Touchless Fingerprint Capturing

Jatin Yuvraj Pendharkar
Masters of Engineering, Information Technology
SRH Hochschule Heidelberg
Heidelberg, Germany
Email Id: 11015240@stud.hochschule-heidelberg.de

Under the Guidance Of:
Prof. Dr.-Ing Christof Joneitz
Professor, SRH Hochschule Heidelberg

Abstract— **Image processing and Machine Learning are the latest technologies' development phase. Fingerprints have been utilized in vision-based applications to track or control safety-related activities. Fingerprint recognition is a natural technique of transmitting a signal to a machine for human identification. A section of the human-machine interface that outlines a fast algorithm for fingertip detection may be found here. This approach is insensitive to hand orientation, and in preprocessing, it removes only the hand portion of the complete image, making subsequent computations significantly faster. Touching a rigid sensing surface produces a fingerprint. This action results in finger skin deformation and necessitates the usage of external physical devices. To avoid the foregoing impacts, we employ cameras to capture a person's fingertips, which are then translated from BGR to HSV color space; this process is known as TOUCHLESS CAPTURING OF FINGERPRINT. We extract the Region of Interest (ROI) with the help of contours utilizing HSV color space, and then store the resulting unique Fingerprints in a data base that may be accessed later.**

Keywords-; HSV (Hue, Saturation and Value); BGR (Blue Green Red) ROI (Region of Interest); Fingertips Detection. Palm Seperation

I. INTRODUCTION

Due to their proven uniqueness and persistence properties, fingerprints are one of the most essential biometric characteristics. Fingerprint recognition systems are employed not only by police enforcement and forensic agencies around the world, but also in mobile devices and worldwide applications. Contact between the finger and the capturing device's surface is required in the great majority of finger-print capturing techniques. Low contrast caused by dirt or humidity on the recording device plate, or latent finger prints of prior users are just a few of the issues that these systems face (ghost fingerprints). Touchless fingerprint recognition systems have been re-searched for more than a decade to address the drawbacks of touch-based schemes. The finger does not make contact with the capturing device in touchless capturing systems.

Skin color segmentation is highly significant in the touchless fingerprint capture process. The color space of the fingertips acquired with a camera is normally RGB or BGR, which is transformed to HSV (Hue, Saturation, and Value) since it is more suitable for image segmentation.

After obtaining the HSV image, we must convert it to a binary picture, also known as the Threshold image. This can be accomplished using segmentation, in which the skin color is white and the background is black.

The contours surrounding the palm region should now be drawn. This can be accomplished using the binary of the Threshold picture that was previously received. We need to get rectangular boxes on the finger tips once the contours are produced, which is the key to this project.

II. IMPLEMENTATION

Implementation for these techniques involves the following steps.

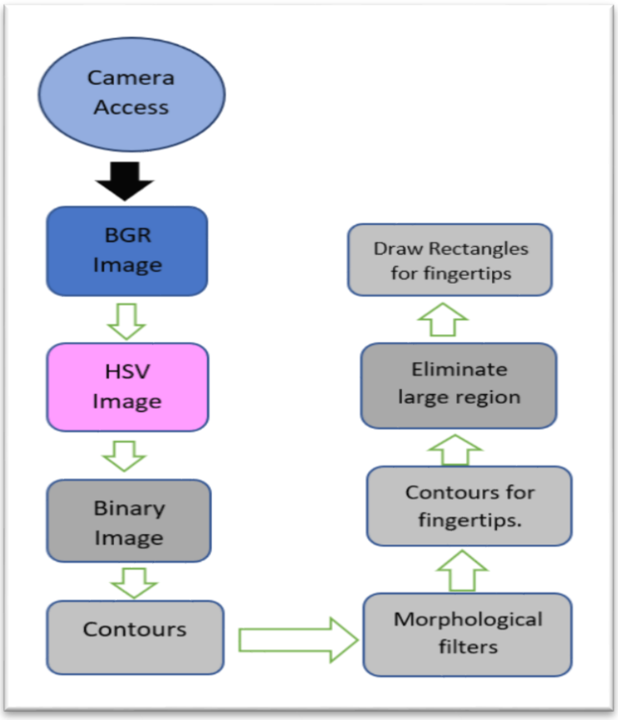

Figure1: Steps followed for this project

i. CAPTURE VIDEO USING CAMERA

The initial step in this procedure is to gain access to the camera in order to capture the video. All of the actions to be performed on a picture are applied only to a single frame in Image Processing, and we repeat this in an unending loop, resulting in a sequence of images that will be shown over and over again and will function as a film. This can be accomplished using the built-in Open.CV functionality (Video Capture). Because only this format is recognized by the electronic equipment, the photographs or video captured will often be in BGR (Blue, Green, and Red) or RGB (Red, Green, and color space formats, since only this format is recognized by the electronic device.

