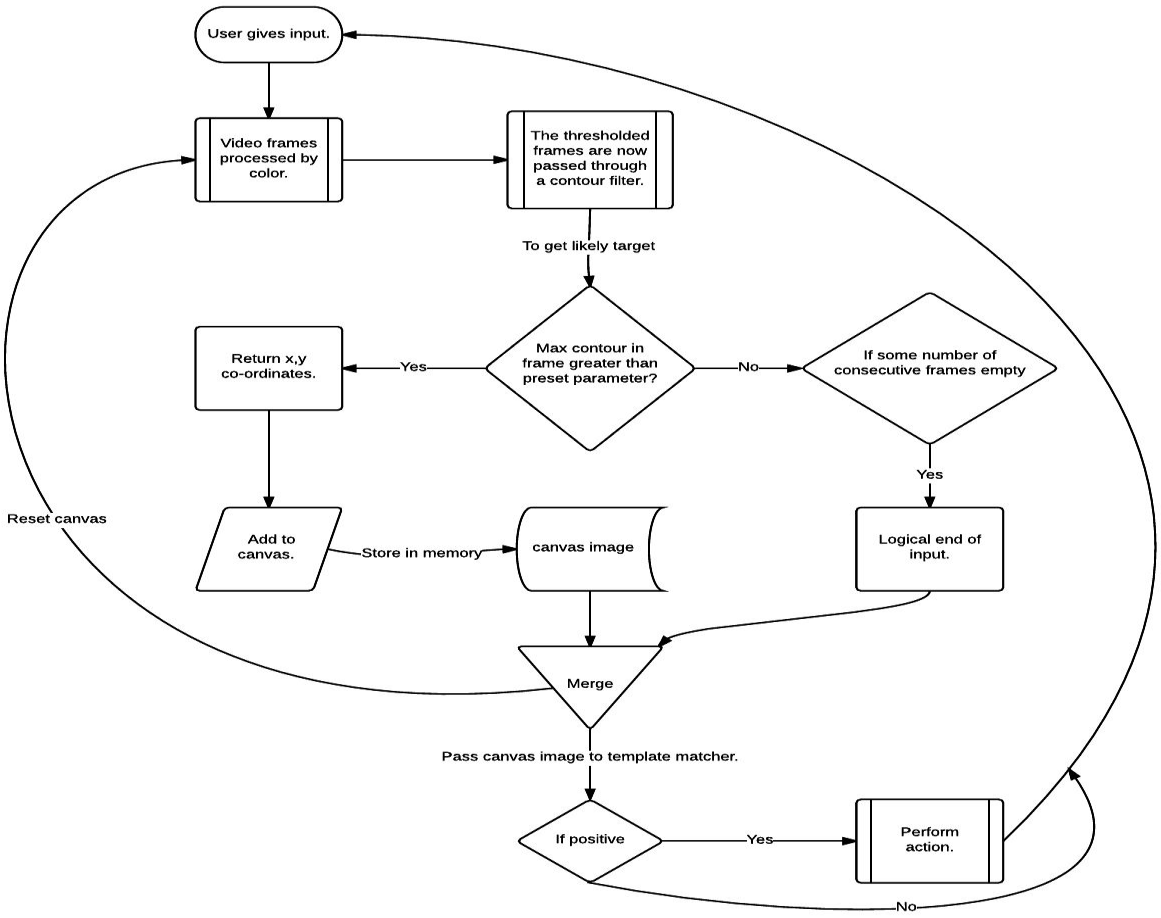
Principles of Programming Languages

Xestos

The World of gestures.



Introduction

Project Xestos is a glimpse into the world of Human - Computer interaction. Using simple gestures to control personal computing devices has been a long area of research. This project aims to replicate some of the concepts to serve as a base for future development.

The team:

- Akhil Alluri
- Chris Sunny Thaliyath
- Shantnu Patlan
- Karan Sharma
- Hoshank Ailani

Things used in Xestos

Python

No more needs to be said about this powerful programming language. Simple to use and very modular, it has enabled Xestos to focus on the actual project rather than the nitty gritty of implementations.

OpenCV

A powerful image processing library that's almost the world standard for visual processing applications.

Numpy & Scipy

Some of the most powerful mathematical and scientific tools all available free to use .

These free libraries are a new phase of hacktivism , which will soon push the human race forth like no other.By allowing people to build on top of each others work rather than competing . This will usher in an era of unprecedented opportunities for everyone.

Linux platform:

Xdotool bash command utility to initiate key commands like pause, play, mute et cetera.

WORK DISTRIBUTION

Chris Sunny Thaliath:

Worked with Akhil & Shantnu to get features like Finger Detection , Background Subtractor. We were able implement a very crude hand tracker using BackgroundSubtractor and Contour mapping .

