# *w*PFP-PCA: Weighted Parallel Fixed Point PCA Face Recognition

Chakchai So-In and Kanokmon Rujirakul
Department of Computer Science, Khon Kaen University, Thailand

**Abstract**: *Principal Component Analysis* (*PCA*) *is one of the feature extraction techniques, commonly used in human facial recognition systems. PCA yields high accuracy rates when requiring lower dimensional vectors; however, the computation during covariance matrix and Eigenvalue Decomposition (EVD) stages leads to a high degree of complexity that corresponds to the increase of datasets. Thus, this research proposes an enhancement to PCA that lowers the complexity by utilizing a Fixed Point (FP) algorithm during the EVD stage. To mitigate the effect of image projection variability, an adaptive weight was also employed added to FP-PCA called wFP-PCA. To further improve the system, the advance in technology of multi-core architectures allows for a degree of parallelism to be investigated in order to utilize the benefits of matrix computation parallelization on both feature extraction and classification with weighted Euclidian Distance (ED) optimization. These stages include parallel pre-processor and their combinations, called weighed Parallel FP-PCA wPFP-PCA. When compared to a traditional PCA and its derivatives which includes our first enhancement wFP-PCA, the performance of wPFP-PCA is very positive, especially in higher degree of recognition precisions, i.e., 100% accuracy over the other systems as well as the increase of computational speed-ups.*

***Keywords***: Face recognition, FP, parallel face recognition, parallel euclidian, parallel PCA, PCA.

*Received December 27, 2013; accepted May 21, 2014; published online March 8, 2015*

## 1. Introduction

Facial recognition systems are currently used in a wide variety of applications including biometrics, pattern recognition and analysis, image processing and computer vision which in turn have led to various real world applications, e.g., robotics, crowd surveillances, access controls and criminal forensics [18].

Facial recognition is the process used to automatically identify and/ or verify a face from an image. The image is usually an unanimated image of a human (person) face in a digital picture snapshot or a video frame [7, 26]. There are several techniques used to achieve the recognition, one of which is to compare selected facial features, i.e., feature based extraction.

Overall, face recognition systems usually consist of two components: Face detection and face recognition [7, 26]. The first component identifies the position of the face before the face identification process can begin in the recognition stage. There has been a number of research projects carried out to achieve higher precision results for the first component [12, 15]. However, although some approaches have been introduced to improve the second one, several issues still remain in the research community interests. Thus, this research focuses on the second component or face recognition.

There are several approaches used to enhance recognition precision and to reduce the computational complexity [24, 29, 35]. Most common image-based face recognition methods with feature-based extraction can be divided into two categories: The appearance-based approach and the model-based approach [32].

Each of which has its own distinctive characteristics. For instance, the first scheme is designed to support images with low resolution and/or poor quality but the second approach considers each standing point of the actual face structure including face variation before forming a face feature model. This approach normally requires human-interaction. Apart from a specific face model, i.e., the expression and position of a human image, the first approach yields a high degree of accuracy and is widely employed in traditional image-based face recognition.

Hence, numerous proposals have adopted the appearance-based approach such as Principal Component Analysis (PCA), Independent Component Analysis (ICA), Linear Discriminant Analysis (LDA), Fisher Analysis (FA), Kernel PCA (KPCA) and Isometric Feature Mapping (ISOMAP) [15, 24].

PCA and its derivatives have been used due to several features. For example, PCA is an optimal linear scheme in terms of mean squared error for compressing a set of high dimensional vectors into a set of lower dimensional vectors. The model parameters in PCA can be directly computed from the data without additional processing steps. This means that PCA only requires matrix manipulation. PCA requires less numbers of features to maintain precision quality. Collectively, these advantages result in high recognition accuracy, even with a small data set [32].

Despite these advantages, one of the drawbacks of PCA is the high degree of complexity due to large matrix manipulation operations. A large amount of memory is also required because the memory

requirement increases with image quality and resolution and the number of training images used for image matching or classification [9, 32].

A number of researchers have attempted to improve the efficiency of PCA, i.e., symmetrical PCA and two-dimensional PCA [32]. Some proposed an alternative approach to lessen the computational complexity of PCA, i.e., applying QR decomposition instead of Singular Value Decomposition (SVD) [9].

Moreover, one possible approach for dealing with high computational tasks is to lessen the serial limitation by using the parallelism concept; therefore, with recent rapid advances in computer chip and integrated circuit technology, multi-cores and their parallelization techniques have become accessible and affordable, which may provide a practical solution to some of the limitations of PCA.

As a result, this research focuses on the feasibility to enhance the performance of a traditional PCA face recognition system in both accuracy and computational complexity. Our proposed enhancements are three-folds: First, to seek for alternate approach for integrating the Fixed Point (FP) algorithm; second, to optimize the first improvement to eliminate the random side-effect and projection variability; and finally, to employ a degree of parallelism over multi-cores including classification optimization, all enhancements are *w*eighted Parallel FP-PCA (*w*PFP-PCA).

This article is organized as follows: Section 2 briefly discusses the background of face recognition systems. Then, a comparative study of a traditional PCA, PCA derivation's optimizations and a parallel PCA face recognition are covered in section 3. Next, section 4 presents our proposal with two PCA enhancements using FPs or *w*FP-PCA and parallelism of *w*FP-PCA or *w*PFP-PCA. The comparative performance of our proposal was discussed in section 5. Finally, section 6 contains the conclusions and possible future work.

## 2. Face Recognition Systems

Typically, there are four main modules in a traditional facial recognition system: Acquisition, pre-processor, feature extraction and classification [1, 17]. To fully support the recognition system, there are two main processes: Training and testing. Training involves the use of images acquired from various inputs, e.g., captured photos and scanned images (Acquisition); face images are fed as inputs into the pre-processing stage, e.g., image sizing, greyscale transformation and background removal including various normalization techniques (pre-processor). Then, the system extracts the main features of the images (feature extraction).

The testing process is quite similar to that of the training but it requires a few more steps. After the test image has passed through the training process to produce a proper normalized face image, the image will be fed into a face image classifier to figure out the least feature matching distance between testing and trained features (Classification).

The last two modules are the key components for the recognition system in both accuracy and computational complexity. Recently, several techniques have been proposed to enhance both of these aspects. PCA, which primarily uses as a feature extraction [34] is one of the better-known approaches for facial recognition systems. PCA applies a line transformation technique over a sample image to reduce the set of large image variances and then projects their variances into a coordinate axis in order to decrease a dimensional image data space [22].

Figure 1 shows an overview of a typical PCA face recognition system. There are two main components to the system i.e., PCA feature extraction and PCA image classification. Before the PCA face recognition processing stage, the system normally requires a pre-processing step. One of the requirements is to load the input images into an initial matrix, each of which will be converted from RGB to a greyscale model and then rescaled into a common size [17].