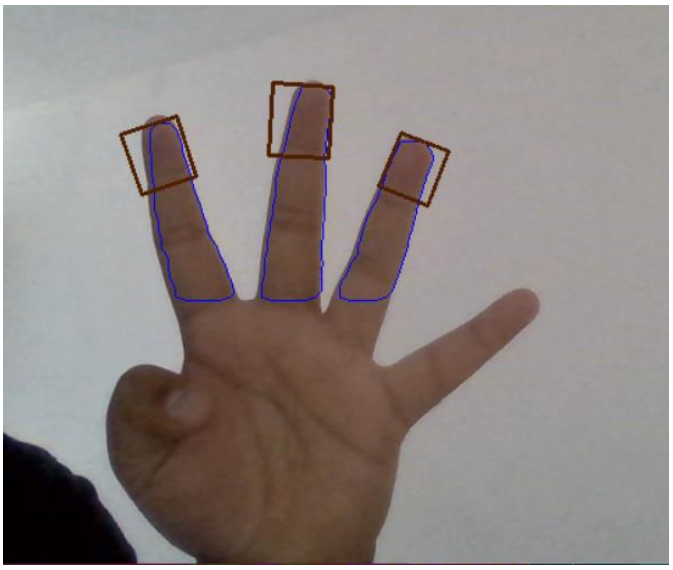
Figure 2: Accessing the Camera in Infinite Loop Syntax

```
VideoCapture cap(0);
      if(!cap.isOpened())
```

The above syntax is used to make the camera run in an infiniteloop. The ‘0’ indicates that we are using the inbuilt camera.

i. CONVERTING BGR COLOR SPACE TO HSV COLOR SPACE

The major goal of our project is to capture only the palm's fingertips and delete the rest of the palm. In order to do so, we must transform the obtained BGR image to an HSV image. We convert to HSV because the color of the image in BGR color space is difficult to distinguish in terms of luminance. As a result, we use the HSV color space to see the image's Intensity, Chroma, and Dominant Wavelength. This can be accomplished with the help of an OpenCV built-in function (cvtColor).

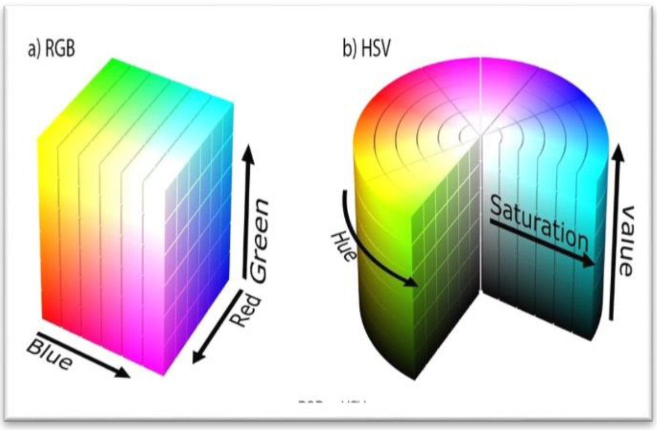

Figure 3: Illustration of RGB and HSV color space

Syntax:
```
cvtColor(Org_Frame, Img_hsv, CV_BGR2HSV);
```

The above syntax shows to which color space is the BGR image is being converted to and also the variable name were this value is stored.

