Some methods for counting the number of raised fingers from an image of a person using OpenCV Python

Prasangsha Ganguly

December 10, 2018

Problem Statement

Given an image of a person with face and a hand gesture, count the number of raised fingers in the image. For example, in the following image, the output will be 2. The problem consists of two

Figure 1: Input image

subproblems:

1. Identification of the hand region (foreground) in the image and absconding all other information of the image as background
2. Counting number of raised fingers present in it

Brief background and methodology

Several works have been done in hand gesture and posture recognition. Sgouropoulos *et al.* [1] proposed a methodology for the same using Hidden Markov Model (HMM) and a neural network. Several techniques are discussed in [3]. The major two categories of the solution methodologies are

1. Geometry based techniques, where the shape of the hand determines the number of raised fingers.
2. Machine learning based techniques, where a model is trained according to a dataset of images and later used to predict the number of fingers.

Here, I used the geometry based technique using **OpenCV Python**. The overall methodology is described below in brief.

Identification of foreground pixels belonging to hand region

I used manual thresholding to identify the pixels belonging to skin or not using the method proposed by Kolkur *et al.* [2]. In the resulting image I have face, hand and some regions of background having near skin color. I identified the face area using Haar Cascade Classifier and made it as background. Now, identify different connected components in the image, and the largest component having the skin color is assumed to be hand.

Length of convexity defects based finger identification

In the image containing only hand, find the hand contour and the convex hull of the hand as in [4]. As, the objective is to identify only the raised fingers, only the vertical convexity defects are considered. If the defect is more than a threshold, it is a finger.

Angle of convexity defects based finger detection

Identify the vertical convexity defects. For each convexity defect, there are two vectors (start to far and end to far). The angle between two vectors ($a1$ and $a2$) is calculated as, $\theta = \arccos \frac{a1.a2}{|a1||a2|}$. If the angle is less than a certain threshold and both start and end points are above far, then it is a defect due to finger.

Pixel intensity pattern based finger detection

In the binary image containing only hand as foreground (1) and remaining part as background (0), identify the change in pattern of pixel intensities. For each row of the image, the pattern 011 (left pixel is background, central and right pixels are foreground) denote the left boundary and 110 denote the right boundary of the fingers or hand. For the palm region, for each row we have one 011 and one 110. For each finger we have the same. If the patterns are found more than once, then more than one fingers are raised, count the number of fingers accordingly. This method can also be applied to skeleton of the hand image with a little bit modification: As skeleton contains a single pixel, for each row of the image count 010 pattern.

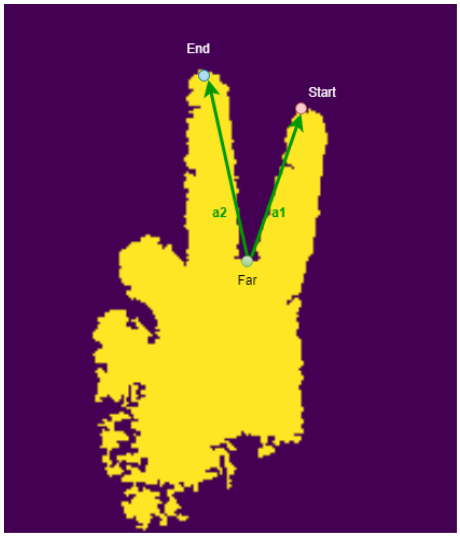

Figure 2: Only hand threshold, and the convexity defect vectors $a1$ and $a2$

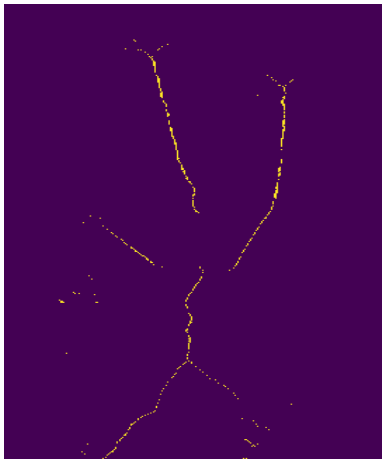

Figure 3: The skeleton image

Different gray intensities and corresponding frequencies

This is a helper function which identifies all the different gray intensities present in an image with corresponding frequencies of that intensity. From this, top k number of intensities according to frequency has been extracted which can be useful for some image processing applications.

References

[1] Kyriakos Sgouropoulos, Ekaterini Stergiopoulou, Nikos Papamarkos *A Dynamic Gesture and Posture Recognition System*. JIRS 2014.

[2] S. Kolkur, D. Kalbande, P. Shimpi, C. Bapat, and J. Jatakia *Human Skin Detection Using RGB, HSV and YCbCr Color Models*. ICCASP, 2017.

[3] `https://www.intorobotics.com/`

[4] OpenCV Contours
`https://docs.opencv.org/3.3.1/d4/d73/tutorialpycontoursbegin.html`