Parallelizing Algorithms in Ada on Clementina II
Face Recognition System

Champredonde Raúl[1]
Chichizola Franco[2]
Laboratory of Research and Development on Computer Sciences[3]
Computer Sciences Faculty - National University of La Plata

Abstract
In the Laboratory of Research and Development on Computer Science of the National University of La Plata, a face recognition system has been developed. This article describes a series of testings based on parallel processing, with the objective of optimizing the said system response times developed in Ada programming language on SGI Origin 2000 parallel architecture known as Clementina II. Then, the results obtained are analyzed

Keywords
Parallel processing, Ada

[1] Mid-time Co-Chair Professor. Advance Scholar CONICET. rchampre@lidi.info.unlp.edu.ar
[2] Mid-time Graduate Assistant. francoch@lidi.info.unlp.edu.ar
[3] 50 St and 115 St. First Floor. 1900 La Plata. Bs.As. Tel/Fax. 54 221 422 7707

1. Introduction

In the frame of the projects carried out by the Laboratory of Research and Development on Computer Sciences of the National University of La Plata, a face recognition system has been developed as dissertation [5].
Basically, the system uses a face image database and it draws certain characteristics from the each of them, in a process called *training.*
The trained system is able to receive a face image and determine whether it corresponds to some of the faces which are found in the image database, and in such case, to which of them it corresponds. This process is known as *recognition.*
Many applications which use this technology can be found. For example, in a security access system to physical or electronic places based on the face of whoever wants to enter.
Furthermore, considering that the processing and recognition of finger prints is expensive as regards time and computing resources, other example could be an application which, given the photo of a person, it finds a set as reduced as possible of candidates, in order to, just then, start the analysis on finger prints.
However, most of the applications, if not all, need the best possible performance, and precisely if we taken into account the fact that they need an image database with a large quantity of faces. The quantity of the necessary calculation, both for training and recognition, increases proportionally to the quantity of faces of the database.
It is clear that the parallelism of thick grain will benefit the application's performance. For example, it is not difficult to prove that in the training process, the eigenvectors computation of an parallel image set requires approximately the same time than the required for the same computation of a single image.
Face recognition system is principally based on matrixes operations. In the testing, the determination of how beneficiary the parallelism could be at the level of these operations using Ada and Clementina II is looked for in order to estimate their effect on the improvement of the performance which would be obtained by parallelizing the face recognition system.
Being the multiplication of the matrixes an operation that could be considered as representative of BLAS level 3 [1], and having several researches been developed about it [2][3][4][6][7], this operation is used for the development of the tests.

2.1. Training and Recognition Algorithms

Face recognition application is based on the EigenFaces method [8][9], which basically consists of a training process and a recognition process. Both processes use a face database composed by a set $\Gamma_1, \Gamma_2, ..., \Gamma_M$ of images of size NxN .
Training consists of the following stages:

1- Each image

$$\Gamma_i \text{ with } i = 1,2,..,M$$

is reorganized as a vector of size N^2, whose value is formed as a concatenation of each of the lines of the images, thus forming a matrix $N^2 xM$

2- Ψ average face is obtained according to the formula

$$\Psi = \frac{1}{M} \sum_{n=1}^{M} \Gamma_n$$

3- Ψ average face obtained is substracted from each of the images Γ_i with i between 1..M thus obtaining a new set of vectors

$$\Phi_i = \Gamma_i - \Psi$$

which compose the matrix $\Lambda = [\Phi_1, \Phi_2, ..., \Phi_M]$ of $N^2 xM$.

4- A reduced covariance matrix is obtained.

$$L = \frac{1}{M} \Lambda^T \Lambda$$

of MxM size.

5- L autovectors are obtained; all of which, ordered in a descending fashion according to their corresponding autovalues, constitute matrix v .