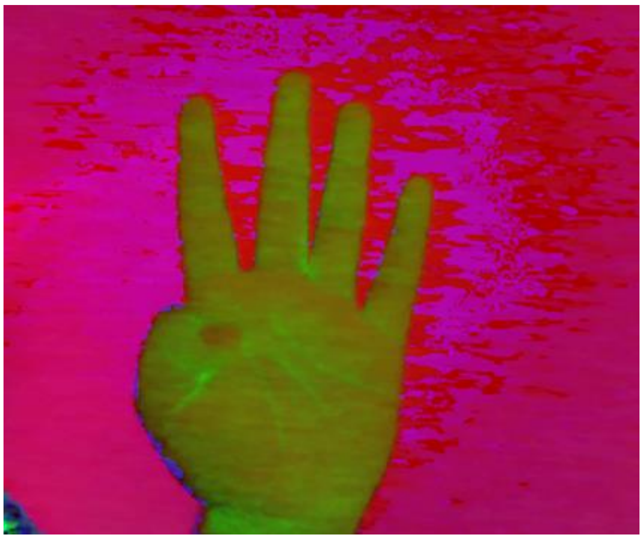
Figure 4: HSV Image

ii. SKIN SEGMENTATION

The primary key to skin recognition from a picture is skin tone segmentation. The skin color is the most important factor to consider. We'll use the OpenCV inbuilt function (in Range ()) to detect the fingertips and draw rectangle boxes over them because we're only interested in the Palm Skin Color, here, we'll pass the scalar values we got from the RGV to HSV conversion to a function that will draw a Threshold or Binary Image with the palm region in white and the rest of the background in black.
Once we have a binary or threshold image, we can utilize the built-in OpenCV functions Median and Gaussian blur to remove undesired pixels and generate a smooth image.

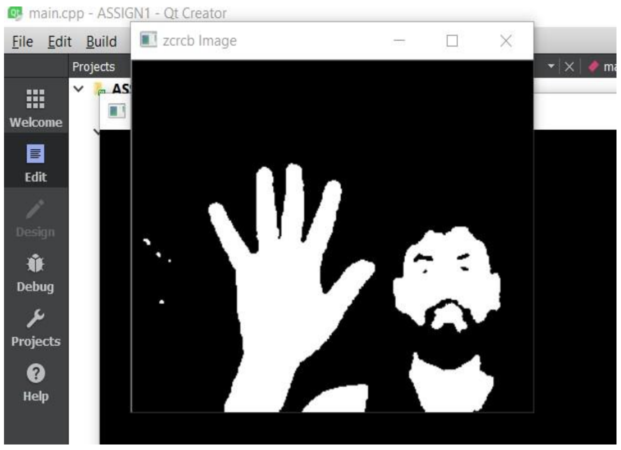

Figure 5: Threshold Image Palm Detection based on skin tone segmentation

Syntax:
```
inRange(Img_hsv, Scalar(Low_H, Low_S, Low
_V), Scalar(High_H, High_S, High_V), Hand
ed_Thres);
```

Here we will be varying the HSV value with help of Track Bar in according with the surrounding environment and this this track bar will be displayed while getting the output of the system.

iii. IMAGE ENHANCEMENT

Three types of Image enhancement techniques are used in this project:

1. *Image Smoothing (medianBlur OpenCV API)*

The median Filter is a non-linear digital Filtering technique, often used to remove noise from an image or signal. Such noise reduction is a typical preprocessing step to improve the results of later processing.

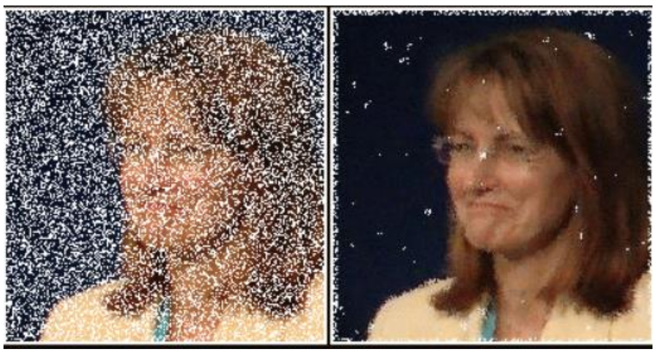
Figure 6. MedianBlur for Image Smoothing.

2. *Erosion (erode OpenCV API)*

The value of the output pixel is the *minimum* value of all pixels in the neighborhood. In a binary image, a pixel is set to `0` if any of the neighboring pixels have the value
Morphological erosion removes islands and small objects so that only substantive objects remain.

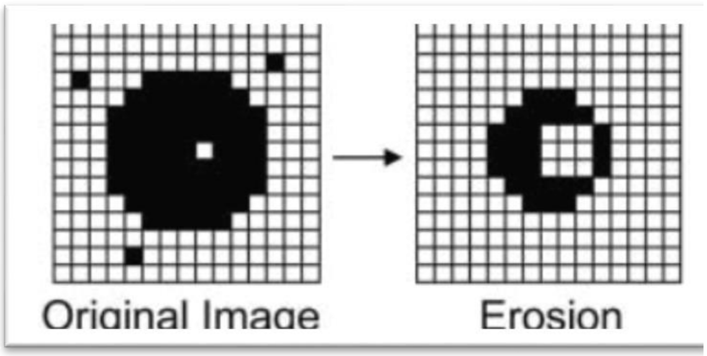

Figure 7. Erosion for Image Enhancement.

