# Operational Efficiency of Face Recognition Algorithms on Raspberry Pi

Kriangkrai Limthong[1,a] and Yajai Charoensuk[1,b]

[1]Department of Computer and Robotics Engineering, School of Engineering,
Bangkok University Main Campus, 9/1 Phaholyothin Road, Klong 1, Klong Luang,
Pathumthani 12120 Thailand

[a]<kriangkrai.l@bu.ac.th>, [b]<yajai.char@bumail.net>

## Abstract

Choosing the appropriate algorithm is one of the key issues in face recognition systems, especially for the Raspberry Pi. As we know, the Raspberry Pi functions as a small computer that contains limits of computation and memory capacity. Therefore, face recognition algorithms mainly affect operational efficiency of the system, especially in real-time detection. In this paper, we evaluated efficiency of three famous algorithms in both training and test phase. The face recognition algorithms are Eigenfaces, Fisherfaces, and Local Binary Patterns Histogram (LBPH). Our experimental results of these three algorithms will help you to select the face recognition algorithm, which eminently suitable for your system.

***Keywords:*** *Eigenfaces, Fisherfaces, LBPH, face recognition, raspberry pi*

## 1. Introduction

Face recognition is a technique to identify or verify the identity people in photos or in a real-time video. This technology has been a quickly developing, and testing in vast application areas in past few years. Facial recognition is being applied in many businesses, such as payments, access and security, criminal identification, advertising, and healthcare. In 2016, MasterCard launched a new selfie pay app called MasterCard Identity Check. Customers launch the app to confirm a payment by using their mobile phone camera. In future, it looks like consumers will be able to get into their cars, houses, and other secure physical locations simply by looking at them. It can be used to keep unauthorized people out of facilities as well. In addition, the US Federal Bureau of Investigation is attempting to do by using a machine learning algorithm to identify suspects from their driver's licenses. Moreover, the ability to collect and compare masses of facial data has given advertisers the chance to get closer to their target markets. Finally, medical professionals could identify illnesses by looking at a patient's face. These are examples of application by using facial biometrics.

The Internet of things (IoT) is one of the devices that has been applied for facial recognition over recent years. The IoT has seen steady growth in the number of devices, smart home appliances, smart personal gears, personal assistants and many more. Devices of the IoT are typically low-powered and limited computational resources. It mean that advanced algorithms, deep learning network or convolutional neural network for example, are hardly possible for facial biometric by using these devices. Therefore, we need a lightweight and streamlined recognition algorithm for coding into these IoT devices. The Raspberry Pi is a low cost and very tiny sized computer that contains capability to do face recognition process. Even though, the new version of Raspberry Pi 4 has been improved a new more powerful processor and more choice of memory, up to 4GB, it seems difficult to apply advanced algorithms for real-time facial recognition without slow frames per second (fps), especially in real-time videos recognition. For these reasons, we need some

evaluation and comparison of simple algorithms that could be employed in restricted IoT devices, such as Raspberry Pi.

In this paper, we present experimental results of three simple and lightweight algorithms for face recognition system on IoT device. The face recognition algorithms are Eigenfaces, Fisherfaces, and Local Binary Patterns Histogram (LBPH). We conducted these experiments upon a Raspberry Pi 3 model B+ as an IoT device. A Raspberry Pi camera module has connected to the experimental device for collecting facial data and for detecting process. We employed Python language and crucial library, OpenCV and DLib for example, to implement face recognition system to the device. In our experiment, we measure operational time on training and testing processes by using several numbers of faces. After that, we compared time efficiency of Eigenfaces, Fisherfaces, and LBPH algorithms.

The rest of this paper is structured as follows. In section 2, we briefly review existing algorithms related to our paper. Section 3 describes our experimental environment and we show the comparison results in section 4. We draw a conclusion from our results in section 5.

## 2. Related Work

In face recognition system, we could divide algorithms in two main parts: first, face detection algorithm and second, face recognition algorithm. In this section, we presented Haar cascade algorithm, which has been used for face detection process, especially in real-time video detection. We also explained three recognition algorithms, namely Eigenfaces, Fisherfaces, and Local Binary Patterns (LBP). In our experiments, we measured time efficiency of these three algorithms in both training and testing steps.

### 2.1 Haar Cascade Algorithm

Haar cascade algorithm [1] consists of a collection of stages, in which each stage is an ensemble of weak learners. The weak learners are simple classifiers called decision stumps. Each stage is trained using a technique called boosting. Boosting provides the ability to train a highly accurate classifier by taking a weighted average of the decisions made by the weak learners. Each stage of the classifier labels the region defined by the current location of the sliding window as either positive or negative. A positive indicates that an object was found and a negative indicates that no objects were found. If the label is negative, the classification of this region is complete, and the detector slides the window to the next location. If the label is positive, the classifier passes the region to the next stage. The detector reports an object found at the current window location when the final stage classifies the region as a positive. Cascade classifier training requires a set of positive samples and a set of negative images. Haar-like features are attributes extracted from images used in pattern recognition.