Akhil Alluri :

Worked on setting up a mechanism to generate inputs for the template matching module set up by Shantnu. Worked on modules to identify target based on Features [Haar] and color.

Karan Sharma :

Worked on integrating the GUI with the backend script and implementing OS based actions performed after detecting hand gestures.Worked on improving the accuracy of hand detection using some optimization tweaks like background subtraction methods.

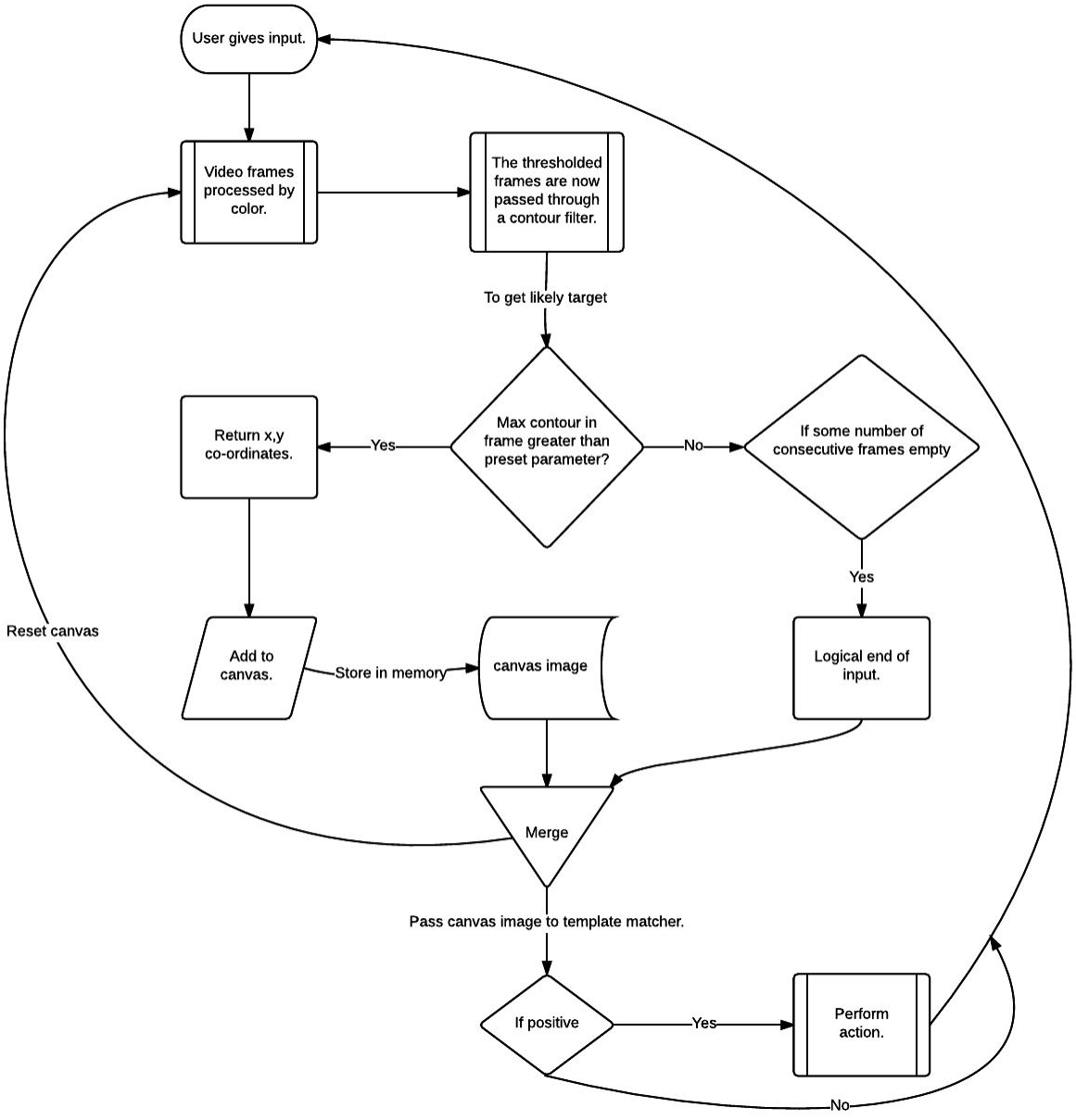
Shantanu Patlan :

Understanding and implementing template matching. Worked with Chris on background subtractor and contour mapping to extract fingers.

Hoshank Ailani :

Worked on GUI with Karan and worked on OpenCV 2 used in the hand and finger detect modules.

Gesture recognition using Color thresholding:



A Specific color is used to find target.