6- u "eigenvectors" are computed as a lineal combination of Λ columns and v columns

$$u_l = \sum_{k=1}^{M} v_{lk} \Phi_k$$

7- A pattern is obtained

$$\Omega_i^T = [\omega_1, \omega_2, ..., \omega_M]_{\text{con}} \quad i = 1,2,...,M$$

where

$$\omega_k = u_k^T (\Gamma_i - \Psi)_{\text{con}} \quad k = 1,2,...,M$$

Given a face image, the recognition process tries to find in the image database that one which corresponds to the given face, for which its Ω pattern is computed using the same procedure preciously described, and the minimum distance is looked for.

$$\min\left(\left\|\Omega - \Omega_i\right\|^2\right)_{\text{with}} \quad i = 1,2,...,M$$

Once the minimum distance is found, if the same is within the limits stipulated by the error percentage admitted which is received as a parameter, the system shows which the corresponding image is. On the contrary, it shows the non-existence of the correspondence.

2.2. Architecture Description

The experiments carried out were carried out in the supercomputer known as Clementina II [11][12][13].
Clementina II is a Cray Origin 2000 system manufactured by the firm SGI (new name for Silicon Graphics). It has forty processors MIPS RISC R12000 of 300 MHz., with a secondary cache of 4 MB, a principal memory of 10 GB shared by all the processors, 360 GB of disk storing.
The operative system of this supercomputer is Irix 6.5 [14], which was developed by Silicon Graphics. Irix is compatible (compliant) with UNIX System V Release 4 and with standards of The Open Group, including UNIX 95, Year 2000 and POSIX.
Irix has a support for symmetric multiprocessing (SMP) scalable up to 128 processors and support for 32 and 64 bits.

3. Testings Carried Out

The testings were carried out by using matrixes multiplication, since this operation is the most representative of BLAS level 3.
Square matrixes of MxM with $M = 250,500,1000,1500,2000$ were used.

The multiplication is divided in several tasks. Each of these is in charge of carrying out the corresponding computations to a part of the resulting matrix. The division of the resulting matrix is carried out by rows, according to the following scheme:
Let it M be the size of the matrix and T be the quantity of tasks used, the resulting matrix is divided in T groups of M/T lines each one. The task T_i is in charge of computing the corresponding values to the line group T_i with $i = 1,2,...,T$.
The tests were carried out by using T with $T = 1,5,10,20,30,40$.
Whit the objective of using the division of the most convenient resulting matrix in relation to the influence this could have on the utility of the cache memory, the way in which Ada language internally represents the matrixes was identified. Ada compiler for the architecture used represents the matrixes per files. The same criterion was then used in order to divide the resulting matrix in as many portions as tasks.
The work conditions were those of normal use, i.e., without having exclusive access to the architecture. These conditions make the quantity of processors used to be dependant on the work charge which each of them has at the time of the execution of the matrix multiplication process, and in the particular case of Clementina II, this charge is commonly high.
The operating system allows to obtain the total time of effective process execution. This time is equal to the sum of the time periods in which each of the tasks composing the process occupy a processor.
On the other hand, the creation time of tasks and of operating and resulting matrixes is not significant in relation to the processing time, as well as the influence of the use of the cache memory.
Consequently, if there existed a free access to the whole architecture of Clementina II, the total time of the processing would approximately be equal to the executing time of only one of the talks which the process uses.
This is the criterion used in order to compute the performance with exclusive access, in function of the performance in normal work conditions.

4. Results obtained

”Raw” results obtained are showed in the following tables. Each table shows in detail (for each one of the different quantities of tasks used): the user time and the system time which the execution of the process took; the total time as the sum of the two previous ones; the time of each task; the time increasing (speedup) of the parallel solutions in respect to the sequential and the achieved Mflops. All time measures are expressed in seconds.

