Computer Vision Final Project Report
Face Identification using Local Binary Pattern

Bagas Abisena Swastanto - 4415345

July 10, 2015

1 Problem

Face recognition is one of the prominent task that computer vision has been trying to solve. Face recognition is believed to be a superior choice to the other identification technique such as fingerprint and iris, because face recognition is non-intrusive, fairly undetected, and able to be performed from afar [1]. There are two main application of face recognition. First, is face verification, which is a one-to-one match between the captured and the enrolled face in the system. The second is face identification, which aims to match a face with a collection of faces in the database. Face identification is the problem that this project attempt to solve. In this report, the term face identification and face recognition are used interchangeably. Face identification can be seen as a one-to-many matching. Because of the nature of one-to-many mapping, the face identification system has to acknowledge the variation that appears in the faces stored in the database, such as illumination, pose, orientation, and the presence of glasses. Next section will discuss the technique used to handle such challenges.

2 Solution

Zou et al. divide face identification methods into two main categories, the *holistic matching method* and the *local matching method*. Holistic method uses the global information available on the face as descriptor. The popular Eigenfaces [2] and Fisherface [3] belong to this category. These global methods perform well on frontal faces, but suffer on classifying faces which exhibit high degree of variance. As an alternative, the local method is developed.

The idea of local method is to split the face into regions. After that, the feature in each region is extracted and then combined to represent the whole face. Since the feature is obtained locally from parts of the face, local method should be more robust to variance. There are several local-based techniques for face identification, two of them are the Local

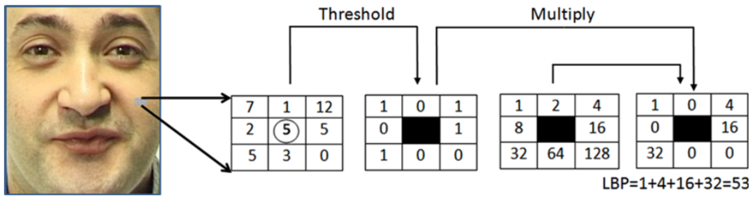


Figure 1: Original LBP implementation

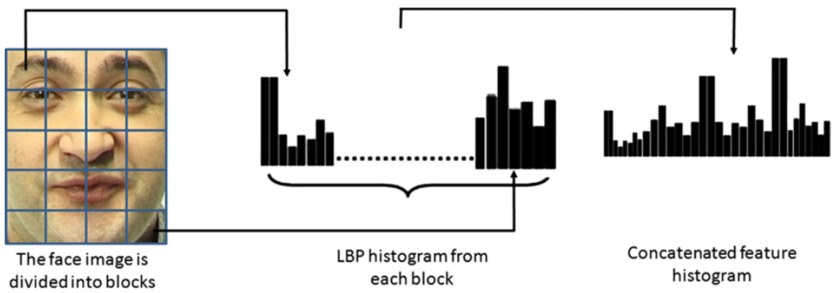


Figure 2: LBP for face identification

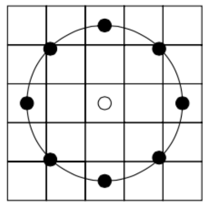


Figure 3: Circular neighbor with 8 points and radius of 2

Binary Pattern (LBP) and Gabor filter. Those two techniques achieve good performance on face recognition hence it is recommended [1]. I will focus on the implementation of LBP for the project.

LBP originally used for texture classification. Each pixel is compared with the its neighbors in 3x3 block, whenever the neighboring pixel value is larger than the pixel value, the neighbor is labeled with 1, otherwise 0. The bits assigned to the neighbors make up a pattern consist of 8 bits, which then converted to decimal value. All decimal values occurrence in the image are counted into a histogram of 256 (2^8) bins which then used as descriptor. Figure 1 shows the illustration of LBP.

