# An Architecture For Real Time Face Recognition Using WMPCA

A. Pavan Kumar V. Kamakoti Sukhendu Das
Department of Computer Science & Engineering
Indian Institute of Technology Madras, Chennai - 600036
{apavan@cse., kama@, sdas@}iitm.ernet.in

## Abstract

*An architecture for real time face recognition using weighted modular principle component analysis (WMPCA) is presented in this paper. The WMPCA methodology splits the test face horizontally into sub-regions and analyzes each sub-region separately using PCA. The final decision is taken based on a weighted sum of the errors obtained from each region. This is based on assumption that different regions in a face vary at different rates with variations in expression and illumination. The WMPCA methodology has a better recognition rate, when compared with conventional PCA, for faces with large variations in expression and illumination. This methodology has a wide scope for parallelism. An architecture which exploits this parallelism is proposed in this paper. We also present a* `System On Programmable Chip` *(SOPC) implementation of face recognition system using this architecture.*

## 1. Introduction

Real time face recognition is an increasingly important area of research today. Its applications are becoming more important, as in ATM machines, criminal identification, access restriction, video conferencing, issuing drivers' license, passports and monitoring public areas for known faces. The task of automated face recognition is very difficult due to two orthogonal reasons:

- Inter-person similarity of faces in general.
- Intra-person variance of faces due to variations in expression, pose and illumination.

Automated face recognition (AFR), being a fundamental problem in computer vision and pattern analysis, is being addressed by several scientists from different areas. Various algorithms have been proposed for the automatic face recognition in last few decades, with varying degrees of success. Rama Chellappa et al. [15] gave a detailed survey of face recognition algorithms based on neural network models, statistical models, and feature-based models. Majority of the contributions are based on *Principle Component Analysis* (PCA) [12], *Linear Discriminant Analysis* (LDA) [9] and *Support Vector Machine* (SVM) [1] techniques. Many variations and improvements have been proposed for these basic methods. Weighted Modular PCA (WMPCA) [8] is one such improvement proposed over PCA. Most of the AFR algorithms evaluate faces as one unit which leads to problems due to variations in expression, illumination and pose. This neglects the important fact that few facial features are expression invariant and others are more susceptible to the expressions.

In weighted modular PCA, different parts of face (eyes, nose, lips) are separately analyzed and the final decision is based on the weighted sum of errors obtained from separate modules. These weights are based on the extent to which each sub-region of a subject is spread in the eigenspace. The weights are the measures of intra-subject variance of the sub-region.

Vision algorithms are classified into high, middle and low-level based on the computation and communication characters of the algorithms [13]. Ratha and Jain [7] use a custom computing approach to meet the computation and communication needs of vision algorithms at all levels. In this approach, hardware architecture is customized at instruction level for every application. This approach can also reuse the same hardware by reconfiguring at software level for different applications at different levels.

Most AFR techniques require large memory and have high computational cost. Due to these characteristics appropriate hardware accelerators are used for real-time applications. The performance is also improved by tailoring the architecture for the application in hand. Examples of such architectures are NETRA [2] developed at University of Illinois, Image Understanding Architecture [14] developed at University of Massachusetts and VisTA [11] developed at University of Texas at Austin.

Face recognition algorithms deal with raw uncompressed images, which leads to millions of operations. We need to

have specialized architectures for real time implementation of these algorithms. Weighted modular PCA modularizes the process of recognition giving room for parallel processing. We present an architecture that exploits the parallel nature of weighted modular PCA, to implement real time face recognition.

This paper is organized as follows: Section 2 gives an overview of WMPCA. Section 3 describes the architectural features. In Section 4, we discuss the design and implementation of a SOPC for face recognition using WMPCA architecture. Finally Section 5 gives the conclusions and future scope of work.