Table 1: Matrixes of 250x250

Tasks	User T	System T	Total Time	Task Time	Speedup	Mflops
1	1,973	0,025	1,998	1,998	1,000	15,609
5	1,977	0,027	2,004	0,401	4,985	77,813
10	1,989	0,023	2,012	0,201	9,930	155,007
20	1,976	0,034	2,010	0,101	19,880	310,317
30	2,007	0,040	2,047	0,068	29,278	457,006
40	2,009	0,056	2,065	0,052	38,695	604,007

Table 2: Matrixes of 500x500

Tasks	User T	System T	Total Time	Task Time	Speedup	Mflops
1	15,889	0,054	15,943	15,943	1,000	15,665

5	15,885	0,061	15,946	3,189	4,999	78,311
10	15,881	0,070	15,951	1,595	9,995	156,573
20	16,019	0,075	16,094	0,805	19,812	310,364
30	16,006	0,082	16,088	0,536	29,729	465,713
40	16,108	0,100	16,208	0,405	39,345	616,350

Table 3: Matrixes of 1000x1000

Tasks	User T	System T	Total Time	Task Time	Speedup	Mflops
1	162,420	0,256	162,676	162,676	1,000	12,288
5	166,869	0,325	167,194	33,439	4,865	59,781
10	163,464	0,276	163,740	16,374	9,935	122,084
20	168,993	0,338	169,331	8,467	19,214	236,106
30	165,542	0,306	165,848	5,528	29,426	361,595
40	166,494	0,390	166,884	4,172	38,991	479,135

Table 4: Matrixes of 1500x1500

Tasks	User T	System T	Total Time	Task Time	Speedup	Mflops
1	661.273	0.646	661.919	661.919	1.000	10.194
5	694.430	1.077	695.507	139.101	4.759	48.510
10	683.000	1.019	684.019	68.402	9.677	98.649
20	688.620	1.259	689.879	34.494	19.189	195.621
30	687.938	1.312	689.250	22.975	28.810	293.700
40	673.474	1.064	674.538	16.863	39.252	400.141

Table 5: Matrixes of 2000x2000

Tasks	User T	System T	Total Time	Task Time	Speedup	Mflops
1	1840,688	1,843	1842,531	1842,531	1,000	8,682
5	1917,263	2,552	1919,815	383,963	4,799	41,660
10	1871,282	2,133	1873,415	187,342	9,835	85,384
20	1884,615	2,851	1887,466	94,373	19,524	169,497
30	1907,235	3,122	1910,357	63,679	28,935	251,199
40	1898,128	3,324	1901,452	47,536	38,761	336,501

These results cannot be shown in a one single graphic since temporal scales are excessively different. Therefore, they are shown in one graphic for each size of matrix. The corresponding values to the tests with 5 tasks are not included in order to avoid distortion which, logically, will introduce in the graphic.

