A Comparative Implementation of PCA Face Recognition Algorithm

Nicolas Morizet, Frédéric Amiel, Insaf Dris Hamed, Thomas Ea
Department of Electronics
Institut Supérieur d'Electronique de Paris (I.S.E.P.)
Paris, France
{nicolas.morizet, frederic.amiel, insaf.dris-hamed, thomas.ea}@isep.fr

Abstract—**This paper presents PCA algorithm used in face recognition system and its implementation on different architectures in order to choose the best solution for designing a real time face recognition system. Benchmarks and comparisons will be given for PC, DSP and FPGA and results will show that FPGA soft core is too slow for this computation. An IP solution is the best choice. But DSP is also a good compromise for implementing easily the algorithm and having acceptable processing time.**

I. INTRODUCTION

For more than 15 years, research in biometric systems has been increasing significantly due to international insecurity environment. Research groups around the world are developing algorithm and systems based on face, iris, fingerprint, palm print or voice… In our research laboratory, recognition with iris [1] and face, and their implementations on embedded platforms are studying.

Face recognition algorithm is mainly based on Principal Component Analysis (PCA) [2]. PCA can be time consuming and this article will give quantitative data for choosing the best platform for implementing this algorithm.

This paper is organized as follows: firstly, PCA algorithm description is presented in section 2. Secondly, reduced accuracy for number coding is presented in section 3. Thirdly, target platforms and results are described in section 4. Finally, a short summary and future work conclude this article in section 5.

II. PCA ALGORITHM

A. Description

Principal Component Analysis (PCA, also known as "Eigenfaces"), is one of the most known global face recognition algorithm. The main idea is to decorrelate data in order to highlight differences and similarities by finding the principal directions (i.e. the eigenvectors) of the covariance matrix of a multidimensional data. For our experiments, we use the FERET database [3]. The Gallery Set contains M=1196 subjects, the Training Set is a subset of the Gallery Set and is composed of N=501 subjects. For testing our system, we use some face images from the FB Probe Set containing P=1195 subjects (same persons of the Gallery Set but with changes in facial expressions).

B. Training the PCA

From a theoretical point of view, a face image Γ_i can be seen as a vector is a huge dimensional space, concatenating the columns.

We work with normalized face images that we preprocessed according to the CSU System 5.0 [7] and compressed with a double 2D-DWT [4]. Those images are of size of (*38x33*) pixels (*Fig. 1*), leading to a vector of dimension 1254, after concatenating columns.

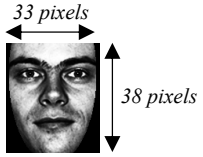

Figure 1. An Example of a FERET normalized face image used in our system.

The first step is to train the PCA using the Training Set, in order to generalize the ability of our system and generate eigenvectors. We compute the mean image of the training data: $\Psi_{train} = \frac{1}{N}\sum_{n=1}^{N}\Gamma_n$. Then each training image is mean-subtracted: $\Theta_n = \Gamma_n - \Psi_{train}, n = 1 \cdots N$.

The covariance matrix (C) of the mean-subtracted training data is then computed (1) (^{T} denotes the matrix transposition operation):

$$C = \sum_{n=1}^{N} \Theta_n \Theta_n^T \quad (1)$$

The next step consists in finding the eigenvectors e_n and the eigenvalues λ_n of *C*. We use the mathematical trick described in [2] to finally solve (2):

$$C \cdot e_n = \lambda_n \times e_n, n = 1 \cdots N \quad (2)$$

A part of the great efficiency of the PCA algorithm is to take only the "best" eigenvectors in order to generate the subspace ("*Face Space*") where the gallery images will be projected onto, leading to a reduction of dimensionality.

Eigenvalues are sorted in decreasing order (a higher eigenvalue captures a higher variance, hence more information). Then, we compute the total energy of the eigenvalues to retain the 240 first eigenvectors (48 % of the total training number, corresponding to a 98.5% energy cutoff, see *Fig. 2*). Each eigenvector is of size of (*1254x1*).