## 2. Weighted modular PCA

Weighted modular PCA modularizes the face into sub-regions and performs recognition on each sub-region individually. This assumes that there are n-tasks of recognition, one for each sub-region and all may be done in parallel. Net error is calculated as a weighted sum of errors in each sub-region, where these weights are obtained based on variations of each feature across various expressions and illuminations. This follows the hypothesis that different regions vary at different rates across various expressions and illuminations.

Each face is horizontally split into a set of sub-regions such as forehead, eye, nose and chin. For each sub-region, $\rho$, of each face, we now compute average sub-region, calculate covariance matrix, eigenvectors and the weight set as in PCA. All these computations can be implemented in parallel. Finally net error is obtained as a weighted sum of the error vectors of individual sub-regions. The given face is classified as to belong to that class which is at nearest euclidean distance in face space.

### 2.1. Training

Let the training set contain $L$ subjects, where each subject is one person. Each person has $N$ different faces. So the training set has $M = LN$ faces. All $M$ faces are divided in to $R$ regions. Hence, each $r^{th}$ partition of $n^{th}$ sample of $l^{th}$ subject is, $\rho_{l,n,r}$, where $l = 1, 2, ..., L;\ n = 1, 2, ..., N;\ r = 1, 2, ..., R$. Thus entire training set can be represented as $T_{set} = \{\ \rho_{l,n,r} \mid \forall l, n, r\}$. The following steps are repeated for each sub-region $r = 1, 2, ..., R$. For each $r^{th}$ sub-region, an average sub-region, $\psi_r$ is computed over all faces.

$$\psi_r = \frac{1}{LN} \sum_{l=1}^{L} \sum_{n=1}^{N} \rho_{l,n,r}$$

This equation can be conveniently rewritten as,

$$\psi_r = \frac{1}{M} \sum_{m=1}^{M} (\Upsilon_m)_r$$

where $r = 1, 2, ..., R,\ M = LN$, and $(\Upsilon_m)_r$ is the $r^{th}$ sub-region of $m^{th}$ face.

The covariance matrix $C_r$ of $r^{th}$ sub-region is calculated as,

$$C_r = \sum_{i=1}^{M} [(\Upsilon_m)_r - \psi_r][(\Upsilon_m)_r - \psi_r]^T$$

where $r = 1, 2, ..., R$. The eigenvectors of this matrix are computed and the most significant $S$ eigenvectors, $((V_1)_r\ , ...,\ (V_S)_r)$, are considered for each sub-region $r$ as mentioned in section 2. Then each sub-region $r$ of face $m$ can be expressed as a linear combination of these eigenvectors.

$$(\Upsilon_m)_r = \psi_r + \sum_{s=1}^{S} (w_{m,s})_r (V_s)_r$$

where $(w_{m,s})_r$ is the weight of $r^{th}$ sub-region of $m^{th}$ face and is calculated as,

$$(w_{m,s})_r = (V_s)_r^T ((\Upsilon_m)_r - \psi_r) \tag{1}$$

where $m = 1, 2, ..., M,\ s = 1, 2, ..., S$, and $r = 1, 2, ..., R$. Similarly weight vector of each sub-region $(\Omega_m)_r$ is generated from these weights as,

$$(\Omega_m)_r = [\ (w_{m,1})_r\ (w_{m,2})_r\ ...\ (w_{m,S})_r\ ] \tag{2}$$

where $m = 1, 2, ..., M$ and $r = 1, 2, ..., R$.

### 2.2. Intra-subject variance of each sub-region

As mentioned, the final decision is based on the weighted sum of error vectors obtained from each sub-region. These weights represent a measure of *the extent of variation in eigenspace for a sub-region of a subject across all samples.* For each sub-region $r$ of each subject, $l$, with $N$ faces, calculate average $(\Phi_l)_r$ as,

$$(\Phi_l)_r = \frac{1}{N} \sum_{n=N*(l-1)+1}^{l*N} (\Omega_n)_r$$

where $r = 1, 2, ..., R$ and $l = 1, 2, ..., L$. For each sub-region $r$, the measure of variance for $l^{th}$ subject is,

$$(P_l)_r = \frac{1}{N} \sum_{n=N*(l-1)+1}^{l*N} [(\Omega_n)_r - (\Phi_l)_r]^2 \tag{3}$$

where $l = 1, 2, ..., L$ and $r = 1, 2, ..., R$. It may be noted that more compact sub-regions have lesser value of $(P_l)_r$ .

