Summer Training Project Report

Fingers Detection and Counting

SUBMITTED BY

Kashish Aggarwal
UE198050

UNDER THE GUIDANCE OF

Dr. Amandeep Verma

SUBMITTED TO THE

Information Technology Department

University Institute of Engineering and Technology,
Panjab University, Chandigarh

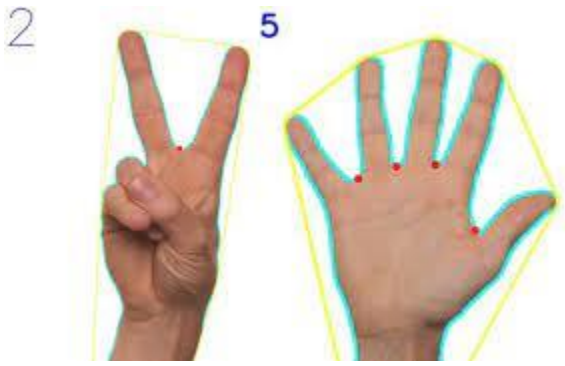
On behalf of Summer Training, July 2021

Table of Content

Introduction:

We will be creating a program that can detect a hand, segment the hand, and count the number of fingers being held up!

This is a system that detects a human hand, segments the hand using thresholding, counts the number of fingers being held up, and displays the finger count, all from live video input.



- This technique can be used in gesture recognition.
- This technique can also be used to detect if a person is left-handed or right-handed.

Code of Fingers Detection and Counting

**********Start of the code**********

Import Libraries:

```
[1]: import cv2

     import numpy as np

     from sklearn.metrics import pairwise
```

- First of all, to make a project in python, we need to import some of the packages of python.
- OpenCV (cv2) is the main package that I used in this project. This is the image processing library.
- At last, I import the pairwise function from the sci-kit learn module which is part of the metrics module.

Assigning Variables:

```
[2]: background = None

     accumulated_weight = 0.5
```

- I declare two variables, one is the background which is initially None and 2nd is accumulated_weight with an initial value of 0.5.

Calculate the accumulated weight:

[3]:
```
def calc_accum_avg(frame, accumulated_weight):

    global background

    if background is None:
        background = frame.copy().astype("float")
        return None

    cv2.accumulateWeighted(frame, background, accumulated_weight)
```

- This function takes two arguments, the first is a frame and the second is accumulated_weight.
- This function updates a running average of the background values in a region of interest.
- This function allows us to detect new objects (hands) in a frame.

Segmenting the hand in ROI:

```
[4]: def find_segment(frame, threshold_min=25):
         global background

         diff = cv2.absdiff(background.astype("uint8"), frame)

         ret , thresholded = cv2.threshold(diff, threshold_min,
                                           255,
                                           cv2.THRESH_BINARY)

         image,contours, hierarchy = cv2.findContours(thresholded.copy(),cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

         if len(contours) == 0:
             return None
         else:
             hand_segment = max(contours, key=cv2.contourArea)

             return (thresholded, hand_segment)
```

- In this, I use the technique of Thresholding to actually grab the hand segment from the region of interest.
- Basically, in the end, we want to have the white hand and then be able to calculate the contour around that against a black background.

Thresholding:

- Thresholding is a technique in OpenCV, which is the assignment of pixel values in relation to the threshold value provided.
- In thresholding, each pixel value is compared with the threshold value. If the pixel value is smaller than the threshold, it is set to 0, otherwise, it is set to a maximum value (generally 255).
- Thresholding is a very popular segmentation technique, used *to separate an object considered a foreground from its background.*.

