Machine learning for finger recognition

Morgan Asnar

29/5/2020

Contents

1 Problem definition **1**

2 Image processing **2**
2.1 Identifying finger 2
2.2 Enhancing pictures 2

3 Model description **3**

4 Results and interpretation **4**

5 Conclusion **5**

6 Annexes **5**
6.1 Regarding the code 5

1 Problem definition

The following report address this particular problem; being able, from a picture of a finger, to know who it belongs.
The database used here is composed of 10 to 30 pictures of the index finger from the left hand of 5 persons, on a white sheet of paper. We divide out set in two, as one part will be used to train our model, and the other one will be used to test it. To this end, we keep three pictures per person as a training dataset, and the others pictures will be used as validation dataset.
The major issue with this is the low resolution of the pictures, making this dataset hard to work with, alongside with a low inter-class variance, as such as a low intra-class variance, which makes hard to extract characteristics that are not context-related.
All the code described in the following report can be accessed on github, at the following address:
https://github.com/MorganAsnar/finger-recognition/tree/master

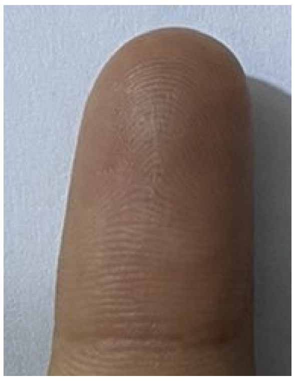

Figure 1: Sample of the pictures from the database

2 Image processing

In order to extract informations from the pictures, we need to enhance our pictures in order to make them easier to work with. A sample of the database can be seen on figure 1.

2.1 Identifying finger

The first thing we want to do is isolate the finger from the background. To this end, we use a HSV decomposition. We then work on the hue channel, as this is the most efficient one to extract colors. We chose to apply an interval from 50 to 100. This was found empirically to give the best results. We create a mask from all pixels in the picture that are in this interval. We then apply this mask to our original picture, which results in a picture where only the finger is visible. We then convert it to grayscale, as we're not interested in colors, but in shapes.

2.2 Enhancing pictures

We would now like to accentuate some characteristics of the pictures we would like to work on. We want to focus on what makes the object unique. Here, we would like to focus on fingerprints. To this end, we shall at first sharpen the edges. We will use for this the following kernel, that we will convolve with the picture.

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Considering the small size of the pictures, using a larger filter would result in information loss, as the edges would cross over each others. It would also result in larger sides effect. The goal of this step is to make ridges stand out better. However, this is not yet sufficient.
In order to make ridges the most prominent thing in our picture, we use height

Gabor filter, with height different directions, and sum them [1]. The parameters of these filters can be found bellow.

- Kernel size : 5*5
- $\sigma = 5$
- $\theta = \{0; 22.5; 45; 67.5; 90; 112.5; 135; 157.5\}$
- $\lambda = 5$
- $\gamma = 1$
- $\psi = 0$

The picture we obtain after this pretreatment can be seen in figure 2. We also resize it, in order to make comparison between pictures more relevant.

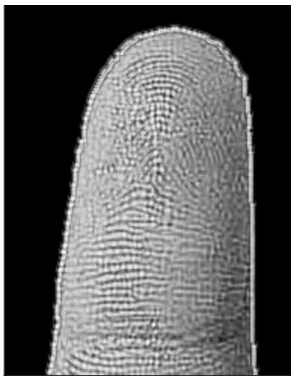

Figure 2: Picture obtained after pre-processing

3 Model description

To extract features from our pictures, we will be using a Scale Invariant Feature Transform algorithm. In order to make our model reliable, we want to try to find some common keypoints in our reference pictures, and use these to sort out query pictures.
We first extract the SIFT descriptors of our reference pictures. For our 3 pictures per person, with 5 persons, this gets us around 48,000 descriptors, which are vectors of dimension 128. We want to find the n centroids of these points, which we will use as a visual vocabulary, for a bag of visual words [2]. This parameter need to be tuned according to the used database. For the one used here, it has been found empirically to give the best results using $n = 750$.

[1]Ross, Arun & Jain, Arjun & Reisman, J.. (2002). A Hybrid Fingerprint Matcher. Pattern Recognition. 36. 795 - 798 vol.3. 10.1109/ICPR.2002.1048138.

[2]Csurka, Gabriela & Dance, Christopher & Fan, Lixin & Willamowski, Jutta & Bray, Cédric. (2004). Visual categorization with bags of keypoints. Work Stat Learn Comput Vision, ECCV. Vol. 1.

Once these centroids have been found, we will now create histograms of the number of centroids for every reference picture. More specifically, for one picture, we considere all of its descriptors, compute the nearest centroid using Euclidean distance, and increment the value for this centroid. Doing so, we get to a space of dimension n, which better fits our problem.
For every picture, we will preprocess them as described in 2. We then compute the SIFT of the picture, and its histogram in the n-dimensional space. We then find the two neirest histograms in our reference database, using Euclidean distance. Let $d0$ and $d1$ be the respective distances from the query histogram to the neirest and the second neirest, we considere the ratio $r = \frac{d0}{d1}$. If this ratio is under a threshold $r0$, we considere that it means the two closest reference histogram are too differents to make a Conclusion. This picture won't be classified. If $r > r0$, we considere that the picture is from the same class as the neirest histogram. Here, it was chosen to apply $r0 = 0.75$, as this would provide a good compromise between discard rate and classification performances.

4 Results and interpretation

With described parameters, the result is a correct classification rate of 90%. However, 52.2% of the pictures were discarded, while only 50% of the pictures that should have not been classified were discarded. On figure 3, we can see the confusion matrix. The line number 4 is the 'trap' class, some random pictures that don't belong to the original database. Appart from this line, every other picture that was classified was correctly classified. A query takes an average time of 1.6 second. It was tried to discard descriptors during histogram creation,

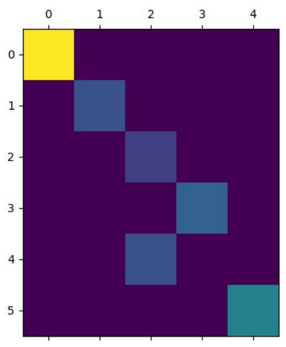

Figure 3: Confusion matrix

using a similar approach as the one described at 3, using the ratio between the distance from the descriptor and the two neirest visual words. If this ratio was too close to 1, the descriptor would be discarded. However, doing so highlighted that most of the time, this ratio is very close to 1, (mean value of 0.95) which is a problem, and makes this approach unusable.

5 Conclusion

This last result is a great concern of this study. The fact that this ratio is to close to 1 would tend to show that the centroids are not significant enough, and that we would need to use more of them. However, using a higher number of centroids leads to lesser results, while maintaining a similar average ratio. However, the results using solely database pictures are usable, as there is no mistake in classified pictures. This, However, can not be used as itself in a security application.

6 Annexes

6.1 Regarding the code

This code was made in python 3, and makes great use of openCV. However, as SIFT was a patented algorithm the SIFT algorithm is not implemented yet in the default versions of openCV. To use this algorithm, we use the opencv_contrib package, available on github. If the creation of the SIFT objects is not working, you may use the following, with package manager pip.

```
pip uninstall opencv-python
pip install opencv-contrib-python
```