Their names are derived from their similarities to Haar wavelets. First, the pixel values inside the black area are added together; subsequently, the values in white area are added. Next, the total value of the white area is subtracted from the total value of the black area. This result is used to categorize image sub-regions. The application of this algorithm varies from face detection to other object recognition applications. During the Haar-cascade algorithm process, the AdaBoost learning algorithm was also applied to boost the performance of the training process. AdaBoost required a large number of examples that had a strong effect on the generalization of the training error. It combined weak classifiers into strong ones using its specific equations. By collecting positive and negative images of a single object, this algorithm can build a completed classifier that can detect an object within a short time (almost real-time) and with high efficiency compared to other algorithms.

### 2.2 Eigenfaces Algorithm

Eigenfaces algorithm [2] is well established that was used to recognize a feature in a facial image. It is based upon Principle Component Analysis (PCA). In this method the fundamental concept is to recognize the face by taking its unique information about the face in question. Then

encode it to compare with the decode result of previously taken image. In Eigenfaces process, decoding is performed with the calculation of eigenvector and then it is represented as a matrix. However, Eigenface based face recognition system is only suitable for images having the frontal faces, but some researches identify a face with different poses have also been made. Analyzing different results drawn from the researchers the accuracy ratio has been much improved in recent years. It is expected to have an effective and efficient output in upcoming years.

Eigenfaces have advantages over other techniques available, such as the system's speed and efficiency; as Eigenface is primarily a dimension reduction method, a system can represent many subjects with a relatively small set of data. As a face-recognition system it is also fairly invariant to large reductions in image sizing; however, it begins to fail considerably when the variation between the seen images and probe image is large. The deficiencies of the Eigenface method are also obvious. It is very sensitive to lighting, scale and translation, and requires a highly controlled environment. Eigenface has difficulty capturing expression changes. In addition, the most significant Eigenfaces are mainly about illumination encoding and do not provide useful information regarding the actual face.

### 2.3 Fisherfaces Algorithm

Fisherfaces is another method similar to the Eigenface technique. It uses linear discriminant analysis, this method for facial recognition is less sensitive to variation in lighting and pose of the face than using Eigenfaces. Fisherface uses labelled data to retain more of the class-specific information during the dimension reduction stage. Face recognition using Fisherfaces algorithm is based on the reduction of face space dimension using principal component analysis (PCA) technique, then apply Fisher's Linear Discriminant (FLD) method or also known as Linear Discriminant Analysis (LDA) [3] method to obtain feature of image characteristic. The algorithm used in the process for image recognition is Fisherfaces algorithm while for identification or matching face image using minimum Euclidean. Since the learning set is labeled, it makes sense to use this information to build a more reliable method for reducing the dimensionality of the feature space.

But in its development, face recognition with Fisherface method still have some problems, such as computation problems and the condition of the face image into input that will be used as image testing. The problem of computation in face recognition using Fisherface method becomes a problem because it has a very complicated and very complex computation process. While the problems that affect the condition of the face image is the diversity of the light of the face image, the attributes of the face image, the expression of the face image, and the variation of the position of the image of the face itself.

### 2.4 Local Binary Patterns Histogram (LBPH)

The original Local Binary Patterns [4] labels the pixels of an image with decimal numbers, called LBP codes, which encode the local structure around each pixel. It proceeds thus. Each pixel is compared with its eight neighbors in a 3x3 neighborhood by subtracting the center pixel value. The resulting strictly negative values are encoded with 0 and the others with 1. A binary number is obtained by concatenating all these binary codes in a clockwise direction starting from the top-left one and its corresponding decimal value is used for labelling. The derived binary numbers are referred to as Local Binary Patterns or LBP codes.

One limitation of the basic LBP operator is that its small 3x3 neighborhood cannot capture dominant features with large scale structures. To deal with the texture at different scales, the operator was later generalized to use neighborhoods of different sizes. A local neighbourhood is defined as a set of sampling points evenly spaced on a circle which is centered at the pixel to be labelled, and the sampling points that do not fall within the pixel are interpolated using bilinear interpolation, thus allowing for any radius and any number of sampling points in the neighbourhood.