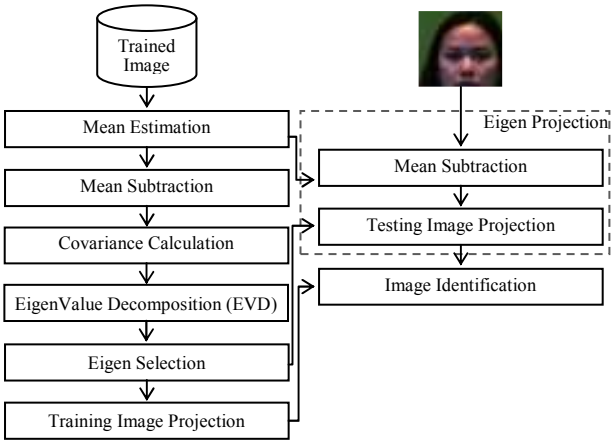



Figure 1. PCA face recognition systems.

Consider the first component, Figure 1-a. There are six sub-components as follows: Estimating the mean vector of trained images, centering the input data around the mean vector by finding the difference between the input image and image's mean value, performing the covariance matrix calculation, applying Eigenvalue Decomposition (EVD) on the covariance matrix to obtain the eigenvectors/ eigenvalues, sorting the eigenvectors in descending order and then selecting non-zero eigenvalues; and finally projecting training images by calculating the dot-product between the trained images and the ordered eigenvectors.

Consider the second component, Figure 1-b. There are three sub-components as follows: Subtracting the testing image by mean vector, projecting Eigen space by calculating the dot-product on test images and making a comparison between training and testing images. In general, PCA does not specifically identify the classifier scheme and there are several techniques available that can perform this operation, e.g.,

Euclidian Distance (ED), Support Vector Machine (SVM) and K-nearest neighbour.

As discussed previously, applying PCA for face recognition has several distinctive features; however, there are some limitations. For example, PCA involves a complex mathematical procedure due to a transformation from a large number of correlated variables to a smaller number of uncorrelated variables. This situation occurs when processing a high resolution image, a large number of images or a combination of both, which will result in a high computational intensity during matrix manipulation [8].

## 3. Related Works

Recently, several approaches have been proposed to optimize both recognition precision and computational complexity including PCA derivations such as the research carried out by Zhao *et al.* [37] who reported that most of surveyed proposals used PCA for different purposes and then obtained several distinctive features. The results of the survey brought about various PCA derivations including optimizations to increase recognition precision rates.

Previously, Gumus *et al.* [10] applied a hybrid approach of PCA and wavelets to extract face features resulting in higher recognition rates. The recognition rates were further improved by Bansal and Chawla [2] who proposed Normalized-PCA (N-PCA), which normalized the face images to remove the lighting variations and background effects by applying SVD.

Moreover, Yue [36] exploited a radial basis function to construct a kernel matrix by computing the distance of two different vectors generated by the parameter of a two-norm exponential and then applying a cosine distance to calculate the matching distance leading to a higher recognition rate.

With regards to computational complexity, Chen *et al.* [5] proposed a local facial feature framework for still images and video-based face recognition using feature averaging, mutual subspace, manifold to manifold distance and affine hull, which resulted in high speed processing for video surveillance but with low recognition rate as a key limitation.

To reduce computational complexity, Bingham and Hyvarinen [3] carried out research into ICA, a competitive approach of PCA, which mathematically analyzed and discussed the effectiveness of FP for ICA in complex valued signals. Sharma and Paliwal [30] discussed a concept of using FP instead of EVD for uniform random vector data, which is a time-consuming stage of PCA, leading to high accuracy but with lower computational complexity. It is worth noting that these researches discussed the effective of FP as a general optimized scheme.

Recently, in order to overcome the major limitations of single-core processing, one of the most promising approaches is multi-core parallel computing, parallelism. Several parallel architectures including parallel algorithms and machines have been explored [25]. Most parallel face recognition systems only applied computer hardware architectures; for example, Wang *et al.* [33] introduced a parallel face analysis platform that used a two-level parallel computing architecture: The first level assigned each individual core for recognition purposes by dividing testing and training images in parallel and the second level was used to only make final decisions based on the results of the recognition process.

Previously, Liu and Su [6] modified a distributed system to support parallel retrieval virtual machines by allowing multi-virtual machines for each slave to run individual face recognition algorithms, the so-called Parallel PCA or P-PCA. To improve recognition accuracy, Meng *et al.* [20] applied a parallel matching algorithm with a multimodal part face technique. This technique uses five main face parts on each face that is bare face based on PCA and then used for recognition process in each individual computer hardware system to enhance the speed-up of the matching process called Multimodel Part PCA (MMP-PCA).

Some proposals focused on applying each individual computer system to run an individual image or a subset of face images; for instance, Cavalcanti *et al*. [4] proposed a recent novel method called weighted Modular Image PCA (*w*MIMPCA) by dividing a single image into different modules to individually recognize human faces for the purpose of complexity reduction of a traditional PCA.

Most approaches discussed previously can achieve some degrees of parallelism by only performing an individual face recognition algorithm either in multi-virtual machines or multi-core processing, but with key limitations on the number of cores. In general, these approaches do not fully utilize parallelism in each recognition stage; and this is one of the focuses of this research - to propose a parallel system that fully utilizes parallelism during face recognition stage.

Apart from the recognition stage, the other two stages; namely, the pre-processor and classification phases are also crucial. For instance, some researches in the first phase were carried out by Haiyang [11] who introduced wavelet transform, discrete cosine transform and colour normalization, as pre-processors in face recognition to achieve better recognition precision. Kavitha *et al.* [16] introduced colour transformation with fractional fourier transform that included local binary pattern to increase recognition rate. However, both of these reports did not fully cover stage's parallelisms.

Several approaches have been also reported on the comparative performance of classification stages, e.g., ED, Manhattan distance, Mahalanobis distance, nearest neighbour and SVM [13]. For example, Melnykov and Melnykov [19] employed Mahalanobis

Distance for the K-mean algorithm to improve the performance when covariance matrices are not properly initialized resulting in an increase in computational complexity. In addition, several techniques based on soft computing-based approaches, i.e., neural networks, are used to compute the matching distance [36, 38].

For instance, Yue [36] used a nearest neighbour method to compute the matching distance to improve classification precision. Zhou *et al*. [38] also investigated a combination of PCA and LDA for image reconstruction which was then classified by SVM.