Code:

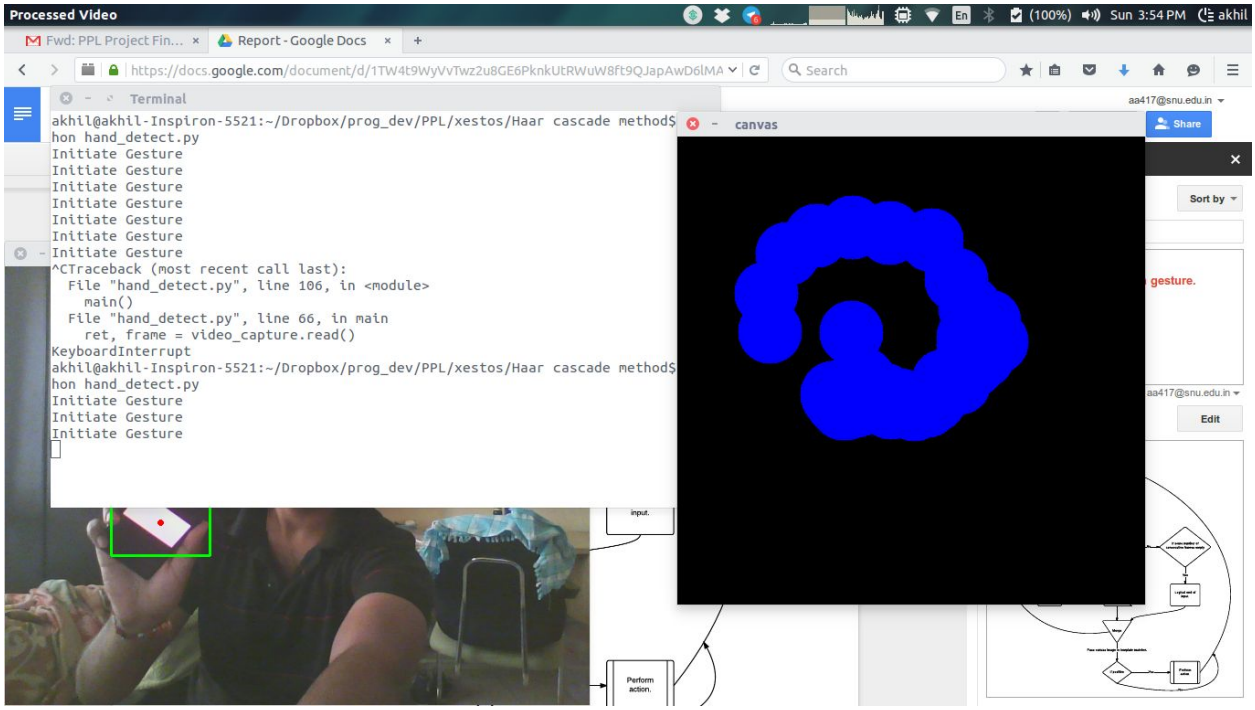
```
1.  import cv2
2.  import sys
3.  import numpy as np
4.  import subprocess
5.  import os
6.  from matplotlib import pyplot as plt
7.  PINK_MIN = np.array([120, 50, 50], np.uint8)
8.  PINK_MAX = np.array([180, 180, 200], np.uint8)
9.  BLUE_MIN = np.array([100,150,0], np.uint8)
10. BLUE_MAX = np.array([140,255,255], np.uint8)
11. def color_detect(img):
12.     hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
13.     #frame_threshed = cv2.inRange(hsv, PINK_MIN, PINK_MAX)
14.     frame_threshed = cv2.inRange(hsv, BLUE_MIN, BLUE_MAX)
15.     contours,hierarchy = cv2.findContours(frame_threshed, 1, 2)
16.     max_area = 350
17.     found = 0
18.     if contours:
19.         for i in contours:
20.             area = cv2.contourArea(i)
21.             if area > max_area:
22.                 found = 1
23.                 cnt = i
24.         if(found == 1):
25.             x,y,w,h = cv2.boundingRect(cnt)
26.             cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
27.             centroid_x = (x + x+w)/2
28.             centroid_y = (y + y+h)/2
29.             cv2.circle(img, (centroid_x, centroid_y), 2, (0,0,255), 2)
30.             #cv2.imshow('Threshold', frame_threshed)
31.             #cv2.imshow('Original', img)
32.             return img,centroid_x,centroid_y
33.     return img,0,0
```

```
34.
35.
36. def detector(image):
37.     img_rgb = image
38.     img_gray = cv2.cvtColor(img_rgb, cv2.COLOR_BGR2GRAY)
39.     template1 = cv2.imread('template.jpg',0)
40.     template2 = cv2.imread('template_line.png',0)
41.     template3 = cv2.imread('template_line_horiz.png',0)
42.     counter = 0
43.     res1 = cv2.matchTemplate(img_gray,template1,cv2.TM_CCOEFF_NORMED)
44.     res2 = cv2.matchTemplate(img_gray,template2,cv2.TM_CCOEFF_NORMED)
45.     res3 = cv2.matchTemplate(img_gray,template3,cv2.TM_CCOEFF_NORMED)
46.     threshold = 0.5
47.     posloc1 = np.where( res1 >= threshold)
48.     for pt in zip(*posloc1[::-1]):
49.         #cv2.rectangle(img_rgb, pt, (pt[0] + w, pt[1] + h), (0,0,255), 2)
50.         counter = 1
51.     posloc2 = np.where( res2 >= threshold)
52.     for pt in zip(*posloc2[::-1]):
53.         #cv2.rectangle(img_rgb, pt, (pt[0] + w, pt[1] + h), (0,0,255), 2)
54.         counter = 2
55.     posloc3 = np.where( res3 >= threshold)
56.     for pt in zip(*posloc3[::-1]):
57.         #cv2.rectangle(img_rgb, pt, (pt[0] + w, pt[1] + h), (0,0,255), 2)
58.         counter = 3
59.     return counter
60.
61. def main():
62.     if len(sys.argv) == 1:
63.         counter = 0;
64.         count = 0;
65.         video_capture = cv2.VideoCapture(0)
66.         img = np.zeros((512,512,3), np.uint8)
67.         oldx = oldy = 0
```