## 3. Methodology

In this section, we present overview of our system. We also show how to collect facial data from university subjects and how to conduct our experiments. We mainly collected the face data from a group of students in our university. The number of students is 41 persons, both male and female in classrooms. Moreover, in our experiments, we divide the tests into two phases, first training phase and second test phase. We slightly varied algorithms from Eigenfaces, Fisherfaces, and LPBH. We have to apply the same algorithm in both training phase and test phases for time measuring and evaluating.

### 3.1 Our Face Recognition System

We developed a face recognition system by using the Python language with OpenCV as a main library for computer vision. Our code has been implemented in a Raspberry Pi 3 model B+ with a Pi camera module. The system consists main three steps, data gathering, train recognizer, and recognition step, as shown in Figure 1

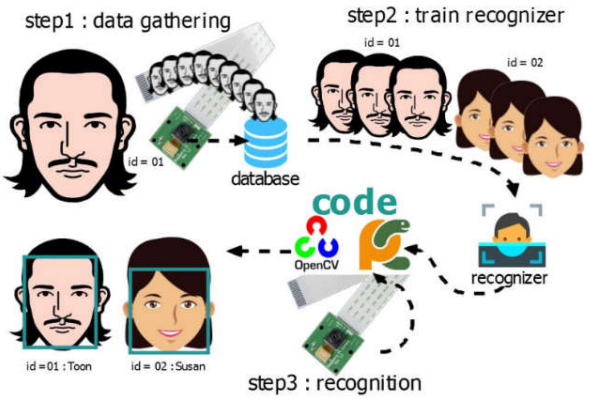


**Figure 1 Overview of our face recognition system.**

Data gathering step, we collected facial data from 2 groups of total 41 students and saved them into a standard data file system of Raspberry Pi. We captured facial images of person with our application from video stream. In this application, we input an identification number for each single person, student id or citizen id for instance. While taking photos, subject was far away from camera about one meter in control environment, such as background and light. It took around one minute or less for each person. Our application captured 100 images for individual. However, the number of image for taking photos can change in our source code. All image file are jpeg format and saved into separated folder for each person.

Second step is train recognizer. Our program processed all images from the data gathering step to learn and train our classifier for recognize each facial person. In this step, we have to select one of the three algorithms via user interface of the application. The program employed selected algorithm to extract features for individual and produced a model or classifier for each person. We also do the same these procedures for all three algorithms. In case of adding a new person, we have to restart these all processes in this step for all existing persons and a new person.

Recognition is the last step of our system. Users have to select an algorithm to detect the subject face. Our application has been processed all time during detection. Haar cascade algorithm has invoked to detect faces in ranges of camera. After that our program recognize the facial data by comparing features of input image to collected data in database from the second step. The algorithm in this step have to be the same algorithm in the second step. For example, if we applied Eigenfaces in second step, we have to use Eigenfaces in recognition step as well. In addition, we also measured recognition time for each algorithm and compared efficiency.

### 3.2 Data

We collected facial data from students of Engineering School, Bangkok University, Thailand. All students are freshmen and junior of Computer Engineering Department. The number of data is 41 students, which consist 25 females and 26 males. For each person, we took 100 facial images per time and did it again in different background, environment for 5 times. Therefore, we collected 500 facial images for individual. All images have been converted to black and white before storing them into a folder for individual.

### 3.3 Experimental Processes

We separated experiments into 3 main items. First items, we varied the number of facial data in training step. The second item, we changed the algorithm among selected three algorithms, namely Eigenfaces, Fisherfaces, and Local Binary Patterns Histogram (LBPH). The last item, we evaluate accuracy of these algorithms. In the first item, we measured the time consumption of both training and detecting step related to the number of facial data. The number of facial data is varied from 10 to 500 images, add 10 images at a time. The second item, we changed the algorithm, which is the same in both training and detecting step, then repeat procedures in the first item again for all algorithms.

## 4. Experimental Results

From experiments, we could show our results in three figures. The first figure exhibits time consumption in training process for each algorithm. The second figure shows time consumption in testing or detection process for each algorithm. The last figure presents detection performance in term of accuracy. All these three figures explain the trend of each value, when we varied the number of training data per person, from 10 to 500 images.