Most facial recognition systems have applied ED for face classification to simplify the process thus yielding acceptable classification precision. To emphasize the feasibility of this technique, Draper [8] compared Manhattan distance, ED and Mahalanobis distance over PCA. They reported that ED has the highest accuracy rate. Recently, Moon and Pan have [23] also compared Manhattan distance, ED, cosine similarity and Mahalanobis distance over LDA leading to a positive performance results for ED.

In summation, PCA yields high face recognition precision but with very high complex mathematical procedures leading to the optimization of several PCAs. Here, our proposal is to investigate the inclusion of FP into PCA to reduce the computational complexity. To mitigate the effect of random process and projection variability, an adaptive weight was added including the use of covariance matrices to increase the precision.

To further enhance the speed-up, although several approaches have focused on parallel architectures, our proposal deals with each individual stage parallelism during the matrix manipulation of our first enhancement by rearranging the matrix manipulation, including the orthogonalization process. The optimization over parallel classification technique was also investigated in this research. All combined, again, this leads to *w*PFP-PCA.

## 4. *w*PFP-PCA Face Recognition Systems

Figure 2 shows an overall of *w*PFP-PCA, as in Figure 1 but in parallel stages consisting of training/ testing processes as follows: Pre-processor and feature extraction; and pre-processor, Eigen projection and image identification using *w*eighted ED, respectively.

In this system, both training and testing processes will be performed in parallel (*w*PFP-PCA). Initially, an input image will be divided into pixels and then be parallelly executed. During a high matrix computation complexity, a manipulation process is also performed in parallel by computing one element of a result matrix per thread. Generally, the parallel efficiency depends upon a number of cores and the relationships between thread and core use. For example, in a case where there are sixteen processes on eight cores, the system performs the first eight processes followed by remaining eight processes, completed iteratively.

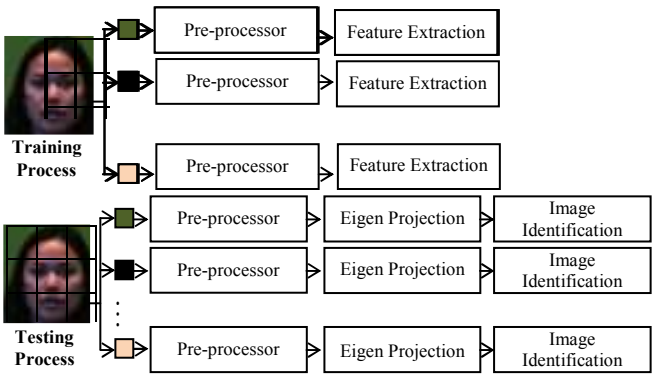



Figure 2. Weighted Parallel Fixed Point PCA Systems: Feature Extraction (*w*FP-PCA) and Image Identification (*w*eighted ED)

### 4.1. Parallel Face Pre-Processor

One of the first key stages, used to adjust/ normalize an input image for further steps, especially for noise reduction, is pre-processing. This leads to computational complexity reduction and increases recognition precision, stated in related work section.

However, parallelism can be considered to further speed-up this process. There are several methods to aid algorithm efficiency, e.g., image scaling, gray scale conversion and histogram equalization. With regards to our previous experiment [28] the recognition precision between colours vs. gray scale images was not significantly different. There was, however, a marked increase in computational complexity and so our focus is to parallelly perform gray scale conversion once the input images are scaled into the same size in both training and testing processes [27].

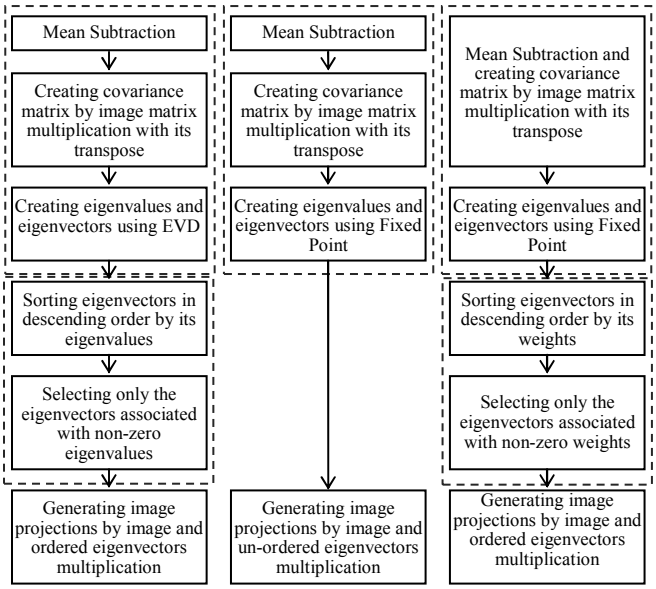
### 4.2. Parallel Face Feature Extraction

As discussed in the related work section, PCA and its derivatives have been proposed to enhance their efficiencies in terms of the increase of face recognition precision and computational time reduction; however, each criterion always comes with a trade-off over one or the other. To mitigate this effect, one of the main contributions of this research is to employ FP-PCA into a traditional PCA during the EVD stage. Another optimization is to explore a degree of parallelism over fixed point PFP-PCA together with employing an adaptive weight as one of the criteria for image matching (*w*eighted or *w*PFP-PCA).

#### 4.2.1. FP-PCA

The EVD stage of a traditional PCA incurs high computational complexity since eigenvectors and eigenvalues are generated according to the number of images. Thus, our first enhancement is to employ FP [23] in this stage as shown in Figure 1 ($4^{th}$ step or $3^{rd}$ step in Figure 3 excluding mean estimation). The covariance matrix calculation was performed first and then FP calculation was applied, instead of EVD, to

obtain only eigenvectors based on a number of pre-defined prospective vectors. Figure 3 illustrates the comparative details of PCA vs. FP-PCA.




a) Traditional PCA. b) FP-PCA. c) *w*FP-PCA.

Figure 3. Feature extraction stages (excluding mean estimation): Mean subtraction, covariance calculation, EVD or FP, eigen selection, image projection: PCA, FP-PCA and *w*FP-PCA.

Algorithm 1 shows the detail as follows (1st to 4th steps in Figure 3):

- Mean Subtraction (line 1): Similar to traditional PCA, this process is to find the difference between the input image and images' mean estimation value.
- Covariance Matrix Computation (line 2): This process is used to perform the arithmetic subtraction of image matrix with its mean. Then, the covariance computation performs the multiplication by image matrix and its matrix transpose; its result will be divided by trained images in the dataset.
- Eigenvector Computation using FP (line 3-9). The eigenvectors will be derived at this stage as follows: Define the expected eigenvectors ($c$); randomly select one dimensional initial eigenvectors; multiply its eigenvector to a covariance matrix; perform the Gram-Schmidt orthogonalization process; normalize the eigenvector; update a new random eigenvector if the normalized eigenvector is not converged within $e$ else re-apply Gram-Schmidt and loop ($L$) through the entire process until the pre-defined counter ($p$) reaches the expected number of eigenvectors.

