# Hybrid MPI/OpenMP Implementation of PCA

Dalia Shouman Ibrahim
Salma Hamdy



Ibrahim, D.S. & Hamdy, S. 2017. Hybrid MPI/OpenMP Implementation of PCA. In: *Proceedings of the 2017 IEEE Eighth International Conference on Intelligent Computing and Information Systems (ICICIS 2017).* IEEE , Cairo, Egypt, pp. 205-211, 8th International Conference on Intelligent Computing and Information Systems, Cairo, Egypt, 5/12/17.
DOI: 10.1109/INTELCIS.2017.8260048

# *Hybrid MPI/OpenMP Implementation of PCA*

Dalia Shouman Ibrahim
Computer Systems Department
Computer and Information Sciences
Ain Shams University
Cairo, Egypt
Email: daliashouman@cis.asu.edu.eg

Salma Hamdy
Computer Science Department
Computer and Information Sciences
Ain Shams University
Cairo, Egypt
Email: s.hamdy@cis.asu.edu.eg

***Abstract*—Most surveillance systems depend on fully automated face recognition applications. The main concern is achieving high accuracy in real time. Principle Component Analysis algorithm is used for reducing the number of variables and getting the maximum variance between low dimensional data.**

**The proposed approaches significantly reduce the algorithm complexity when implemented over a cluster with parallel computing architecture. The first approach achieves 2975X and 102X faster than the sequential implementation in the training and recognition phases, respectively. However, the second approach achieves 74X faster than the sequential implementation in the recognition phase.**

***Keywords—Face Recognition; PCA; hybrid MPI/OpenMP; Parallel Programming; hybrid memory architecture;***

## I.INTRODUCTION

Face recognition is a significant application used in many important areas. Specifically, in security and biometrics, the decision often should be highly accurate and fast. Because of the importance of recognizing human faces in real time, many researchers put the effort into speeding-up the recognition process. This imposes a problem since face recognition is usually computationally expensive, and involves large amounts of data and features to process. One way of getting around this is feature reduction, which obviously presents a trade-off for accuracy. Another way to enhance speed is parallel architectures. In a multi-core system solution, this is usually done by either parallelizing the recognition algorithm, or distributing the datasets, or both. One of the traditional face recognition algorithms that tend to exploit feature reduction is the Principal Component Analysis (PCA) [1]. Calculating the covariance matrix to extract the eigenvectors and eigenvalues is computationally challenging and consumes a lot of time; slowing down the entire system [2].

Hence, a large part of the literature is dedicated for developing algorithms and architectures to improve PCA speed by exploiting parallelism. In addition, open source APIs exist to support high-performance computing and parallel programming paradigms like Message Passing Interface (MPI) [3] and Open Multi-programming (OpenMP) [4]. MPI can be used for communication between nodes and execute different tasks on different machines simultaneously [5], while OpenMP supports multi-platform shared-memory multiprocessing programming. Moreover, hybrid MPI/OpenMP architectures reduce memory consumption and communication between different nodes. These architectures can distribute tasks and data between cores within a node, and over the nodes inside the cluster [5].

In this paper, a hybrid MPI/OpenMP paradigm is used to speed-up the sequential and earlier published MPI implementations of PCA [6]. Two architectures are presented to parallelize traditional face recognition PCA in two different situations to achieve higher speed-up. The proposed architectures are experimentally evaluated on the Face Recognition Technology (FERET) database [7].

The rest of this paper is organized as follows. Related work and other solutions to speeding-up PCA are reviewed in section II. Section III introduces common parallel programming architectures. The two proposed architectures for improving PCA for face recognition are introduced in section IV. Section V presents the experimental results and evaluations, while conclusions and future work are presented in section VI.

## II.RELATED WORK

The architecture in [8] used the OpenMP library to develop parallel implementations of some selected face recognition algorithms including PCA. The execution was carried on an Intel dual core processor, so the parallelism was done on four concurrent working cores. The AT&T [9] database was used to evaluate the proposed solutions. The training dataset contained forty images, while ten to hundred images were used for testing. In addition, the images were scaled to 100×100 pixels. Results show the parallelized implementation being able to recognize up to 100 images in 610 milliseconds as opposed to 722 milliseconds in the sequential implementation.

