```python
# step -1 import libraries
#  2- covert frame into hsv
# 3 - track hand on color basis
# 4 -  create mask on the basis o color and filter actual color
# 5 - invert pixcel value and then enchance the result for the better output
# 6 - find contour for spacific colored object
# 7 - find max area contour and draw it on live feed
# 8 - find convexity detect for counting value and apply cosin mehod
# 9 - put counting value on fingers on the basis of convexity defects
# 10 - Enjoy your output

import cv2
import numpy as np
import math

cap = cv2.VideoCapture('https://192.168.10.5:8080/video')


def cross(x):
    pass


cv2.namedWindow('color_Adjestments', cv2.WINDOW_NORMAL)
cv2.resizeWindow('color_Adjestments', (300, 300))
cv2.createTrackbar('Thresh', 'color_Adjestments', 0, 255, cross)
#  Color Detection Track

cv2.createTrackbar('Lower_H', 'color_Adjestments', 0, 255, cross)
cv2.createTrackbar('Lower_S', 'color_Adjestments', 0, 255, cross)
cv2.createTrackbar('Lower_V', 'color_Adjestments', 0, 255, cross)
cv2.createTrackbar('Upper_H', 'color_Adjestments', 255, 255, cross)
cv2.createTrackbar('Upper_S', 'color_Adjestments', 255, 255, cross)
cv2.createTrackbar('Upper_V', 'color_Adjestments', 255, 255, cross)


while True:
    res, frame = cap.read()
    if res == True:

        cv2.imshow('frame', frame)
        frame = cv2.flip(frame, 2)
        # print('--------------> ', frame.shape)
        frame = cv2.resize(frame, (500, 400))

        # get hand dat from the rectangle sub window
        cv2.rectangle(frame, (0, 1), (300, 500), (255, 0, 0), 0)
        crop_image = frame[1:500, 0:300]

        # step2
        hsv = cv2.cvtColor(crop_image, cv2.COLOR_BGR2HSV)

        # detecting hand
        l_h = cv2.getTrackbarPos('Lower_H', 'color_Adjestments')
        l_s = cv2.getTrackbarPos('Lower_S', 'color_Adjestments')
        l_v = cv2.getTrackbarPos('Lower_V', 'color_Adjestments')

        u_h = cv2.getTrackbarPos('Upper_H', 'color_Adjestments')
        u_s = cv2.getTrackbarPos('Upper_S', 'color_Adjestments')
        u_v = cv2.getTrackbarPos('Upper_V', 'color_Adjestments')

        # step-3
        lower_bound = np.array([l_h, l_s, l_v])
        upper_bound = np.array([u_h, u_s, u_v])

        # step 4
        # creating the mask
        mask = cv2.inRange(hsv, lower_bound, upper_bound)

        # filter mask with imgage
        filtr = cv2.bitwise_and(crop_image, crop_image, mask=mask)

        # step-5
        mask1 = cv2.bitwise_not(mask)
        m_g = cv2.getTrackbarPos('Thresh', 'color_Adjestments')
        ret, thresh = cv2.threshold(mask1, m_g, 255, cv2.THRESH_BINARY)
        dilata = cv2.dilate(thresh, (3, 3), iterations=6)

        # step 6
        cnts, hier = cv2.findContours(
            thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

        try:
            print('try')

            # step-7
            # Find contour with maximum area
            cm = max(cnts, key=lambda x: cv2.contourArea(x))

            epsilon = 0.0005*cv2.arcLength(cm, True)
            data = cv2.approxPolyDP(cm, epsilon, True)

            hull = cv2.convexHull(cm)
            cv2.drawContours(crop_image, [cm], -1, (50, 50, 150), 2)
            cv2.drawContours(crop_image, [hull], -1, (0, 255, 0), 2)

            # ste 8-
            # find convexity defects
            hull = cv2.convexHull(cm, returnPoints=False)

            # step 9
            defects = cv2.convexityDefects(cm, hull)
            count_defects = 0

            for i in range(defects.shape[0]):
                s, e, f, d = defects[i, 0]

                start = tuple(cm[s][0])
                end = tuple(cm[e][0])
                far = tuple(cm[f][0])

                # cosin rule
                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 ** -a ** 2)/(2*b*c))*180)

                if angle <= 50:
                    count_defects += 1
                    cv2.circle(crop_image, far, 5, [255, 255, 255], -1)

                print('count==', count_defects)
                if count_defects == 0:
                    cv2.putText(frame, '1', (50, 50), cv2.FONT_HERSHEY_SIMPLEX)
```

```
                elif count_defects == 1:
                    cv2.putText(frame, '2', (50, 50), cv2.FONT_HERSHEY_SIMPLEX)

                elif count_defects == 2:
                    cv2.putText(frame, '3', (50, 50), cv2.FONT_HERSHEY_SIMPLEX)

                elif count_defects == 3:
                    cv2.putText(frame, '4', (50, 50), cv2.FONT_HERSHEY_SIMPLEX)

                elif count_defects == 4:
                    cv2.putText(frame, '5', (50, 50), cv2.FONT_HERSHEY_SIMPLEX)

                else:
                    pass

            except:
                pass

            cv2.imshow('thresh', thresh)
            cv2.imshow('filter', filtr)

            key = cv2.waitKey(25) & 0xFF
            if key == 27:
                break
        else:
            print('frame not found')


cap.release()
cv2.destropyAllWindows()
```