*Algorithm* 1: FP algorithm: eigenvector generation.

*Input*: *Matrix x*
*Output*: $\varphi_p$ *(Eigenvector Matrix)*

*Subtract x by its mean*
*Compute covariance of* $\sum x$
*Choose c, the number of principal axes or eigenvectors required to estimate and set* $p \leftarrow 1$
*Initialize eigenvector* $\varphi_p$ *of size* $d \times 1$ *randomly*
*Update* $\varphi_p$ *as* $\varphi_p \leftarrow \sum x \, \varphi_p$
*Do the Gram–Schmidt orthogonalization process*

$$\varphi_p \leftarrow \varphi_p - \sum_{j=1}^{p-1} (\varphi_p^T \varphi_j) \varphi_j$$

*Normalize* $\varphi_p$ *by dividing it by its norm:*

$$\varphi_p \leftarrow \varphi_p \div \|\varphi_p\|_T$$

*If* $\varphi_p$ *is not converged (within* $\varepsilon = |\varphi_p \varphi_p - 1|$*), go to step* 4
*Increment counter* $p \leftarrow p+1$ *and go to step* 3
*until p equals c*

In addition to these three states, instead of generating image projections by trained images and ordered eigenvector multiplication operations, FP-PCA projection is generated with (un-ordered) original eigenvectors. Following these stages, for testing purposes, the normal PCA facial recognition will be followed (image classification as shown in Figure 1-b).

Algorithm 1 shows FP-PCA's complexity in the order of $O(d^2c+d^2n)$ [28] vs. $O(d^3+d^2n)$ for the EVD stage, traditionally used in PCA face recognition such that $d$ and $n$ are the number of dimensions of trained and training images, respectively. The accuracy of FP will be based on the selection process of initial eigenvectors and Epsilon with a time complexity trade-off.

### 4.2.2. wFP-PCA

Without an eigenvalue generation process similar to PCA with EVD, the recognition precision may be underestimated due to probable outliers of zero eigenvalues. FP-PCA precision may also be affected by the random process. Thus, the optimization over two issues is also one of our main contributions. Figure 3-c illustrates detailed description of *w*FP-PCA. Algorithm 1 also shows overall structure with the modifications stated below:

- Mean Subtraction and Covariance Matrix Computation (line 1-2): *w*FP-PCA combines these two steps to reduce the computation complexity with detailed discussion stated in section 4.2.3.2.
- Eigenvector Initialization Selection (line 4): Instead of randomly selecting initial eigenvectors in one dimension which may be the side-effect of random process leading to lower accuracy, our proposal is to select the initial values from covariance matrix.
- Weight Derivation (before line 5): This extra step is to adjust the ordering of eigenvectors to mitigate the image projection variability using an adaptive weight derived from the convergence values of Epsilon stated in FP-PCA as shown in Equation 1.

$$Weight = \left| \varphi_p^T \varphi_p - 1 \right| \quad (1)$$

- Ordered Eigenvector Generation (after line 8): Once $p$ equals to $c$, an additional process sorts the eigenvectors in descending order of their weights and eliminates the vectors by their zero weights.

Instead of the unordered eigenvectors multiplication, used in FP-PCA, image projection was performed by trained images and ordered eigenvectors multiplication operations according to their weights.

Once again, after following these stages, for testing purposes, the normal PCA processes will continue. Looking at this algorithm's complexity compared to FP-PCA, instead of $O(d^2c+d^2n)$, *w*FP-PCA leads to additional steps but with an insignificantly different degree of complexity, i.e., $O(d^2c+d^2n+c)$.

#### 4.2.3. wPFP-PCA

To further speed-up of *w*FP-PCA with its limitation of serial computation on single core architecture, a parallelism was investigated for a weighted FP-PCA algorithm consisting of two main stages: Parallel matrix computation and parallel mean subtraction and covariance computation. All combined is *w*PFP-PCA.

##### 4.2.3.1. Parallel Matrix Computation

Based on analysis and experimental observation, there are five matrix operations that occur during *w*FP-PCA: Matrix multiplication and division with constant values, matrix subtraction and transpose, matrix multiplication with matrix and matrix multiplication with its transpose. Each of these operations can utilize parallelism depending on its distinctive characteristics. In addition, an extra process, Orthogonalization, can be performed in parallel.

1. Parallel Matrix Multiplication and Division with Constant Values: This method involves multiplying or dividing every element of the input matrix by a constant number. The result at one position can be retrieved by multiplying or dividing the same position of an input matrix by the constant, which leads to an equalized dimensional matrix. It should be noted that each element value in the result matrix can be individually computed. These calculations are independent; therefore, the fine-grained parallelism can be achieved as shown in Algorithm 2 leading *O(mn)* complexity where *m* and *n* are the number of row and column matrices.

*Algorithm* 2: Parallel matrix manipulation with constant values.

*Input*: *Matrix input, String operation, Double constant_values*
*Output*: *Matrix result*

```
// Matrix Multiplication and Matrix Division
parallel for i from 0 to rows
    for j from 0 to cols
        if (operation == "Multiplication")
            Multiplication:
            result[i, j]=input[i, j]×constant_values
        else if (operation == "Division")
            Division:
            result[i , j]=input[i , j]÷constant_values
        endif
    endfor
endfor
return result
```

2. Parallel Matrix Subtraction and Transpose: To calculate the difference of each element at the same position, the subtraction and transpose of dimensional matrices can be also performed in parallel similar to Algorithm 2 with modifications. Either line (6 or 9) will be changed to these operations as shown in Equation 2 or 3. Since, each individual element of subtracted matrix is independent and the transformation from row to column is not dependent on each other, some degrees of parallelism can be utilized.

$$result[i, j] = left[i, j] - right[i, j] \quad (2)$$

$$result[j, i] = input[i, j] \quad (3)$$

3. Parallel Matrix Multiplication with Matrix: With multiplying with matrix, each operation can be performed in parallel as shown in Algorithm 3 [27] leading to $O(qrp)$ complexity where $q$ and $r$ are the number of left matrix rows and columns, and the right one for $p$. The row dimensions of the right matrix or the multiplier is equal to the column dimension of the left one. The row dimension of the result matrix is equal to the left row dimension while the column dimension of the result matrix is equal to the multiplier column dimension.

*Algorithm* 3: Parallel matrix multiplication with matrix computation [27].

