Palm print feature extraction.

Sagaf Youssouf

Université Paris-Est Créteil (UPEC)
International Master of Biometrics and Intelligent Vision
https://www.international-master-biometrics-intelligent-vision.org/

Abstract. Palm print carries a set of features that may be used to describe a person's hand. Measures of palm characteristics differ from one hand to the other one and from one person to another. For this reason, we have studied and developed an image processing feature extraction algorithm that takes as input an image of the hand. The method used is mainly based on the convex hull algorithm. We added some functionalities by comparing the acquired signature from the extracted features with one sample that have been previously captured and stored. The designed application obtained is promising and can be improved.

Keywords: Feature extraction · Palm Print · Measurements

1 Introduction

Feature extraction is a crucial step in any biometric system pipeline. In the case of palm print, one seeks to extract reliable and useful characteristics that should help to identify an individual. The objective of our study is to develop an algorithm to extract the necessary information contained in the palm print to characterize the hand. Raw data acquired bypassing the palm of the hand in front of the laptop web-camera. The method adopted consists of a series of calculations and measurements based on the characteristics obtained by the algorithm in order to create a signature that could be used as a query in an biometric authentication system. This report aims to show you the strategy adopted in order to achieve a functional palmprint feature extraction system. The methods and materials used are outlined in the next section. This is followed by a section highlighting the results obtained.

2 Methods and materials

Main components of the system : The key components that made the feature extraction algorithm are illustrated in figure 1. A simple overview of the system is presented. The algorithm is coded in Python language using OpenCV library. We used Scipy, the scientific computing tool for scientific calculus. The system is made up of four mains steps :

- **Image acquisition** : the image to process is acquired with a laptop webcamera. Live capture stream is launched for a sequence of 600 frames and one image is selected. By default, the last image is chosen. We suppose that the user had the time to calibrate his hand in the field of view to get a suitable pose.
- **Preprocessing** : a series of pre-processing methods are applied to the selected image in order to make it easily exploitable for the algorithm.
- **Hand segmentation** : the main idea behind this step is to take out the hand region by eliminating unwanted regions.
- **Palmprint feature extraction** : this is the final key component of the system. Features are extracted from the segmented image. At the end, a set of calculations and measurements are applied to the characteristics extracted to form a signature vector.

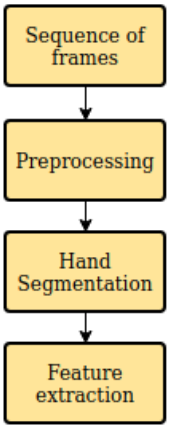

Fig. 1: Mais components of the system

Algorithm : The flow shart in figure 2, shows how the process is carried out to reach the final solution by connecting each step. The initial specifications consisted of feature extraction only. We decided to add other functionalities such as the score calculation part, to make it as a complete biometric matching system. Once all the intermediate steps have been completed, the system can determine the score based on the computed measurements. This allows us not only to characterize the hand but also to deliver access authorization or not.

Selection of the hand image : as we mentioned earlier, a laptop webcam is used for video capture. The frames acquired are saved as images in jpg format and the last one is selected and cropped considering only the hand region. This

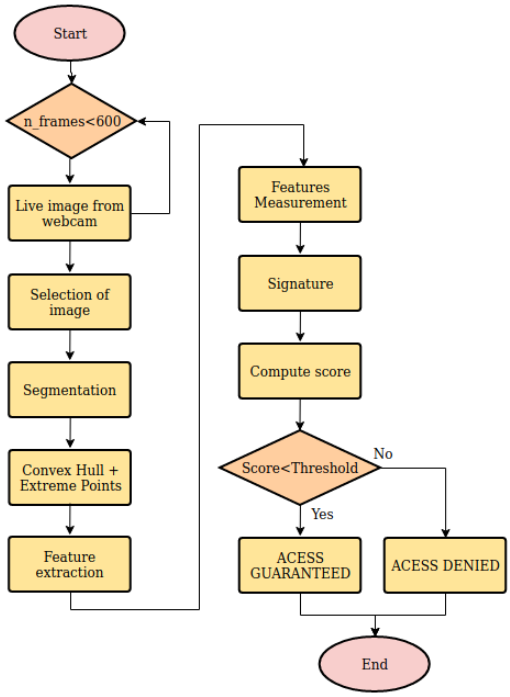

Fig. 2: System flow chart

image is used as the input of the system. The user should make sure that the last image is very good while calibrating.

Preprocessing :

Resizing of image : This step is necessary to make sure we have a reference size of our image. For this reason, the palm print image is resized using a given scale percentage. The size of the frame provided by the webcam is 480x640. Size have been increased and the size of the final hand region image is 778x700.

Segmentation of the hand : to create a binary image with where white will be skin colores (hand) and the rest is black, thresholding operation is performed. This is followed by a series of operations to filter the background noise dilation and erosion increase the skin color area. Figure 3 illustrates the obtained result.

Calcul of midpoint between two coordinates A function is created to calculate the minimum enclosing rectangle of the incoming contour, calculate the midpoint of each side of the enclosing rectangle, and the distance between the midpoints of the long and wide sides, and display the calculated value in the image

Fig. 3: Thresholding and Segmenting Hand region from

Feature extraction : the first step here is to find the contours of the image. This is done using `cv2.findContours()` OpenCV function. Each contour is looped over and if it is not extending to a large enough size, the contour is ignored. And then, one can compute Convex Hull using the approprite functions `cv2.convexHull` first and `cv2.convexDefects()`. Any deviation of the object from this hull is considered as convexity defect. The return value defects of `convexityDefects()` is an array of N*1*4, where N represents the number of convex defects.

