Parallel Architecture for Face Recognition using MPI

Dalia Shouman Ibrahim
Computer Systems Department
Computer and Information Sciences
Ain Shams University
Egypt

Salma Hamdy
Computer Science Department
Computer and Information Sciences
Ain shams University
Egypt

Abstract—**The face recognition applications are widely used in different fields like security and computer vision. The recognition process should be done in real time to take fast decisions. Principle Component Analysis (PCA) considered as feature extraction technique and is widely used in facial recognition applications by projecting images in new face space. PCA can reduce the dimensionality of the image. However, PCA consumes a lot of processing time due to its high intensive computation nature. Hence, this paper proposes two different parallel architectures to accelerate training and testing phases of PCA algorithm by exploiting the benefits of distributed memory architecture. The experimental results show that the proposed architectures achieve linear speed-up and system scalability on different data sizes from the Facial Recognition Technology (FERET) database.**

Keywords*—*Face Recognition; PCA; MPI; Parallel Programming; Distributed memory architecture

I. Introduction

Over the last decade, face recognition has become one of the most important issues in computer vision and machine learning. Face recognition involves applications like airport security [1], surveillance systems, automated student attendance, album organization, computer entertainment, virtual reality, online banking and video indexing.

One of the goals of surveillance systems for example, is to identify a particular person among large crowds with no physical interaction. The matching results do not require an expert to be interpreted, as the target person is compared against images from a database. Such automated process can be used for finding known criminals and terrorists if their images are stored in the database. One of the basic algorithms to solve face recognition problems is the Principle Component Analysis (PCA)Fig.1.

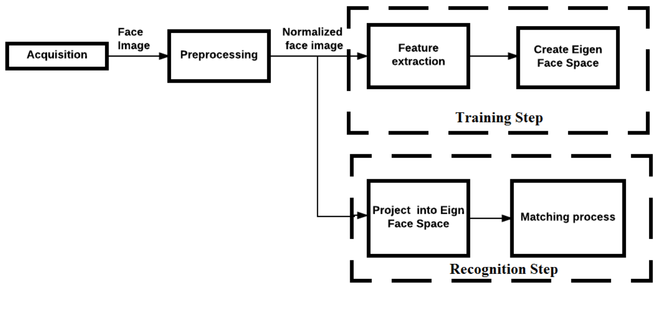

Fig. 1. PCA Recognition System

PCA is used in many fields and have a lot of applications in machine learning and data mining. Any application that has data sets of numerical dimensions can apply the PCA algorithm beacuse PCA is basically used to reduce the dimensions to the lower number of independent dimensions by understanding the correlations from multi-dimensions [2]. In this paper, address the problem face recognition based on PCA, and suggest two different approaches for speeding-up the process.

The rest of this paper is organized as follows. We introduce categories of face recognition algorithms and present the related work in section II. Section III elaborates on the use of PCA for face recognition. In section IV two approaches are proposed for speeding-up the PCA with distributed databases. Experimental results are presented in Section V. Finally, section VI concludes the paper and discuss the future work.

II. Related Work

Face recognition algorithms can be categorized according to how the features are extracted from the face image.

- In appearance based methods, features are mainly from pixel intensity values [3] [4]. These methods can be organized into linear or nonlinear analysis methods [5]. In linear methods, matching depends on the calculated distance between the projected testing face image and the projected training faces in a face space. The smaller the distance between the projected vectors, the more similar the two images are. Example for linear methods include PCA, Independent Component Analysis (ICA) and Linear Discriminant Analysis (LDA). Nonlinear analysis methods try to compensate for the ill performance of the linear methods in some cases when the image has variations in illumination, viewpoint and/or facial expression. Therefore, the relations between pixels are not linear. Examples of such methods include Kernel PCA (KPCA), Kernel ICA (KICA) and Kernel Fisher Discriminant Analysis (KFLD) [6].
- On the other hand, geometrical feature matching [5]. The geometrical feature approach construct feature vectors based on the geometrical representation of the face. The feature vector contains the face outline, eyes, mouth and nose positions.

