# Image Processing using Python and OpenCV

Name : Aastha Joshi & Aaron Christian.

Roll No : 1891015 & 179003 .

Guide : Manish Solanki .

# Topics To Be Covered

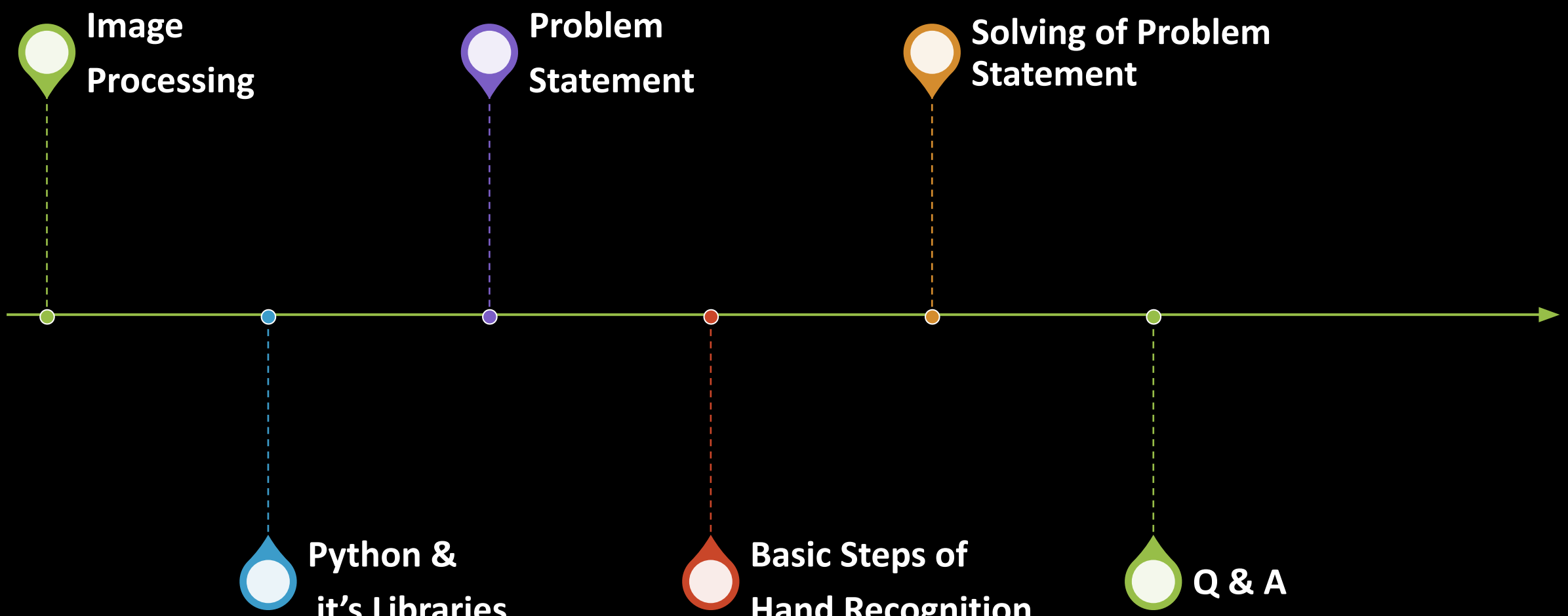
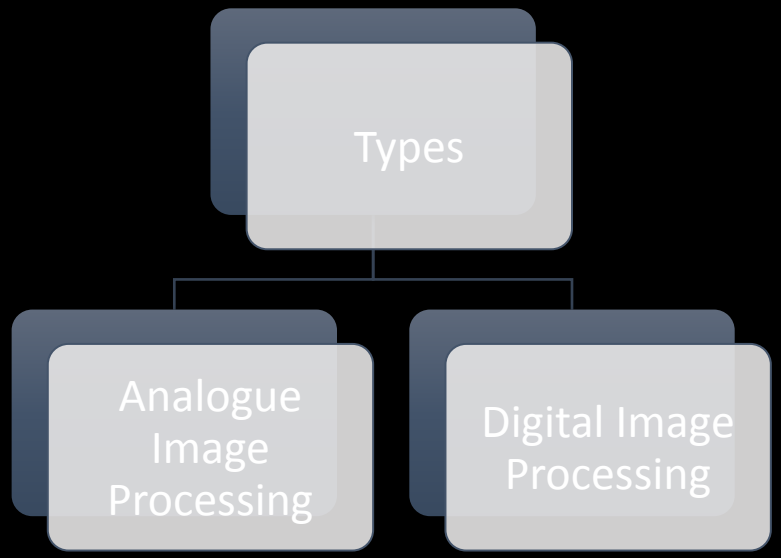


**Image processing 3 Steps**

1. Importing the image via image acquisition tools;
2. Analysing and manipulating the image;
3. Output in which result can be altered image or report that is based on image analysis;