3. *Dilation (dilate OpenCV API)*

The value of the output pixel is the *maximum* value of all pixels in the neighborhood. In a binary image, a pixel is set to 1 if any of the neighboring pixels have the value 1.
Morphological dilation makes objects more visible and fills in small holes in objects.

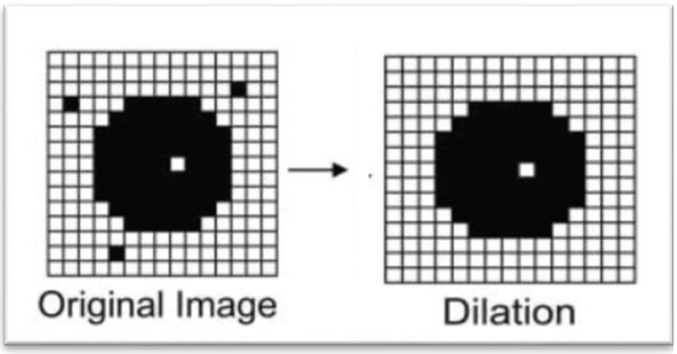

Figure 8. Dilation for Image Enhancement.

iv. FINDING THE CONTOURS

Contours are essentially lines that connect high-intensity points to generate a single large-area image. These contours are used to determine the greatest area or desired area of an object in an image, in this case our palm. We define two variables, contours and hierarchy, to find contours. Where each contour is an array of (x, y) coordinates representing the image's boundary points. Hierarchy is an optional output vector that contains topology information for the image. We now provide this contours variable to a draw contours variable, which will draw or combine all of the contours' coordinates.

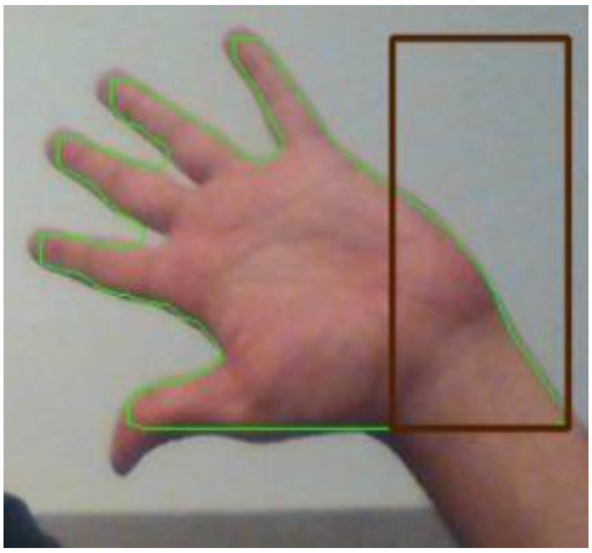
Figure 9: Contour Formation along with rectangle with disproportionate dimensions

v. DRAWING CONTOURS ONLY FOR THE FINGER (PALM SEPERATION)

The contours will initially be produced for the entire palm region, but the major purpose of this project is to discover capture fingertips, so we must filter the palm region, which is the largest region, and ensure that only contours for the fingertips are generated. This can be accomplished by drawing a line along the finger region to prevent contours from appearing below the finger region (palm).
In this Technique, a Dark line is made from the bottom of the image until the row it has more than 1 contour. If it separates the palm with fingers than it will get contours according to the number of fingers in the camera. This technique helps to separate the Palm from the fingers

Steps for Palm separation:

- ➢ Find the total number of contours in the input image.
- ➢ Make a row dark from the bottom of the hand.
- ➢ Store the latest image in current_image to calculate contours
- ➢ Find the number of contours in current_image. If the number of contours are more than last time then store it in nContours.
- ➢ At this stage we know how many rows needs to be removed from the input image to separate the fingers from palm.
- ➢ To do the above task on the output_image. output_image at the start of the function was same as the input_image.

vi. DRAWING THE RECTANGLE BOXES OVER THE FINGERTIPS.

After the fingertips have moulded their shapes, the next step in this procedure is to draw rectangles across the region of the fingertip. We can figure out the rectangle's dimensions and reduce it into smaller rectangles by drawing rectangles along the contour over the entire finger area at first. This can be done with the help of the built-in function (minAreaRect ( )), which returns the angle, height, width, and center point of the rectangle.

