RCV Project2 Report:

Siyuan Zhong 168005338

In this project, I use C++ and openCV 2.49.

1. BOW:

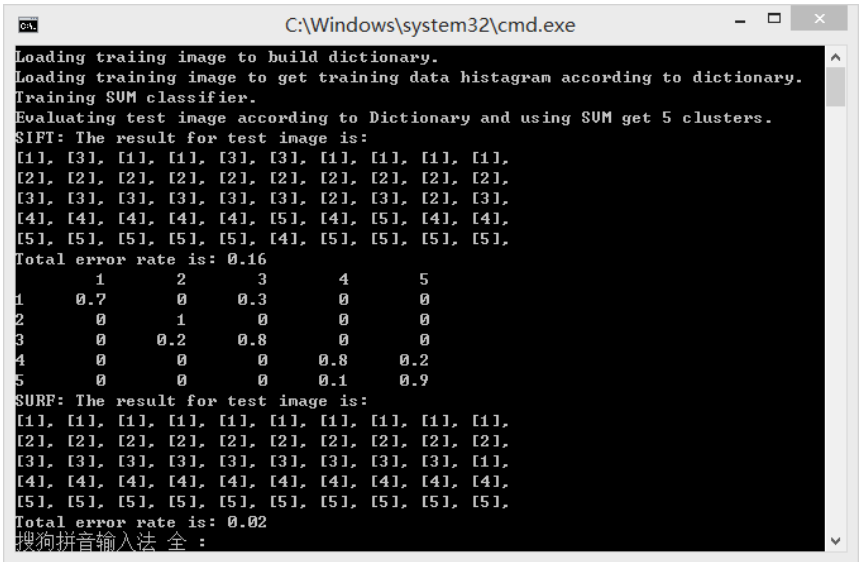
In this project, I use the library from openCV itself. In the program, lots of classes and function can be found in features2d. Totally I use three methods to get result and display confusion matrix.

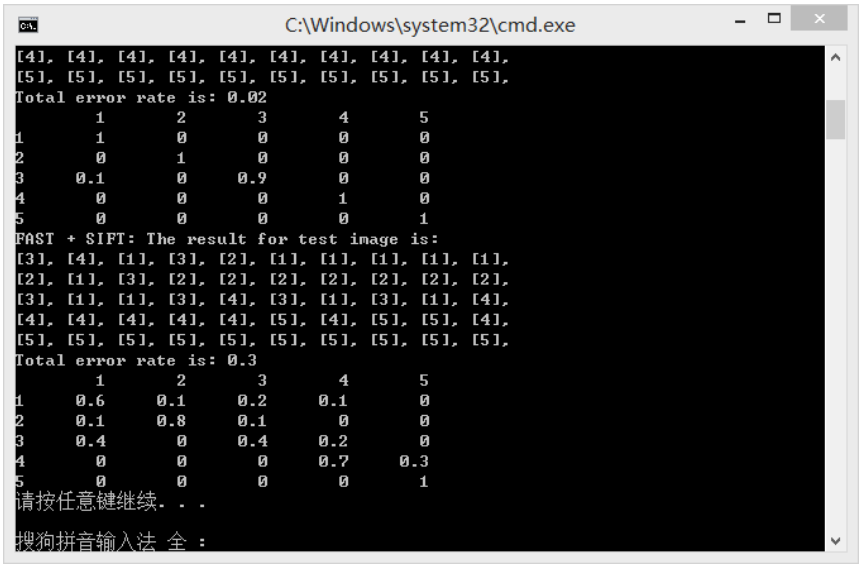
The first method uses SIFT to detect key points and uses SIFT to extract descriptor for BOW. The second method is use SURF to key points and uses SURF to extract descriptor for BOW. The third one is use FAST (One algorithm can be find in openCV document and the key points usually are corners) to detect key points and use SIFT to extract descriptor for BOW.

For all these three methods we follow this step:

1) Detecting key points from training image and extract descriptor for those key points.
2) Decided a dictionary size (k) and use BOWKMeansTrainer to train those key points and it can get k×i (i is decided by its own descriptor) matrix as a dictionary. This step can get K centroids for those keypoints.
3) Put the dictionary to BOWImgDescriptorExtractor and get BOW descriptor for train data each one is k×1.
4) Because we already know those train data class, so we use SVM to train those BOW descriptors and divide it to 5 classes.
5) Input test image to BOWImgDescriptorExtractor and get BOW descriptor for test data.
6) Use SVM and to predict it belongs to which class. The input for SVM is those test image BOW descriptor.
7) Get confusion matrix.