### 2.3. Recognition

The training phase, consists of obtaining (i) Weight vector of each sub-region $r$ of each face $m$ in the training set: $(\Omega_m)_r$, as in eqn.(2), and (ii) measure of variance of each sub-region $r$ for each subject $l$: $(P_l)_r$, as in eqn.(3). Now given a test face, $\Gamma_{test}$, it is split into $R$ horizontal sub-regions as in the training phase. These regions can be represented as $(\Upsilon_{test})_r$ where $r = 1, 2, ..., R$. These regions are then projected onto face space and weights are calculated as in eqn.(1), $(w_{test,s})_r = (V_s)_r^T((\Upsilon_t est)_r - \psi_r)$. The corresponding weight vector is built as,

$$(\Omega_{test})_r = [(w_{test,1})_r \ (w_{test,2})_r \ \ldots \ (w_{test,S})_r]$$

where $r = 1, 2, ..., R$.

The error vector for a region $r$ which is the euclidean distance between $(\Omega_{test})_r$ and $(\Omega_m)_r$ is computed as,

$$(E_m)_r = (P_l)_r \sum_{i=1}^{S} [(w_{test,i})_r - (w_{m,i})_r]^2 \qquad (4)$$

where $m = 1, 2, ..., M,\ r = 1, 2, ..., R$ and $l$ is the subject of $m^{th}$ sample.

For each subject, the sub-region that is more invariant to expressions and illuminations is given more priority in the net error function. This is implemented by multiplying each error of the sub-region with the measure obtained in eqn.(3). The net error function for comparing a test face $\Gamma_{test}$ with $\Gamma_m$ is,

$$(F_{test})_m = \sum_{r=1}^{R} (E_m)_r \qquad (5)$$

where $m = 1, 2, \ldots, M$. The test face is said to have matched with face $m'$, for which $(F_{test})_{m'} = min(F_{test})_m, \forall m$. Suitable threshold is used to reduce false acceptance.

### 2.4. Performance

The algorithm was tested on *Yale database* [6]. The database had 11 samples each of 15 different subjects. We performed PCA on the actual samples and modular PCA, weighted modular PCA (WMPCA) on the partitioned set. The experiment was repeated with different number of eigenvectors. It was observed that, as number of eigenvectors increase the recognition rate also increases, but at the cost of computational complexity. It was observed that WMPCA is able to achieve higher rates of recognition than PCA at lower number of eigenvectors itself.

Using a training set of 6 samples per subject, and other 5 for testing, WMPCA achieved an accuracy of over 87% while PCA achieved only 76%. Using modular PCA, described in [10], the recognition rate reached only 80%. The recognition rate of WMPCA improved to 89%, if 7 samples of each subject were used for training. The figure 1 gives a plot of results obtained in the experiment with 7 samples per subject used as training set and other 4 samples used for testing.

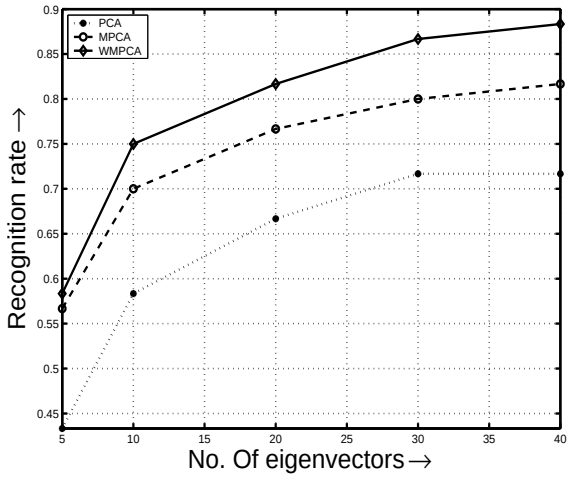


**Figure 1.** *Results of WMPCA compared to MPCA and PCA for different number of eigenvectors.*

## 3. Hardware Implementation

We propose an architecture that exploits the inherent parallelism of the WMPCA methodology. The WMPCA methodology has a two fold parallelism.