By using mathematical calculations, we can shrink the rectangle to the size of our fingertips. Consider the following scenarios:

Case 1: If Height is greater than width,

```
if
(minRect[i].size.height > minRect[i].size.width)
{
minRect[i].center     =     (rect_points[1]     +
rect_points[2])    /    2    +    (rect_points[0]    -
rect_points[1]) / 6;

       minRect[i].size.height  =  (float)scale  *
(minRect[i].size.height);
 }
```

Height Value and Center Value Manipulation.

Case 2: If Width is greater than height

```
else
{
minRect[i].center     =     (rect_points[2]     +
rect_points[3])    /    2    +    (rect_points[0]    -
rect_points[3]) / 6;
minRect[i].size.width     =     (float)scale     *
(minRect[i].size.width);
 }
```

Width Value and Center Value Manipulate.

We will be able to obtain the desired rectangle over the finger tips by using the above logic along with mathematical operations.

II. RESULTS

The final result obtained for this project can be seen in the figure 11, which shows the rectangular boxes over the fingerprints. So, this output can be obtained by following the steps shown in this paper.

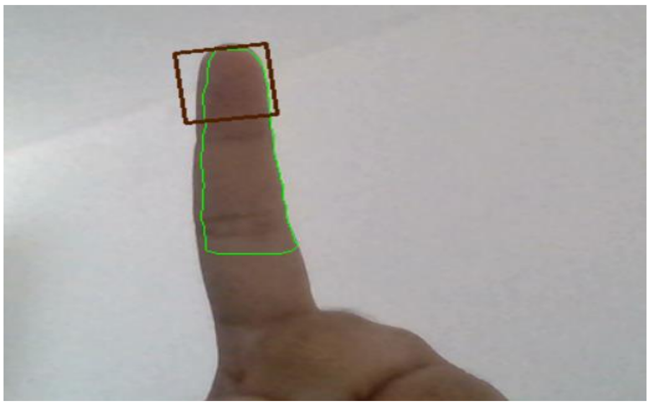
Figure 8: result for one finger in frame

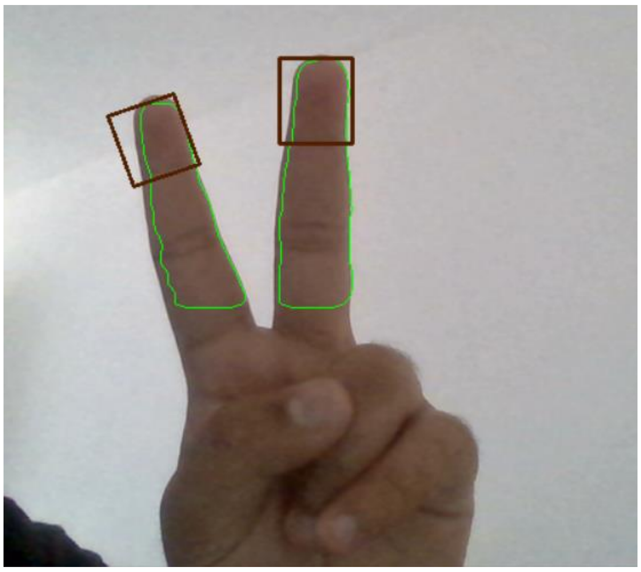
Figure 9: Result for two fingers in frame

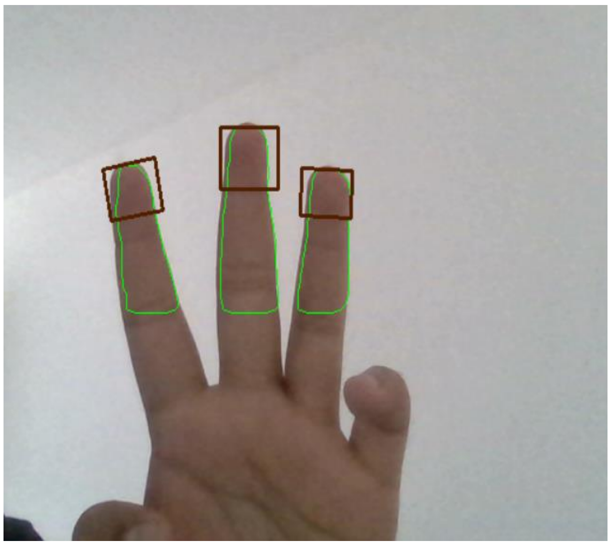
Figure 10: Result for three fingers in frame