Ahonen et al. [4] make several adjustments to the original LBP technique to make it appropriate for face identification problem. First, the LBP histogram is extracted per image block instead of whole image. The histogram is normalized by its sum then concatenated with the histogram from other region. Figure 2 shows the illustration. Moreover, instead of using 3x3 block as neighbor, the LBP label or pattern is determined by its circular neighbor as shown in 3. When the neighbor points are not located on the center of the pixel, the value is determined using bilinear interpolation. Lastly, uniform pattern is applied. A LBP label is called uniform when the pattern contains at most two 0-1 and 1-0 transition. Bitstring 0001000 is uniform (2 transitions) while 10101000 is not (5 transitions). Uniform patterns have separated bins on the histogram, while the occurrence of non-uniform patterns are combined into one bin. Uniform pattern greatly reduce the number of bins and thus keep the feature vector compact. My LBP implementation will be based on above mentioned techniques.

Before extracting the LBP feature, an effective preprocessing chain proposed by Tan and Triggs [5] for the image is performed. The normalization consists of three steps, those are the gamma correction, difference-of-gaussian filtering, and contrast equalization. It is shown by the author that the normalization should make the LBP-based face identification more robust to extreme illumination condition while very fast to compute.

3 Experiment

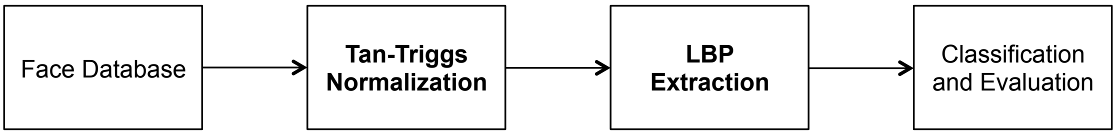


Figure 4: Flow of the experiment. Bolded blocks indicated my own implementation.

Diagram in Figure 4 shows the flow of project. Refer to Table 3 for the overview of all implementations and data, both developed by myself and by others.

Face Database

I use two face databases for the experiment. First is the YALE face database. It consists of 165 grayscale images with 15 subjects. There are 11 images per subject, each image show different facial expression or configuration. I use the preprocessed version where all the images are already aligned and rotated such that the eyes are in the center of image and horizontally placed. The face size is 200x160 pixel. This dataset represent a controlled image recognition situation where all images are frontally faced.

Second database is the Labelled Faces in the Wild (LFW) [6]. It contains 13,000 images of 1680 public figure scraped from the internet. I only choose the person who has more than 20 distinct images in the dataset. This cuts down the number of images to 3023. The face size in this dataset is 60x45 pixel. I have to resize the image to keep the training time reasonable. The images in this dataset are unconstrained with many poses and variations, therefore it let to a more difficult problem than the YALE dataset.

Normalization and LBP Extraction

Before feature extraction, the images are prepressed using the Tan-Triggs normalization. Both YALE and LFW dataset are preprocessed using the same parameter: $\gamma = 0.2, \sigma_0 = 1, \sigma_1 = 4, \alpha = 0.1, \tau = 10$.

In this project, I use the circular neighbors with radius of 2. The number of neighbor points are 8 per-pixel. Uniform pattern is applied, therefore each image block histogram will contains 59 bins instead of 256. The locality is obtained by splitting the image into non-overlapping rectangular grid, all with the same size. I use 40×40 and 10×10 block for YALE and 5×5 block size for LFW dataset.

I also extract the eigenfaces from the dataset, train the same classifiers, and use that as performance comparison with the LBP feature. I use the first 10 principal components as a feature for YALE dataset and 100 principal components for LFW dataset.

Classification and Evaluation

The identification task is done by two classifiers, 3-nearest neighbors and linear support vector machine (SVM) with parameter $C = 1$. I choose these classifiers because the the length of LBP feature vector is larger than the number of the available image, especially in the YALE dataset, where the length is equal to 18880 when using 10x10 block. For kNN distance measure, the χ^2 distance is used, which is a commonly use metric for comparing two different histograms. After extracting the LBP feature for both dataset, the classifiers are trained with all available images. I use 10-fold cross validation to evaluate the face identification accuracy of the classifiers.