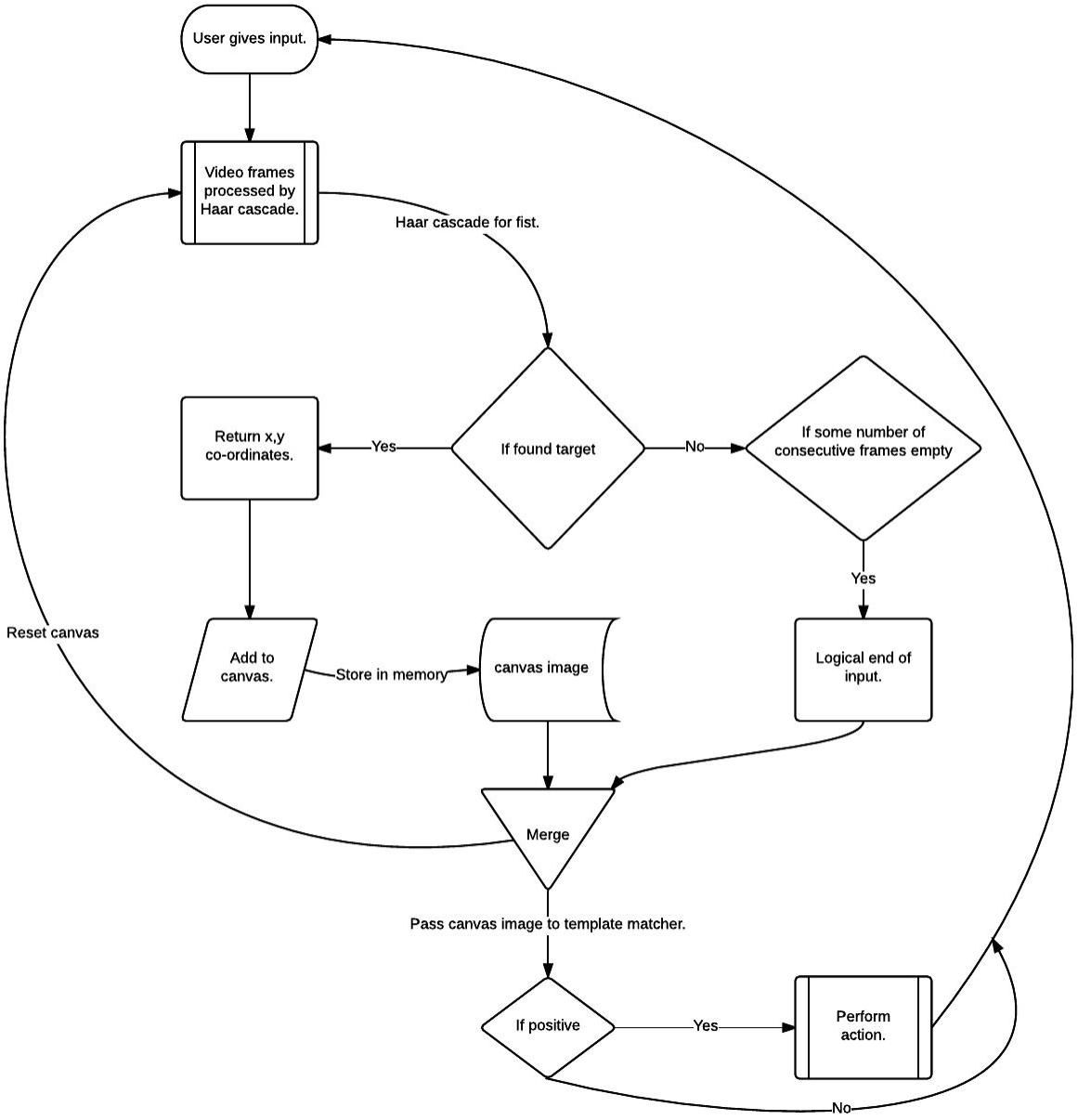
```
68.         x  = 0
69.         y= 0
70.         write = 0
71.         while True:
72.             ret, frame = video_capture.read()
73.             frame,x,y = color_detect(frame)
74.             if(frame == None or x==None or y==None):
75.                 continue
76.             if(x==0 & y==0):
77.                 count = count + 1;
78.             else:
79.                 count = 0
80.                 cv2.circle(img,(x,y),35,0xff,-1)
81.                 cv2.imshow('canvas',img)
82.                 if(write ==2):
83.                         write = 1
84.                 else:
85.                         write = 2
86.             if(count == 5):
87.                 #print('should work')
88.                 #if(write == 1):
89.                 result = detector(img)
90.                 if(result == 1):
91.                     print("Initiate Gesture")
92.                     os.system("xdotool key XF86AudioMute")
93.                 if(result == 2):
94.                     print("gesture 2")
95.                     os.system("xdotool key XF86AudioPlay")
96.                 if(result == 3):
97.                     print("Gesture 3")
98.                     os.system("xdotool key XF86AudioLowerVolume")
99.                 write = 0
100.                    img = np.zeros((512,512,3), np.uint8)
101.                    cv2.imshow('canvas',img)
```

```
                    count = 0
                #cv2.imshow('Processed Video', frame)
                oldx = x
                oldy = y
                counter = counter+1;
                if cv2.waitKey(1) & 0xFF == ord('q'):
                    break
            video_capture.release()
            cv2.destroyAllWindows()

    if __name__ == '__main__':
        main()
```

