid, ret = r , img = b, crop_img = Croppedimg,
grey=color_grey , value = kernel
   r, b = vid.read()              //ret:- "Ret" will obtain return value from getting the camera
frame, either true of false.

   cv2.rectangle(b, (400,400), (100,100), (0,255,0),0)     //Syntax: cv2.rectangle(image,
start_point, end_point, color, thickness)
   Croopedimg = b[100:400, 100:400]          `          //image manipulater  resizing aspect ratio


   color_grey = cv2.cvtColor(Croppedimg, cv2.COLOR_BGR2GRAY)
```

```
kernel = (35, 35)          //kernel size initialized in place of SigmaX and SigmaY

blurred = cv2.GaussianBlur(grey, value, 0)

_, thresh_value = cv2.threshold(blurred, 127, 255,
                          cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU) // thers1 =
thresh_value

(version, _, _) = cv2.__version__.split('.')

if version == '4':
    con, hr = cv2.findContours(thresh_value.copy(),cv2.RETR_TREE,
           cv2.CHAIN_APPROX_NONE)   // contours = con, hierarchy = hr

max_area = max(contours, key = lambda x: cv2.contourArea(x))  //cnt = max_area

m, n, o, p = cv2.boundingRect(max_area)      // x,y,w,h = m,n,o,p
cv2.rectangle(Croppedimg, (m, n), (m+o, n+p), (0, 0, 255), 0)

hl = cv2.convexHull(max_area)    // hull = hl

points = np.zeros(Croppedimg.shape,np.uint8)     //drawing = points
cv2.drawContours(points, [max_area], 0, (0, 255, 0), 0)
cv2.drawContours(points, [hl], 0,(0, 0, 255), 0)

hl = cv2.convexHull(max_area, returnPoints=False)#  return point false to find convexity
defects

gaps = cv2.convexityDefects(max_area, hl)   //defects = gaps
count_defects = 0
cv2.drawContours(thresh_points, con, -1, (0, 255, 0), 3)# to draw all contours pass -1

for i in range(gaps.shape[0]):
    q, r, s, t = gaps[i,0]                // s,e,f,d = q,r,s,t

    st = tuple(max_area[q][0])       // start = st , end = en , far = ar
    en = tuple(max_area[r][0])
```

```
        ar = tuple(max_area[s][0])

        a = math.sqrt((en[0] - st[0])**2 + (en[1] - st[1])**2)
        b = math.sqrt((ar[0] - st[0])**2 + (ar[1] - st[1])**2)
        c = math.sqrt((en[0] - ar[0])**2 + (en[1] - ar[1])**2)

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

        if finger_angle <= 90:
            count_defects += 1
            cv2.circle(Croppedimg, ar, 1, [0,0,255], -1)

        cv2.line(Croppedimg,st, en, [0,255,0], 2)

    if count==0:
        cv2.putText(b,””, (50, 100), cv2.FONT_HERSHEY_SIMPLEX, 3, 3)

    if count_defects == 1 and count!=2 and tabs<=8:
        wb.open_new_tab('http://www.'+finger_2+'.com')
        tabs=tabs+1
        cv2.putText(b,””,finger_2, (50, 100), cv2.FONT_HERSHEY_SIMPLEX, 3, (255,0,0), 3)
        count=2
    elif count_defects == 2 and count!=3 and tabs<=8:
        wb.open_new_tab('http://www.'+finger_3+'.com')
        tabs=tabs+1
        cv2.putText(b, “”, finger_3, (50, 100), cv2.FONT_HERSHEY_SIMPLEX, 3, (0,0,255), 3)
        count=3
    elif count_defects == 3 and count!=4 and tabs<=8:
        wb.open_new_tab('http://www.'+finger_4+'.com')
        cv2.putText(b, “”,finger_4, (50, 100), cv2.FONT_HERSHEY_SIMPLEX, 3, (255,165,0),
3)
        tabs=tabs+1
        count=4
    elif count_defects == 4 and count!=5:
        cv2.putText(b, “”, (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 3, 3)
        os.system("taskkill /im chrome.exe /f")
        tabs=0
        count=5
    else:
        cv2.putText(b,"", (50, 100),\
                    cv2.FONT_HERSHEY_SIMPLEX, 3, 3)


    if count==2:
        cv2.putText(b, fingers2, (50, 100), cv2.FONT_HERSHEY_SIMPLEX, 3, (255,0,0), 3)
    elif count==3:
```