- Parallelism across the sub-regions.
- Parallelism with in each subregion across various eigenfeatures.

To exploit this two-fold parallelism and for computational simplicity,we bring out the following changes in the equations used in WMPCA. Considering eqn.(4) , we replace the square operation by the absolute value operator. The weight values of test region are computed as,

$$(E_m)_r = (P_l)_r \sum_{i=1}^{S} |V_i((\Gamma_{test})_r - \psi_r) - (w_m)_r|$$

where $m = 1, 2, ..., M,\ r = 1, 2, ..., R$ and $\psi_r$ is the average sub-region.

For any test region, the term $[(\Gamma_{test})_r - \psi_r]$ is a constant and computed only once per recognition,

$$(E_m)_r = (P_l)_r \sum_{i=1}^{S} |V_i((\Gamma_{diff})_r - (w_m)_r)| \qquad (6)$$

where $m = 1, 2, ..., M$ and $r = 1, 2, ..., R$. Thus, we need to calculate R errors, where R is the number of subregions used, and each error is a sum of S partial sums, where S is the number of eigenvectors.

### 3.1. Architecture

Figure 2 shows the proposed architecture with $r$ processing elements. Recall that r represents the number of subregions into which the face image is divided. The host sends the test face image to the cropping unit. The cropping unit divides the face image into sub-regions and place each of these sub-regions onto an R-BUS. As seen in figure 2, the R-bus in turn feeds the corresponding $PE_{PCA}$ processing elements with the sub-region. The function of the processing element $PE_{PCA}$ is to perform PCA on the sub-region and multiply the estimated error of the sub-region with the corresponding variance measure which is obtained from the training phase. Each $PE_{PCA}$ has its own local memory where the eigenvectors, intra-subject variance measures and weights of each sub-regions obtained in the training phase are stored. The $PE_{PCA}$ places the error of the input image with respect to each of those in the database on the O-BUS. The summing unit collects these errors from each O-BUS and generates the net error. The host sorts all the net errors and outputs the best of the matches, whose net error falls below the rejection threshold.

Each processing element has a local memory where the following data obtained during the off-line training phase is stored:

- Average of each sub-region.
- Eigenvectors of the covariance matrix obtained during the off-line training phase.
- Each sub-region used in the training phase is stored as a set of $S$ weights, $S$ being number of eigenvectors used.
- Intra-subject variance of the sub-region for each of the subjects.

### 3.2. The PCA processing element

Each PCA processing element is designed to implement the above modified equation (6). we employ $S$ parallel blocks to compute the partial sums of this equation and pass onto the *region summing unit*. Each of these parallel blocks multiply $\Gamma_{diff}$ with a eigenvector and generate corresponding weight. The difference between this weight and the corresponding weight of the sub-region in the database is computed. This difference is multiplied with corresponding intra-subject variance measure obtained from the database. The absolute value of this product is passed onto the *region summing unit*. Figure 3 shows the internal architecture of each of $PE_{PCA}$ with $S$ parallel blocks.