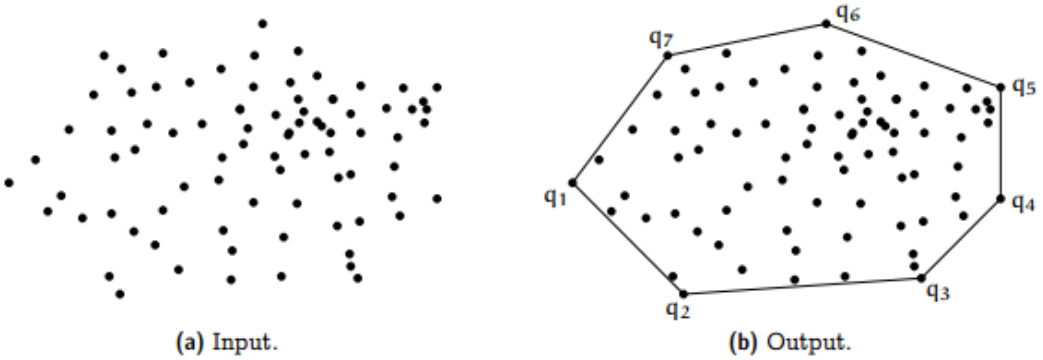
Finger Counting with Convex Hull:

```
[ ]: def count_fingers(thresholded, hand_segment):

    conv_hull = cv2.convexHull(hand_segment)
    top    = tuple(conv_hull[conv_hull[:, :, 1].argmin()][0])
    bottom = tuple(conv_hull[conv_hull[:, :, 1].argmax()][0])
    left   = tuple(conv_hull[conv_hull[:, :, 0].argmin()][0])
    right  = tuple(conv_hull[conv_hull[:, :, 0].argmax()][0])

    cX = (left[0] + right[0]) // 2
    cY = (top[1] + bottom[1]) // 2

    distance = pairwise.euclidean_distances([(cX, cY)], Y=[left, right, top, bottom])[0]

    max_distance = distance.max()

    radius = int(0.8 * max_distance)
    circumference = (2 * np.pi * radius)
    circular_roi = np.zeros(thresholded.shape[:2], dtype="uint8")

    cv2.circle(circular_roi, (cX, cY), radius, 255, 10)

    circular_roi = cv2.bitwise_and(thresholded, thresholded, mask=circular_roi)

    image, contours, hierarchy = cv2.findContours(circular_roi.copy(), cv2.RETR_EXTERNAL,
                                                  cv2.CHAIN_APPROX_NONE)
    count = 0

    for cnt in contours:
        (x, y, w, h) = cv2.boundingRect(cnt)

        out_of_wrist = ((cY + (cY * 0.25)) > (y + h))

        limit_points = ((circumference * 0.25) > cnt.shape[0])

        if  out_of_wrist and limit_points:
            count += 1

    return count
```

Convex Hull:

Hull means the exterior or the shape of the object. Therefore, the Convex Hull of a shape or a group of points is a tight-fitting convex boundary around the points or the shape

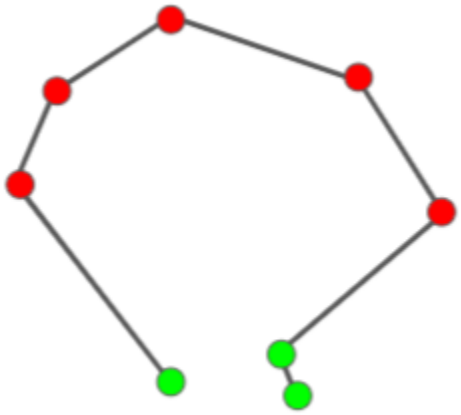
- The Convex Hull of a convex object is simply its boundary.
- The Convex Hull of a concave shape is a convex boundary that most tightly encloses it.



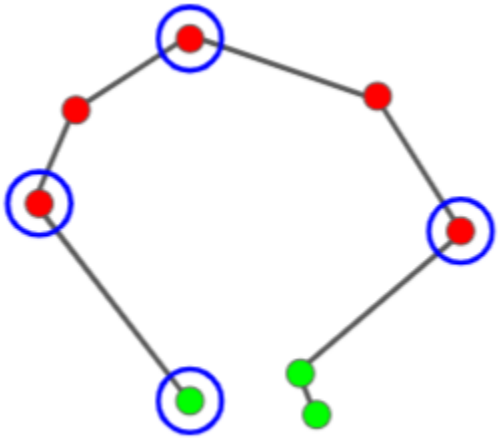
- Now that we have the hand segment, the next step is to count the fingers.
- We can do this by utilizing a Convex Hull.
- A convex hull draws a polygon by connecting points around the most external points in a frame.