## Image Processing



Scipy

PIL/ Pillow

Numpy

OpenCV-Python

Scikit Image

SimpleCV

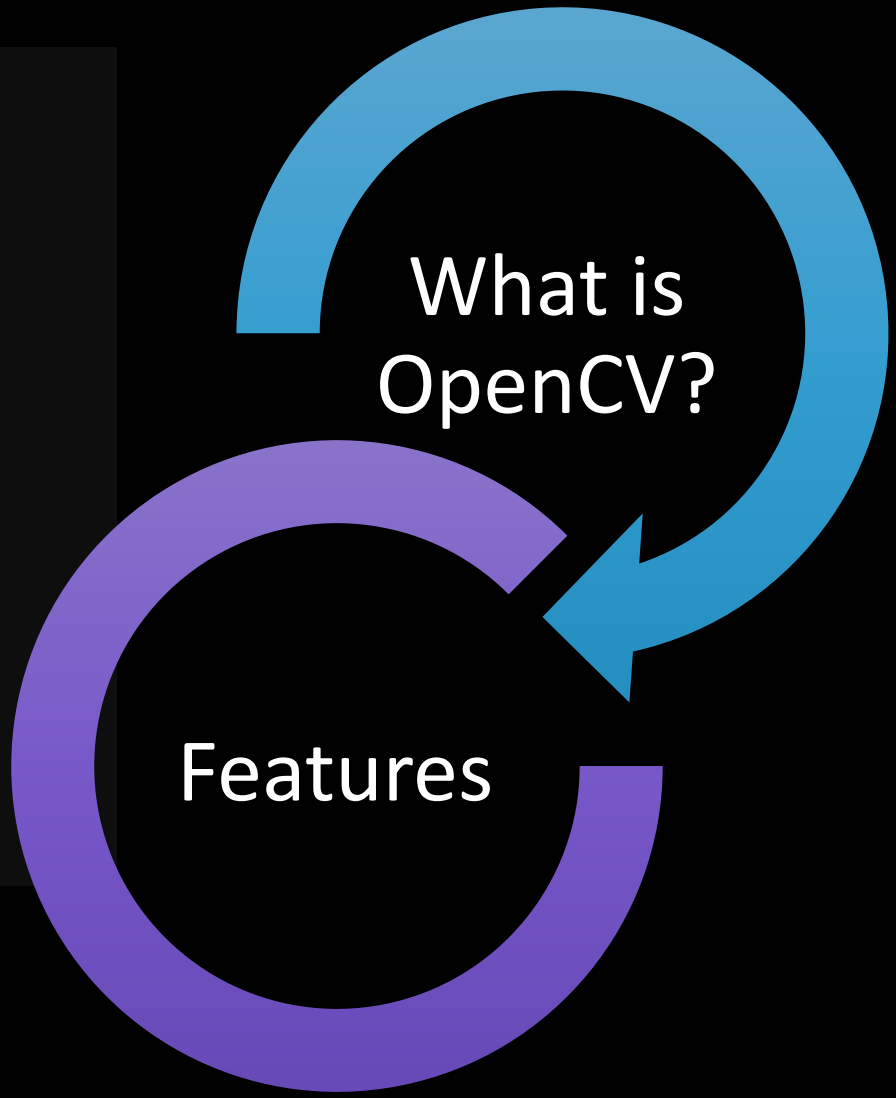
Pycairo

Mahotas

pgmagick

SimpleITK


OpenCV


What is OpenCV?
Features

# HAND GESTURE RECOGNITION

# Problem statement

Recognize hand gestures and count number of fingers from a  live video sequence.

 To recognize these gestures from a live video sequence, we first need to take out the hand region alone removing all the unwanted portions in the video sequence. After segmenting the hand region, we then count the fingers. Thus, the entire problem could be solved using 2 simple steps

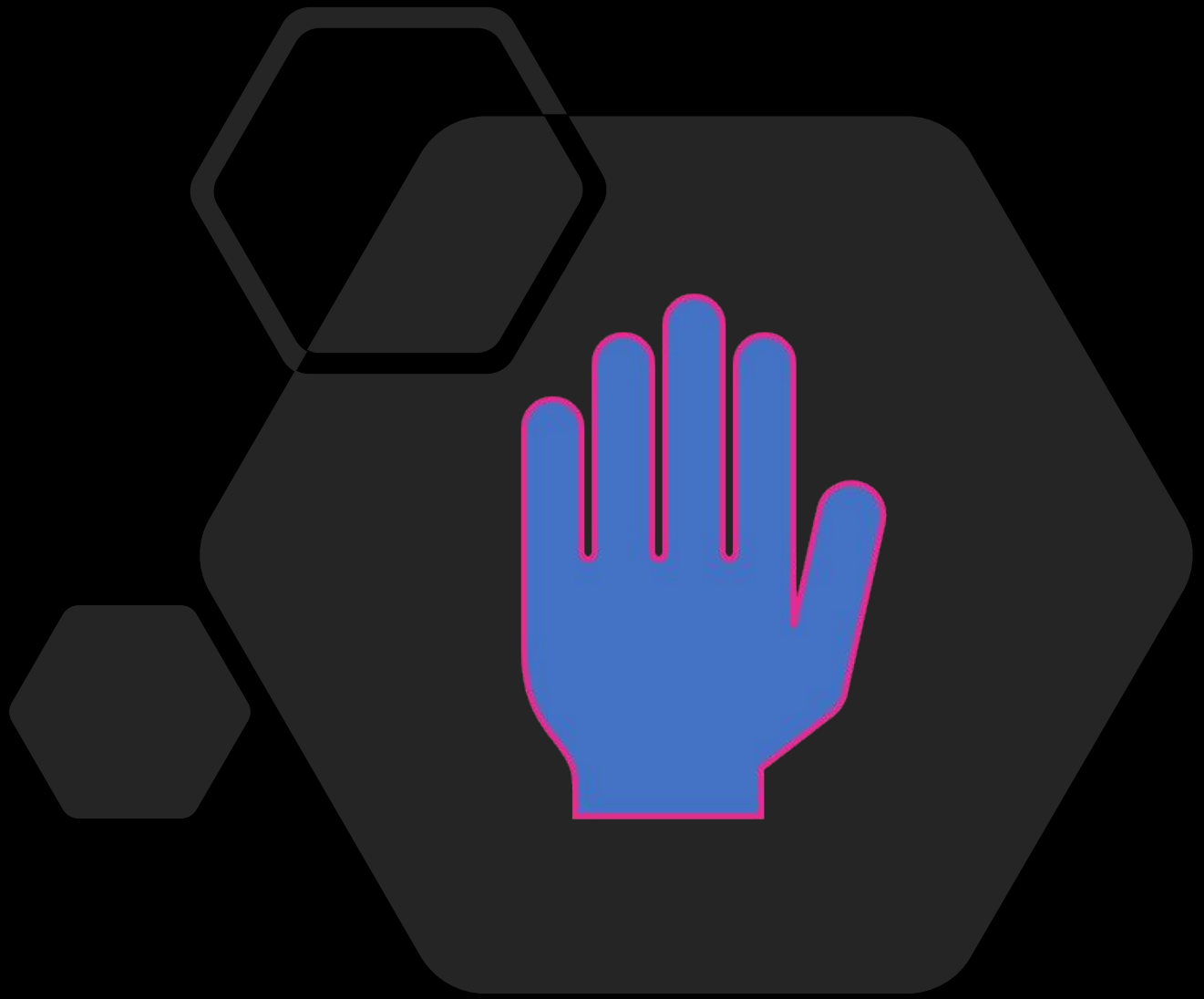
Step1:Find and segment the hand region from the video sequence

