Contactless fingerprint capture using OpenCV

Tushant
Dept. of Information Technology
SRH Hochschule Heidelberg
Heidelberg, Germany
11015370@STUD.HOCHSCHULE-HEIDELBERG.DE

Abstract— **Image processing and machine learning are the current technologies' development eras. Fingerprints have been utilized in vision-based applications to track or control safety-based applications. Fingerprint recognition is a natural approach to send a signal to a machine for human identification. Here's a section of the human-machine interface that outlines a time-efficient approach for detecting fingertips. This approach is invariant to hand direction, and in preprocessing it removes only the hand section from the complete image, thus subsequent calculation would be considerably faster than processing the entire image.**

Keywords- Image Processing, Contactless, Binary image, Image enhancement, Rotated rectangles, Skin tone segmentation

I. INTRODUCTION

Computer vision is a branch of computer science that focuses on developing digital systems that can process, interpret, and comprehend visual input (pictures or videos) in the same manner as people can. The notion of computer vision is centered on training computers to analyze and analyze images at the pixel level. Technically, machines use sophisticated software algorithms to retrieve visual input, process it, and interpret the findings.

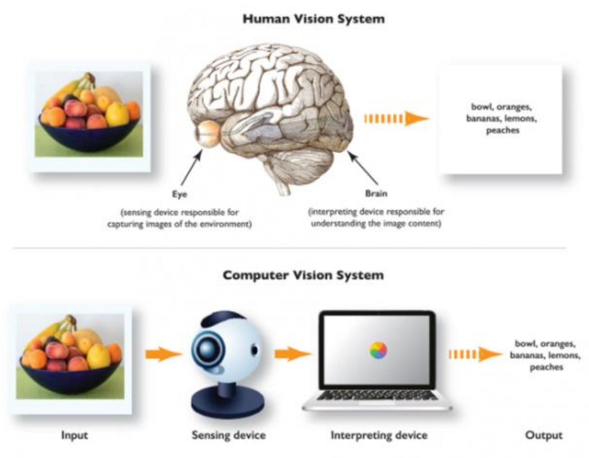
Human Vision System
bowl, oranges, bananas, lemons, peaches
Eye
(sensing device responsible for capturing images of the environment)
Brain
(interpreting device responsible for understanding the image content)
Computer Vision System
bowl, oranges, bananas, lemons, peaches
Input
Sensing device
Interpreting device
Output

Figure 1: Computer vision

Source: https://xd.adobe.com/ideas/principles/emerging-technology/what-is-computer-vision-how-does-it-work/

According to the paper, accuracy with contactless devices matching a single finger was somewhat poor, at 60 to 70%, but one contactless system matching several fingers performed within a fraction of a percent of the 99.5 percent accuracy that is the benchmark for most contact scanners. A smartphone app reached 95% accuracy, while other devices were close to 90% accuracy. All contactless systems demonstrated minimal false positive match rates, bolstering their potential.

In a laboratory environment, the agency collected fingerprints from 200 participants using all six touchless devices and two cutting-edge contact devices for comparison. In 2018, the NIST produced a paper on contactless fingerprints. Four of the touchless systems used smartphone applications, while the other two used stand-alone devices.

The findings of the 48-page paper, titled “Interoperability Assessment 2019: Contactless-to-Contact Fingerprint Capture,” are anonymized, although all were commercially accessible as of May 2019. Idemia and Veridium are two companies that provide contactless fingerprint technology, and the market for contactless biometrics is expected to reach $70 billion by 2030.

The technology is projected to be in more demand as a result of hygiene issues around shared surfaces such as touch fingerprint scanners, however the NIST claims the work began before the novel coronavirus arose.

In this project, I wrote code (In C++ using OpenCV library) to create a contactless fingerprint detection software for biometric applications that will use inbuilt video camera. Binary Image (HSV Image) and Ultimately Region of Interest (i.e., Fingerprint area of all fingers) will be shown in real time.

II. IMPLEMENTATION

A. Flow Chart

To explain the sequence of steps taken for implementing this project is described in flow chart below. Flow chart basically give the idea of code flow and a general overview of steps taken to achieve this goal.