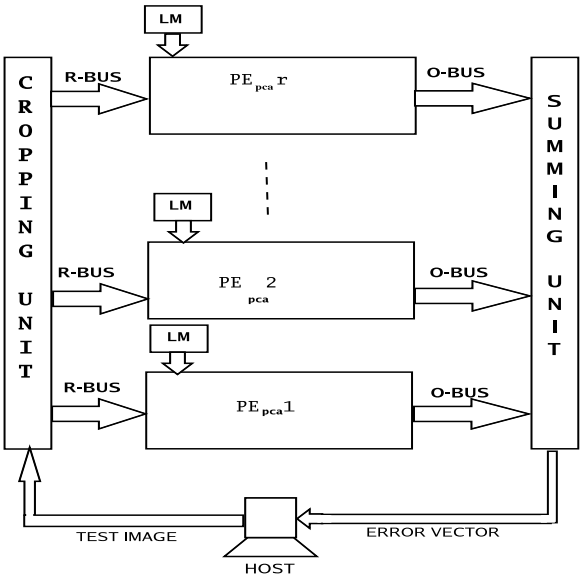


**Figure 2.** *The WMPCA architecture.*

The output of *region summing unit* is the error obtained in that region of the test image. The errors of each subregion are added together in the summing unit of the circuit. This is the net error of test face from the given face in the database. This is repeated for all the images in the database and net errors with respect to all the images in the database are obtained.

The host collects all these net errors, and sorts them in order, and displays the top matches, whose error is below the threshold value.

## 4. SOPC realization of the architecture

The architecture proposed in the previous section was realized as a *System On Programmable Chip*. The device used was ALTERA EP20K200EFC484-2X [3].

### 4.1. Design of the system

The system was designed using Quartus-II tool. The SOPC builder of Quartus-II was used to add following cores to the architecture.

- Nios-processor core [5].
- On-board memory of 2KB (Boot-ROM).
- Flash memory.
- 128MB SDRAM module through SODIMM controller.
- UART (RS-232 interface).

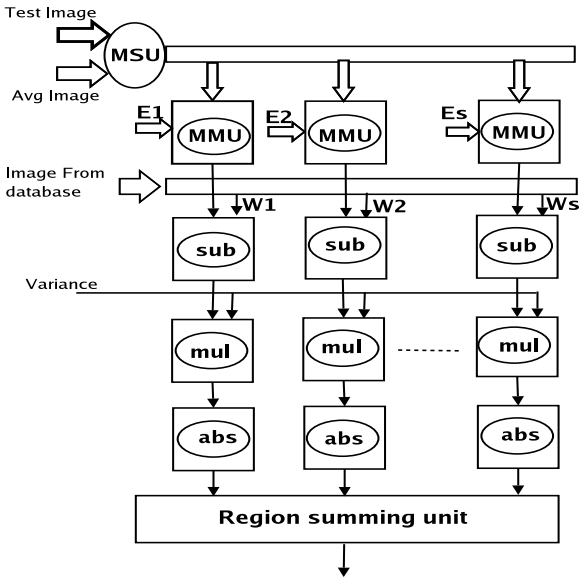


**Figure 3.** *The $PE_{PCA}$ processing element.*

- Custom logic of WMPCA.
- Parallel I/O interface to LCD display.
- Timer.

These components are connected by Avalon bus [4]. The Nios-processor core was used as the host in the architecture. The UART port of board was connected to the PC using RS-232 cable to transfer image files. The SDRAM was used to store test image, results and other intermediate data. A LCD panel was added to view results. Figure 4 shows a block diagram of interconnected components used in the system. The data obtained during off-line training is also stored as a part of custom-logic of the system.

The processor reads the test image from UART-port and stores it in the SDRAM. Once the entire image is downloaded, the processor invokes the custom-logic of the system to perform WMPCA on the image. Figure 5 shows detailed view of the custom logic.

The custom-logic has two inputs, *data* and *index*. The multiplexer **M1**, based on the value of *index*, places the data either in the *Test register sets R1, R2* or *select register, S*. The test register sets R1 and R2 are used to store the test image weights of regions 1 and 2 respectively. The select register is used to store a number, the index of image in the database with which the test image should be compared with. The multiplexers **MA** and **MB** multiplex one of the subregions in the database to the corresponding *Error Generating unit EGU*, based on the value of *select register*. The architecture of EGU follows the one shown in Figure 3. the EGU's compute the error with respect to the image corresponding to the contents of *S* register. The error is obtained from *result* port of the custom logic.