In our previous work [6], we presented a distributed system over five working nodes. The MPI library was used to implement parallel approaches to handle two scenarios: when the training dataset requires frequent updates; hence training is distributed and performed locally at each node, and when the training database is somehow fixed with several testing images fed to the recognition system. The first architecture achieves superlinear speed-up, reaches up to 25X in the training phase, whereas in both scenarios, the speed-up increases linearly to 5X in the recognition phase.

Another approach presented Modular Principal Component Analysis (MPCA) [10], where the image is partitioned into regions of the same size and traditional PCA is used on each region. The image division improves the recognition rate with large variations of head pose, light, and facial expressions. This reduces the computational effort since all calculations were done on smaller matrices compared to traditional PCA.

In addition, a two-dimensional PCA approach (2DPCA or IMPCA) was presented [11] which did not require transforming each image into a one-dimensional vector. The size of the covariance matrix was much smaller when constructed directly from the original image matrices.

Modular Image PCA (MIMPCA) [12] combined the two techniques; MPCA and IMPCA, to achieve lower recognition time under different conditions. Authors made an evaluation of the proposed technique against MPCA and IMPCA techniques. The MIMPCA took 19 seconds, while MPCA and IMPCA required 443 seconds and 34 seconds respectively.

Whereas weighted Modular Image PCA (wMIMPCA) [13] minimized the effects of illumination, rotation, and head pose in automatic face recognition systems, by giving weights to each region. Authors evaluated the feature extraction techniques (MPCA , IMPCA, MIMPCA, and wMIMPCA) using three different databases (Yale, ORL, and Sheffield) under the same conditions and changing the image partition sizes. The partition size 3 × 3 achieves the best performance independent of the database. Using this partitioning size, if the same task of feature extraction procedure is performed on one face, the wMIMPCA required s seconds while MPCA took 1.5 s, MIMPCA took 0.84 s and MPCA is three times slower than wMIMPCA.

## III.PARALLEL PROGRAMMING ARCHITECTURES

Most high-performance computing systems provide a hierarchical hardware design; either based on shared memory with multi-core CPUs, or distributed memory between CPUs connected over a network [5]. This opens the opportunity to exploit the resources in the hybrid paradigm to combine shared memory and distributed memory architectures. Such model reduces communication needs and memory consumption [5].

***Distributed Memory Architecture:*** every processing node has its own local memory and synchronization is achieved by exchanging messages between each other. The master node initiates and finalizes the communication. The important issue here is data decomposition, that is, how to distribute data between working nodes while minimizing the communication overhead. Distributed memory systems can be categorized into two main models. In the *massage-passing* model, data packets are sent and received between working nodes, and communication is controlled by the application program through message-passing libraries (MPI, PVM, etc.). In *distributed-shared memory* model, the memories are physically separated, although it logically appears as one shared address space to the programmer. Consequently, the distributed memory spaces are manipulated as a single shared resource [3].

***Shared Memory Architecture:*** in this architecture, the same memory is shared and accessible to multiple

processors, and synchronization is done by controlling the reading and writing from every processor. In addition, each working node has its own local memory. All processors work concurrently to solve the task. The most commonly used library for this type of memory architecture is OpenMP [4].

***Hybrid Architecture:*** this architecture combines the benefits of the distributed and shared memory models. The code and data are distributed over the working nodes using the MPI library. Moreover, within each working node, tasks are distributed over all cores and run simultaneously using the OpenMP library. The advanced hyperplane scheme is used in hybrid models to set synchronization between threads inside processing nodes and the communications between working nodes inside the cluster [14].

## IV.PROPOSED ARCHITECTURE

As mentioned so far, face recognition algorithms are usually of high complexity and the ultimate goal is faster implementation. In this work, two scenarios for face recognition are re-implemented using hybrid memory architecture and compared against the distributed memory architecture from [6]. In a hybrid PCA model, all cores within all connected servers can be exploited to accelerate the training and recognition phases.

### *A. One Test Image*

The first approach deals with a frequently changed database. Consequently, the training phase is performed a lot. By using the multicore system and hyper threading, the training database can be distributed over the network between nodes by giving small training chunk over its cores of the connected nodes.

This experiment hosts one master node and four working slaves. The master node distributes the training database evenly over the working nodes, including itself. Every node initiates its concurrent threads and individually starts PCA training on its sub-database. When the master node broadcasts the normalized test image, which is received from a monitoring camera connected to a user interface system (Fig. 1), every thread processes this image against its local sub-database. The working nodes collect the results from their cores and select the best match among them. Local results are sent back to the master node, as shown by the dashed lines in (Fig. 1). The master node searches through them, including its own, for the closest image based on their distances. Finally, the master sends the final results to the user interface for displaying. The closest results are determined by the shortest distance between the projected test image and projected training images.