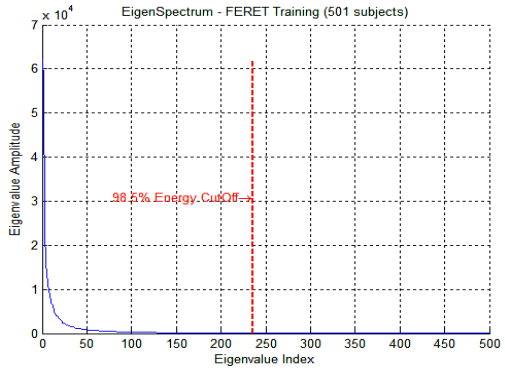

Figure 2. Eigenspectrum of the FERET Training Set.

C. *Projecting the Gallery*

The mean image Ψ of the Gallery Set is computed. Each mean-subtracted gallery image $\Theta_i = \Gamma_i - \Psi, i = 1 \cdots M$ is then projected onto the "Face Space" spanned by the *M'=240* eigenvectors deriving from the Training Set.

This step leads to a simple *dot product* (3):

$$\omega_k = e_k^T . \Theta_i, k = 1 \cdots M' \quad (3)$$

Scalars ω_k are called "weights" and represent the contribution of each eigenvector for the input image. Thus, for each gallery image, we have a "Weights Vector": $\Omega_i^T = [\omega_1, \omega_2, \ldots, \omega_{M'}]$ of size of (*1x240*) (*Fig. 3*).

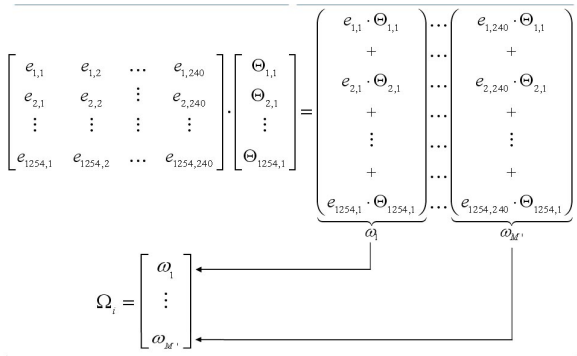

Figure 3. Scheme used to compute the different dot products for one image, for our hardware implementation.

The "Weights Matrix" $\Omega^T = [\Omega_1^T, \Omega_2^T, \ldots, \Omega_M^T]$ of size of (*1196x240*) is then generated and stored in the database and will be used during the *recognition step* we focused on.

D. *Recognition*

1) *Dot Product*

The *dot product* is the first basic operation that must be done during the recognition step. A normalized probe image is projected onto the "*Face Space*", in order to obtain a vector Ω'^T of size of (*1x240*).

2) *Distance Measure*

Once the incoming probe image has been projected onto the Face Space, we have to see whether it is a known face or not. To proceed, we compute the *Squared Euclidean Distance* (SED) (4) between the weights from the probe image and the Weights Matrix Ω^T of the entire Face Space:

$$\varepsilon_i = (\Omega'^T - \Omega_i^T), i = 1 \cdots M \quad (4)$$

We left the square-root of the basic Euclidean distance in order to save some computation time on the FPGA. This process has no influence on the recognition performance.

3) *Final Decision*

At this stage, we have *M* SEDs. Since we work in verification mode (the subject's identity is claimed) and since the Probe Set contains persons who are registered in the database, we simply take the minimum SED (5) as the final decision rule:

$$\varepsilon_k = \min(\varepsilon_i), i = 1 \cdots M \quad (5)$$

Assume the ID of the probe image is the *l* (from 1 to *P*) and *k* is the index corresponding to the minimum SED, a subject is considered as a genuine if *l=k*, otherwise he is considered as an impostor.

III. ACCURACY REDUCTION

We are prospecting to implement the algorithm on platforms which do not implement floating point calculus. Then we need to study for all the calculation the fixed point precision required.

The final platform will implement only the second part of the algorithm (phase 2) i.e. projection of the image on eigenvectors and distance calculation (*Squared Euclidean Distance*).

We have established the algorithm on MATLAB, all the calculations are done in double format. But, to reduce time, the projection on "face space" (*dot product*) was written in C language with double variables. The distance calculation was already written in C.

In order to estimate the precision, * and + operations are replaced with *mul* and *add* functions. Inside these functions, we have placed some minimum and maximum test, and some truncate operation on passed and used arguments in

order to simulate the effect of fixed point arithmetic with a determined precision.