4 Result

Table 1 and 2 show the identification accuracies on cross validation tests for YALE and LFW dataset respectively. It is clear from results that LBP outperforms eigenfaces in both dataset. On the 'easier' YALE dataset, LBP able to attain perfect identification accuracy. The performance drops to 80% accuracy on the more difficult LFW dataset, but LBP still performs better identification compared to eigenfaces feature by 20% margin. Using a smaller block helps increase the identification accuracy of LBP feature. That is understandable because by using smaller block we can capture more precise local information in the face. Interestingly, using Tan and Triggs normalization does not really help the performance. Also from the result we can see that SVM classifier more suitable for both dataset.

Table 1: Identification Accuracy on YALE dataset

	Accuracy (%)	
Feature	3-NN	SVM
LBP-40x40	97 ± 5.3	88 ± 18.8
LBP-40x40-norm	97.3 ± 6.1	89.3 ± 17.2
LBP-10x10	100 ± 0	98 ± 4.3
LBP-10x10-norm	99.3 ± 2	100 ± 0
Eigenfaces	4 ± 5.3	90.3 ± 20

Table 2: Identification Accuracy on LFW dataset

	Accuracy (%)	
Feature	3-NN	SVM
LBP-5x5	70.2 ± 2.4	79 ± 2.7
LBP-5x5-norm	66.8 ± 3	76.7 ± 2.5
Eigenfaces	8.2 ± 2.3	59.3 ± 2.3

References

[1] Stan Z Li and Anil K Jain. Handbook of face recognition. 2011.

[2] Matthew Turk, Alex P Pentland, et al. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 586–591. IEEE, 1991.

[3] Peter N Belhumeur, João P Hespanha, and David J Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):711–720, 1997.

[4] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. Face description with local binary patterns: Application to face recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(12):2037–2041, 2006.

[5] Xiaoyang Tan and Bill Triggs. Enhanced local texture feature sets for face recognition under difficult lighting conditions. *Image Processing, IEEE Transactions on*, 19(6):1635–1650, 2010.

[6] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

A Extras

Table 3: Project component implementation

Component	Implementation
Dataset-YALE	Use preprocessed image available from http://vismod.media.mit.edu/vismod/classes/mas622-00/datasets/
Dataset-LFW	Use software routine from scikit-learn Python library to automatically download, resize, and format image in Numpy array
Tan and Triggs normalization	Use my own implementation on top of Numpy
LBP feature	Use my own implementation on top of Numpy
Eigenfaces	Use software routine from scikit-learn Python library
Classification	Use software routine from scikit-learn Python library