### *B. Multiple Test Images*

The second approach is used when the monitoring system is attached to a video camera, for example, surveillance systems in parking, hotels, or airports. The camera captures the video and sends it the connected computer. The video is split into frames and the detected faces extracted from that frames. The master node distributes P test images evenly over the n nodes according to (1). Furthermore, the master node is involved in the searching process and it takes a chunk of testing images. The beginning and end of the chunk are determined by (2) and (3) respectively. The searching in each node is done against the whole database which is pre-trained and stored in the local storage as an XML file. As a result, all working nodes' outputs considered to be final results. The master node collects the results and sends them as they are to the user interface for display, as shown in the dashed line in (Fig. 2).

$$H = \frac{F}{N} \quad (1)$$

Where:
H is the number of images per node.
F is the number of detected faces in all video frames.
N is the number of working nodes.

$$ChunkBeginning = (N_{i-1}H) + 1 \quad (2)$$

$$ChunkEnding = N_iH \quad (3)$$

Where
$N_{i-1}$ is the id of the previous working node.
$N_i$ is the id of the working node.

## V.EXPERIMENTAL RESULTS

The Facial Recognition Technology (FERET) database [7, 15] is used in our experiments. The color

PPM images are in 32-bit floating point number, and all are of the same size (512×768).

The experiments were conducted on a cluster in the high-performance laboratory of Faculty of Computer and Information Science, Ain Shams University, Egypt. The cluster has twelve servers connected over an infinite band network. The experiments use five servers, each having two Quad-Core Intel® Xeon® Processor E5520 @ 2.27GHz with hyper-threading enabled. Thus, each core can run two threads and a node can reach up to sixteen concurrent threads. In addition, these servers are equipped with 24 GB of RAM. VMware ESXi has installed on the servers and the Vsphere client post the jobs to this cluster.

### *A. One Test Image*

In the first scenario, the execution time was calculated for the training phase against different sizes of the database when the PCA algorithm was run several times. It is noted that execution time decreases when the number of servers is increased. In addition, when each is re-distributing its partition over its cores, the execution time is significantly decreased. For instance, when running the training phase with 1000 images on two servers with two concurrent threads, the execution time was 474.391 seconds, as shown in (Fig. 3). However, it took 21.406 seconds on two servers with sixteen cores making use of hyper-threading. Furthermore, the training phase took only 2.593 seconds on five running servers with sixteen concurrent threads.

As shown in (Fig. 4), the proposed architecture achieves super linear speed-up. It reaches up to 2975X relative to one core. In addition, by comparing these results to our previous architecture in [6], the hybrid programming model MPI/OpenMP achieves the higher speed-up. As shown in (Fig. 5), pure-MPI took from 12.14 to 303.5 seconds when the training image database had 200 to 1000 images, respectively. However, in the hybrid MPI/OpenMP implementation using five servers each with sixteen cores; the training process took only from 0.109 to 2.593 seconds, respectively, for the same number of training images.

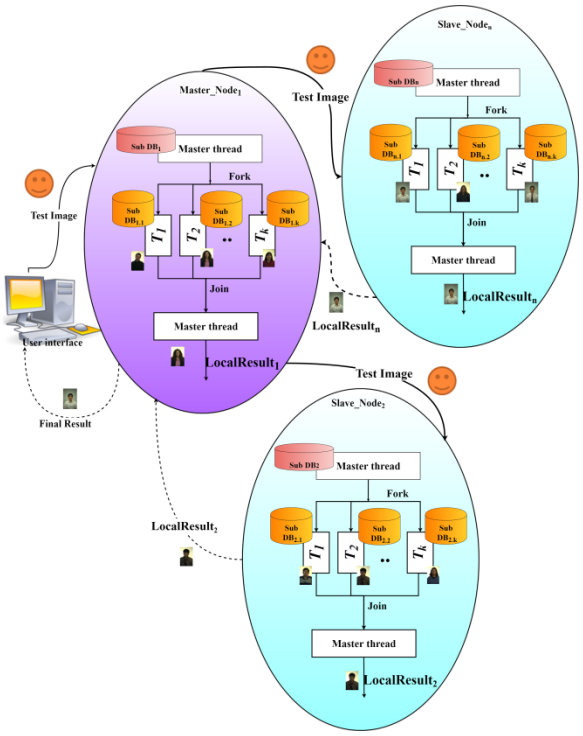


Fig. 1 The execution model of broadcasting the captured test image over the working nodes.