ScreenShot:



Gesture Recognition using Haar Cascade:



The target is found using a Haar Cascade.

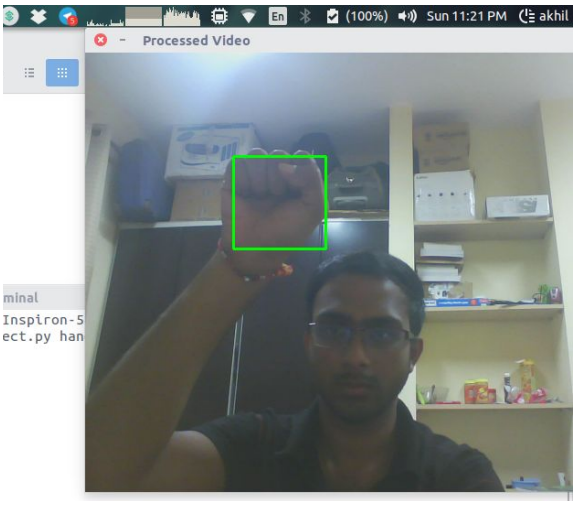
Code:

```
import cv2
import sys
import numpy as np
from matplotlib import pyplot as plt

def fist_detect(image):
    cascPath = sys.argv[1]
    HaarCascade = cv2.CascadeClassifier(cascPath)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    faces = HaarCascade.detectMultiScale(
        gray,
        scaleFactor=1.1,
        minNeighbors=5,
        minSize=(40, 40),
        flags=cv2.cv.CV_HAAR_SCALE_IMAGE
    )
    xx = 0
    yy = 0
    for (x, y, w, h) in faces:
        cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)
        #cv2.circle(image,(x+w/2,y+h/2),2,0xff)
        xx = x+w/2
        yy = y+h/2
    return image,xx,yy;

def detector(image):
  img_rgb = image
  img_gray = cv2.cvtColor(img_rgb, cv2.COLOR_BGR2GRAY)
  template = cv2.imread('template.jpg',0)
  w, h = template.shape[::-1]
  counter = 0
```