Interested readers can refer to [7] [8] for a more comprehensive categorization of face recognition

One common linear algorithm is the PCA, which is considered a feature extraction technique and a dimensional reduction method. PCA is widely used for face detection and recognition beacuse of achieving high accuracy while requiring a small number of features [9].

The main problem of PCA is consuming a huge amount of time because of its computationally intensive nature [10]. A major step of the algorithm is calculating the covariance matrix of the dataset to be able to get the eigenvectors and eigenvalues. This step intensively increases the execution time of the algorithm when the training data-base contains a lot of pictures.

Many researches focus on how to accelerate this step to improve the overall recognition performance. One approach is calculating the eigenvectors and eigenvalues without calculating the covariance matrix. Methods like the expectation maximization algorithm (EM) to reduce the determinant matrix manipulation [9].

There are some tries to accelerate PCA. One approach in [11] suggest a use of distributed computing to improve recognition on huge databases, specifically, TH-FACE [12]. Their proposed algorithm uses special distributed parallel architecture with mmx technology to speed-up the matching process in PCA algorithm. Their PC cluster uses a Parallel Virtual Machine (PVM) that consists of one host and five slaves. Consequently, the database is divided into five parts. Furthermore , they divide the face image into five sub parts to improve the recognition rate. This method is called multimodal face recognition method (MMP-PCA). However, they enforce that the input images be labeled with information about gender and age. The input image are normalized to the standard image size (360×480 pixel) to fit the facial key points such as eyes and nose.

In [2], the proposed approach compute PCA in one pass on a large data set based on summarization matrices. Furthermore, that algorithm is applied on database management systems(DBMS). They use parallel data set summarization via user-defined aggregations and solve Singular Value Decomposition (SVD) by using Math Kernel Library (MKL) parallel variant of the Linear Algebra PACKage (LAPACK) library.

Authors in [13], proposed a distributed parallel system consisting of one host, four slaves and some clients. The Parallel Virtual Machine (PVM) is established by the host. In addition, the communication is done over TCP socket and the whole system is communicated and linked over 100M network switch, and achieves an acceleration ratio of 4.133 if the whole system works together.

III. Principle Component Analysis in Face Recognition

The main goal of PCA algorithm is to improve the computational complexity by reducing the dimensionality of the presented data and keeping the significant variations among data points [14].

Assuming a set of data points represented in $(x - y)$ coordinate system. Training and recognition require finiding a relation between these points that enables classification or clustering based on a distance metric like euclidean distance, cosine angle distance, mean square error (MSE) distance, Manhattan distance or correlation distance [15]. Finding such a relation is not easily done and may be erroneous. To centralize the points around the origin, their mean is calculated and subtracted from all points to get them closer. Since the main goal is to reduce the dimensionality another coordinate system is used to represent the centralized points in one instead of two dimensions.

The new coordinate system consists of one long vector split between points. The data is then re-distributed around this new vector such that each point represented with two values is now transformed and projected to a proper location in the new reduced system.

Covariance is one method to model the relation between data points. In this case, the covariance between each data point and every other point is calculated in a matrix C. Consequently, the eigenvector and eigenvalue of C are calculated and used to re-distribute the points into one dimensional space, that is the points should be re-drawn around this eignvector. Besides, the eigenvalues represent the vectors lengths. Therefore the longest vector is chosen to represent the data. This is done by simply applying dot product between each point and the new vector. This achieves the main goal of reducing the space from two to one dimension only.

The above procedure is applicable also to three dimensional systems. These steps can be applied on images. If the image has $M \times N$ pixels, and there are P images in the training database, then, each image is considered the image as a point in $M \times N$ dimensional spaces, with a total number of P points (images). A relation should be established between every pixel in the image and every image in the training database. However, this consumes a lot of time working in this original space.