```
    cv2.putText(b, fingers3, (50, 100), cv2.FONT_HERSHEY_SIMPLEX, 3, (0,0,255), 3)
elif count==4:
    cv2.putText(b, fingers4, (50, 100), cv2.FONT_HERSHEY_SIMPLEX, 3, (255,165,0), 3)
elif count==5:
    cv2.putText(b, "", (50, 100), cv2.FONT_HERSHEY_SIMPLEX, 3, 3)



# show appropriate images in windows
mg = np.hstack((points, Croppedimg))
#not necessary to show contours and can be skipped
cv2.imshow('Con', mg)

k = cv2.waitKey(10)
if k == 27:
    break
```

References

[1] Zhou Ren, Jingjing Meng and Zhengyou Zhang, “Robust Part-Based Hand Gesture RecognitionUsing Kinect Sensor”, IEEE Transactions on multimedia, vol. 15, no. 5, august 2013

[2] Hand posture recognition with the fuzzy glove Author links open overlay panel T.Allevard, E.BenoitL.Foulloy LISTIC-ESIA, Université de Savoie, B.P. 806, 74016 Annecy, France online 9 May 2007 https://www.sciencedirect.com/science/article/pii/B9780444520753500352

[3] Jobin Francis,Anoop B K,”Significance of Hand Gesture Recognition Systems in Vehicular Automation- A Survey” International Journal of Computer Applications (0975 – 8887) Volume 99– No.7, August 2014

[4] Juan P. Wachs, Helman Stern and Yael Edan, “Cluster Labeling and Parameter Estimation for the Automated Setup of a Hand-Gesture Recognition System”, IEEE Transactions on Systems, Man, and Cybernetics—part a: Systems and Humans, vol. 35, no. 6, november 2005

[5] Regina Lionnie, Ivanna K. Timotius & Iwan Setyawan, “Performance Comparison of Several Pre-Processing Methods in a Hand Gesture Recognition System based on Nearest Neighbor for Different Background Conditions” ITB J. ICT, Vol. 6, No. 3, 2012, 183-194

[6] Yuan Yao and Yun Fu, “Contour Model based Hand-Gesture Recognition Using Kinect Sensor”, 10.1109/TCSVT.2014.2302538, IEEE Transactions on Circuits and Systems for Video Technology.

[7] Robert Y. Wang and Jovan Popovic, “Real-Time Hand- Tracking with a Color Glove”,

[8] International Journal of Engineering and Technical Research (IJETR)ISSN: 2321-0869, Volume-3, Issue-5, May2015

[9] Premaratne P., Nguyen Q., Premaratne M. (2010) Human Computer Interaction Using Hand Gestures. In: Huang DS., McGinnity M., Heutte L., Zhang XP. (eds) Advanced Intelligent Computing Theories and Applications. ICIC 2010. Communications in Computer and Information Science, vol 93. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-14831-6_51

[10] https://www.geeksforgeeks.org/python-thresholding-techniques-using-opencv-set-1-simple-thresholding/

[11] Satoshi Suzuki and others. Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing, 30(1):32–46, 1985.

[12] Jack Sklansky. Finding the convex hull of a simple polygon. Pattern Recognition Letters, 1(2):79–83, 1982.

[13] https://docs.opencv.org/master/d3/dc0/group__imgproc__shape.html#ga103fcbda2f540f3ef1c042d6a9b35ac7

[14] Andrew W Fitzgibbon and Robert B Fisher. A buyer's guide to conic fitting. In *Proceedings of the 6th British conference on Machine vision (Vol. 2)*, pages 513–522. BMVA Press, 1995. https://docs.opencv.org/master/d3/dc0/group__imgproc__shape.html#ga3d476a3417130ae5154aea421ca7ead9

[15] https://medium.com/analytics-vidhya/hand-detection-and-finger-counting-using-opencv-python-5b594704eb08