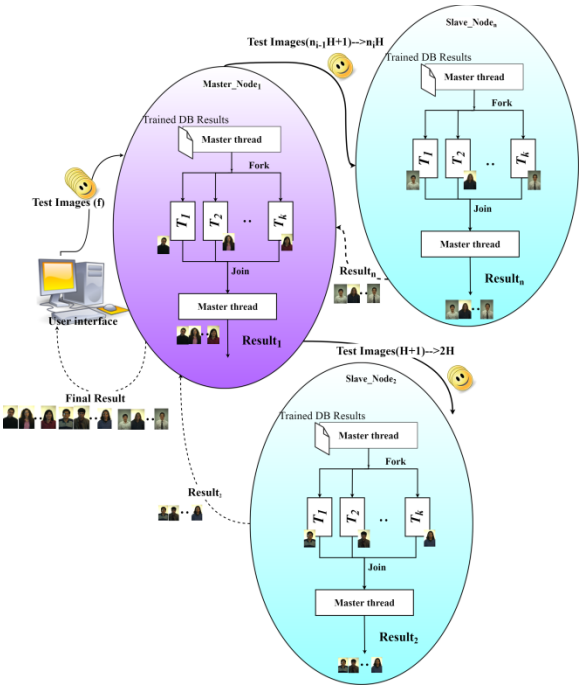


Fig. 2 The execution model of distributing multiple test images extracted from video camera over working nodes.

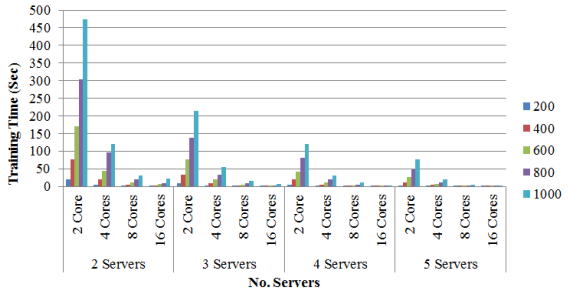


Fig. 3 The execution time for training different sizes of face databases.

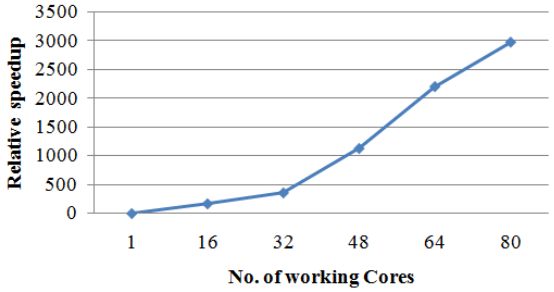


Fig. 4 Relative speed-up for training phase of sub-databases.

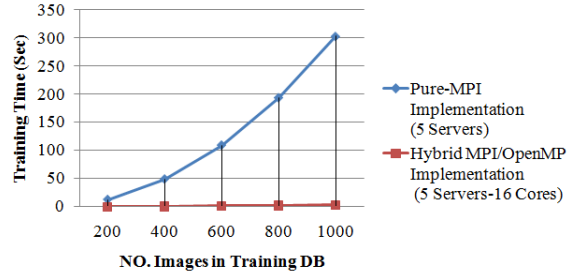


Fig. 5 Comparison between pure-MPI and hybrid MPI/OpenMP in training sub-database process.

On the other hand, (Fig. 6) shows the significant performance improvement of the recognition phase. It only required 0.094 seconds on two servers with sixteen concurrent threads, while on five servers with sixteen threads, it took 0.031 seconds. The relative speed-up reaches up to 102X, as shown in (Fig. 7), when running the hybrid implementation on eighty cores represented in five servers, with sixteen working cores. In addition, it is noted that this approach outperforms the one from [6] since searching is faster when performed on a smaller sub-database on concurrent threads (Fig. 8). In pure-MPI, for recognizing one test image against 1000 images, it took 0.609 seconds versus 0.031 seconds in hybrid MPI/OpenMP implementation.