We can summarize the steps of applying PCA to an image dataset as follows:

- In the training phase

Step 1 : For the purpose of computing the mean, convert the image to be a column in a 2D matrix called A. This matrix represents all images. Therefore, the matrix size is $((M \times N) \times P)$.

$$X = [\ img_1 \quad img_2 \quad ...... \quad img_n\ ]_{((MxN)xP)} \tag{1}$$

Step 2 : Calculating the average of all pixels in all images (each pixel with its corresponding pixels) and subtracting the average from X to remove any lighting. consequently, all pixels are returned back to the origin of the system.

$$avg = \frac{1}{P} \sum_{i=1}^{P} img_i \tag{2}$$

$$A = X - avg \tag{3}$$

Step 3 : Get the eigenvalues and eigenvectors by calculating the covariance matrix. The number of eigenvalues and eigenvector is equal to P-1. The covariance matrix consumes a lot of time, therefore, if the number of images in the training database is less than the number of data points in the image, it is better to calculate the linear covariance L, and then calculate the eigenvalues and vectors based on L [16].

$$C = AA^T \tag{4}$$

$$L = A^T A \tag{5}$$

$$[V, D] = eig(L) \tag{6}$$

Where:
V is the matrix of eigenvectors.
D is the diagonal matrix for eigenvalues.

Step 4 : Sort the eignvectors based on their eigenvalues. The longest vectors which have the largest eigenvalues can split the points in the proper way.
Step 5 : Project the training images to this new space by applying dot product between A and V matrices.
Step 6 : Store the resulting face space.

- In recognition phase

Step 1 : Convert image into a $((M \times N) \times 1)$ vector.
Step 2 : Subtract the mean calculated in the training phase from this image vector.
Step 3 : Project the testing image in the face space.
Step 4 : Calculate the Euclidean distance between the projected test image and all training images. Images having the distance less than or equal to a predetermined threshold is matched to, the testing image belong to that image.

IV. Proposed Approach

We exploit distributed parallel programming models to improve the execution of PCA for face recognition. This enables the distribution of either data or tasks over a network of connected computers. The Message Passing Interface (MPI) [17] enables us to run different MPI tasks concurrently on different machines. In addition, MPI handles the communication by sending message between nodes. MPICH2 [18] implementation is a high-performance and portable implementation of the MPI standard. MPICH2 is provided by Argonne National Laboratory [19].

We use MPICH2 with one master and four slaves nodes to implement a distributed database environment to handle two scenarios.

- Having one test image and a very large database, searching for a matching face could take too long if done on a single processing node. Moreover, if the database is updated frequently, due to system feedbacks or regular new entries, the training steps will be repeated as frequent which will also consume much time on one processor.
  The proposed approach is handled this by splitting the training database and distributing a subset to each node. Every single node, including the master node, uses PCA algorithm to train on its allocated subset and stores the resulting face space locally. During testing, copies of the target image is sent to every node for recognition and each local match is sent back to the master node. The Master concatenates the results with its own and sorts the result and appears the final match based on the smallest Euclidean distance.
- When the training database is somehow fixed, or rarely updated and the training steps are done once or infrequently. Moreover, if the system is receiving a streamed video, there will be a large number of frames, and in each frame there could be several faces to identify, like in airport surveillance systems. Hence, there are a lot of testing images fed into the recognition system.
  Every processing node has a complete copy of the database. This way, each node performs the training phase once and stores the results in its local memory. When the master receives a stream of testing images, they are distributed to the slaves. Each slave runs the PCA algorithm for recognition and retrieves the closest match from the face space which will be the final result for that input image. Finally, the master node collects and displays the results.

V. Experimental Results

The two proposed approaches are evaluated by applying them to different database sizes and measuring the speed-up. The main metric for evaluation is the execution time in each approach separately.

We deploy the proposed architectures on a cluster hosted on the Faculty of Computer and Information Sciences, Ain Shams University, Egypt. The cluster has two blade chassis; each has six blade servers. Each blade server has two Quad-Core Intel®Xeon®CPU E5520 @ 2.27GHz with 24GB RAM. All twelve servers are connected together on an infinite band network. Moreover, VMware ESXi is installed directly on each server. Consequently, the Vsphere client is used to post jobs on these servers. Five servers are used in the presented experiments and the Facial Recognition Technology (FERET) database is used in training and testing [20] [21]. All images are colored, of the same size; 512 x 768 pixels, and stored as 32-bit floating point number in PPM form. We use around 1.5 GB worth of images in the experiments. The 1000 images are used in the first approach but only 500 for training in the second approach because loading time consumes a lot of time and our focus is recognition time.