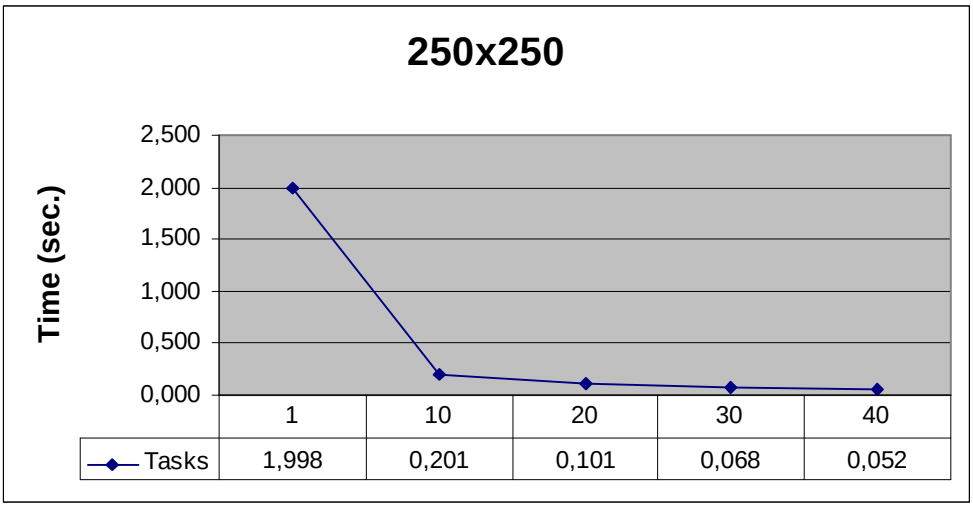

Figure 1: Executing Times for matrixes of 250x250

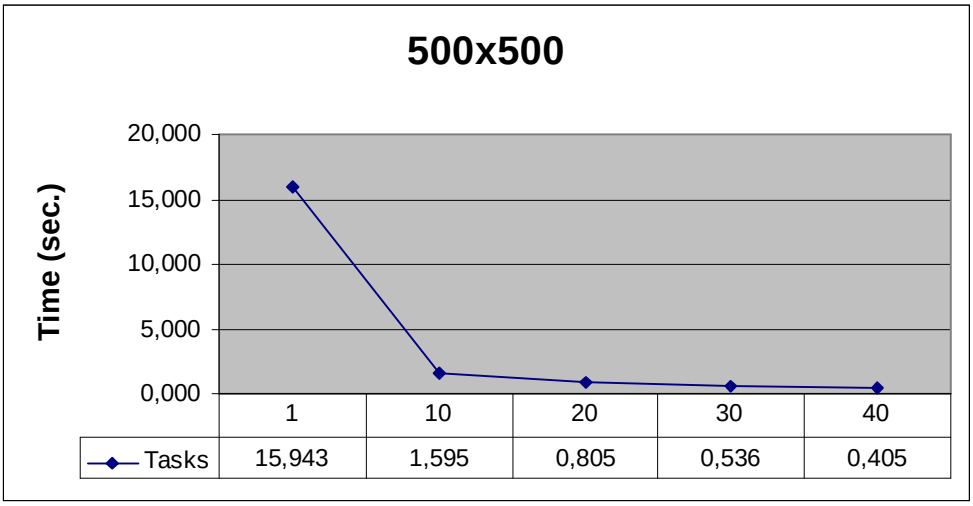

Figure 2: Executing Times for matrixes of 500x500

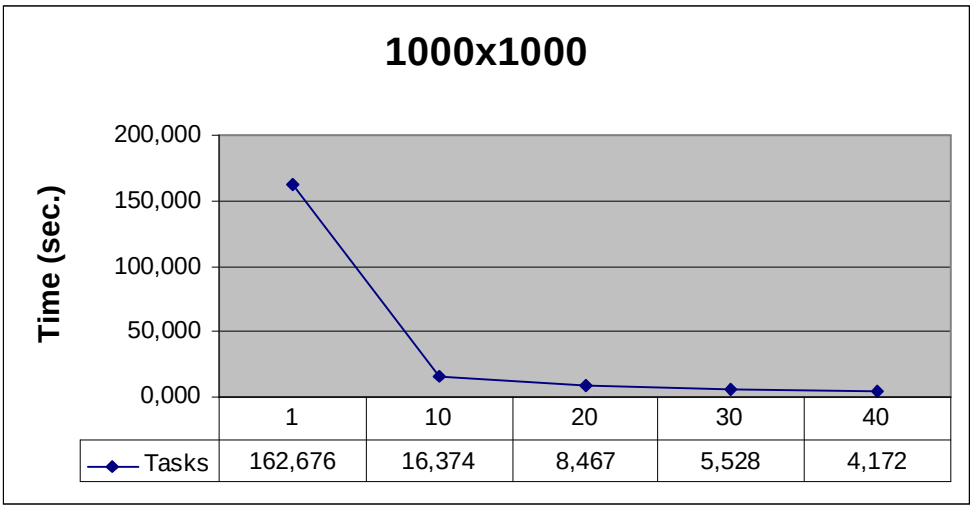

Figure 3: Executing Times for matrixes of 1000x1000

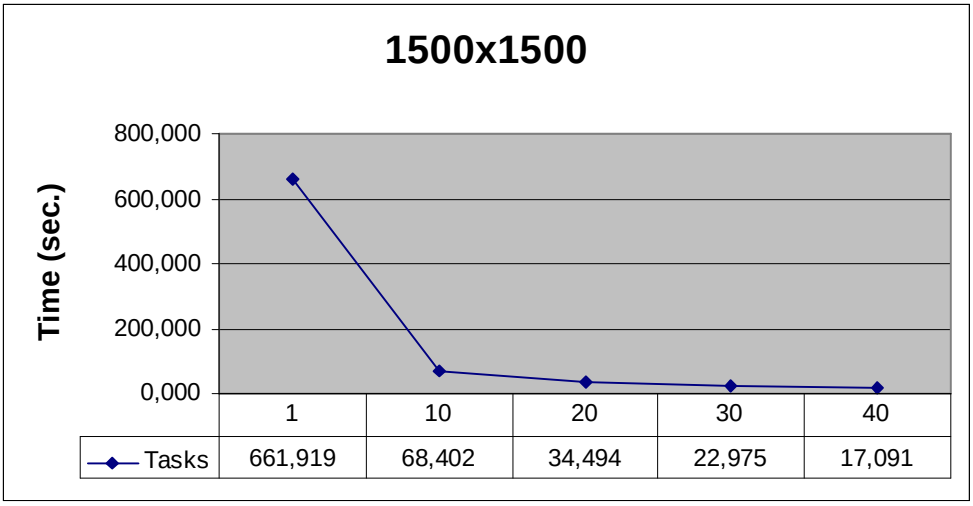

Figure 4: Executing Times for matrixes of 1500x1500

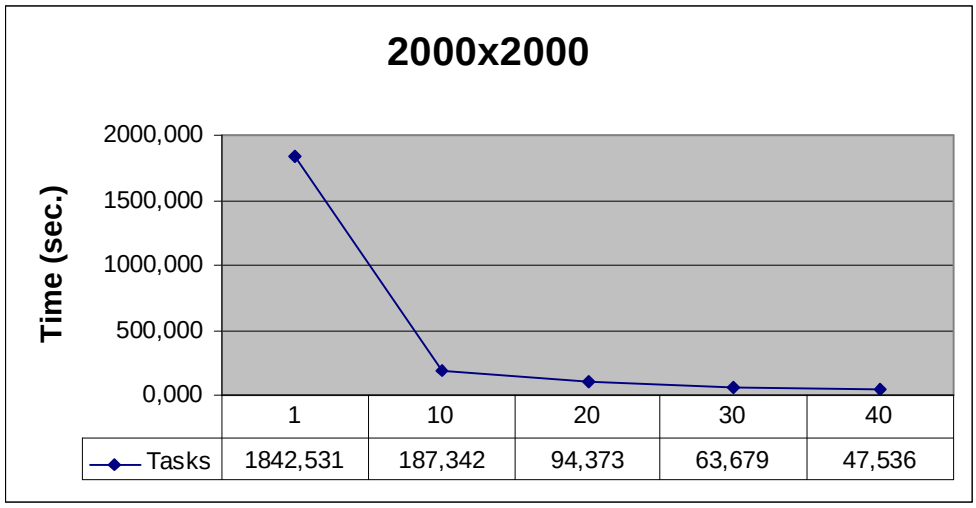

Figure 5: Executing Times for matrixes of 2000x2000

The previous graphics show curves whose shapes are very similar between them. This shows that the performance has approximately the same improvement for all the tests as the number of processors used increases.
In order to observe more precisely the improvement of the performance, the following graphic shows speed increasings of the parallel solutions with respect to the sequential solution (speedup) obtained in the different tests.