*Input*: *Matrix left, matrix right, matrix input, string operation*
*Output*: *Matrix result*

```
// Matrix multiplication
for i from 0 to leftRows
    parallel for j from 0 to rightColumns
        for k from 0 to leftColumns
            Calculate sum:
            sum+=left[i, k]×right[k, j]
        endfor
        result[i, j]=sum
    endfor
endfor
return result
```

4. Parallel Matrix Multiplication with its Transpose: Looking specifically at matrices with symmetrical and square properties, the multiplication of this matrix with its transpose can be performed in parallel. By the nature of the computational matrix in FP-PCA, the square matrix, one of which is the matrix multiplication with its transpose, $A \times A^T$, could be optimized as stated in Algorithm 4 leading to $O(mn)$ complexity.

*Algorithm* 4: Parallel matrix multiplication with its transpose.

*Input*: *Matrix input*
*Output*: *Matrix result*

```
//(1)Diagonal Calculation
for i from 0 to Rows
    for j from 0 to Columns
        result[i, i]+=input[i, j]^2
    endfor
endfor
//(2)Lower and Upper Diagonal Calculations
b=0, c=0
for i from 1 to Rows
    repeat
```

```
            result[i, c]+=input[i, b]×input[c, b]
            b=b+1
        if (b>=Columns)
            c=c+1
            b=0
            //Copy lower to upper-diagonal
            result[c-1, i]=result[i, c-1]
        endif
    until ( !((c<i) and (b<Columns)))
endfor
c=0
b=0
return result
```

First, the diagonal computational elements can be further derived as stated in Equation 4. The original matrix could be reused as shown in Figure 4 by removing the transpose process, i.e., accessing and reserving memory, but performing the summation of square of the original matrix in diagonal positions instead. Here, *c* is the number of columns.

$$\begin{bmatrix} a[0,0] & a[0,1] & a[0,2] \\ a[1,0] & a[1,1] & a[1,2] \\ a[2,0] & a[2,1] & a[2,2] \end{bmatrix} \times \begin{bmatrix} a[0,0] & a[1,0] & a[2,0] \\ a[0,1] & a[1,1] & a[2,1] \\ a[0,2] & a[1,2] & a[2,2] \end{bmatrix}$$

Figure 4. Examples reused matrix transpose.

$$Dia[i,j]=\Sigma_{j=0}^{c} a[i,j]^2 \quad (4)$$

Second, to figure out non-diagonal values, since the matrix is symmetry, each element in the upper-triangle is the same as that in the lower-triangle (Figure 5), which would half the computational complexity (Equation 5). This calculation is also used in both FP-PCA and *w*FP-PCA for convergence evaluation process.

$$\begin{bmatrix} a[0,0]^2+a[0,1]^2+a[0,2]^2 & A & C \\ A & a[1,0]^2+a[1,1]^2 & B \\ C & B & a[2,0]^2+a[2,1]^2+a[2,2]^2 \end{bmatrix}$$

Figure 5. Optimized matrix multiplication with its transpose.

$$Upper=Lower=\Sigma_{i=0}^{n}\Sigma_{i=0}^{n} a[i,j] \quad (5)$$

5. Parallel Orthogonalization: Gram-Schmidt is an orthogonalization process used to convert a set of vectors into orthonormal vectors. The process starts by normalizing the first vector and iteratively calculating a weight vector of the remaining vectors, then normalizing them. The matrix is normalized as follows: Powering every matrix component by two, summarizing the result in each column and dividing each matrix component in each column by the square root of the summary.

The orthogonalization process proposed in most of the previous works generally performed modified Gram-Schmidt including our preliminary work with FP [28]. However, in this research, a traditional algorithm, Classical Gram Schmidt, was selected in order to gain a higher degree of parallelism as shown in Figure 6. Algorithm 5, $O(n^2)$ complexity, especially shows an additional step to support the epsilon convergence (lines 3 and 10).

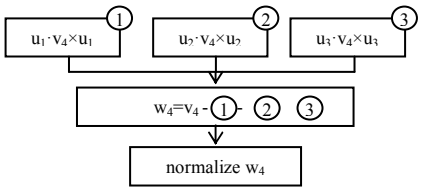



Figure 6. Examples weight calculation of the $4^{th}$ vectors.

*Algorithm* 5: Parallel matrix orthogonalization computation.

*Input*: *Matrix a*
*Output*: *Matrix q*

1. *for j from 1 to n*
2. $v_j = a_j$
3. *repeat*
4. *parallel for i from* 1 *to* (*j*-1)
5. $r_{ij} = q_i \times v_j$
6. *Rewrite Vectors:* $v_j = v_j - r_{ij}\, q_i$
7. *endfor*
8. *Normalize vector:* $r_{jj} = \|v_j\|_2$
9. *Rewrite Remaining Vectors:* $q_j = v_j \div r_{jj}$
10. *until* $|(v \cdot q)-1|<\varepsilon$
11. *endfor*

#### 4.2.3.2. Parallel Mean Subtraction and Covariance Computation

From Figure 3-c, *w*PFP-PCA first merges two steps to reduce the computational complexity, i.e., mean subtraction and covariance computation by the input matrix from mean estimation stages.

Traditionally, given matrix $T[i, j]$ as the training images and $m[j]$ as the image mean in each column, the result of mean subtraction, $a[i, j]$, is stated in Equation 6. In addition, the covariance computation stage process (*Cov*) is the matrix multiplication with its transpose for matrix size $N \times N$ then divided by number of trained images ($n$) as shown in Equation 7.

$$a[i,j]=T[i,j]-m[j] \quad (6)$$

$$Cov=(a \times a^T) \div n \quad (7)$$

This covariance calculation is similar to that of parallel matrix multiplication with its Transpose (Algorithm 4 and Equations 4 to 5) but additionally dividing by $n$ (lines 4 and 12) and again, the original matrix could be reused by removing the transpose process, i.e., accessing and reserving memory, but performing the square of the original matrix instead. To combine these two steps, Equations 6 and 7 are merged into 4 and 5 to reduce the complexity from $O(n^2)$ to $O(n)$ as shown in Equations 8 and 9.

$$Diag[i,i]=(\textstyle\sum_{j=0\ to\ N}(T[i,j]-a[j])^2) \div n \quad (8)$$

$$Upper=Lower=(\textstyle\sum_{i=0\ to\ N}\sum_{j=0\ to\ N}(T[i,j]-a[j])^2) \div n \quad (9)$$

### 4.3. Parallel Face Classification

Following the face feature extraction process, the face classification stage will be performed. The first two components function in the same way as those of PCA for both FP-PCA and *w*FP-PCA. However, for the third component, PCA and FP-PCA employ ED, which is a commonly used classification technique. It

is worth noting that the main reason using ED, Equation 10, is because of several distinctive characteristics and results in an identified image with the least distance. Here, *m* is the projection matrix of a one dimensional image matrix (trained and testing).

