Python Fingerprint Recognition

Introduction

Biometrics is a technology that can identify a person based on his physical characteristics. Identification and fingerprint recognition is a biometric method that is widely used in different types of applications because of its accuracy and reliability. The main goal of this project is to develop a system that will be able to recognize whether 2 prints come from the same person or not. For this purpose, the images are first collected from a public data set. Then digital imaging techniques are applied to the same images in order to improve their quality. Once the image is reprocessed, the so-called image is searched. critical points that are later compared according to their Hamming distance.



(Example image of a placard in different positions)

Data set

The dataset I used is called FVC2002 [1] and is published by the University of Bologna. The site has 4 different datasets, but for this project I used the DB1 datasets.

Minarization

Image bias allows us to clear the image from unnecessary noise and helps us to contrast between the "wrinkled" surface of the print and the other lines.

Otsu's doorstep

Otsu's threshold will automatically select the best generic image threshold to get a good contrast between foreground and background information. This is because the image contains a bimodal distribution of pixel values. For that picture, we can roughly take the value in the middle of the peaks of the histogram as a threshold.

(For non-modal images, the binary will not be accurate.) Otsu allows us to avoid using a fixed blank value, thereby making the system more general for any recording device. The result of the threshold can be seen below.



Image Skeletonization

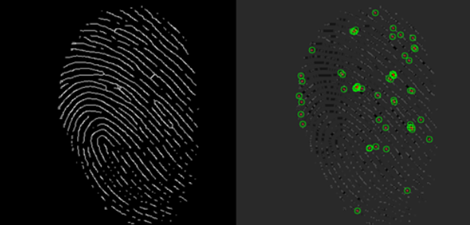
In order to improve the process of searching for critical points on the printout, it is advisable to sketch the image itself. This creates more unique and stronger critical points. The skeletonization is based on Zhang-Suen's algorithm. [2]



The result before and after skeletonization.

Requesting minutiae points

Once you get the skeleton image, the next step would be to find crossing points on the print reefs, which are then called minutiae points. This can be done with the help of a critical point detector, which requires a great deal of local contrast change. Such is the Harris Corner detector [3]. Because the Harris Corner detector is capable of detecting strong angles and edges, this is ideal for the fingerprint problem, where the most important minimums are short edges and bifurcations, where the edges are collected. The critical points found can be seen in the figure.



Definition of a formal descriptor

Once the list of critical points is obtained, a formal descriptor of the local region around that point should be created so that it can be uniquely identified among the other critical points. Since the orientation of the printout may vary (not a fixed position), we need a descriptor that is robust to such small differences. One of the most commonly used descriptors for this purpose is the ORB descriptor.

With this method we can obtain a descriptor for each critical point and form a matrix of such descriptors that identifies the printout.

Comparison of prints

Once the two-fingerprint descriptor matrices are obtained, an algorithm for their comparison is needed. The simplest way is to search with so called coarse force along the Hamming distance between descriptors at 2 different points. This way we get a rating that indicates how similar those 2 prints are. Setting a threshold can determine whether or not the prints are the same.

Another way is to compare the original fingerprint with all the fingerprints found in our fingerprint database and to select the one that has the best score of all - in order to identify the input fingerprint.

Conclusion

This project developed a fingerprint recognition system based on the method of finding critical points. These points are then used to find formal descriptors of the region around them and thereby form a matrix that identifies the impression itself. I tested the system on the FCV2002 DB1 dataset to determine if it successfully recognized the prints.

Reference:

[1] http://bias.csr.unibo.it/fvc2002/databases.asp

[2] T.Y. Zhang and C.Y. Yuen. “A Fast-Parallel Algorithm for Thinning Digital Patterns”. Communications of the ACM. Vol. 27 1984

[3] Harris, Chris, and Mike Stephens. "A combined corner and edge detector." Alvey vision conference. Vol. 15. 1988