Generally for these three methods, SURF perform best its total error rate is 0.02. SIFT total error rate is 0.16. FAST+SIFT total error rate is 0.3. The results are display as below (1, 2, 3, 4, 5 represent five class object which are mouse, pens, razors, vacuum cup, glass bottle. In each class there are two or three different objects):





For SIFT method we can see that mouse are easily to confuse with razors. It may because that they are similar in the outward and background also similar. And the vacuum cup and glass bottle also similar to each other.

For SURF method every classification performs well.

FAST + SIFT doesn't perform very well. Because FAST extract corners. So for some object like mouse and razor which are much more rounded in the outward.

When I use my own method, at first I want to use another extractor to extract BOW descriptor like FAST + BriefDescriptorExtractor. But the program always aborts in BriefDescriptorExtractor. I spent a lot of time debugging for this exception abort and finally find that openCV BOWImgDescriptorExtractor can only support descriptor are float except SIFT and SURF. I search this information in the internet (http://answers.opencv.org/question/24835/is-there-a-way-of-using-orb-with-bow/).

Here are some sample images from data:

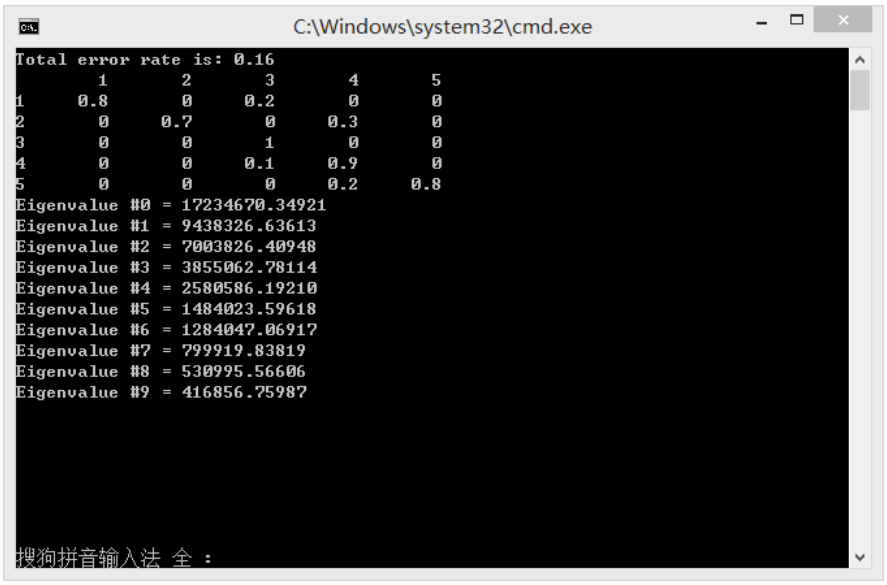
2. Eigenface (extra credit for undergraduate)

In this question, I use Extended Yale Facedatabase B. I randomly choose 5 different people, and each people have 20 faces (the differences of this faces are angles and illuminations). In each class 10 faces for training and 10 faces for testing. And then use openCV FaceRecognizer to train input image. The procedure for this question as below:

1) Input the data to two vector, one is store the image, and one is store the labels.
2) Create eigenface recognizer. Use two vectors as input.
3) Input the train data. Then use face recognizer predicts it belongs to which class. And print confusion matrix.
4) Display mean face, and ten eigenfaces (the largest 10 eigenvalues).
5) Use eigenface to reconstruct one test face (select different number of eigenface as base vector. In this question I select 10, 20, 30, 40.).

The confusion matrix is display as below, we can see that the total error rates for this 50 faces is 0.16. In some case we can use eigenface method to recognize people.

The result is showed as below:

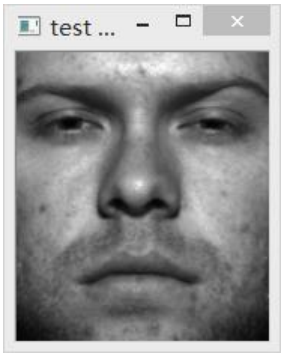


Error rates and largest 10 eigenvalue

Mean face

Ten Eigenfaces

4 reconstruct faces according to Eigenface (10, 20, 30, 40)



Actual image

So from the results, we can see that eigengace method could reconstruct images very well. And the error rate for eigenface method is not too high.