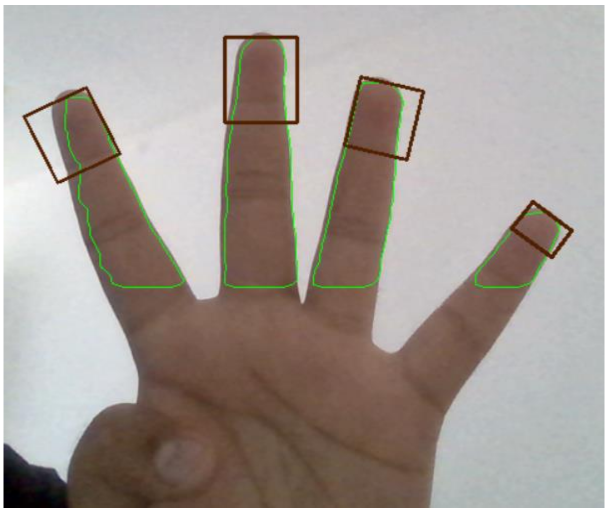
Figure 11: Final Result for four fingers in frame

III. APPLICATIONS

The touchless fingerprint technology has the following applications:

a. Reduces the external physical hardware.

b. After witnessing recent events around the world, this technology can be adopted in airports to substitute the hardware systems for scanning the fingers.
c. This technology can be used in forensics lab.
d. Prevents finger deformity
e. It uses the simple web cam to detect the fingerprints
f. Fingerprints are captured with real time moving human bodies

IV. CONCLUSION

Using the above method, we were able to acquire fingerprints without touching them. This system is primarily reliant on the camera and the CPU used. The better the processing, the greater the output. Various alternative approaches or image processing technologies can be used to increase precision and efficiency. During the early stages of this investigation, background noise and distortion in the output were discovered. This was reduced, though, by using the native OpenCV function medianBlur.
The bounding boxes can also show when the palm enters the frame, depending on the lighting and other factors. This problem was solved by employing a line segmentation procedure, in which the palm's area was removed and only the fingers were processed. Project is simple and easy for detecting the lesser finger. As we consider more finger the project becomescomplicated.

V. FUTURE SCOPE

The fingerprint can be discovered and utilized for authorization and security verification by creating a new code to enhance the region recorded and employing other techniques. By enhancing the application for identification, This can be used everywhere as it is important to implement touchless algorithm after COVID pandemic.

VI. REFERENCES

1. Volume V, Issue II, February 2016 IJLTEMAS ISSN 2278 – 2540 Touchless Fingerprint Recognition System by Prof. A. C. Suryawanshi1, Assistant Professor, Electronics Department, Umrer College of Engineering, Umerer, Nagpur, India.
2. S. B. Jemaa, M. Hammami, and H. Ben-Abdallah. Data-mining process: application for hand detection in contact free settings. Image Processing, 7(8):742–750, November 2013
3. Human Skin Detection Using RGB, HSV and YCbCr Color Models, S. Kolkur D. Kalbande, P. Shimpi C., Bapat J., Jatakia, Department of Computer Engineering, Thadomal Shahani Engineering College, Bandra, Mumbai, India and Department of Computer Engineering, Sardar Patel Institute of Technology, Andheri, Mumbai, India.
4. A. Farrukh, A. Ahmad, M. I. Khan, and N. Khan. Automated segmentation of skin-tone regions in video sequences. In *Students Conference, 2002. ISCON '02. Proceedings. IEEE*, volume 1, pages 122–128, Aug 2002.
5. https://www.sciencedirect.com/science/article/pii/S1 018363914000063 A.choudary , "Finger_stylus for Non Touch Enable Systems ",JKSU Engineering Sciences ,2015
6. https://www.uow.edu.au/~phung/docs/2005-TPAMI-skin-segmentation.pdf
7. https://docs.opencv.org/2.4/doc/tutorials/imgproc/shape descriptor/find_c ontours/find_contours.html.
8. https://www.learnopencv.com/color-spaces- in- opencv-cpp-python/
9. https://answers.opencv.org/question/44580/how-to- resize-a-contour/
10. https://docs.opencv.org/2.4/doc/tutorials/imgproc/shaped escriptors/bounding_rotated_ellipses/bounding_rotated_ ellipses.html