WARSAW UNIVERSITY OF TECHNOLOGY
Faculty of Mathematics
and Information Science

IASR Final Report

Gender Recognition by Fingerprint

Omer Berk Gencer
Mirac Can YUKSEL
Anil Pinar OZDEMIR

26.01.2020

Table of Contents

Task description

In this project, our aim was to use image recognition methods to identify gender based on fingerprint images. We used some image processing methods before feature selection, then used logistic regression to classify it.

Algorithm description

Before feature selection, we used binarization to reduce the complexity of the images. Then we applied skeletonizing to reduce the image. After that we used feature detection to extract features from the skeletonized image. For feature detection, we used ORB, Kaze and Hu feature detection and binary description algorithms. We used ORB, Kaze and Hu responses converted to a list as our features. Then we separated our data into training/testing parts and trained our logistic regression classifier on the training data. The classifier was tested on the test part of our data and the results were plotted.

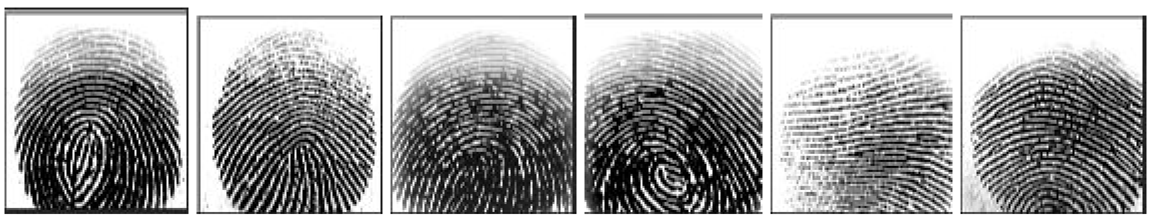
ORB is a fusion of FAST keypoint detector and BRIEF descriptor with many modifications to enhance the performance. First it uses FAST to find keypoints, then applies Harris corner measure to find top N points among them. ORB uses BRIEF descriptors.

In KAZE the points of interest as found using nonlinear diffusion filtering which preserves edges and improves the distinctiveness, while it is computationally expensive compared to the algorithms discussed before it has better performance and a more stable repeatability score than FAST/AGAST based detectors. They do well with textured images. AKAZE is the faster version of KAZE. The KAZE/AKAZE descriptors only work with KAZE/AKAZE keypoints in OpenCV.

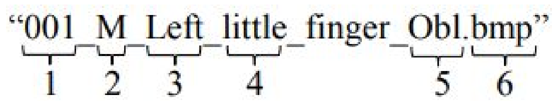
Hu Moments are a set of 7 numbers calculated using central moments that are invariant to image transformations. The first 6 moments have been proved to be invariant to translation, scale, and rotation, and reflection. While the 7th moment’s sign changes for image reflection.
Hu Moments are normally extracted from the silhouette or outline of an object in an image. By describing the silhouette or outline of an object, we are able to extract a shape feature vector to represent the shape of the object.

Testing images/recordings selection

For our project we used Sokoto Coventry Fingerprint Dataset (SOCOFing). It is a biometric fingerprint database designed for academic research purposes that is made up of 6,000 fingerprint images from 600 African subjects and contains unique attributes such as labels for gender, hand and finger name(100__M_Left_index_finger_CR.BMP) as well as synthetically altered versions with three different levels of alteration for obliteration, central rotation, and z-cut. We used these labels for labeling our training data. All file images have a resolution of 1 × 96 × 103 (gray × width × height) and are on bmp format.



The file format provides the labels for each individual image and has the naming convention of:



where:

1. Identifies the number of the subject: 001 to 600.
2. Indicates the gender of the subject: M – male, F – female.
3. Denotes the hand: Left or Right.
4. Indicates the finger name: little, ring, middle, index, or thumb.
5. Indicates the type of alteration type (altered images only): Obl – obliteration, CR – central rotation, or Zcut.
6. File extension: ".bmp" for all images.

Example Results

Tests with 1200 data and 10000 epoch..

```
orb-----------------------------------------
Accuracy on test set: 82.77%
Loss on test set: 0.48
--------------------------------------------
hum-----------------------------------------
Accuracy on test set: 78.15%
Loss on test set: 0.53
--------------------------------------------
kaz-----------------------------------------
Accuracy on test set: 77.73%
Loss on test set: 0.53
--------------------------------------------
```