We adopt a central database approach to enable easy modification in one location. The training and testing images reside on the server, and each slave loads these images via an infinite band network. Thus, the communication overhead does not have a high impact on the performance of the overall system because of the network speed. In the experiments each slave can access the images and stores the training results as an XML file in its local storage. These XML files are used in the recognition steps.

A. Distribute Training Database and Duplicate Test Image

The experiments are carried out using one master node and four slaves with five different database sizes. In the training phase, the master divides the training database equally over the slaves and itself. In addition, it creates text files containing the indices of the images associated with each slave, depending

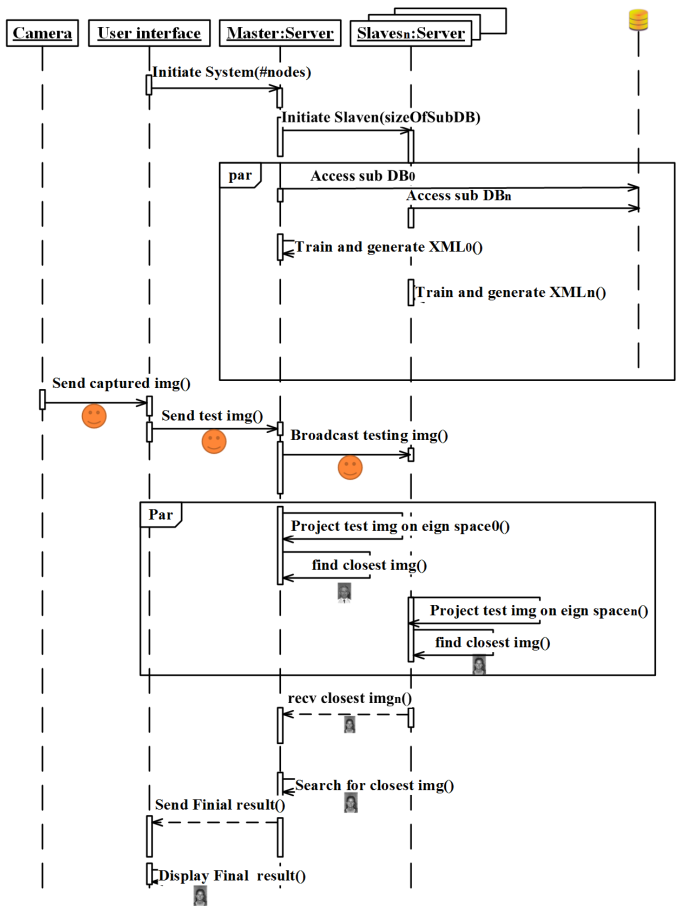

Fig. 2. Sequence diagram to recognize 1 test image vs 1000 images in training DB

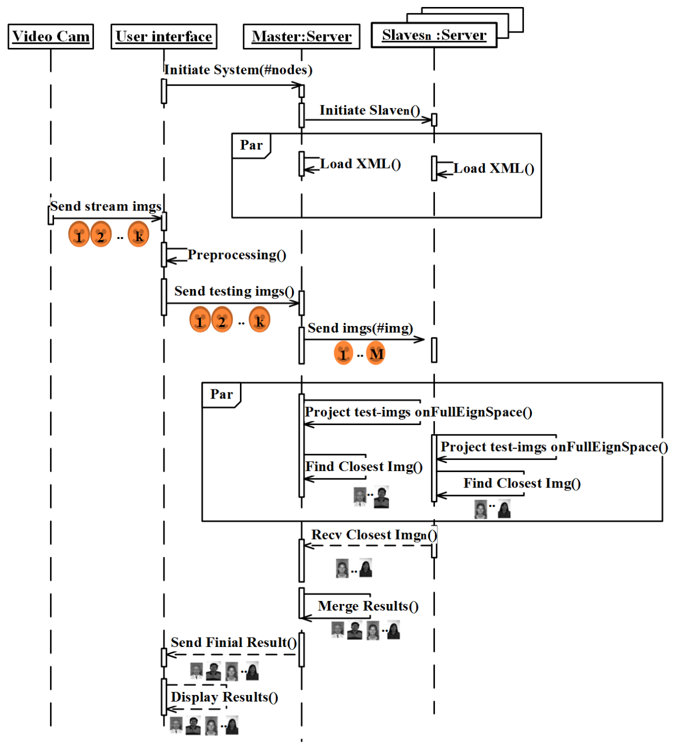

Fig. 3. Sequence diagram to recognize stream of testing images in training DB

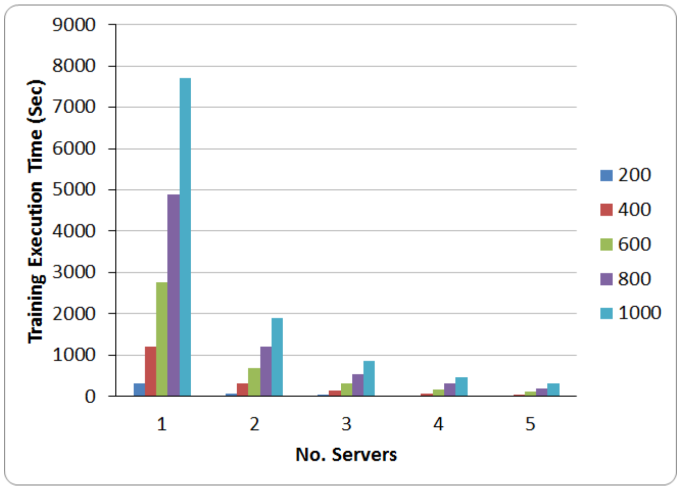

Fig. 4. The execution time for training different sizes of face DB

on the training set size, and the number of slaves. Each slave accesses its assigned text file and loads the associated images into its local storage. Training is done locally for each machine and all (P-1) eigenvectors are considered ignoring the zero eigenvector. The training results are stored as XML files on local storage. A sequence diagram shows this architecture in Fig.2 For each experiment, the training database is increased from 200, 400, 600, 800 to 1000 images. As