Algorithm:

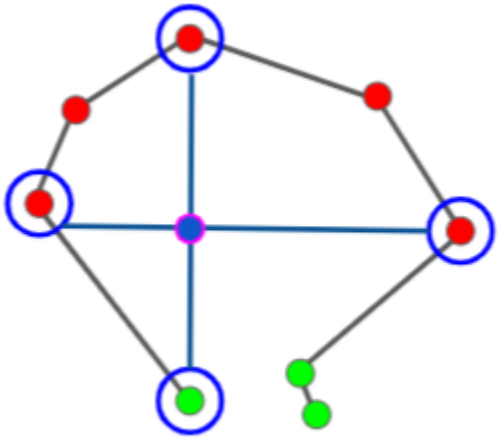
1. In our case, this set of points is just our thresholded image of a hand (and the external contour information.
2. We can expect the general shape of our polygon to be something like this:



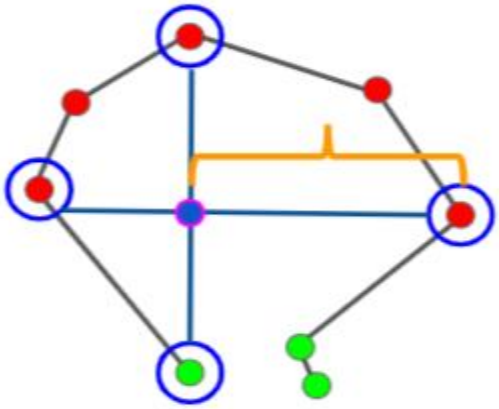
3. First, we will calculate the most extreme points (top, bottom, left, and right).