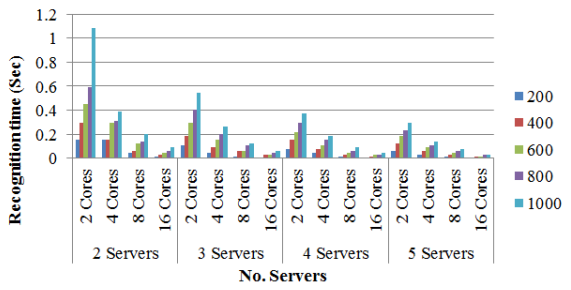


Fig. 6 The execution time for recognition phase with different sizes of the database against one test image.

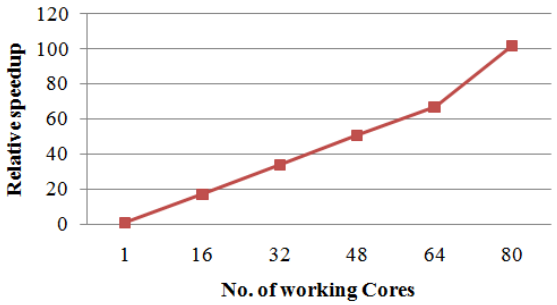


Fig. 7 Relative speed-up for recognizing test image against sub-databases.

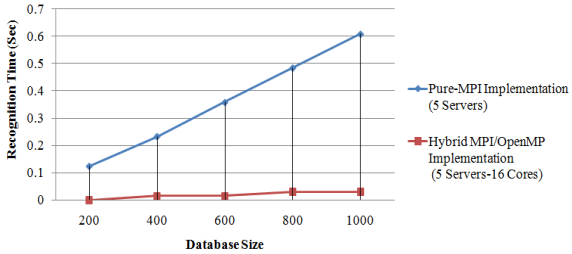


Fig. 8 Comparison between pure-MPI and hybrid MPI/OpenMP in recognition one test image against sub-databases.

### *B. Multiple Test Images*

In the second scenario, the pre-training database had 500 images and the suggested approach was tested by running the recognition phase several times with a different number of normalized testing images. As shown in (Fig. 9), when recognizing 200 images against the pre-trained database using two working nodes each of which distributing the testing images over two cores, the execution time was 75.578 seconds in total. However, when the number of testing images is increased to 500, the execution time was 188.984 seconds and the speed-up reaches up to 74X relative to the sequential implementation (Fig. 10).

On the other hand, comparing the hybrid implementation against pure-MPI [6] when using five working nodes with sixteen concurrent threads to recognize 500 images, the execution time significantly dropped to only 10.21 seconds instead of 151.313 seconds (Fig. 11).

## VI. CONCLUSION

Face recognition is used in surveillance systems applications which require accurate and fast results. PCA gives promising results under constrained environments. This paper presents two hybrid MPI/OpenMP implementations for the PCA algorithm; handling one input image or multiple input images at the same time.

In the first scenario, the hybrid model allows increasing the size of training database images, and any updates in the database and feedbacks could be handled easily by distributing the database over working nodes and cores. This approach achieves 2975X and 102X superlinear relative speed-up for training and recognition processes, respectively.

In the second scenario, the system deals with video streaming and distributes testing images over cores, and recognizes them against the whole pre-trained local databases. The relative speed-up for recognition process reaches up to 74X. A comparison between the results shows that the hybrid MPI/OpenMP outperforms the early published pure-MPI implementations [6] in both scenarios.

Future work includes studying and deploying the face recognition system using PCA algorithm on heterogeneous systems such as GPU (Graphics Processing Units) and making a system performance comparison between the proposed architectures and heterogeneous system.

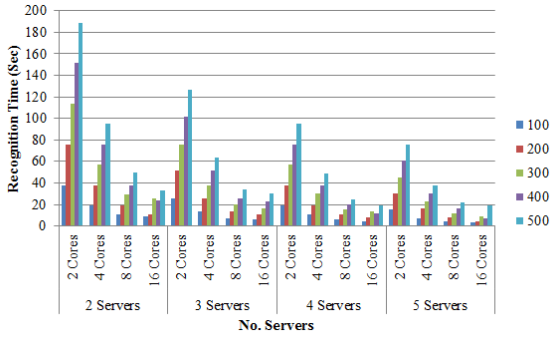


Fig. 9 The execution time for recognition phase of a different number of test images against fixed training database.