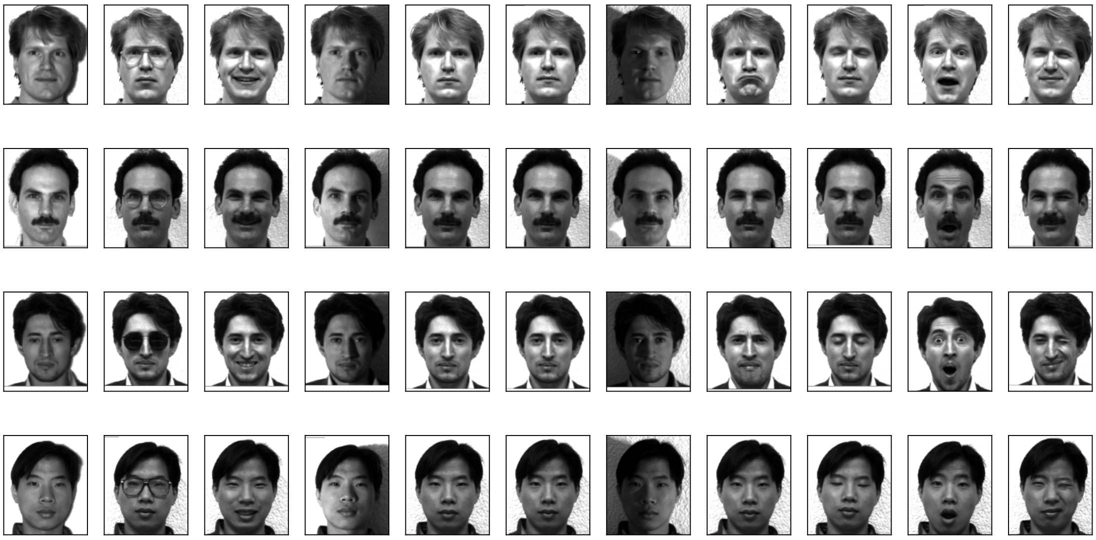
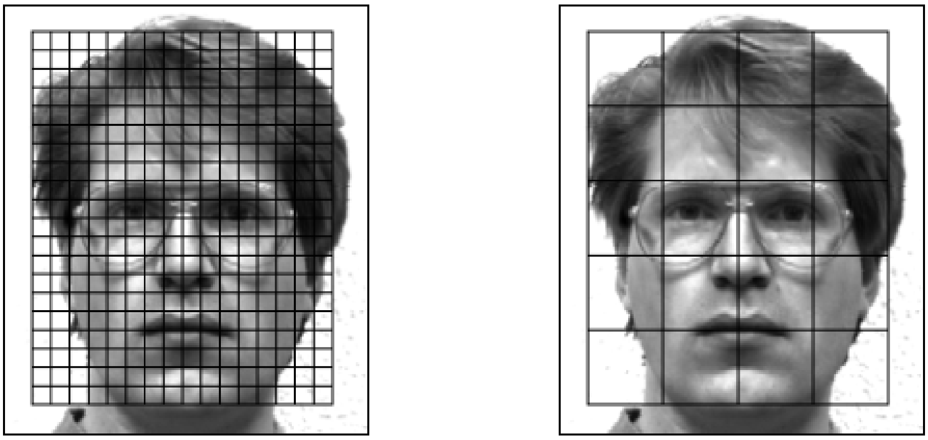


Figure 5: YALE face dataset

Figure 6: Labeled face in the Wild dataset



(a) 10x10 block size (b) 40x40 block size

Figure 7: Image block for LBP feature extraction on YALE dataset

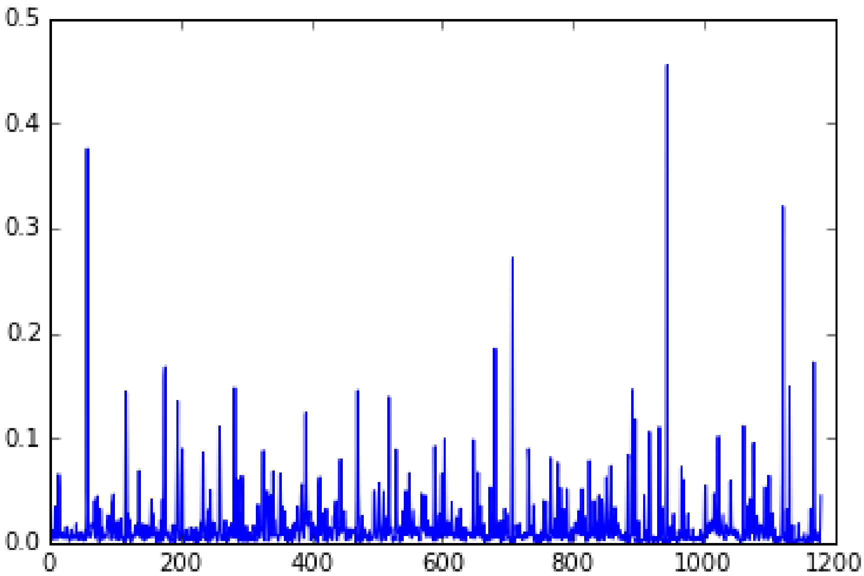


Figure 8: Example of concatenated LBP histogram for one image