Step2:Count the number of fingers from the segmented hand region in the video sequence.

# The Basic Steps of Hand Recognition

STEP 1: **Segment the Hand region**

STEP 2: **Background Subtraction**



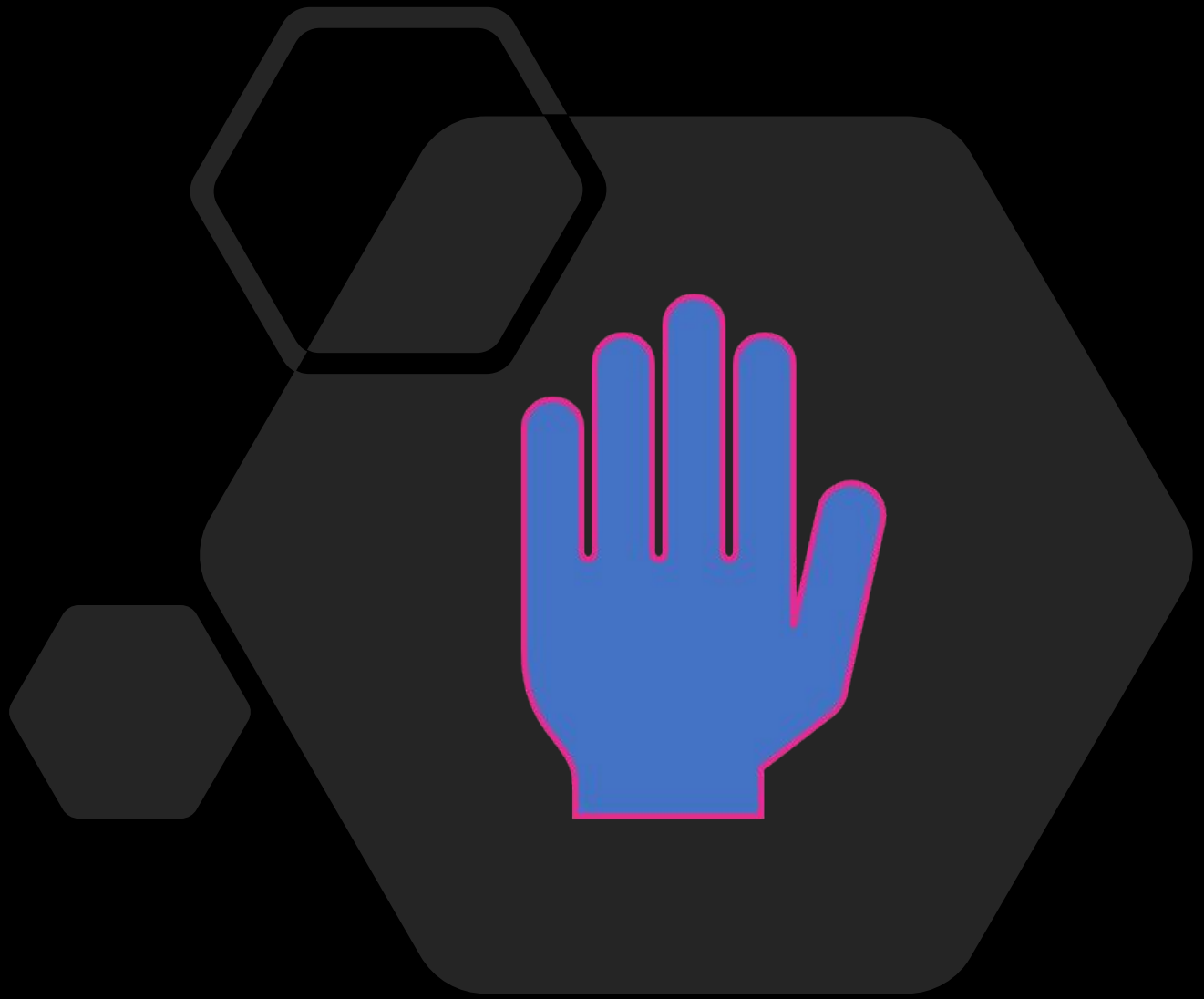
# Background Subtraction.