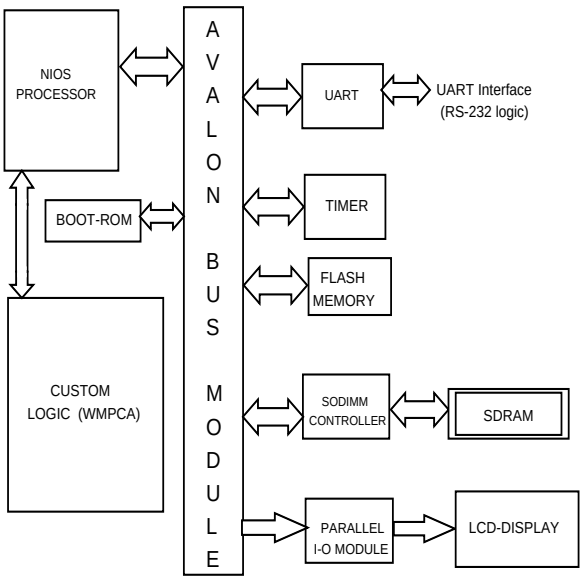


**Figure 4.** *The components of SOPC for Face Recognition.*

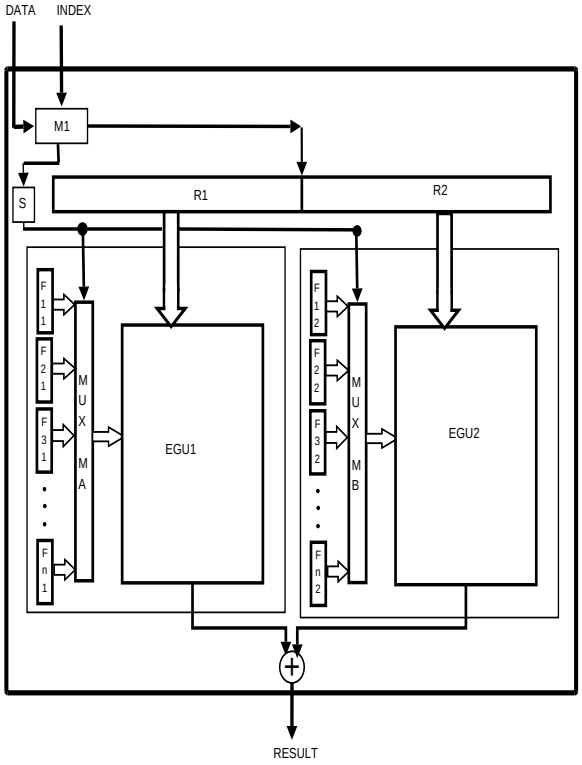


**Figure 5.** *Detailed view of each processing element.*

The Nios-processor, after placing the test image into register sets, starts placing the values *1, 2, 3,. . .,N* (N=No. of faces used in training) in the S register sequentially, and collects the errors *Er1, Er2, . . ., ErN*, which are errors corresponding to each face in the database. The host then sorts these errors. If the minimum of these is below the threshold, the output is displayed onto LCD panel.

### 4.2. Experiments and Results

The experiment had two phases:

**1. Off-line training:** Off-line training was done on MATLAB model of WMPCA training methodology. During off-line training, a covariance matrix for each sub-region was built using training images, eigenvectors were computed and each training image was projected into eigenspace and represented as a set of weights. Intra-subject variance measure of each subject was also computed. All this data was stored onto local memories of each processing element.

**2. On-line testing:** On-line testing was done on the SOPC realization of the architecture presented in previous section. On-line testing was done by writing the test-image's sub-regions onto *test register sets*. These sub-regions were read by the processing elements which projected the test sub-regions onto eigenspace and net error with respect to each of training images was computed. These error values were read, sorted and the best matches were obtained.