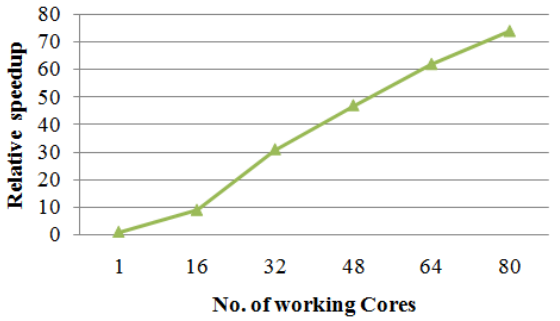


Fig. 10 Relative speed-up for recognizing 500 test images against the whole pre-trained database.

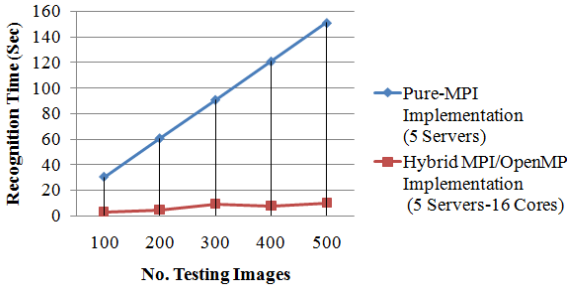


Fig 11 Comparison between pure-MPI and hybrid MPI/OpenMP in recognition 500 testing images against the pre-trained database.

## REFERENCES


[1] M. Abdullah, M. Wazzan, and S. Bo-Saeed, "Optimizing face recognition using PCA," in International Journal of Artificial Intelligence & Applications (IJAIA), vol. 3, no. 2, 2012.

[2] J. Liu, S. Chen, and Z.-H.Zhou, "Progressive Principal Component Analysis," in International Symposium on Neural Networks. Springer, 2004, pp. 768–773.

[3] D. V. Krishna and P. Sammulal, "Advances in parallel computing from the past to the future," in International Journal of Advance Research in Computer Science and Management Studies, vol. 1, no. 4, 2013, pp. 16–23.

[4] R. Chandra, R. Menon, L. Dagum, D. Kohr, D. Maydan, and J. McDonald, "Parallel Programming in OpenMP." Morgan kaufmann, 2000.

[5] R. Rabenseifner, G. Hager, and G. Jost, "Hybrid MPI/OpenMP parallel programming on clusters of multicore SMP nodes," in a 17th Euromicro international conference on parallel, distributed and network-based processing. IEEE, 2009, pp. 427–436.

[6] D. Ibrahim and S. Hamdy, "Parallel architecture for face recognition using MPI," in (IJACSA) International Journal of Advanced Computer Science and Applications, vol. 8, no. 1, 2017, pp. 425–430.

[7] 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.

[8] J. Suryaprasad, D. Sandesh, I. Priyanka, G. Pravalika, and A. Kumar, "Parallel implementation and performance evaluation of facial recognition algorithms using open source technologies," in Electrical, Electronics, and Optimization Techniques (ICEEOT), International Conference on. IEEE, 2016, pp. 177–182.

[9] B. Dai, D. Zhang, H. Liu, S. Sun, and K. Li, "Evaluation of face recognition techniques", in International Conference on Photonics and Image in Agriculture Engineering (PIAGENG). Proc. of SPIE, vol. 7489, 2009, pp. 74890M-1-74890M-7.

[10] R. Gottumukkal and V. K. Asari, "An improved face recognition technique based on modular PCA approach," in Pattern Recognition Letters, vol. 25, no. 4. Elsevier, 2004, pp. 429–436.

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

[12] J. F. Pereira, G. D. Cavalcanti, and T. I. Ren, "Modular image principal component analysis for face recognition," in International Joint Conference on Neural Networks (IJCNN). IEEE, 2009, pp. 2481–2486.

[13] George D.C. Cavalcanti, Tsang Ing Ren, and José Francisco Pereira, "Weighted modular image principal component analysis for face recognition," in Expert Systems with Applications, vol. 40, no. 12. Elsevier, 2013, pp. 4971–4977.

[14] N. Drosinos and N. Koziris, "Performance comparison of pure MPI vs hybrid MPI-OpenMP parallelization models on SMP clusters," in 18th International Parallel and Distributed Processing Symposium, IEEE, 2004, p. 15.

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