FIG1.

# The Basic Steps of Hand Recognition

STEP 3 : **Motion Detection and Thresholding**

STEP 4 : **Contour Extraction**



1

Segment the Hand region

2

Background Subtraction

3

Motion Detection and Thresholding.

4

Contour Extraction

**Step1:Find and segment the hand region from the video sequence.**

**Is further Subdivided into 4 parts .**


Hand-Gesture Recognition algorithm to count the fingers

- Find Convex Hull
- Find Center of Palm
- Euclidean Distance
- Bitwise And

FIG2.

```python
# organize imports
import cv2
import imutils
import numpy as np
from sklearn.metrics import pairwise

# global variables
bg = None

# To find the running average over the background
def run_avg(image, accumWeight):
    global bg
    # initialize the background
    if bg is None:
        bg = image.copy().astype("float")
        return
  # compute weighted average, accumulate it and update the ba
ckground
    cv2.accumulateWeighted(image, bg, accumWeight)

  # To segment the region of hand in the image
def segment(image, threshold=25):
    global bg
    # find the absolute difference between background and current frame
    diff = cv2.absdiff(bg.astype("uint8"), image)

    # threshold the diff image so that we get the foreground
    thresholded = cv2.threshold(diff, threshold, 255, cv2.THRESH_BINARY)[1]

    # get the contours in the thresholded image
    ( cnts, _) = cv2.findContours(thresholded.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    # return None, if no contours detected
    if len(cnts) == 0:
        return
    else:
        # based on contour area, get the maximum contour which is the hand
        segmented = max(cnts, key=cv2.contourArea)
        return (thresholded, segmented)
```

```python
def count(thresholded, segmented):
    # find the convex hull of the segmented hand region
    chull = cv2.convexHull(segmented)

    extreme_top    = tuple(chull[chull[:, :, 1].argmin()][0])
    extreme_bottom = tuple(chull[chull[:, :, 1].argmax()][0])
    extreme_left   = tuple(chull[chull[:, :, 0].argmin()][0])
    extreme_right  = tuple(chull[chull[:, :, 0].argmax()][0])

    # find the center of the palm
    cX = int((extreme_left[0] + extreme_right[0]) / 2)
    cY = int((extreme_top[1] + extreme_bottom[1]) / 2)


    distance = pairwise.euclidean_distances([(cX, cY)], Y=[extreme_left, extr
eme_right, extreme_top, extreme_bottom])[0]
    maximum_distance = distance[distance.argmax()]

      radius = int(0.8 * maximum_distance)

    # find the circumference of the circle
    circumference = (2 * np.pi * radius)

    # take out the circular region of interest which has
    # the palm and the fingers
    circular_roi = np.zeros(thresholded.shape[:2], dtype="uint8")

    # draw the circular ROI
    cv2.circle(circular_roi, (cX, cY), radius, 255, 1)

    # take bit-wise AND between thresholded hand using the circular ROI as th
e mask
    # which gives the cuts obtained using mask on the thresholded hand image
    circular_roi = cv2.bitwise_and(thresholded, thresholded, mask=circular_ro
i)

    # compute the contours in the circular ROI
    ( cnts, _) = cv2.findContours(circular_roi.copy(), cv2.RETR_EXTERNAL, cv2
.CHAIN_APPROX_NONE)
```

A circle with center point and radius obtained from
convex hull and
bit-wise AND with thresholded image