Then we applied a criterion to determine the quality of the result. Because our application is to identify people, we have applied the following process:

- For a determined precision, we quantize the eigenvectors matrix, we project an image and we proceed to the distance calculation with all the signatures of our database. We determine then the two first minimums.
- If the face of the person is in the database, the first minimum should identify her (identification). The second minimum references the next face which is the most likeness.
- We verify with the accorded precision if those two minimum identify the same face. On the other hand, we tolerate 10% of scattering.

Those measurements established the following results:

- all the number are between [-30;60]
- 18 bits are enough to obtain good results accorded to our criterion.

Then we can apply at the minimum the following scheme: *7 bits for integer part, 11 bits for decimal part.*

IV. TARGET PLATFORM DESCRIPTION AND RESULTS

To determine which kind of platform should implement our system we have to consider:

- Memory size required
- Computation time

For our biometric identification system, we estimate that a total time of one second is acceptable.

The complete algorithm is described on *Fig. 4*.

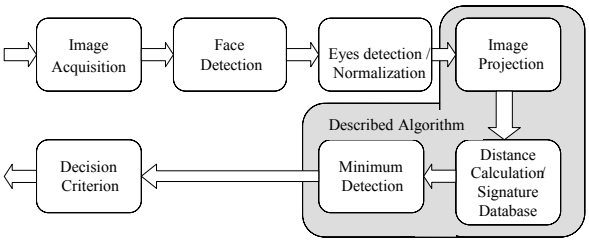

Figure 4. Face recognition algorithm.

After image acquisition, eyes detection permits to normalize and to crop image (translation and rotation), to have a standardized scale.

We keep ½ seconds to execute those two algorithms and we attribute ½ seconds for the algorithms described in this article.

Our algorithm is used with a Gallery database of 1200 signatures. We will test it with a Probe Set of 1200 signatures.

A. *Memory*

Each image has a 120x140 pixels resolution, on 8 bits (16800 pixels), but is reduced to 33x38 pixels after double wavelet compression: 1254 data on fixed precision with 18 bits.

- Eigenvectors matrix:

This matrix is approximately 20% the size of original images from the database. To represent nearly 1200 different faces, it has 240 lines (*Fig. 3*) with 1254 data coded on 18 bits per column.

- Signature Set:

We generate the Signature Set from the Probe Set. It contains the signature for the different persons in the database. In our case, 1200 vectors, each with 240 elements of 18 bits.

- Total memory with 18 bits data is summarized on Tab. 1

TABLE I. MEMORY NEEDED

Original Image	1254 * 1
Eigenvectors	1254 * 240
Signature Set	240 * 1200
TOTAL	*590214*

For 18 bits data it means approximately 10 Mbits = 1.3 Mbytes. With a 24 bits wide processor it will be 1.7 Mbytes, and with a 32 bits wide processor: 2.3 Mbytes.

This is not a large memory and we can use some static memory components to implement it. (Database can be stored inside a flash memory).

B. *Computation time*

We can estimate the number of needed basic operations:

Projection of image on eigenvectors space:

- Matrix multiplication 1254 * 240 / vector 1254 = 300960 MACs (mul add).
- Distance calculation / Probe set: 2 multiplications / 1 addition: 240 * 1200 * 2 = 576000 MACs.
- Minimum calculation: negligible.

We have chosen different platforms to test this system:

- PC: To establish and to verify algorithm. We will not use this platform for final system because we're looking a small and low power platform.
- DSP: We have put the program on a TI6713 DSP [5] processor (with floating point calculation).

- FPGA: We have run the program on a soft processor (NIOS II on Altera device [6]). An IP is also designed and mapped on FPGA (*Fig. 5*).

We have tested different versions for the NIOS:

- Floating float calculation (emulated by software)
- Floating point calculation (with custom instructions)
- Fixed point calculation (with native 32 bits wide integer)

The IP (*Fig. 6*) we have designed on FPGA use only one MAC operator, with a specific sequencer to extract data from a local memory. Then, it is not space consuming and it has enough accuracy. Because we have others IP on the component which need memory, the memory device is shared between different blocks at different time.