a result, the XML file size of the training results is 1364024 , 2728934, 4095359, 5463706, and 6828681 KB, respectively. Consequently, storing and loading time of these XML files consume a lot of time. In addition, they need workaround to deal with that time which is out of scope now. The training results are shown in the Table I. The training time includes the time of calculating PCA eigenfaces and projecting all training faces into the eigenspace. In case of 200 training images, the sequential time is 303.766 seconds, and decreases significantly when the training database is distributed over 2, 3, 4 and 5 nodes respectively, as shown in Fig.4.

The maximum speed-up scored is 25X when the training database is distributed on five servers, achieving superlinear speed-up. The supperlinear speed-up is achieved because the large size of the XML does not fit in cache for sequential processing.

In the recognition phase, the master node sends copies of the testing image to the four slaves. Each slave tries to recognize the testing image against its local training set and sends the result back to the master. The master node selects the image having the least Euclidian distance among these results as shown in Fig. 2. The recognition time increases from 0.609 to 3.172 seconds when the training database size is increased from 200 to 1000 images in the sequential algorithm as shown in Table II. However, after distributing the databases on different numbers of servers, the recognition time decrease and the speed-up increases linearly to 5X when using five servers as shown in Fig. 5. The experimental results show that proposed architecture achieves accelerating ratio 5.208 instead of 4.133 in [13].