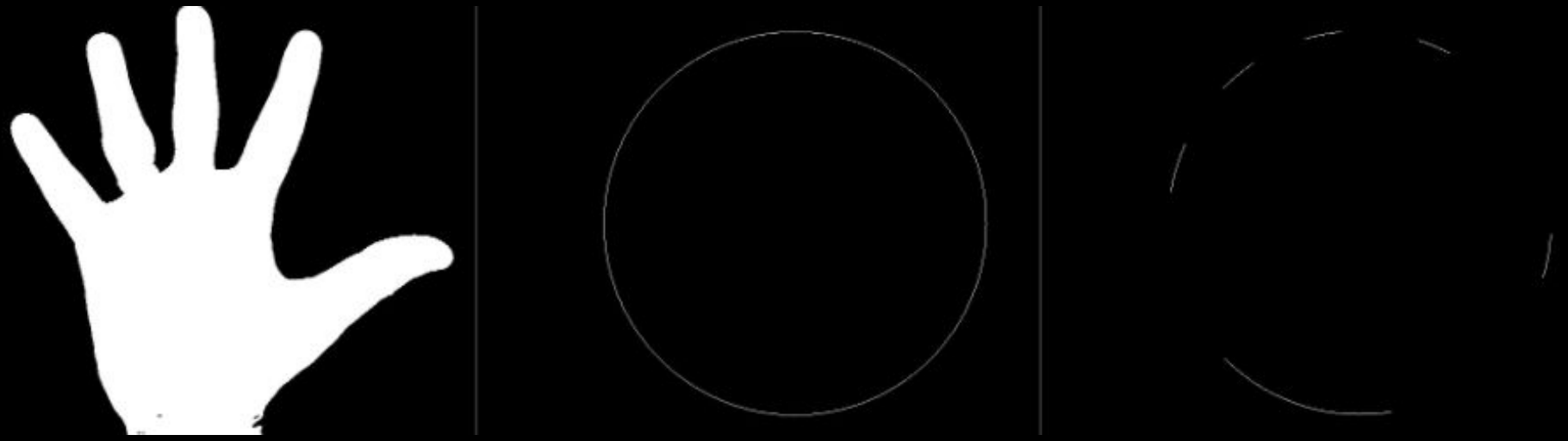
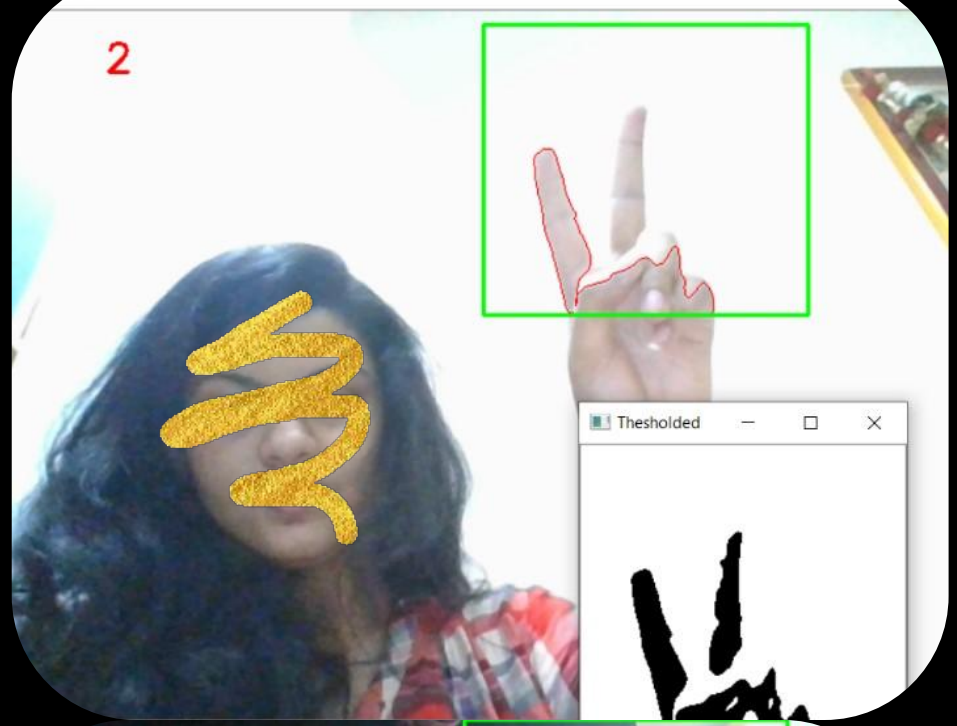