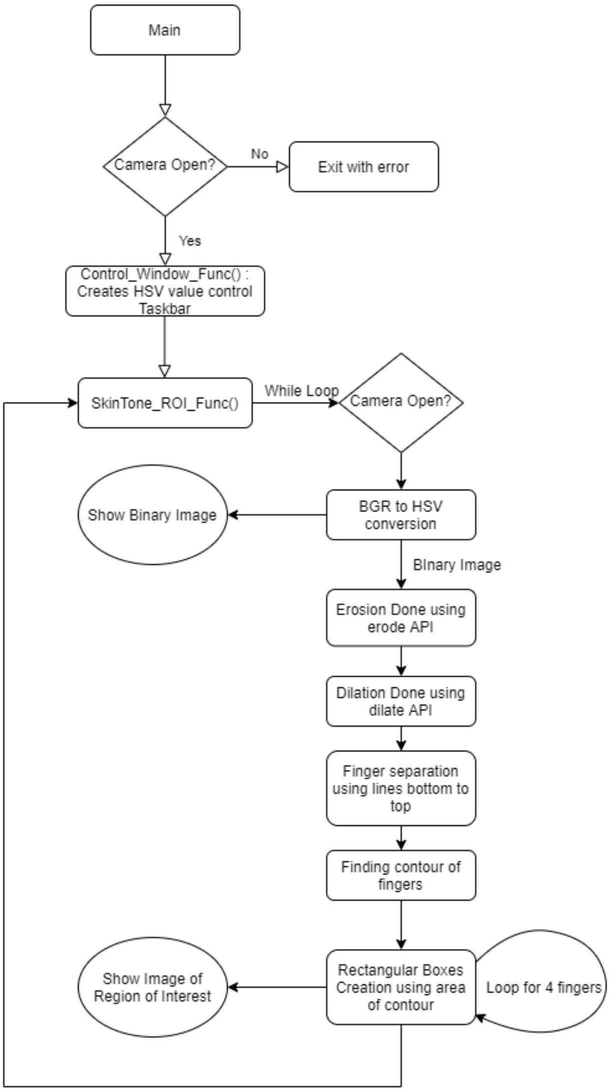
Main
Camera Open?
No
Exit with error
Yes
Control_Window_Func() : Creates HSV value control Taskbar
SkinTone_ROI_Func()
While Loop
Camera Open?
Show Binary Image
BGR to HSV conversion
BInary Image
Erosion Done using erode API
Dilation Done using dilate API
Finger separation using lines bottom to top
Finding contour of fingers
Show Image of Region of Interest
Rectangular Boxes Creation using area of contour
Loop for 4 fingers

Figure 2: Flow chart of code flow

B. *Skintone Segmentation*

Camera by default give output in BGR format. To differentiate the skin tone, we need HSV (Hue, Saturation and value). cvtcolor OpenCV API is used to do so.

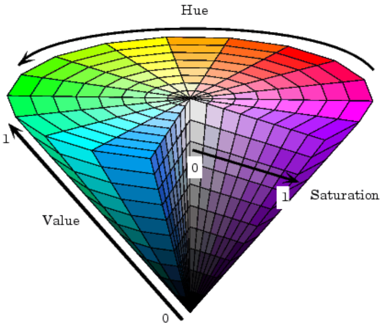

Figure 3: HSV Color Space

Source: http://www.ece.northwestern.edu/local-apps/matlabhelp/toolbox/images/color11.html

RGB colors are coded using three channels, it is difficult to segment the item depending on its color when utilizing the RGB format. Using the HSV color standard is a much better alternative since in HSV (hue, saturation, and lightness). The color space behaves similarly to how the human eye sees colors (saturation, value). As a result, the RGB scale is transformed to HSV. The hue channel in HSV represents the color, with hue being an angle ranging from 0 to 360 degrees. Saturation describes the grey component of the color space, ranging from unsaturated gray colors to highly saturated.

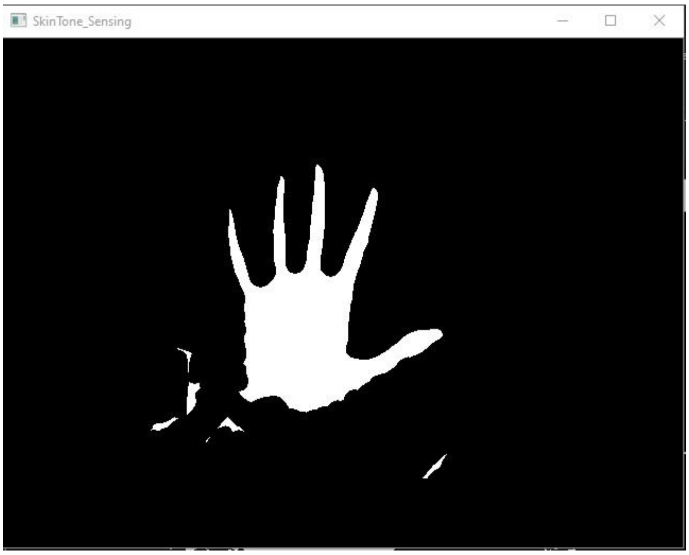

Figure 4 : HSV (Binary) Image

C. *Image Enhancement*

Three types of Enhancement techniques are used:

1. Image Smoothing:

The median filter (medianblur OpenCV API is used) is a non-linear digital filtering technique, often used to remove noise from an image or signal. Such noise reduction is a typical pre-processing step to improve the results of later processing (for example, edge detection on an image). Below applying smoothing algorithm image is shown in “Fig 5”.After applying image smoothing a blurred image frame as shown in the ”Fig. 6”.

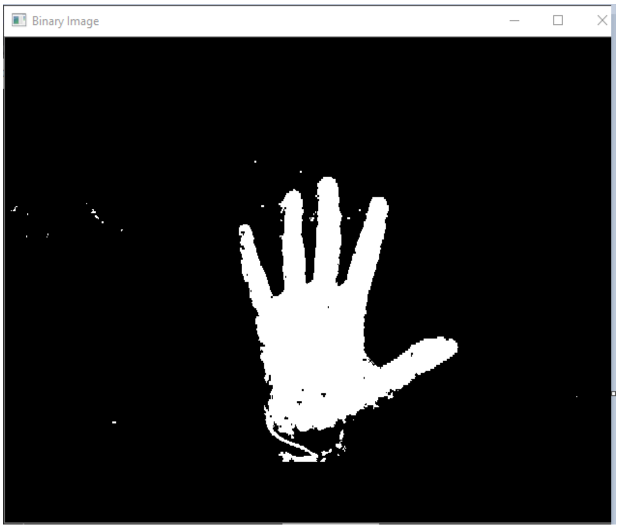

Figure 5: Before Image smoothing

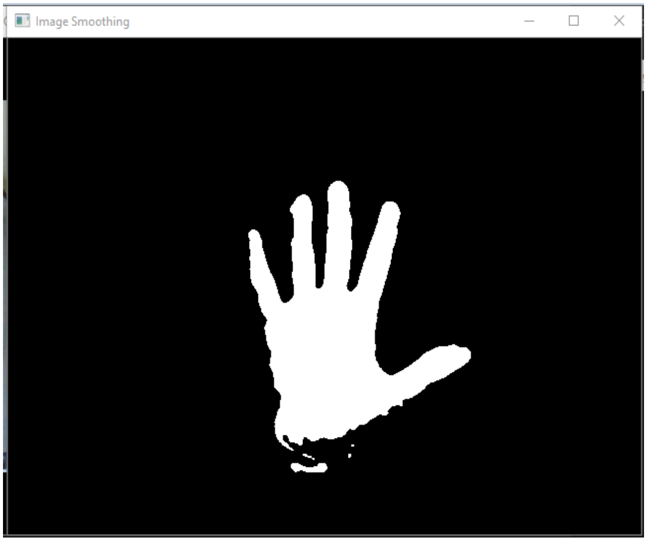

Figure 6: After Image Smoothing

2. Erosion (erode OpenCV API is used):

In erosion, we do an AND operation on the pixel values existing inside this element structure, and it replaces the anchor point pixel with 1 if all of the pixels within the structure are 1, else it erodes/changes the pixel to 0. Hence the object in white becomes smaller as shown in “Fig. 7”.

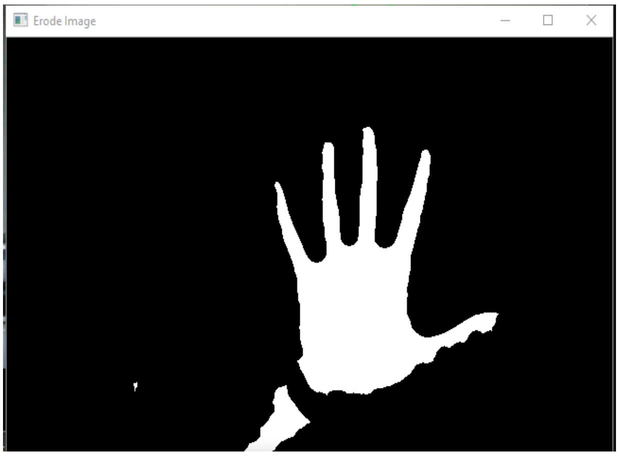

Figure 7: Image After Erosion

3. Dilation (dilate OpenCV API is used):

In dilatation, we execute an OR operation on the pixel values included inside this element structure, and if the result is 1, the anchor point pixel is replaced by 1. As a result, the item in white pixels expands with dilation, as illustrated in "Fig. 8."