- defects[N][0][0]: the starting point of the Nth convex defect (startPoint);
- defects[N][0][1]: the end point of the Nth convex defect (endPoint);
- defects[N][0][2]: the farthest point of the Nth convex defect (farPoint);
- defects[N][0][3]: the depth of the Nth convex defect (depth);

The starting point, the end point, and the farthest point of the convex defect representing the root of the finger are respectively taken out and stored in the corresponding array. The X-axis coordinates of the farthest point are sorted from left to right. The start point, end point, and farthest point are sorted in this order. In the above cycle, use the starting point of the current convex defect, the farthest point, the middle point, the end point, the middle point, and the farthest point of the next convex defect to form a contour. The area surrounded by this contour is drawn in white in the ROI with black background. Next, perform the AND operation between the ROI and the binarized image, and the image of the finger contained in the current convex defect can be taken out separately.The taken out image containing the finger part may have a little stickiness, so the image is corroded to separate the sticky part. The image ROI after separation is shown in the figure 4.

Fig. 4: ROI finger image

Measure of characterization : to achieve measurements to characterize the hand, we choose to compute different features like centroid and perimeter. For this reason, image moments of the largest contours must be calculated. This will help us to compute the center of mass. The centroid is given by the following formulas :

$$C_x = \frac{M_{10}}{M_{00}}$$

and

$$C_y = \frac{M_{01}}{M_{00}}$$

In python, we use the following code :

```
cx = int(M['m10']/M['m00'])
cy = int(M['m01']/M['m00'])
```

Definition of signature vector : To characterize the palm print, we decided to use the Euclidian distance between midpoints for all the finger bouding box which is supped to be the width and the length of the finger. This extend with the coordinates of the center mass. The signature vector is then made up fo 12 values.

3 Results

The image 5, gives a final view of the result reached. The plot of the different characteristics is given. In purple, the distance are width and length of the fingers and in red center mass which is supposed to be the center of the hand.

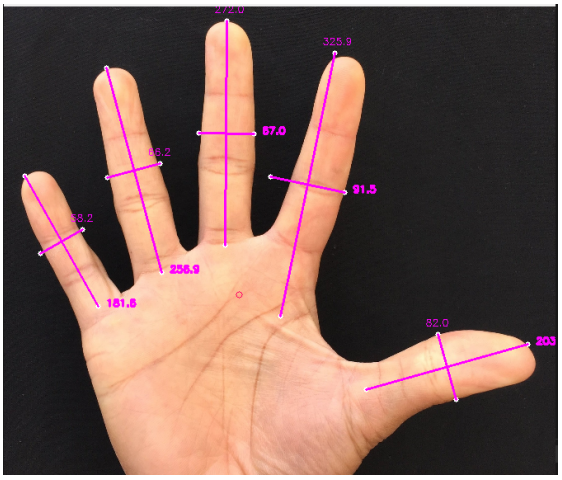

Fig. 5: Plot of extracted features. I

Signature The values of our feature vector constituting the signature is given in the figure 6. The values are stored in a python list. The list is made up of the euclidian distance of the midpoint of the hand bounding box which is supposed to be the width and the length of each fingers. The two last values are the coordinates of the center mass computed using the centroid. This center mass is the center of the palm which can be an additional information.

Matching of signatures This last step refers to the process of the degree of match using a concept of match score. This process consists of comparing two signatures: the computed signature from the palm print features calles query and one collected and stored as a list in the code. This will be used for identification. We compute the cosine similarity from the two lists and we set a threshold of 0.009. If the score is smaller than the threshold, we consider that the signature is validated and can get access otherwise the access is denied.

4 Conclusion

In this report, we presented the implementation of a simple biometric system for feature extraction using OpenCV. We took the opportunity to add other func-

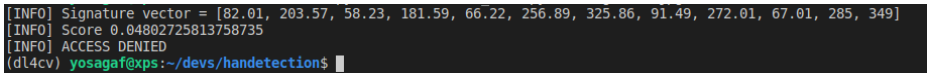

Fig. 6: Created signature which will be used as a query

tionalities such as matching of two signatures. Many points need to be evaluated and improved, especially at the level of signature matching. We must also think about using biometric metrics to evaluate the system objectively. We could also define a criterion for the selection of the image to be processed for video sequence module. This lab enabled me to review the basics of image processing and to highlight my skills in terms of feature extraction.

References

1. Yeo, Zi Xian Justin. “Hand Recognition and Gesture Control Using a Laptop Webcamera.”
2. Francis, Jobin and K. AnoopB. “Significance of Hand Gesture Recognition Systems in Vehicular Automation-A Survey.” International Journal of Computer Applications 99 (2014): 50-55.
3. S. Mitra and T. Acharya, ”Gesture Recognition: A Survey,” in IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 37, no. 3, pp. 311-324, May 2007, doi: 10.1109/TSMCC.2007.893280.
4. Hand Gesture Recognition and Servo Control tutorial, https://gogul.dev/
5. Medium article, https://medium.com/analytics-vidhya/hand-detection-and-finger-counting-using-opencv-python-5b594704eb08
6. Measuring size of objects in an image with OpenCV, https://www.pyimagesearch.com/2016/03/28/measuring-size-of-objects-in-an-image-with-opencv/