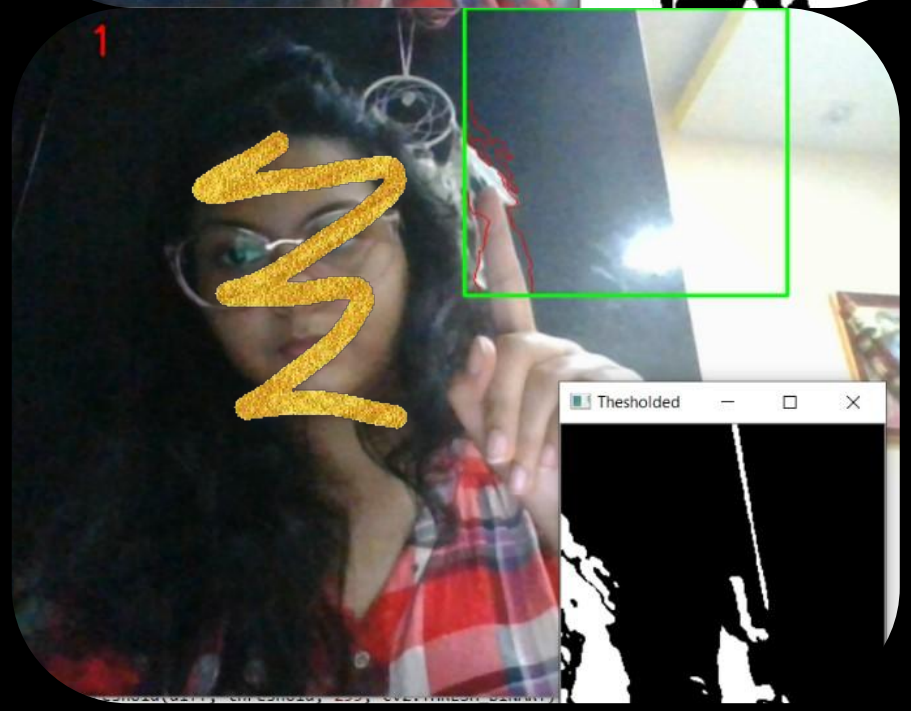
FIG3.

```
        count = 0

    # loop through the contours found
    for c in cnts:
        # compute the bounding box of the contour
        (x, y, w, h) = cv2.boundingRect(c)


        if ((cY + (cY * 0.25)) > (y + h)) and ((circumference * 0.25) > c.shape[0]):
            count += 1

    return count


# MAIN FUNCTION
if __name__ == "__main__":
    # initialize accumulated weight
    accumWeight = 0.5

    # get the reference to the webcam
    camera = cv2.VideoCapture(0)

    # region of interest (ROI) coordinates
    top, right, bottom, left = 10, 350, 225, 590

    # initialize num of frames
    num_frames = 0

    # calibration indicator
    calibrated = False

    # keep looping, until interrupted
    while(True):
        # get the current frame
        (grabbed, frame) = camera.read()

        # resize the frame
        frame = imutils.resize(frame, width=700)

        # flip the frame so that it is not the mirror view
        frame = cv2.flip(frame, 1)

        # clone the frame
        clone = frame.copy()

        # get the height and width of the frame
        (height, width) = frame.shape[:2]

        # get the ROI
        roi = frame[top:bottom, right:left]

        # convert the roi to grayscale and blur it
        gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
        gray = cv2.GaussianBlur(gray, (7, 7), 0)

        # to get the background, keep looking till a threshold is reached
        # so that our weighted average model gets calibrated
        if num_frames < 30:
            run_avg(gray, accumWeight)
            if num_frames == 1:
                print("[STATUS] please wait! calibrating...")
            elif num_frames == 29:
                print("[STATUS] calibration successfull...")
        else:
            # segment the hand region
            hand = segment(gray)
```

```
            # check whether hand region is segmented
             if hand is not None:
                # if yes, unpack the thresholded image and
                # segmented region
                (thresholded, segmented) = hand

                # draw the segmented region and display the frame
                cv2.drawContours(clone, [segmented + (right, top)
], -1, (0, 0, 255))

                # count the number of fingers
                fingers = count(thresholded, segmented)

                cv2.putText(clone, str(fingers), (70, 45), cv2.FO
NT_HERSHEY_SIMPLEX, 1, (0,0,255), 2)

                # show the thresholded image
                cv2.imshow("Thesholded", thresholded)

        # draw the segmented hand
        cv2.rectangle(clone, (left, top), (right, bottom), (0,255
,0), 2)

        # increment the number of frames
        num_frames += 1

        # display the frame with segmented hand
        cv2.imshow("Video Feed", clone)

        # observe the keypress by the user
        keypress = cv2.waitKey(1) & 0xFF

        # if the user pressed "q", then stop looping
        if keypress == ord("q"):
            break

# free up memory
camera.release()
cv2.destroyAllWindows()
```