Table I. The Execution Time for Training of Different Training Dataset Size and Tested by 1 Image

DB Imgs	1 Server	2 Servers	3 Servers	4 Servers	5 Servers
200	303.766	75.703	33.968	18.891	12.14
400	1216.172	303.328	136.063	75.75	48.719
600	2749.71	683.672	303.718	170.75	109.203
800	4887.65	1216.313	541.751	303.39	194.047
1000	7713.812	1901.389	847.235	474.312	303.5

Table II. The Execution Time for Recognition Step of Different Training Dataset Size and Tested by 1 Image

DB Imgs	1 Server	2 Servers	3 Servers	4 Servers	5 Servers
200	0.609	0.313	0.204	0.156	0.125
400	1.203	0.61	0.407	0.297	0.234
600	1.828	0.921	0.594	0.453	0.359
800	2.734	1.203	0.812	0.594	0.485
1000	3.172	1.531	1.015	0.766	0.609

B. Duplicate Training Database and Distributed test Image

The training database is fixed and the training phase is done once at the master node. Therefore, the training time is ignored. In addition, copies of the resulting XML file are distributed to the four slaves as shown in Fig.3.

In the recognition phase, a number of test images are captured from video camera attached to the master node node which distributes them to the slaves in a manner similar to the one used in the previous section. As mentioned earlier, there is no communication overhead in the proposed architecture.

The recognition time decreases significantly when the test images are distributed over an increasing number of slaves. As shown in the Table III, in case of 500 test images when searching in training database have 500 images , the sequentially required 757.172 seconds and recognition time dropped to 151.313 seconds on five machines. The presented distributed approach is 5X faster than the sequential algorithm and the linear speed-up is reached as shown in Fig. 6.

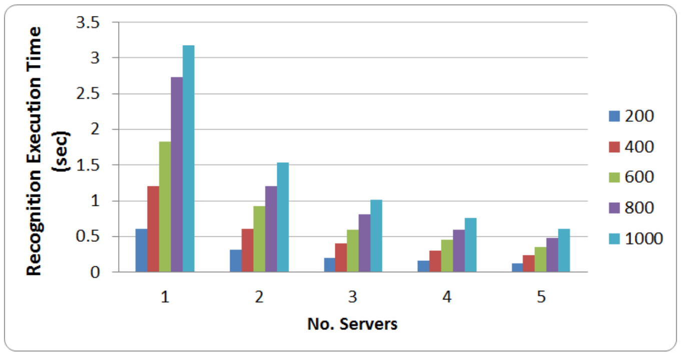

Fig. 5. The execution time for recognition one test image vs different sizes of face DB

Table III. The Execution Time for Recognition Step of 500 Images in Training Database and Different Number of Tested Images

No.Testing Imgs	1 Server	2 Servers	3 Servers	4 Servers	5 Servers
100	151.875	75.954	51.578	37.969	30.282
200	302.547	151.891	101.36	75.64	60.562
300	455.172	227	151.297	113.469	90.781
400	605.297	302.578	202.766	151.282	121.032
500	757.172	378.203	252.765	189.125	151.313

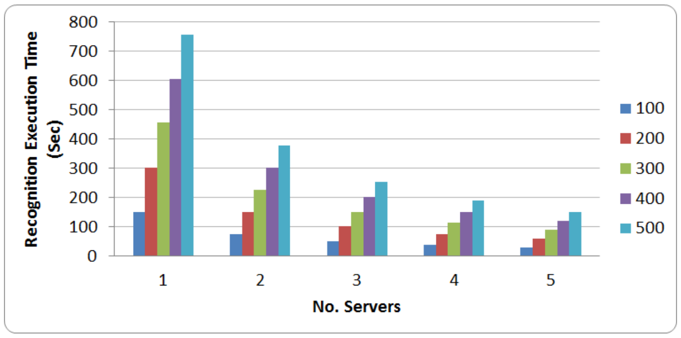

Fig. 6. The execution time For recognition step of 500 image in training database and different number of tested images

VI. Conclusion and Future Work

In this paper, two MPI-based approaches are proposed to handle different scenarios in face recognition systems. Distributing the training set and duplicating the test image is most likely the best solution when the stored training images are updated regularly and there is only input test at a time. On the other hand, having a large number of test faces or processing a video stream for recognition, is best handled by centralizing the training set and distributing the test images. The proposed systems improve execution time up to 25X in training and 5X in recognition phase, reaching superlinear and linear speed-up. Experiments considered all P-1 eigenvectors in the PCA algorithm. However, taking a fewer number of eigenvectors is expected to improve the expected execution time even further.