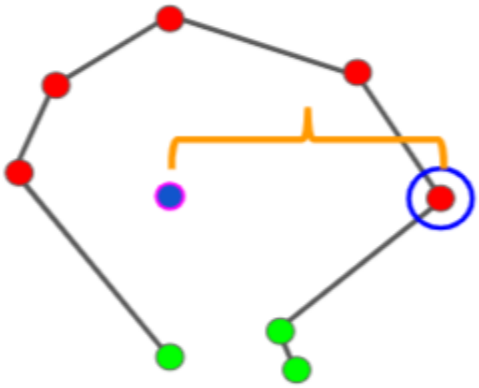


4. We can then calculate their intersection and estimate that as the center of the hand

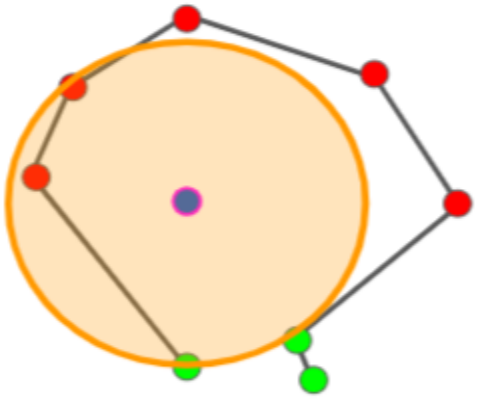


5. Next, we will calculate the distance for the point furthest away from the center

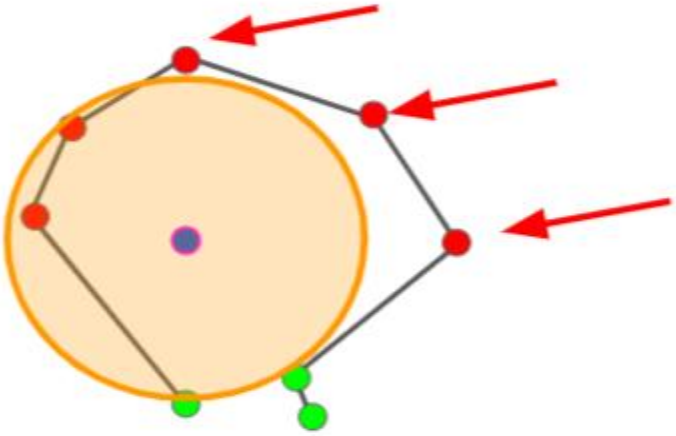




6. Then using a ratio of that distance we create a circle.



7. Any points outside of this circle and far away enough from the bottom should be extended fingers!



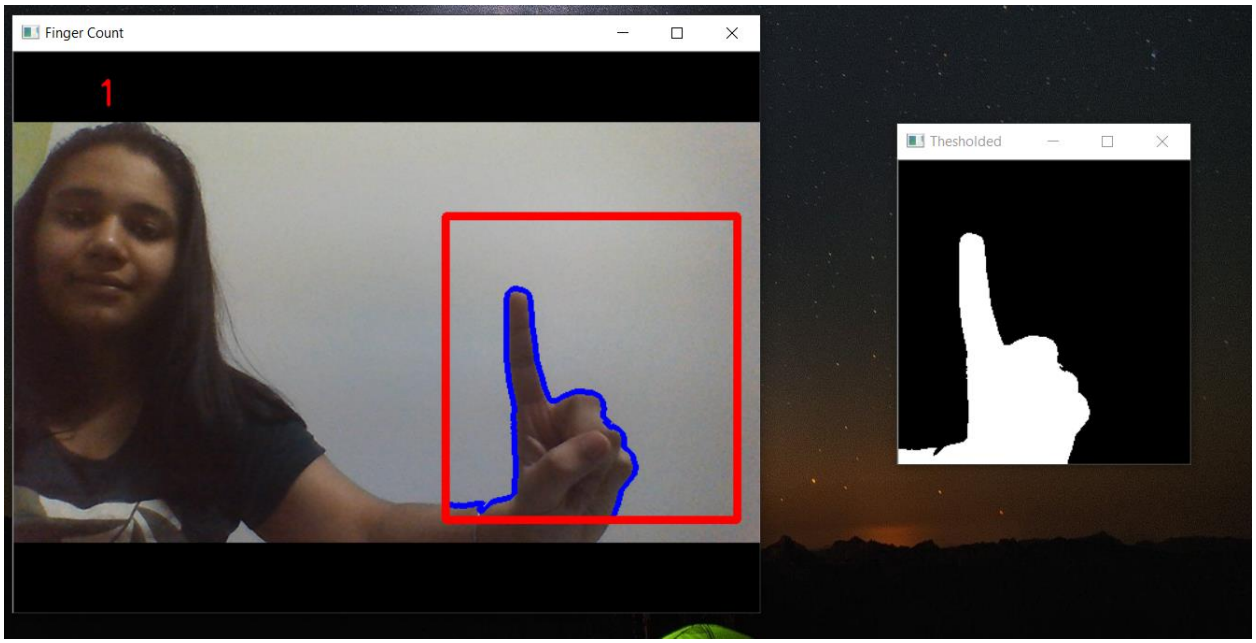
Main Code:

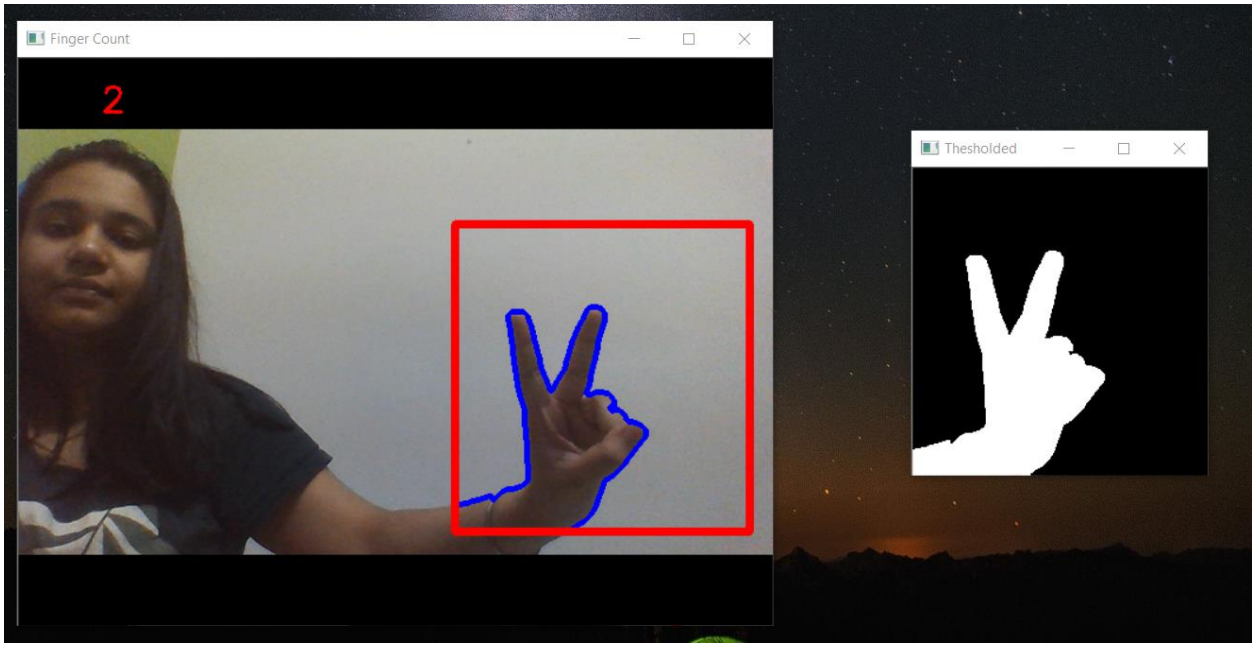
```
[ ]: roi_top = 140
roi_bottom = 400
roi_right = 370
roi_left = 620
cam = cv2.VideoCapture(0)
num_frames = 0
while True:
    ret, frame = cam.read()
    frame = cv2.flip(frame, 1)
    frame_copy = frame.copy()
    roi = frame[roi_top:roi_bottom, roi_right:roi_left]
    gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
    gray = cv2.GaussianBlur(gray, (7, 7), 0)
    if num_frames < 60:
        calc_accum_avg(gray, accumulated_weight)
        if num_frames <= 59:
            cv2.putText(frame_copy, "WAIT! GETTING BACKGROUND AVG.", (200, 300), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 2)
            cv2.imshow("Finger Count",frame_copy)
    else:
        hand = find_segment(gray)
        if hand is not None:
            thresholded, hand_segment = hand
            cv2.drawContours(frame_copy, [hand_segment + (roi_right, roi_top)], -1, (255, 0, 0),4)
            fingers = count_fingers(thresholded, hand_segment)
            if fingers <=5:
                cv2.putText(frame_copy, str(fingers), (70, 45), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 2)
            cv2.imshow("Thesholded", thresholded)
    cv2.rectangle(frame_copy, (roi_left, roi_top), (roi_right, roi_bottom), (0,0,255), 5)
    num_frames += 1
    cv2.imshow("Finger Count", frame_copy)

    k = cv2.waitKey(1) & 0xFF

    if k == 27:
        break

cam.release()
cv2.destroyAllWindows()
```