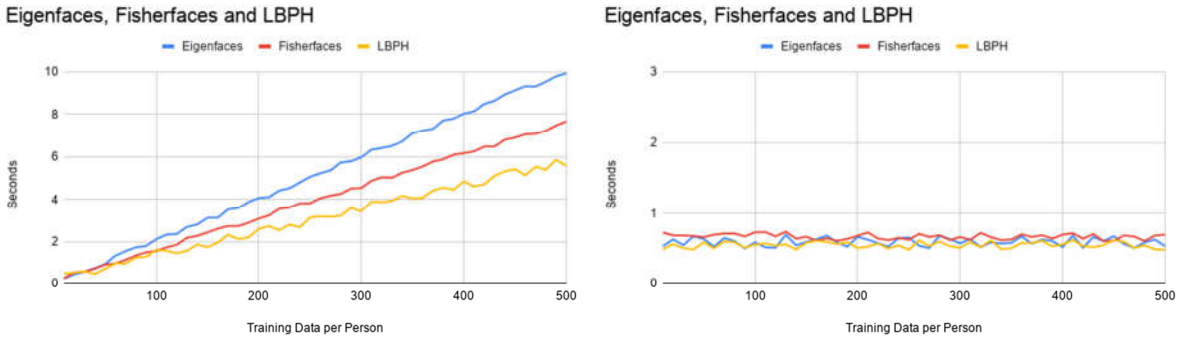


**Figure 2 Time consumption in training process (left) and detection process (right).**

Figure 2 (left) presents time consumption in training process. The y-axis shows the time in seconds. The x-axis shows the number of facial images for each person. The result suggests that all three algorithms spent almost the same time for training data between 10 to 50 images per person. Between 50 to 100 images, the time consumptions are quite different for each algorithm. After 100 images, order of time consumption in training process, from long to short time period, are Eigenfaces, Fisherfaces, and LBPH respectively.

Figure 2 (right) presents time consumption in test or detection process. The result suggests that all three algorithms use time for test or detection process nearly the same. All algorithms spent a short period less than a second, even though the number of training images is increasing.

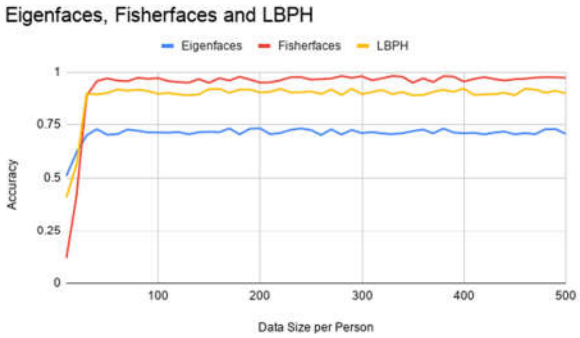


**Figure 3 Detection performance in term of accuracy.**

Figure 3 presents performance in term of accuracy for each algorithm. The result suggests that Eigenfaces performs quite well than Fisherfaces and LBPH when the number of training data is low. However, both fisherfaces and LBPH gain better performance than Eigenfaces when the number of training data larger than 30 images. After 50 images of training data, all three algorithms perform in steady stage until experiment finished in 500 images per person. This figure also shows that the order of best performance is Fisherfaces, LBPH, and Eigenfaces respectively, when we train the system with the number of images larger than 50 images.

## 5. Conclusions

From experimental results, we conclude that Eigenfaces is suitable for a system that contains few numbers of training data about 10 to 30 images. However, if the system could collect training images rather than 50 images, we suggest Fisherfaces or LBPH rather than Eigenfaces, due to they gain the better performance. Whereas, LBPH spent time in training data less than those using Fisherfaces.

## References

[1] P. Viola and M. Jones. (2001) "Rapid object detection using a boosted cascade of simple features". **IEEE Int. Conf. Computer Vision and Pattern Recognition (CVPR)**. pp.511-518.
[2] L. Sirovich and M. Kirby. (1987) "Low-dimensional procedure for the characterization of human faces". **Journal of the Optical Society of America A 4**. 1987. pp.519–524.
[3] Klecka and William R. (1980). **Discriminant analysis. Quantitative Applications in the Social Sciences Serie**. No. 19. Thousand Oaks, CA: Sage Publications.
[4] DC. He and L. Wang. (1990). "Texture Unit, Texture Spectrum, And Texture Analysis"., **IEEE Transactions on Geoscience and Remote Sensing**. vol. 28. pp.509 - 512.

**Kriangkrai Limothong:** He received a B.Eng (2nd Honors) degree in Computer Engineering from Sripatum University; and a M.Eng degree in Computer Engineering from Kasetsart University, Thailand. After that he pursued the Ph.D. degree in the Department of Informatics from the Graduate University of Advanced Studies (Sokendai), Japan. He has currently be a lecturer in the Department of Computer Engineering, School of Engineering, Bangkok University, Thailand, since 2009. His research interests are network traffic measurement, computer security, signal processing techniques, and machine learning methods and applications.

**Yajai Charoensuk:** She received a B.Eng degree in Computer Engineering, School of Engineering from Bangkok University, Thailand in 2018. She also got a second round award from the twenty-first National Software Contest (NSC 2019) under the title of 'BU Face Recognition and Identification (BU-FRI)'. She research interests are technology innovation for better life, development of IoT such as Raspberry Pi, image processing techniques, and computer network.