Future work includes studying the nonlinear methods for feature extraction and enhancing their performance. We will try to design robust algorithms for face recognition with occlusions in unconstrained environment. In addition, using the benefits of shared memory architecture and parallel distributed memory architecture to get better performance.

References

[1] C. Liu, “Gabor-based kernel PCA with fractional power polynomial models for face recognition,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 5, pp. 572–581, 2004.

[2] C. Ordonez, N. Mohanam, and C. Garcia-Alvarado, “PCA for large data sets with parallel data summarization,” *Distributed and Parallel Databases*, vol. 32, no. 3, pp. 377–403, 2014.

[3] R. Gross, I. Matthews, and S. Baker, “Appearance-based face recognition and light-fields,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 4, pp. 449–465, 2004.

[4] J. Yang, D. Zhang, A. F. Frangi, and J.-y. Yang, “Two-dimensional PCA: a new approach to appearance-based face representation and recognition,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 1, pp. 131–137, 2004.

[5] L. Xianwei and Z. Haiyang, “A survey of face recognition methods,” *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013)*, 2013.

[6] M.-H. Yang, “Kernel eigenfaces vs. kernel fisherfaces: Face recognition using kernel methods.” *Fgr*, vol. 2, 2002.

[7] G. Girish, “A comparative study on face recognition using MBLBP and PCA,” Ph.D. dissertation, VISVESVARAYA TECHNOLOGICAL UNIVERSITY, 2014.

[8] N. Goyal and H. Dev, “A comparative study of face recognition techniques: Asurvey.” *International Journal of Advanced Research in Computer Science*, vol. 5, no. 8, 2014.

[9] K. Rujirakul, C. So-In, and B. Arnonkijpanich, "PEM-PCA: A parallel expectation-maximization PCA face recognition architecture," *The Scientific World Journal*, vol. 2014, no. 4, pp. 47–58, 2014.

[10] J. Liu, S. Chen, and Z.-H. Zhou, "Progressive principal component analysis," *International Symposium on Neural Networks*, pp. 768–773, 2004.

[11] K. Meng, G.-D. Su, C.-C. Li, B. Fu, and J. Zhou, "A high performance face recognition system based on a huge face database," *International Conference on Machine Learning and Cybernetics*, vol. 8, pp. 5159–5164, 2005.

[12] C. Li, G. Su, K. Meng, and J. Zhou, "Technology evaluations on the TH-FACE recognition system," *International Conference on Biometrics*, pp. 589–597, 2006.

[13] J. Chunhong, S. Guangda, and L. Xiaodong, "A distributed parallel system for face recognition," *Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003. Proceedings of the Fourth International Conference*, pp. 797–800, 2003.

[14] G. Dashore and D. V. C. Raj, "An efficient method for face recognition using principal component analysis (PCA)," *International Journal of Advanced Technology & Engineering Research (IJATER)*, vol. 2, no. 2, 2012.

[15] N. H. Tuan and T. T. N. Huong, "Distance metrics for face recognition by 2d PCA," *PROCEEDING of Publishing House for Science and Technology*, 2016.

[16] M. L. Toure and Z. Beiji, "Intelligent sensor for image control point of eigenface for face recognition," *Signal Processing Systems (ICSPS), 2010 2nd International Conference*, vol. 2, pp. V2–769, 2010.

[17] I. Foster and N. T. Karonis, "A grid-enabled MPI: Message passing in heterogeneous distributed computing systems," *Proceedings of the 1998 ACM/IEEE conference on Supercomputing*, pp. 1–11, 1998.

[18] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A high-performance, portable implementation of the MPI message passing interface standard," *Parallel computing*, vol. 22, no. 6, pp. 789–828, 1996.

[19] "MPICH2: A new start for MPI implementations."

[20] P. J. Phillips, S. Z. Der, P. J. Rauss, and O. Z. Der, *FERET (face recognition technology) recognition algorithm development and test results*. Army Research Laboratory Adelphi, MD, 1996.

[21] P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss, "The FERET evaluation methodology for face-recognition algorithms," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 10, pp. 1090–1104, 2000.