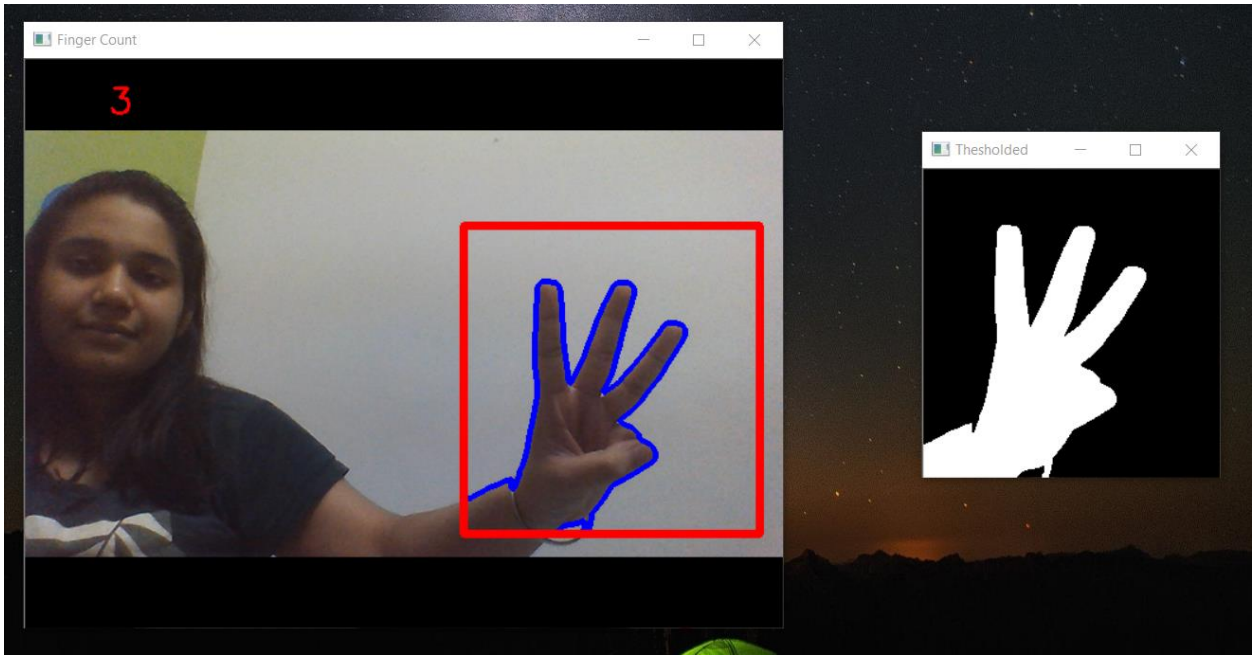
**********End of Code**********

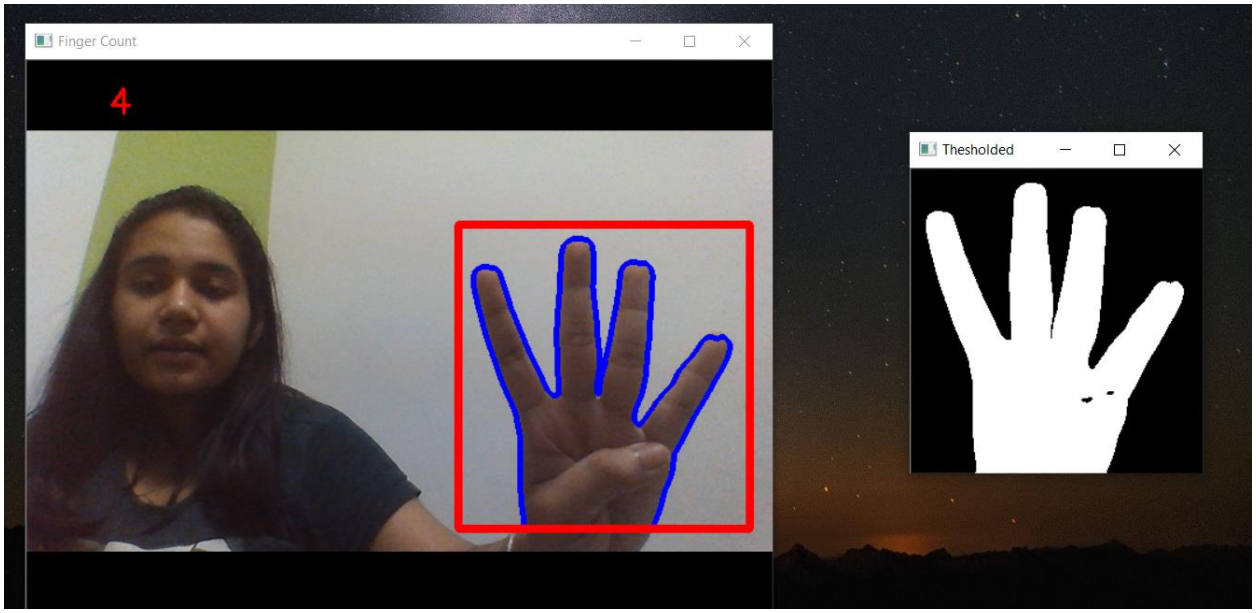
Output:

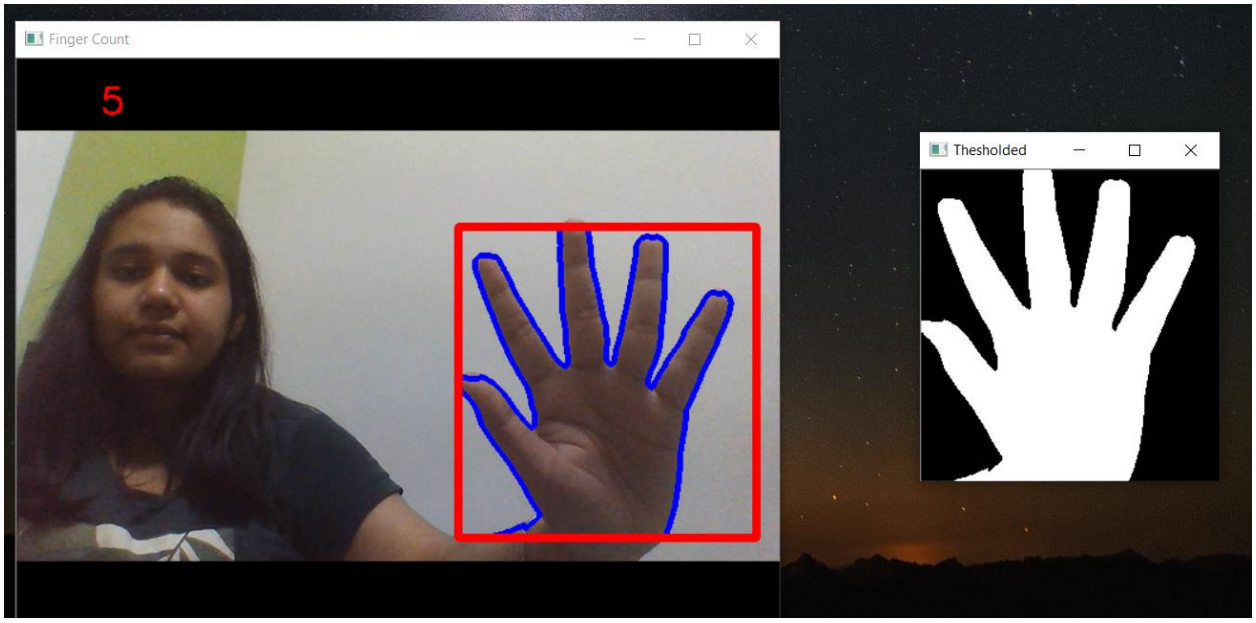
The following snapshots are some examples of the input.