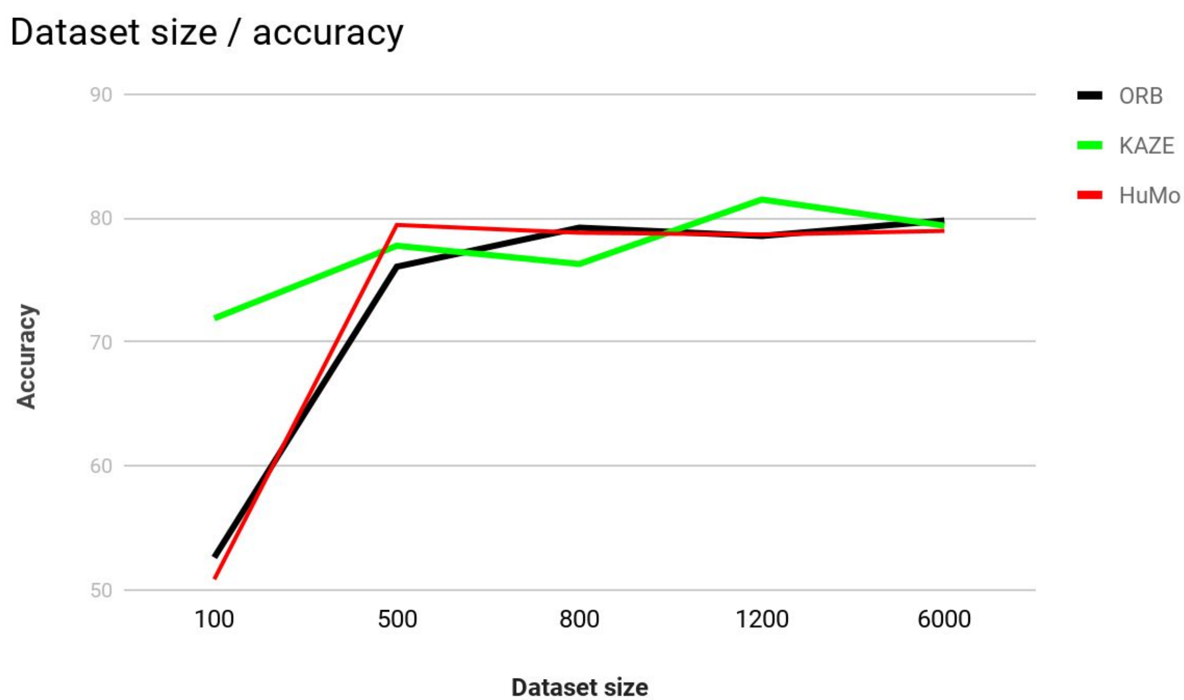
```
orb-----------------------------------------
Accuracy on test set: 80.67%
Loss on test set: 0.71
--------------------------------------------
hum-----------------------------------------
Accuracy on test set: 78.99%
Loss on test set: 0.52
--------------------------------------------
kaz-----------------------------------------
Accuracy on test set: 84.03%
Loss on test set: 0.45
--------------------------------------------
```

```
orb-----------------------------------------
Accuracy on test set: 76.05%
Loss on test set: 0.56
--------------------------------------------
hum-----------------------------------------
Accuracy on test set: 77.31%
Loss on test set: 0.53
--------------------------------------------
kaz-----------------------------------------
Accuracy on test set: 87.39%
Loss on test set: 0.43
--------------------------------------------
```

```
orb-----------------------------------------
Accuracy on test set: 74.79%
Loss on test set: 0.58
--------------------------------------------
hum-----------------------------------------
Accuracy on test set: 80.25%
Loss on test set: 0.50
--------------------------------------------
kaz-----------------------------------------
Accuracy on test set: 76.89%
Loss on test set: 0.54
--------------------------------------------
```

Result Analysis

We tested our data with different feature extractors and different data sizes. For data sizes we used 100, 500, 800, 1200 and 6000. The results on the same data sizes were similar and were around 75-80% most of the time. Accuracy was low on small data size and became increasingly stable as data size increased. At 6000, all feature extractors gave very similar results around 80%.



Additional thoughts

We have noticed that increasing feature vector size didn’t have a significant impact on accuracy beyond a point and around 10-15 were enough. Since Hu moments always returns vectors with length seven, it wasn’t changed. And, we thought, when the dataset size increases, accuracy would decrease however it increased and at some point accuracy stayed the same.

Sources

https://www.kaggle.com/ruizgara/socofing#100__M_Left_index_finger_CR.BMP

https://docs.opencv.org/3.4/db/d95/classcv_1_1ORB.html

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_orb/py_orb.html

https://docs.opencv.org/3.4/d3/d61/classcv_1_1KAZE.html

https://blog.annaphilips.com/feature-detectors-descriptors-opencv/

https://medium.com/machine-learning-world/feature-extraction-and-similar-image-search-with-opencv-for-newbies-3c59796bf774

https://towardsdatascience.com/comparing-the-feature-extraction-algorithms-for-images-e27c3c662874