$$distance=\sum(m_{train} - m_{test})^2 \quad (10)$$

In a similar way to PCA and FP-PCA, the third component is modified [27, 28] by ED integrating the derived weighted matrices ($w[i]$), *w*FP-PCA, as stated in Equation 11. Once again, *C* denotes as the expected number of eigenvectors.

$$distance=\sum_{i=0\ to\ C} w[i]\times(m_{train} - m_{test})^2 \quad (11)$$

To further enhance *w*FP-PCA, for testing purposes, our parallel classification is based on ED being used to parallelly figure out the closest distance of face images due to the algorithm simplicity while retaining high recognition precision. By employing the parallelism of optimized ED, our enhancement utilizes the input matrix characteristic (one dimensional scale) during the classification stage leading to reduction in complexity of $O(n^3)$ to $O(n^2)$ as shown in Algorithm 6.

Figure 7 also shows the feasibility of utilizing the degree of parallelism over our optimized ED in each of ED's matrices. The matrices are simultaneously computed in parallel, e.g., with four processes, the computation at $d_{11}$, $d_{12}$, $d_{13}$ and $d_{21}$ will be performed in parallel. Also, Algorithm 6 and Figure 7 show the *w*FP-PCA embedded parallelism (line 4), but for FP-PCA and P-PCA, the algorithms exclude the weights.

*Algorithm* 6: Weighted parallel Euclidean distance computation.

*Input*: *Array weight; Matrix A, B*
*Output*: *Matrix result*

```
1. parallel for row from 1 to A_Rows
2.     sum = 0.0;
3.     for col from 1 to A_Cols
4.         sum+=weight[col]×(A[row, col]-B[col])^2
5.     endfor
6.     result[row]=sum
7. endfor
8. return result
```

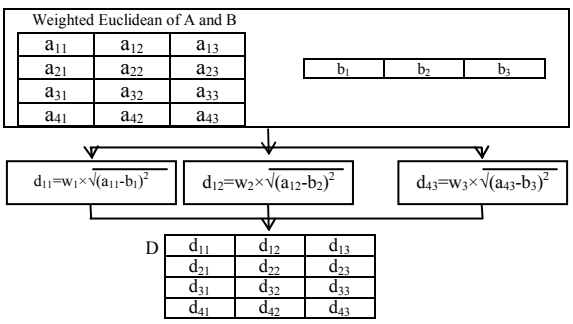



Figure 7. Parallel matrix operation computation for weighted.

## 5. Performance Evaluation

To evaluate the feasibility of *w*PFP-PCA, a comparative performance was carried out using two different scenarios: *w*FP-PCA and its degree of parallelism or *w*PFP-PCA.

### 5.1. Experimental Configuration

The evaluation testbed was a standard configuration on Windows 7 operating systems (64bits): CPU Intel(R) Core (TM) i-3770K 8-Cores 3.50 GHz (8MB L3 Cache), 8192×2MB DDR3-SDRAM, and 500GB 5400 RPM Disk with .NET C# environments [21].

A public face database from FACE94 and FACE95 [31] was used for testing purposes. A set of colours was selected as 24 bits RGB, PNG images, ranging from 100 to 500 images of 180×200 pixels. In all evaluations, there were two main metrics: Average (computational time and recognition precision) and standard deviation for varied random loads (a number of trained images) over five trails [14].

In each evaluation, a classic statistical method was selected, i.e., simple random selection, by randomly selecting nine testing images. Four images were chosen from within the training image dataset and five from the outside [27]. Computational time will be measured during EVD stages, including covariance computation.

The two testing scenarios are described as follows: First, to demonstrate the performance of the first two enhancements with integrated FP-PCA and its added *w*FP-PCA. These were then compared to a traditional PCA by ranging a number of images for scalability purposes. The number of eigenvectors in FP-PCA may result in a computational time trade-off with accuracy. This stage first used 10% of the total images followed by testing with 20% to 90%, but the improvement was not significant, and so 10% was selected for testing purposes.

The accuracy of FP-PCA may be affected by Epsilons so various numbers of Epsilons were also investigated, i.e., $10^{-k}$, where *k* is in the range of 1, 3, 5, 7 and 12, respectively. To maintain recognition accuracy, $k$=12 was selected based on the observation that the accuracy is not significantly improved.

The second scenario investigated the degree of parallelism. The evaluation process was based on a variation of the number of cores that were given a set of numbers of trained images, i.e., 100 images, in the range of 1, 2, 4 and 8 cores, respectively.

The performance of *w*PFP-PCA was evaluated using two main metrics and then comparing the results over the other three parallel face recognition systems: The optimization of PCA integrating its parallelism or Parallel-PCA (P-PCA) [6], MMP-PCA [20] and our first improvement with parallelism integration, i.e., PFP-PCA [28].

### 5.2. Experimental Results and Discussions

Figure 8 illustrates the computational speed of FP-PCA and *w*FP-PCA in that at 500 images, FP-PCA and *w*FP-PCA outperform a traditional PCA by a factor of two. In addition, by ranging the number of training images, the computational speed increases.

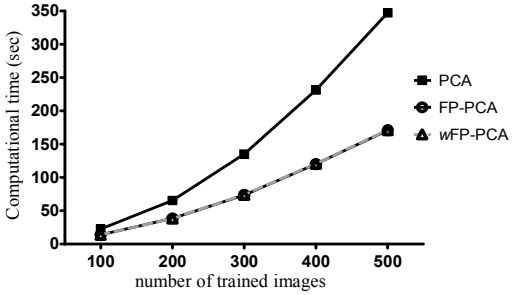



Figure 8. Computational time over number of trained images (PCA, FP-PCA and *w*FP-PCA).

For example, with 100, 200, 300 and 400 images, the speed-up of the first two enhancements is in order of 1.7, 1.7, 1.8 and 1.9, respectively. The mean standard deviation for all cases was within 3 and the computational complexity of our two mechanisms is insignificantly different, leading to similar computational speeds.

In terms of recognition precision, Figure 9 shows that all three systems produced high percentages of accuracy, but FP-PCA had a fluctuation in precision due to the random characteristic. Using weights to absorb this variation, *w*FP-PCA produces excellent performance results, i.e., 100% in accuracy, when compared to PCA and FP-PCA in all cases.