The UART-port was connected to P-IV 3GHz, Linux based PC. The test images were sent to the board and the outputs were available at the LCD panel. The LCD panel displayed the index of the best matched face and the time taken to compute it. The time to transfer the image from PC to board was not considered in calculating time for recognition. Table 1 gives details of the experiment. The

| Device | EP20K200EFC484-2X |
|---|---|
| Tool | Quartus-II |
| Total Logic Elements | 7760 |
| Total Memory Elements | 26,496 |
| Worst case Tco | 12.108 nsec |
| Clock | 33.33Mhz |
| Time for recognition | 38 msec |

**Table 1.** *Details of the experiment.*

SOPC designed above was able to achieve a speedup of 2.15 times when compared its software counterpart at a clock frequency 75 times slower than the P-IV system on which its software counterpart was executed. This shows the efficiency achieved by the hardware implementation that exploited the parallelism of the WMPCA approach.

## 5. Conclusions

We proposed an architecture to implement face recognition in real time using weighted modular PCA. The WM-PCA algorithm was selected owing to its potential for parallelism and improved accuracy in recognizing faces. Using this architecture, a **System On Programmable Chip for Face Recognition** was developed. We were able to achieve a speed suitable for real-time applications. To make the system complete, we are presently working to add a face detection and a pre-processing module.

## References


[1] Bernde Heisele, Purdy Ho and Tomoso Poggio. Face recognition with support vector machines: Global Vs Component based approach. In *Proc. International Conference on Computer Vision*, volume 2, pages 688–694, 2001.

[2] Choudhary A. N., Patel J.H. and Ahuja N. NETRA: A Heirarchical and Partitionable Architecture for Computer Vision Systems. *IEEE Trans. Parallel and Distributed Systems*, 4 , No. 10:1092–1104, 1993.

[3] http://altera.com/literature/ds/ds_nios_board_apex_20k200e.pdf. *Nios development board datasheet*.

[4] http://altera.com/literature/manual/mnl_avalon_bus.pdf. *Avalon bus specification manual*.

[5] http://altera.com/products/ip/processors/nios/nio−index.html. *Nios embedded processor system developement manual*.

[6] http://cvc.yale.edu/projects/yalefaces/yalefaces.html.

[7] Nalini K. Ratha and Anil K. Jain. Computer Vision Algorithms on Reconfigurable Logic Arrays. *IEEE Trans. on Parallel and Distributed Systems*, 10 , No. 1:29–43, 1999.

[8] Pavan Kumar A., Sukhendu Das and Kamakoti V. Face Recognition Using Weighted Modular PCA. *To appear in proc. International Conference on Neuro Information Processing*, 2004.

[9] Peter N. Belhumeur and P. Hesphana. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19, No.7:711–720, 1997.

[10] Rajkiran G. and Vijayan K. An improved face recognition technique based on modular PCA approach. *Pattern Recognition Letters*, 25 , No. 4:429–436, 2004.

[11] Sunwoo M.H. and Aggarwal J.K. *VisTA-An Image Understanding Architecture*, pages 121–153. Calif: acad. press, 1991.

[12] Turk M. and Pentland A. Eigen faces for recognition. *Journal of Cognitive Neuroscience*, 3 No. 1:71–86, 1991.

[13] Weems C.C. Architectural Requirements of Image Understanding Architecture with Respect to Parallel Processing. *Proc. IEEE*, 79 , No. 4:537–547, 1991.

[14] Weems C.C. , Levitan S.P., Hanson A.R., Riseman E.M., Shu D.B. and Nash J.G. The Image Understanding Architecture. *Int'l J. on Computer Vision*, 2 , No. 3:251–282, 1989.

[15] Zhao W., Chellappa R. and Phillips P. J. Face Recognition: A Literature survey. *ACM Computing Surveys*, 35:339–458, 2003.