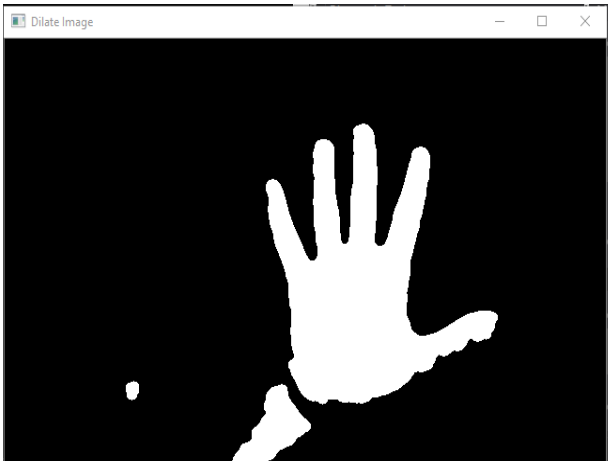

Figure 8: Image After Dilation

D. Segregation of Fingers

To obtain Segregation of fingers from palm, horizontal black lines must be drawn from bottom to top on the binary image.
For this purpose, connectedComponents() OpenCV API is used. If connected components reaches more than 2 this means fingers are segregated. In this way we erased the palm region and we are only left with finger. The thickness of line is considered as 30 it gives stationarity to the frame.
Segregated fingers are shown in “Fig. 9”

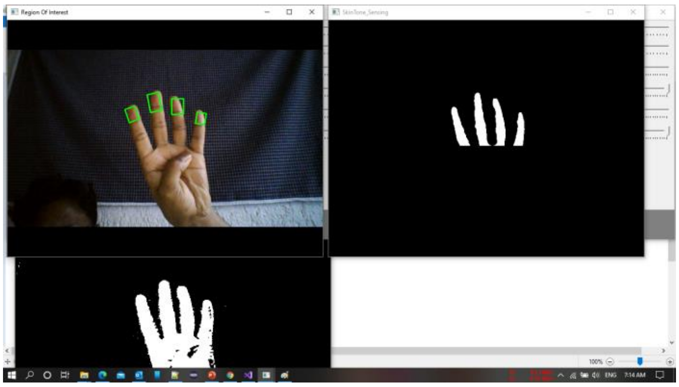

Figure 9: Segregated fingers (On right side corner)

E. Region Of Interest

Region of interest-Finger tips can be extracted by doing two methods in sequence:

a. Finding the Contour:

findContour() OpenCV API is used with RETR_EXTERNAL and CHAIN_APPROX_SIMPLE macros. For our approach, we utilize RETR EXTERNAL to retrieve just the segmented picture's extreme outside pixels, with which we may construct a boundary/contour around the segmented picture. In our approach, we utilize CHAIN APPROX SIMPLE as our approximation technique because it compresses horizontal, vertical, and diagonal segments and leaves just their end points. This approach also saves memory by deleting superfluous points and compresses the contour.

Figure 10 depicts contour lines surrounding the region following skin tone segmentation.

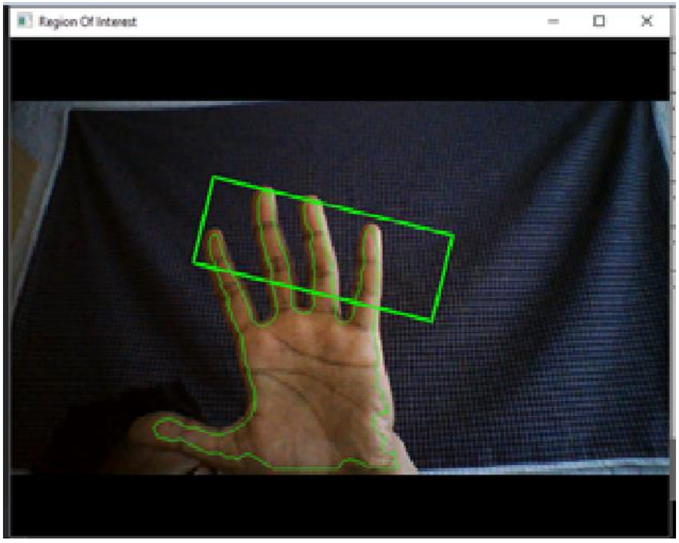

Figure 10: Contour image

b. Using rotated rectangle:

minArearect() OpenCV API is used to get the center, Height, width and angle in RotatedRect type of variable.

