The Outex TC12 Classification

Muaz KURT

Implemented all parts with cpp. Requirements are OpenCV, CMake, Make, G++17.

Build:

To build this directory follow the next terminal codes:

mkdir build && cd build && cmake .. && make

Information:

In this project, I am supposed to classify the given test images wrt. train images and their labels. To use it

```
./Texture_Classifier Outex_TC_00012/
```

Ps: Now the process prints each guess to the console.

How to:

- I have done all the problems that declared from dataset.
- I read all declared problems train images and extracted their features with calculating LBP and the histogram of it, then stored those features with their label into the memory.
- After that, I read all the test images one by one and calculated their features with the same way as train cases.
- Then, calculated L2 distances of a test image with each instance train image features.
- Finally guessed the working images label by the minimum L2 distance of those calculations.

Comparison:

For comparing the effect of different approaches to the problem, I could only made ignoring least significant bits of a feature. Then calculated the result.

It doesn't improve the result at all. Even reduces the accuracy after ignoring 4 and more lsb.

Result:

At the beginning of this project, I calculated each train images features as now I am doing it. But with a simple difference, I calculate the mean of each feature column for every label of images. Then I got a (24 x 255) feature vector that describes all the images. With this approach I got ~68% of accuracy.

And then I realised that I am making a mistake. And then I calculated each train image feature vector, with the label of image and stored it to the memory. Now I got
(<Number of images>x(255x1)) feature vector that could describe each image. As a result, I got ~90.5% of accuracy to guess the label of an image.

```
For the dataset000, after ignoring 0 lsb of features
rate for 000= 91.6667. Overall miss gap: 6.7
For the dataset000, after ignoring 1 lsb of features
rate for 000= 90.625. Overall miss gap: 5.46667
For the dataset000, after ignoring 2 lsb of features
rate for 000= 91.875. Overall miss gap: 3.30769
For the dataset000, after ignoring 3 lsb of features
rate for 000= 82.9167. Overall miss gap: 36.2683
For the dataset000, after ignoring 4 lsb of features
rate for 000= 82.9167. Overall miss gap: 104.463
For the dataset000, after ignoring 5 lsb of features
rate for 000= 81.25. Overall miss gap: 11.8889
For the dataset000, after ignoring 6 lsb of features
rate for 000= 75. Overall miss gap: 10.1167
For the dataset001, after ignoring 0 lsb of features
rate for 001= 91.6667. Overall miss gap: 18.375
For the dataset001, after ignoring 1 lsb of features
rate for 001= 92.5. Overall miss gap: 21.3889
For the dataset001, after ignoring 2 lsb of features
rate for 001= 91.4583. Overall miss gap: 16.8049
For the dataset001, after ignoring 3 lsb of features
rate for 001= 84.1667. Overall miss gap: 28.5395
For the dataset001, after ignoring 4 lsb of features
rate for 001= 73.75. Overall miss gap: 37.3968
For the dataset001, after ignoring 5 lsb of features
rate for 001= 76.25. Overall miss gap: 39.5
For the dataset001, after ignoring 6 lsb of features
rate for 001= 71.6667. Overall miss gap: 19.0735
```

The miss gap means the mean of the first encounter of actual type for each test image.