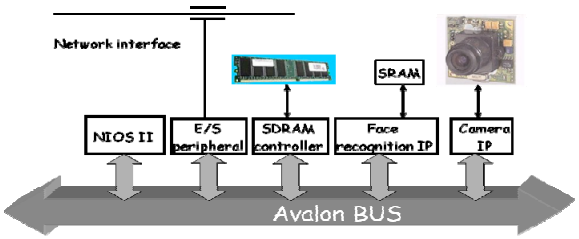

Figure 5. Architexture of the system with IP.

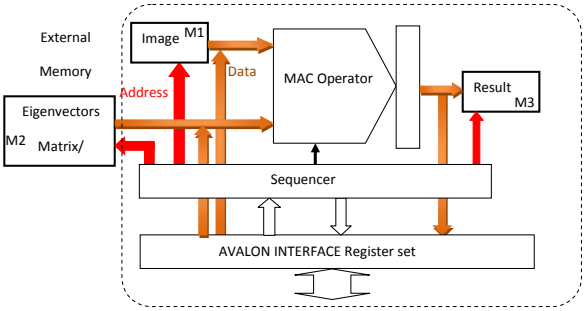

Figure 6. IP internal architecture.

This IP is pipelined and works at 50 MHz with the same clock of NIOS system. This IP is used to compute the projection onto the Face Space spanned by the eigenvectors, and to compute the distance calculation from the Probe Set.

In the first case, the result is a vector 1254 elements long. For the second operation, the result is all the distance from database. The NIOS system loads first the memory M1 with the eigenvectors and Probe Set, and loads the image to M2.

Then the program starts the projection calculation by writing into a control register, and pulls a status register to wait for the end of calculation.

The results are read from M3, and stored to M1 by NIOS program. The control register is set to distance calculation configuration and the cycle restart. After doing IP calculation, all the distances are available on M3. The NIOS system reads the results and computes the minimum required values.

C. *Results*

Computation times are shown in Tab. 2

TABLE II. COMPUTATION TIME

Platforms	Processing Time
PC 3GHz	3 ms
DSP 225 MHz	5 ms
NIOS Soft Floating point operators	62 s
NIOS Hard Floating point operators	28 s
NIOS fixed point	15s
IP @50 MHz	60 ms

Results show bad performances for soft core even with fixed point calculation. DSP gives acceptable time processing and IP presents for FPGA a good compromise to obtain performances with little additional hardware.

V. CONCLUSION AND FUTURE WORK

In this article, we present the PCA algorithm used in our face recognition system. This algorithm, after studying the reduction of accuracy for coding numbers, is implemented on different target platforms. The results demonstrate that IP is the best choice if we need programmable parts and little hardware area but DSP is a good challenger because it offers a good trade-off between processing time and implementation easiness. In a near future, we plan to use C2H, an Altera tool for generating automatically VHDL code from C code. We plan to study the implementation of some preprocessing steps such as automated face detection, eyes detection and face normalization. We are also working on fusion of iris and face and we are trying to develop a bimodal biometric system for recognition.

ACKNOWLEDGMENT

Portions of the research in this paper use the FERET database of facial images collected under the FERET program, sponsored by the DOD Counterdrug Technology Development Program Office. The authors would also like to thanks Sami Romdhani and Yves Meyer for giving precious advices concerning PCA training and preprocessing.

REFERENCES

[1] T. Ea, F. Amiel, A. Michalowska, F. Rossant, A. Amara, "Contribution of Custom Instructions on SoPC for iris recognition application", ICECS 2006, Nice, France.

[2] M. Turk, A. Pentland, "Eigenfaces for Recognition", Journal of Cognitive Neuroscience, vol. 3, no. 1, 1991.

[3] The Color FERET Database, website : http://www.itl.nist.gov/iad/humanid/colorferet/home.html, 2003.

[4] B. Burke Hubbard, "Ondes et ondelettes, la saga d'un outil mathématique", Belin pour la science, 1995.

[5] Analog Technologies; Semiconductors, Digital Signal Processing: Texas Instruments. http://www.ti.com.

[6] FPGAs, CPLDs & Structure ASICs: Altera, the Leader in Programmable Logic. http://www.altera.com.

[7] The CSU Face Identification Evaluation System user's Guide (version 5.0), Internet address: http://www.cs.colostate.edu/evalfacerec/algorithms/version5/faceIdUsersGuide.pdf, 2003.