# Output



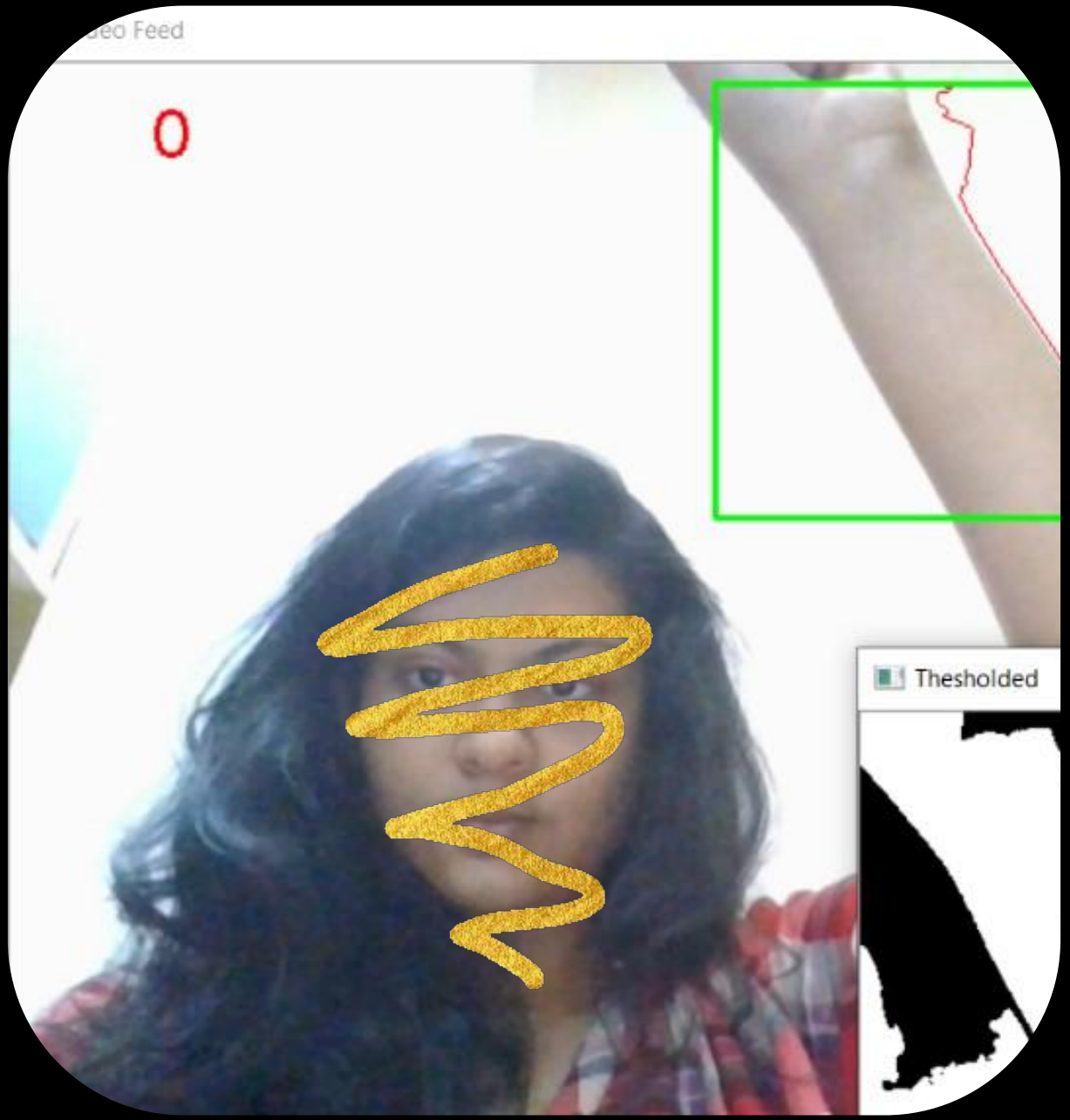
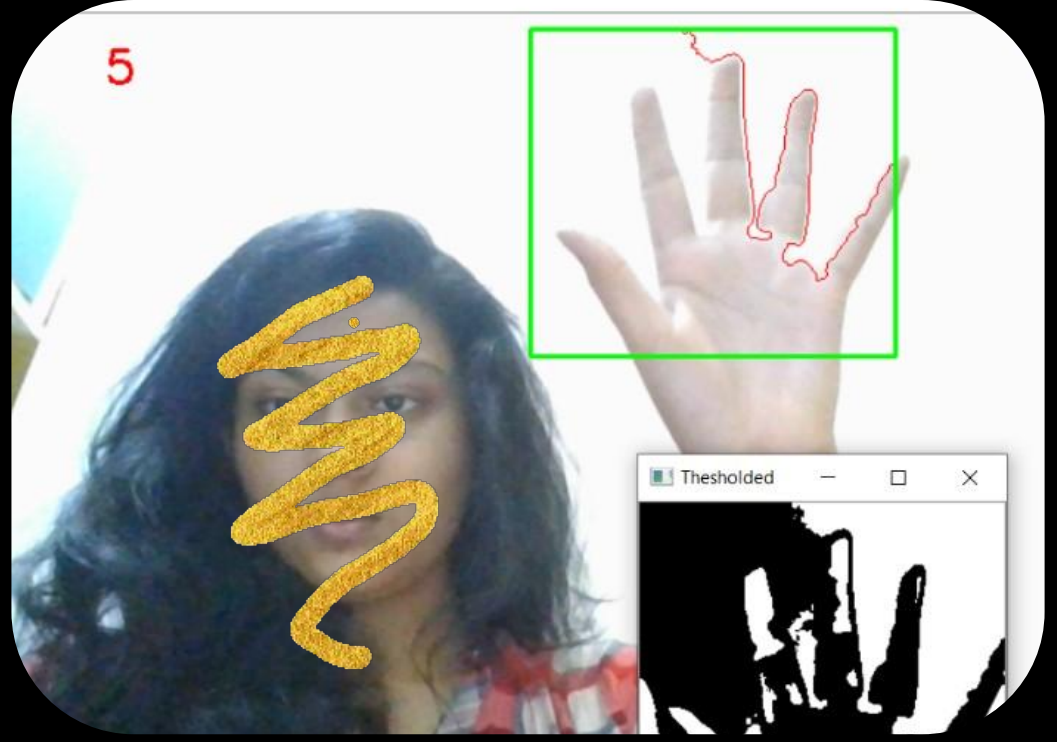
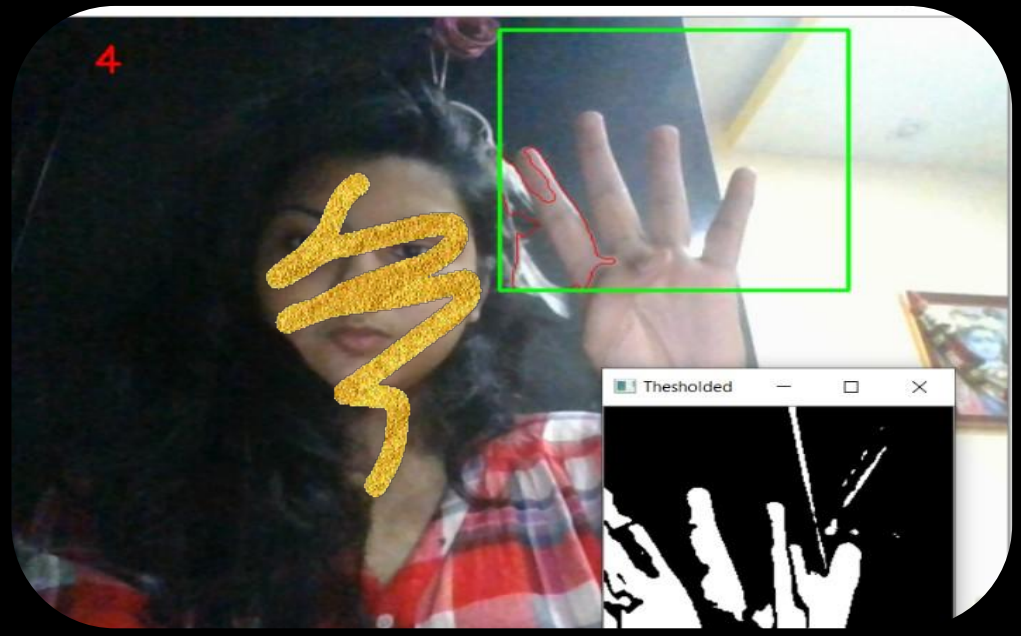