Finger Count
2
Thesholded


Finger Count
3
Thesholded



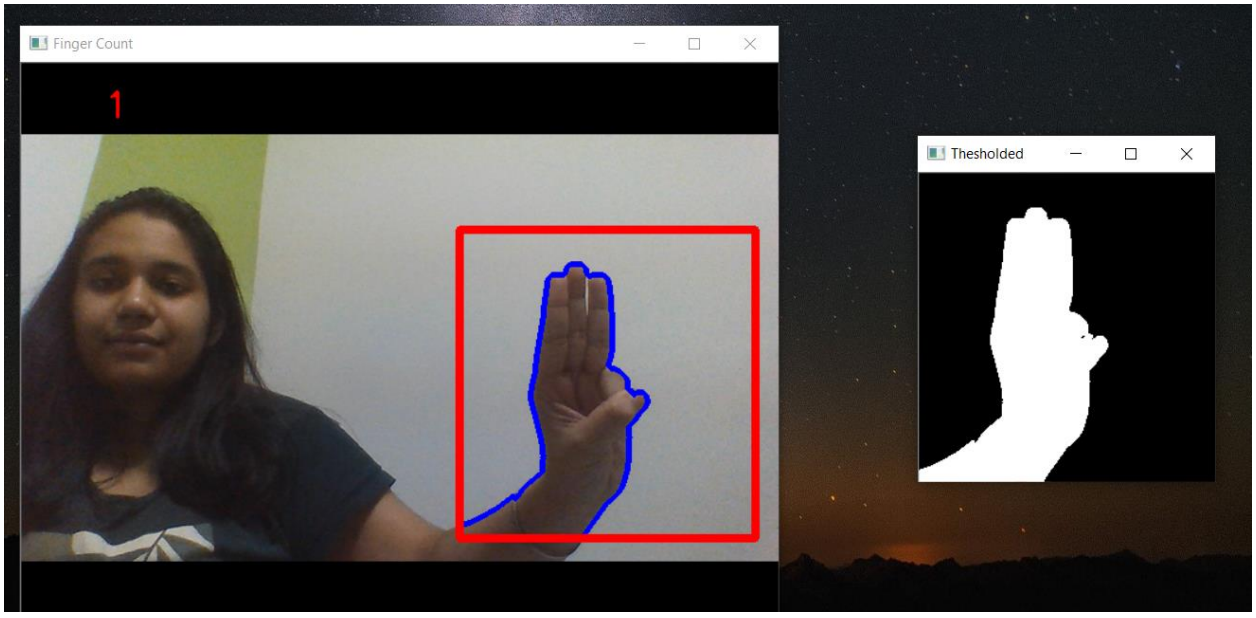


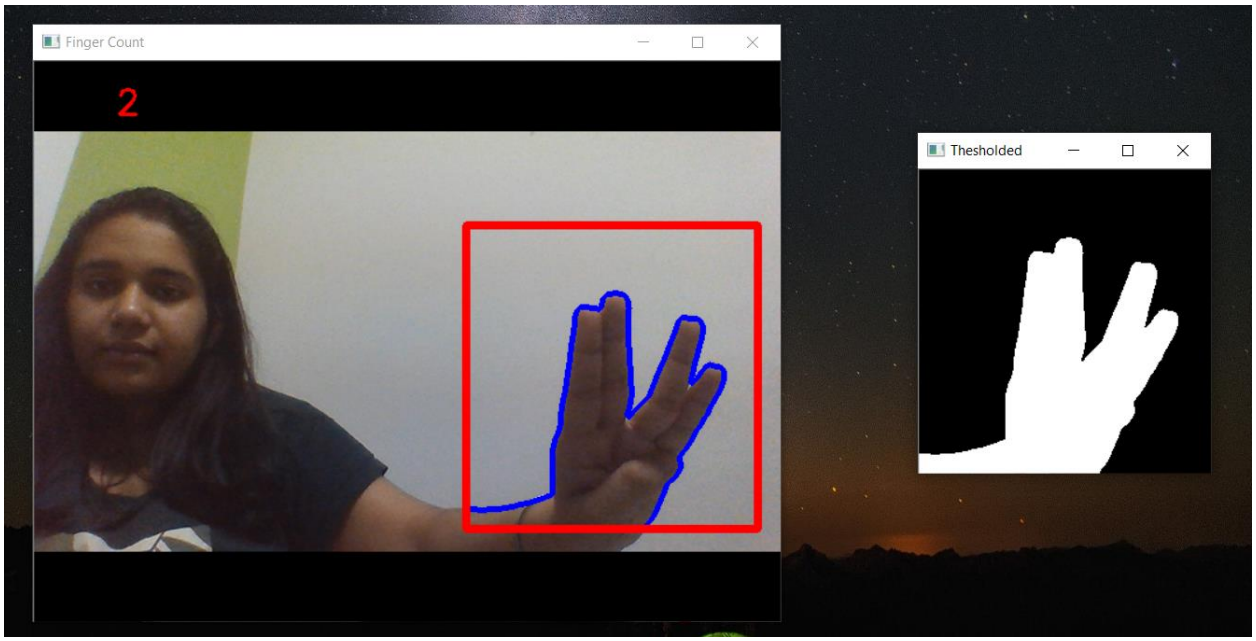
Clearly, we can see that this program is able to find the desired result.

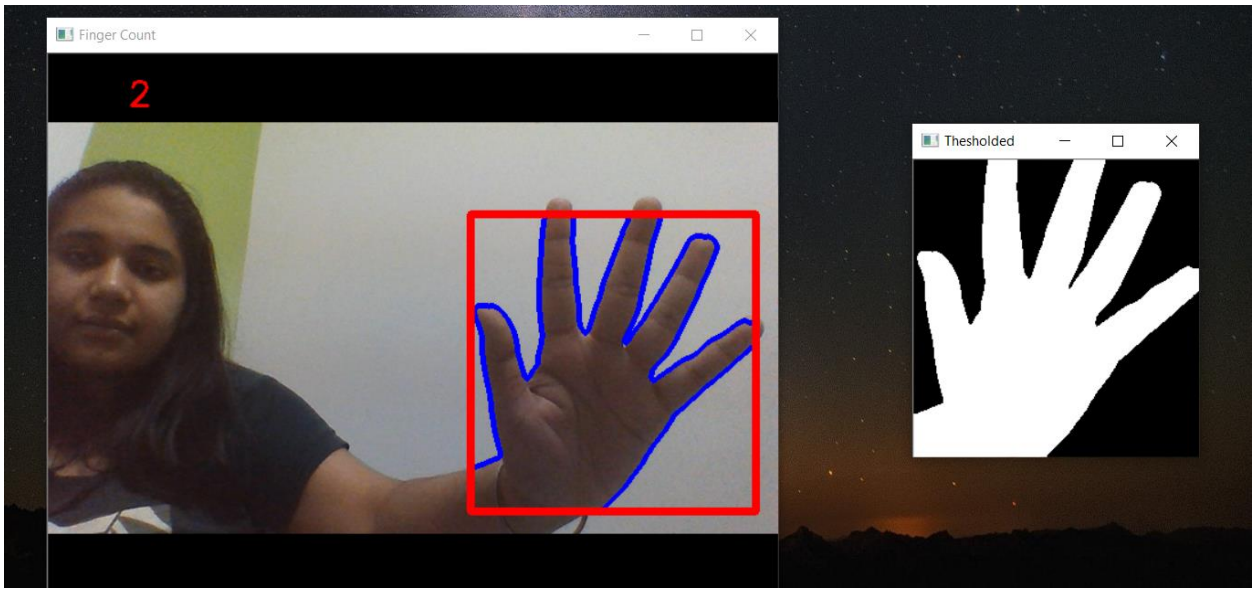
Conclusion and Future Scope:

This was a great experience to make this project. I learned a lot from this project. This project can be extended to a further level. The logic applied in this project can make accurate.

This program may wrongly detect in some cases. Some of the cases are shown below:







By modifying the above algorithm we can make this much better.

References:

- www.python.org
- www.stackoverflow.com
- https://docs.opencv.org/4.5.2/d6/d00/tutorial_py_root.html