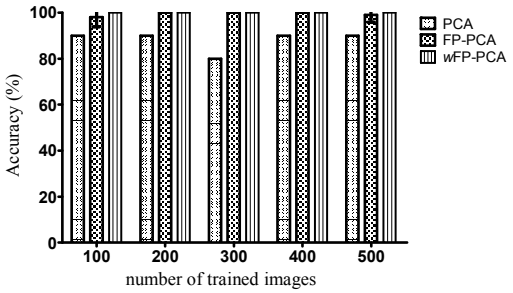



Figure 9. Percentage of recognition precision over number of trained images (PCA, FP-PCA and *w*FP-PCA).

In addition, FP-PCA precision can fluctuate due to the random process, i.e., 98% to 100%, on average, including non-zero standard deviation, but zero for the others. With traditional PCA, the accuracy is in ascending order according to the number of trained images, i.e., 80% to 90%. In addition, the parallelism integration experiment showed that increasing the number of cores, Figures 10 and 11, the parallel face recognition systems produced in-significantly different results with regards to recognition precision.

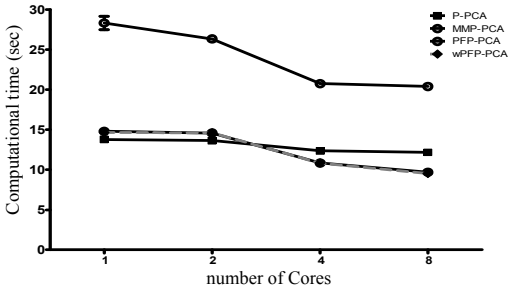



Figure 10. Computational time over number of cores (P-PCA, MMP-PCA, PFP-PCA and *w*PFP-PCA).

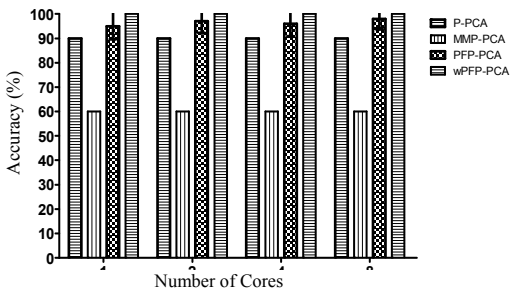



Figure 11. Percentage of recognition precision over number of cores (P-PCA, MMP-PCA, PFP-PCA and *w*PFP-PCA).

Figure 10 also shows that *w*PFP-PCA speed-up is not significantly affected by the additional weights. Also, whenever the number of cores is increased, although P-PCA performance shows a degree of parallelism effect, both PFP-PCA and *w*PFP-PCA still produce excellent performance results, averaging about 15, 15, 11 and 10 seconds; with P-PCA around 14, 14, 12 and 12 seconds; and with MMP-PCA about 28, 26, 21 and 20 (time consuming due to recognition multiplication processes) for 1, 2, 4 and 8 cores, respectively. All of these come with the standard deviation within 1.

Consider recognition precision. *w*PFP-PCA, Figure 11, again, provides the excellent performance results, i.e., 100% in accuracy. PFP-PCA, P-PCA and MMP-PCA were 97%, 90% and 60%, respectively. The standard deviation of P-PCA, MMP-PCA and *w*PFP-PCA are 0s but 5 for PFP-PCA.

## 6. Conclusions and Future Work

The investigation carried out in this research is primarily on facial recognition systems based on PCA. Although, PCA can improve recognition precision rates, a limitation still exists, especially in terms of computational complexity due to large matrix manipulation. Thus, this research proposes an alternative approach by integrating FP into PCA during the EVD stage to produce a high-speed recognition system. To further improve the accuracy, to mitigate the random effect and image projection variability, the concept of adaptive weights was also embedded into FP-PCA yielding *w*FP-PCA.

Due to the advances in parallel computing using multi-cores, there is an opportunity to eliminate the limitations of single-process computation. A novel methodology for face recognition was proposed by applying parallelism into our *w*FP-PCA for large matrix manipulation. This included pre-processor, recognition, and classification in parallels, all of which refer to *w*PFP-PCA.

Based on our parallel algorithm implementations, the parallel system outperforms serial computations, i.e., by a factor of more than 1.7, with in-different in accuracy but considering a variation of parallelism, *w*PFP-PCA and PFP-PCA outperformed parallel PCA derivatives, P-PCA and especially MMP-PCA. These two optimization's yields were insignificantly

different for speed-ups but *w*PFP-PCA produced outstanding performance results, i.e., 100%.

Even though *w*PFP-PCA can achieve a high degree of speed-up, more investigation could be performed, e.g., improving pre-processing stages, enhancing the degree of parallelism, not only in parallel matrix manipulation, reducing sensitivity outliers and testing heterogeneous number of various images including scalability concerns. To show the efficient use of parallelism, autonomously separating recognition tasks can be performed over message passing interfaces or specific processing units. Finally, the face detection can be also investigated in future work.

## Acknowledgements

This research project was in part of grant from the department of Computer Science, Faculty of Science, Khon Kaen University (CSKKU 2556/#03).

## References

[1] Azeem A., Sharif M., Raza M., and Murtaza M., "A Survey: Face Recognition Techniques under Partial Occlusion," *the International Arab Journal of Information Technology*, vol. 11, no. 1, pp. 1-10, 2011.

[2] Bansal K. and Chawla P., "Performance Evaluation of Face Recognition using PCA and N-PCA," *International Journal of Computer Applications*, vol. 76, no. 8, pp. 14-20, 2013.

[3] Bingham E. and Hyvarinen A., "A Fast Fixed-Point Algorithm for Independent Component Analysis of Complex Valued Signals," *International Journal of Neural Systems*, vol. 10, no. 1, pp. 1-8, 2000.

[4] Cavalcanti C., Ren I., and Pereira F., "Weighted Multi Model Part Image Principal Component Analysis for Face Recognition," *Expert Systems with Applications*, vol. 40, no. 12, pp. 4971-4977, 2013.

[5] Chen S., Mau S., Harandi T., Sanderson C., Bigdeli A., and Lovell C., "Face Recognition from Still Images to Video Sequences: A Local-Feature-Based Framework," *Journal on Image and Video Processing*, vol. 2011, no. 11, pp. 1-14, 2011.

[6] Chunhong J., Guangda S., and Xiaodong L., "A Distributed Parallel System for Face Recognition," *in Proceedings of the 4th International Conference on Parallel and Distributed Computing, Applications and Technologies*, Chengdu, pp. 797-800, 2003.

[7] Dandotiya D., Gupta R., Dhakad S., and Tayal Y., "A Survey Paper on Biometric based Face Detection Techniques," *International Journal of Software and Web Science*, vol. 2, no. 4, pp. 67-76, 2013.

[8] Draper A., Baek K., Bartlett S., and Beveridge R., "Recognizing Faces with PCA and ICA," *Computer Vision and Image Understanding*, vol. 91, no. 1, pp. 115-137, 2003.