To find the height, width and center of fingertip rectangle(Upper most rectangle).Width is already know. Height will be the 1/3 of the height the full finger rectangle.

To find the center of the fingertip rectangle we have a formula

Coordinate of center(Fingertip rectangle)= $((x_1,y_1)+(x_2,y_2))/2 +(((x_0,y_0)-(x_1,y_1))/6$

Final output as region of interest is Show in "Fig. 11" below.

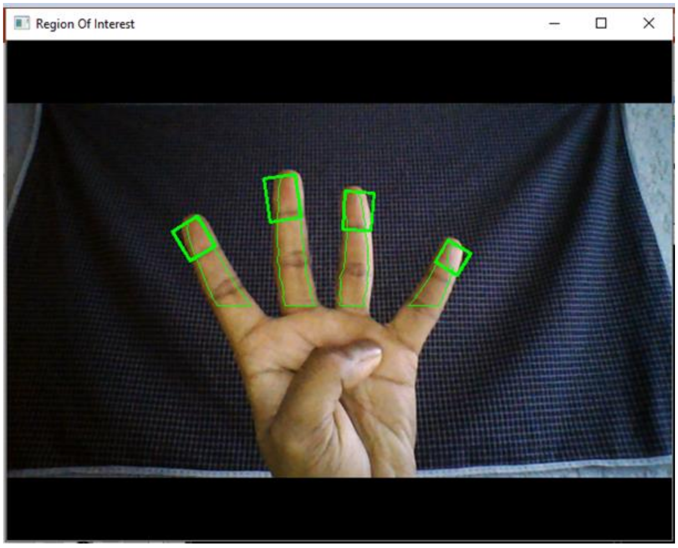

Figure 11: Final ROI : Fingertip in rectangle

III. APPLICATION

- Fingerprints are collected with real-time moving human bodies. Implementation is low-cost.
- OpenCV is simple to create and install.
- Made use of basic C++ programming and a user interface.
- For safety-related problems, it can be employed in biometric and forensic applications.
- Doesn't allow to spread infectious virus due to contactless access.
- Computer and cell phone application security.
- It detects the fingertip using a basic web cam.

IV. RESULT AND CONCLUSION

The camera is placed in few centimeters from the input surface, in a specific angle from the input. Making the model able to detect the skin regions on a video. The main feature of the skin tone segmentation is to choose the HSV color space. And final result that is Region of interest (FINGERTIP

IN RECTANGLE) is visible in below “FIG.12”

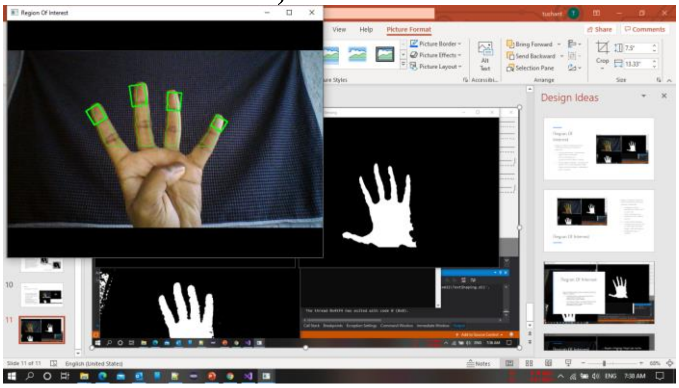

Figure 12: FINAL OUTPUT DETECTING THE FINGERTIPS

REFERENCES

[1] OpenCV,”Contours : Getting Started” [Online], Available: https://docs.opencv.org/master/d5/daa/tutorial js contours begin.html Nov 2020.

[2] Tutorial: Skin Detection Example using Python and OpenCV (pyimagesearch.com)

[3] Files - SRH Cloud

[4] http://opencvpython.blogspot.com/2013/05/threshol ding.html

[5] Gaussian Blur - OpenCV Tutorial C++ (opencv-srf.com)

[6] https://docs.opencv.org/3.2.0/d4/d73/tutorial_py_co ntours_begin.html

[7] https://www.docs.opencv.org/trunk/d4/d73/tutorial_ py_contours_begin.html

[8] OpenCV,”HSV skin colour extraction”OpenCV Forum’ [Online], Available: https://answers.opencv.org/question/128190/detect-hsv-color for-skin-color-and-extract-the-color-in-that-region/ Feb 2017

[9] OpenCV,”Operations on arrays ’ [Online], Available: http://https://docs.opencv.org/3.4/d2/de8/group core array.html.