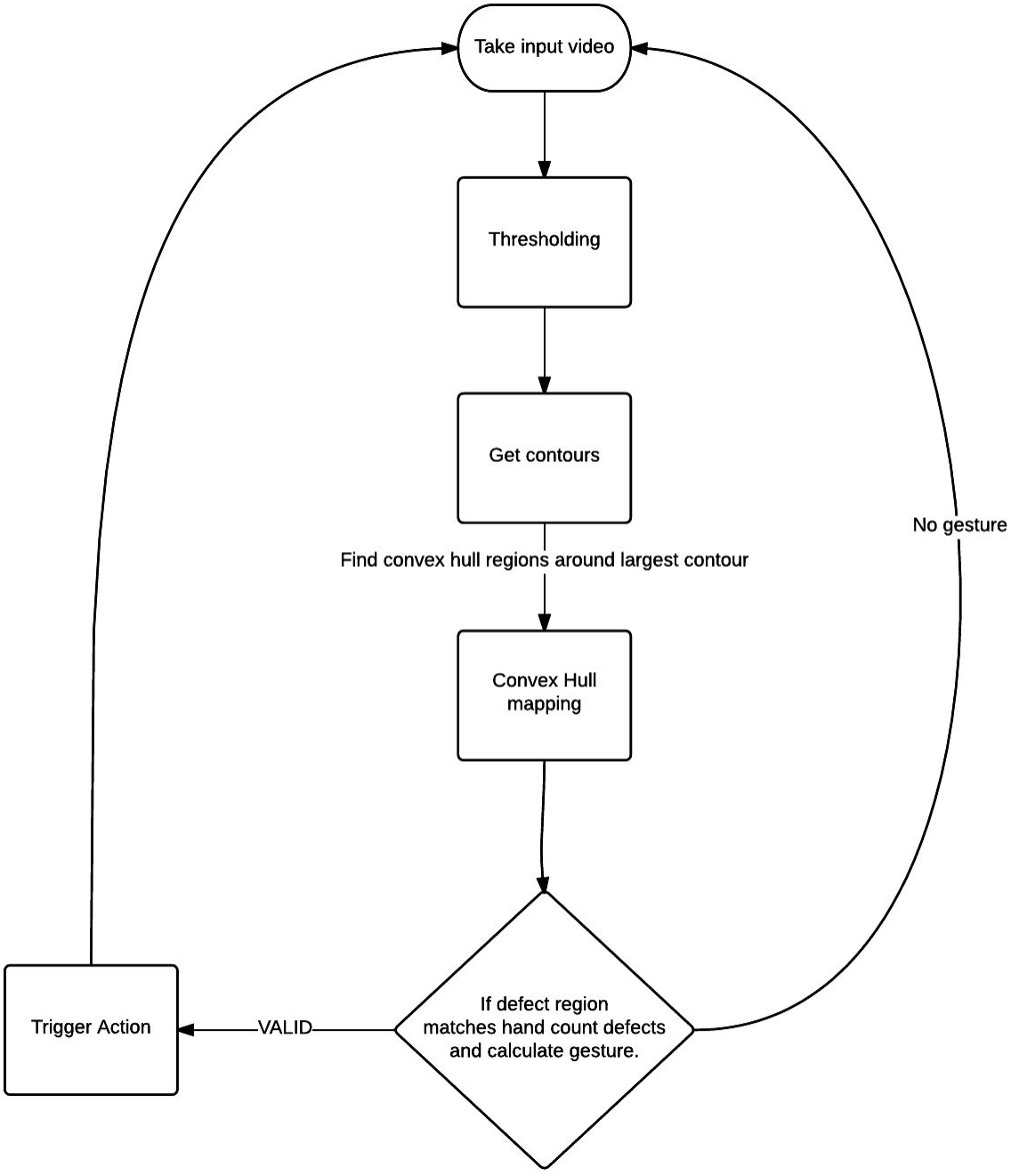
```
    result = cv2.matchTemplate(img_gray,template,cv2.TM_CCOEFF_NORMED)
    threshold = 0.5
    loc = np.where( result >= threshold)
    for pt in zip(*loc[::-1]):
      cv2.rectangle(img_rgb, pt, (pt[0] + w, pt[1] + h), (0,0,255), 2)
      counter = 1
    return counter

def main():
    if len(sys.argv) == 2:
        counter = 0;
        count = 0;
        video_capture = cv2.VideoCapture(0)
        img = np.zeros((512,512,3), np.uint8)
        oldx = oldy = 0
        x  = 0
        y= 0
        write = 0
        while True:
            ret, frame = video_capture.read()
            if(counter%2!=0):
                frame,x,y = fist_detect(frame)
            if(x==0 & y==0):
                count = count + 1;
            else:
                cv2.circle(img,(x,y),35,0xff,-1)
                if(write ==2):
                        write = 1
                else:
                        write = 2
            if(count == 5):
                if(write == 1):
                    result = detector(img)
```

```
                    if(result == 1):
                        print("Working")
                    else:
                        print("no gesture")
                write = 0
                img = np.zeros((512,512,3), np.uint8)
                count = 0
            cv2.imshow('Processed Video', frame)
            oldx = x
            oldy = y
            counter = counter+1;
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
        video_capture.release()
        cv2.destroyAllWindows()
    else:
        print("usage : python hand_detect.py [har CascadeClassifier file]")

if __name__ == '__main__':
    main()
```