FIG4.



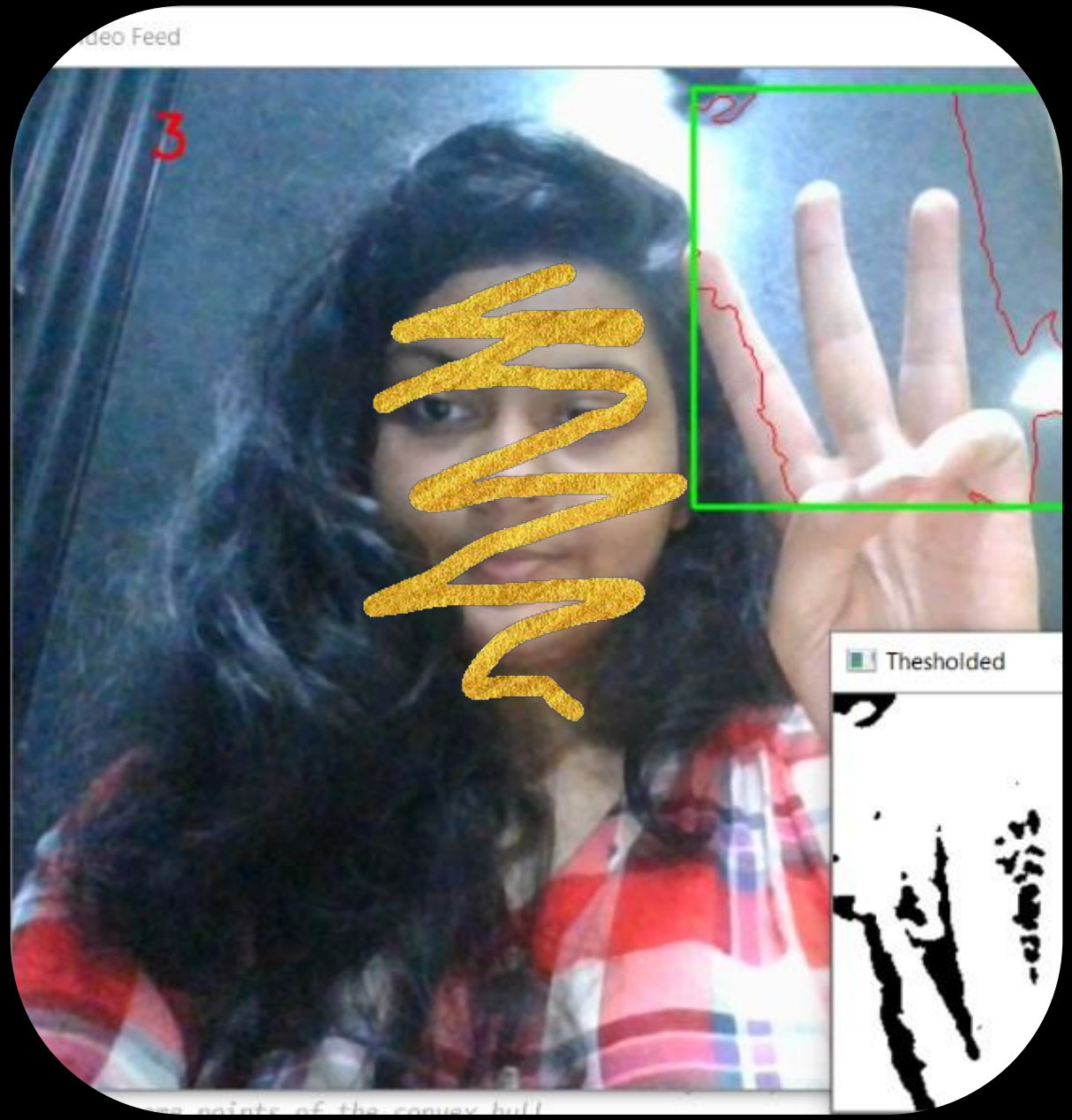




FIG5.

# CONCLUSION

•


•REFERENCES:

1. http://www.ijfrcsce.org/download/browse/Volume_4/June_18_Volume_4_Issue_6/1530176523_28-06-2018.pdf

2. https://sisu.ut.ee/imageprocessing/book/1

3. background subtraction image: https://pysource.com/2018/05/17/background-subtraction-opencv-3-4-with-python-3-tutorial-32/

# QUESTION AND ANSWER.

# THANK YOU.