[9] Gao J., Fan L., and Xu L., "Solving the Face Recognition Problem using QR Factorization," *WSEAS Transactions on Mathematics*, vol. 11, no. 8, pp. 712-721, 2012.

[10] Gumus E., Kilic N., Sertbas A., and Ucan N., "Evaluation of Face Recognition Techniques using PCA, Wavelets and SVM," *Expert Systems with Applications*, vol. 37, no. 9, pp. 6404-6408, 2010.

[11] Haiyang Z., "Image Pre-processing Methods in Face Recognition," *in Proceedings of Symposium* on *Photonics and Optoelectronics*, Wuhan, pp. 1-4, 2011.

[12] Hjelmas E., "Face Detection: A Survey," *Computer Vision and Image Understanding*, vol. 83, pp. 236-274, 2001.

[13] Hu Q., Yu D., and Xie Z., "Neighborhood Classifiers," *Expert Systems with Applications*, vol. 34, no. 2, pp. 866-876, 2008.

[14] Huiyuan W., Xiaojuan W., and Qing L., "Eigenblock Approach for Face Recognition," *International Journal of Computational Intelligence Research*, vol. 3, no. 1, pp. 72-77, 2007.

[15] Jafri R. and Arabnia H., "Survey of Face Recognition Techniques," *Journal of Information Processing Systems*, vol. 5, no. 2, pp. 41-68, 2009.

[16] Kavitha S., Paul V., and Gananasekaran T., "Colour Face Recognition using Texture Features and Fractional Fourier Transforms," *International Journal of Scientific and Engineering Research*, vol. 4, no. 8, pp. 1-3, 2013.

[17] Kshirsagar P., Baviskar R., and Gaikwad E., "Face Recognition using Eigenfaces," *in Proceedings of the 3rd International Conference on Computer Research and Development*, Shanghai, pp. 302-306, 2011.

[18] Lu X., "Image Analysis for Face Recognition," *Personal Notes*, available at: http://www.face-rec.org/interestingpapers/general/imana4facrcg_1u.pdf, last visited 2003.

[19] Melnykov I. and Melnykov V., "On K-Means Algorithm with the Use of Mahalanobis Distances," *Statistics and Probability Letters*, vol. 84, pp. 88-95, 2014.

[20] Meng K., Su D., Li C., Fu B., and Zhou J., "A High Performance Face Recognition System based on a Huge Face Database," *in Proceedings of International Conference on Machine Learning and Cybernetic*, China, pp. 5159-5164, 2005.

[21] Microsoft MSDN, "Parallel for Method," *.NET Framework 4.5*, 2013.

[22] Min L., Bo L., and Bin W., "Comparison of Face Recognition based on PCA and 2DPCA," *Advances in Information Sciences and Service Sciences*, vol. 5, no. 6, pp. 545-553, 2013.

[23] Moon M. and Pan B., "The LDA-based Face Recognition at a Distance using Multiple Distance Image," *in Proceedings of International Conference on Innovation Mobile and Internet Services in Ubiquitous Computing*, Taichung, pp. 249-255, 2013.

[24] Murtaza M., Sharif M., Raza M., and Shah H., "Face Recognition using Adaptive Margin Fisher's Criterion and Linear Discriminant Analysis," *the International Arab Journal of Information Technology*, vol. 11, no. 2, pp. 149-158, 2014.

[25] Rajasekaran S. and Reif J., *Handbook of Parallel Computing*: *Models, Algorithms and Applications*, Chapman and Hall/CRC, 2007.

[26] Rathore N., Chaubey D., and Rajput N., "A Survey on Face Detection and Recognition," available at: http://www.ijcam.com/paperdownload.php?filename=A%20Survey%20On%20Face%20Detection%20and%20Recognition.pdf, last visited 2015.

[27] Rujirakul K., So-In C., and Arnonkijpanich B., "PEM-PCA: A Parallel Expectation-Maximization PCA Face Recognition Architecture," *The Scientific World Journal*, vo1. 2014, pp. 1-16, 2014.

[28] Rujirakul K., So-In C., Arnonkijpanich B., Sunat K., and Pualsanguan S., "PFP-PCA: Parallel Fixed Point PCA Face Recognition," *in Proceedings of the 4th International Conference on Intelligent Systems Modelling and Simulation*, Bangkok, pp. 409-414, 2013.

[29] Shah H., Sharif M., Raza S., and Azeem A., "A Survey: Linear and Nonlinear PCA based Face Recognition Techniques," *the International Arab Journal of Information Technology*, vol. 10, no. 6, pp. 536-545, 2013.

[30] Sharma A. and Paliwal K., "Fast Principal Component Analysis using Fixed-point Algorithm," *Pattern Recognition Letters*, vol. 28, no. 10, pp. 1151-1155, 2007.

[31] Spacek L., "Face Recognition Data," available at: http://cswww.essex.ac.uk/mv/allfaces, last visited 2015..

[32] Turk T. and Pentland A., "Eigenfaces for Recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71-86, 1991.

[33] Wang L., Liu K., Zhang T., Wang Q., and Ma Y., "Parallel Face Analysis Platform," *in Proceedings of International Conference on Multimedia and Expo*, Singapore, pp. 268-269, 2010.

[34] Wang W. and Wang W., "Face Feature Optimization based on PCA," *Journal of Convergence Information Technology*, vol. 8, no. 8, pp. 693-700, 2013.

[35] Wang Z., Sun X., Sun L., and Huang Y., "Semisupervised Kernel Marginal Fisher Analysis for Face Recognition," *The Scientific World Journal*, vol. 2013, pp. 1-13, 2013.

[36] Yue X., "Human Face Recognition based on Improved PCA Algorithm," *Journal of Multimedia*, vol. 8, no. 4, pp. 351-357, 2013.

[37] Zhao W., Chellappa R., Phillips P., and Rosenfeld A., "Face Recognition: A Literature Survey," *ACM Computing Surveys*, vol. 36, no. 4, pp. 399-459, 2003.

[38] Zhou C., Wang L., Zhang Q., and Wei X., "Face Recognition base on PCA Image Reconstruction and LDA," *Optik-International Journal for Light and Electron Optics*, vol. 124, no. 22, pp. 5599-5603, 2013.

**Chakchai So-In** received PhD from WUSTL in 2010 and is currently an assistant professor at the department of Computer Science, Khon Kaen University. His research interests include mobile computing, signal processing, network systems and parallel and distributed systems.

**Kanokmon Rujirakul** received MSc. from NIDA in 2008 and is currently a PhD candidate in the Department of Computer Science, Khon Kaen University. Her research areas include image processing, mobile computing, soft computing and distributed systems.