Screen Shot:



Finger Detection convex hull



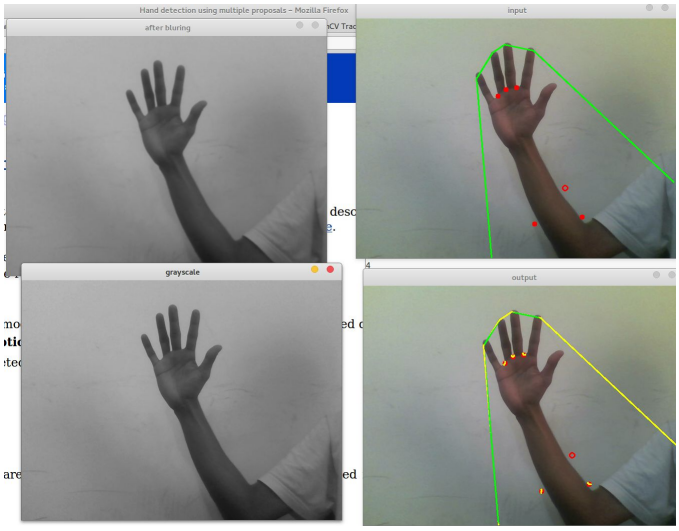
Code:

```
import cv2
import numpy as np
import math
cap = cv2.VideoCapture(0)

while(cap.isOpened()):
    print("\n1.Getting the VideoCapture")
    ret, img = cap.read()
    print("\n2.drawing a rectangle in the main frame")
    cv2.rectangle(img,(400,400),(100,100),(0,255,0),0)
    print("\n3.Croping the MainFrame")
    crop_img = img[100:300, 100:300]
    print("\n4.Converting it into greyscale")
    grey = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
    value = (35,35)
    print("\n5.Blurring the greyscale image")
    blurred = cv2.GaussianBlur(grey, value, 0)
    print("\n6.Thresholded on the blurimage")
    _, thresh1 = cv2.threshold(blurred, 200, 255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
    print("\n7.Image ")
    cv2.imshow('Thresholded', thresh1)
    print("\n8.Finding the Largest Contour")
    contours, hierarchy = cv2.findContours(thresh1.copy(),cv2.RETR_TREE, \
            cv2.CHAIN_APPROX_NONE)
    max_area = -1

    for i in range(len(contours)):
        cnt=contours[i]
        area = cv2.contourArea(cnt)
```

```
        if(area>max_area):
            max_area=area
            ci=i

    cnt=contours[ci]
    print("\n9.Drawing a rectangle around max Contour")
    x,y,w,h = cv2.boundingRect(cnt)
    cv2.rectangle(crop_img,(x,y),(x+w,y+h),(0,0,255),0)
    hull = cv2.convexHull(cnt)
    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)
    hull = cv2.convexHull(cnt,returnPoints = False)
    defects = cv2.convexityDefects(cnt,hull)
    count_defects = 0
    cv2.drawContours(thresh1, contours, -1, (0,255,0), 3)
    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])
        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)
        angle = math.acos((b**2 + c**2 - a**2)/(2*b*c)) * 57
        if angle <= 90:
            count_defects += 1
            cv2.circle(crop_img,far,1,[0,0,255],-1)
        #dist = cv2.pointPolygonTest(cnt,far,True)
        cv2.line(crop_img,start,end,[0,255,0],2)
        #cv2.circle(crop_img,far,5,[0,0,255],-1)
    if count_defects == 1:
        cv2.putText(img,"This is 1", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
```

```
63.     elif count_defects == 2:
64.         str = "This is 2"
65.         print str
66.         cv2.putText(img, str, (5,50), cv2.FONT_HERSHEY_SIMPLEX, 1, 2)
67.     elif count_defects == 3:
68.         cv2.putText(img,"This is 3 ", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
69.     elif count_defects == 4:
70.         cv2.putText(img,"This is 4", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
71.     else:
72.         cv2.putText(img,"Hello World!!!", (50,50),\
73.                     cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
74.     #cv2.imshow('drawing', drawing)
75.     #cv2.imshow('end', crop_img)
76.     cv2.imshow('Gesture', img)
77.     all_img = np.hstack((drawing, crop_img))
78.     cv2.imshow('Contours', all_img)
79.     k = cv2.waitKey(10)
80.     if k == 27:
81.         break
```

Screen shot:



BackGround Subtractor Module:

Code:

```
#The needed libraries OpenCV , Numpy & Math
import cv2
import numpy as np
import math
#Object for using the WebCam
cap = cv2.VideoCapture(0)
#Object for using the BackGroundSubtractor
backSub = cv2.BackgroundSubtractorMOG(history=3,nmixtures=5,backgroundRatio=0.5,
noiseSigma=1)
backSub2 = cv2.BackgroundSubtractorMOG2()


while(cap.isOpened()):
                #Getting the image from the WebCam  Edit  on 7th-OCt
        ret, img = cap.read()
                #Applying the Background Subtractor Edit on 7th Oct
        back_img = backSub.apply( img , learningRate = .7 )
        back_img2 = backSub2.apply( img , learningRate = .1 )


         #Showing the Background Subtractor Edit on 7th Oct
        cv2.imshow( 'BackGroundSubtractor' , back_img )
        cv2.imshow( 'BackGroundSubtractor2' , back_img )

        contours, hierarchy = cv2.findContours(back_img.copy(),cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)
        max_area = -1
        for i in range(len(contours)) :
                cnt=contours[i]
                area = cv2.contourArea(cnt)
                if(area>max_area ):
                        max_area=area
                        x,y,w,h = cv2.boundingRect(contours[i])
                        temp=cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),0)

        k = cv2.waitKey(10)
```

GUI - User Side:

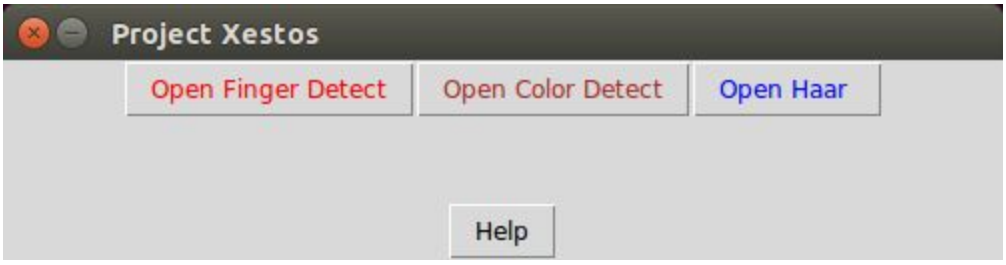
CODE :

```
1. from Tkinter import *
2. root = Tk()
3. root.resizable(width=FALSE, height=FALSE)
4. root.geometry('{}x{}'.format(500, 100))
5. root.wm_title("Project Xestos")
6. frame = Frame(root)
7. frame.pack()
8. bottomframe = Frame(root)
9. bottomframe.pack( side = BOTTOM )
10. def callback():
11. execfile("/home/hoshank/Downloads/ppl/xestos/final/Gestures/color/hand_gesture.py")
12. def haar():
13. execfile("/home/hoshank/Downloads/ppl/xestos/final/Gestures/Haar/hand_detect.py")
14. def finger():
15.   execfile("/home/hoshank/Downloads/ppl/xestos/final/finger gesture
    detection/gesture.py")
16. fingerdetect = Button(frame, text="Open Finger Detect", fg="red",command=finger)
17. fingerdetect.pack( side = LEFT)
18. color = Button(frame, text="Open Color Detect", fg="brown",command=callback)
19. color.pack( side = LEFT )
20. haar = Button(frame, text="Open Haar ", fg="blue",command=haar)
21. haar.pack( side = LEFT )
22. help = Button(bottomframe, text="Help", fg="black")
23. help.pack( side = BOTTOM)
24. root.mainloop()
```

Integrating OS Actions in Python from GUI

```
with open('file.txt', 'r') as the_file:
        choice=the_file.read()

if choice == 1:
        #do option 1
        subprocess.call("firefox")
if choice == 2:
        # do option 2
        subprocess.call("vlc")
if choice == 3:
        # do option 3
        subprocess.call("bash")
if choice == 4:
        # do option 4
        subprocess.call("ls")
if choice == 5:
        # do option 5
        subprocess.call("python")
```

Screen Shot:



Bibliography

"OpenCV-Python Tutorials¶." *OpenCV-Python Tutorials — OpenCV 3.0.0-dev documentation*. Web. 15 Nov. 2015. <http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html>

"OpenCV: Template Matching." *OpenCV: Template Matching*. Web. 15 Nov. 2015. <http://docs.opencv.org/master/d4/dc6/tutorial_py_template_matching.html#gsc.tab=0>

"Playing Pacman With Gestures: Python OpenCV." *Medium*. N.p., Sep. 2015. Web. 15 Nov. 2015. <https://medium.com/@vipul_sharma/playing-pacman-with-gestures-python-opencv-f498306b24fa>

"Python GUI Programming (Tkinter)." *www.tutorialspoint.com*. Web. 15 Nov. 2015. <http://www.tutorialspoint.com/python/python_gui_programming.htm>

"Stack Overflow." *Stack Overflow*. Web. 15 Nov. 2015. <http://stackoverflow.com/>