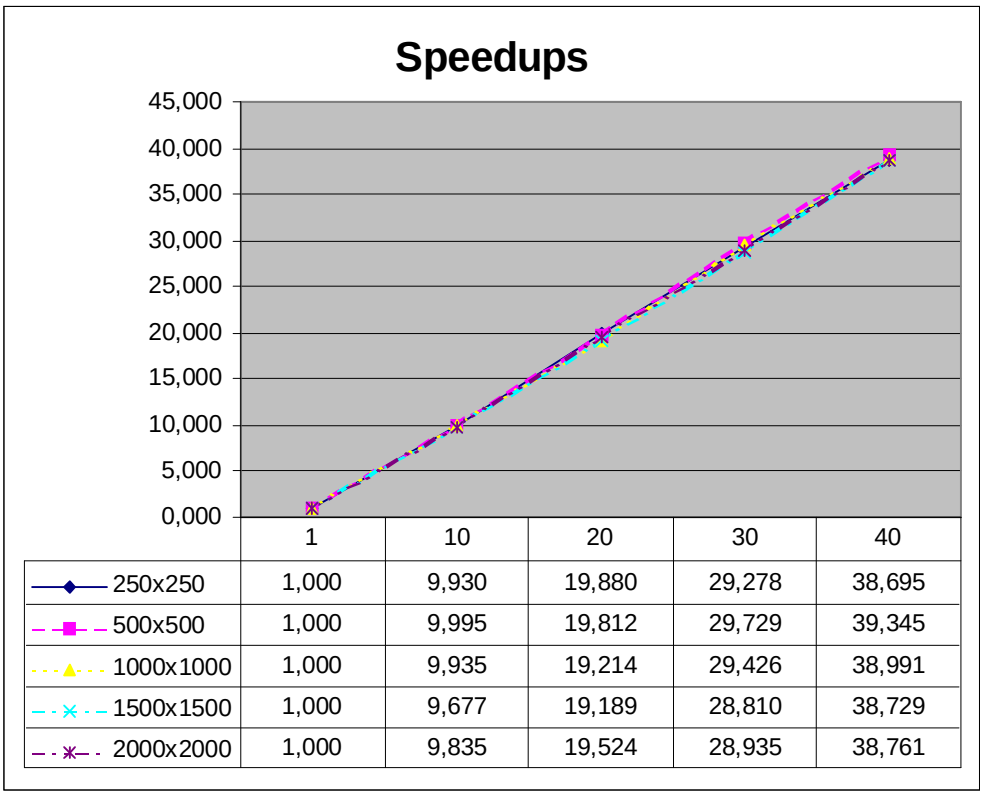

Figure 6: Speedups obtained

According to the shape of the curves which represent the different speedups, it is observed that this is lineal and very close to the optimum. Thus, the improvement of the performance is directly proportional to the quantity of processors used.
This lineal, almost optimum, speedup is achieved by simply dividing the processing of tasks in order to take advantage of the availability of several processors. If tuning techniques of the application are also used in order to make the most of the possibilities of the architecture, extraordinary improvements can be obtained.
In the previous tables a column is included with the Mflops achieved in the processing. These columns are used in order to build up the following table and in order to make the computing of the average of Mflops achieved by a task, i.e., Mflops of a processor.

Table 6: Mflops

Tasks	250x250	500x500	1000x1000	1500x1500	2000x2000	Total Mflops	Average by amount of tasks	Average by tasks
1	15,609	15,665	12,288	10,194	8,682	62,439	12,488	12,488
5	77,813	78,311	59,781	48,510	41,660	306,075	61,215	12,243
10	155,007	156,573	122,084	98,649	85,384	617,697	123,539	12,354
20	310,317	310,364	236,106	195,621	169,497	1221,905	244,381	12,219
30	457,006	465,713	361,595	293,700	251,199	1829,213	365,843	12,195
40	604,007	616,350	479,135	394,814	336,501	2430,806	486,161	12,154
						Average of Mflops by tarea		**12,275**

By using ATLAS [10], the maximum potential of the computation of an architecture in particular can be practically obtained by means of automatic tuning which it provides. On a processor R10000, it reaches some 306,2 Mflops for multiplication of matrixes of 500x500, thus it surpasses that quantity on processors R12000. Therefore, the performance obtained can be improved in 25 times.

5. Conclusions

The use of the programming language Ada for the development of parallel applications which require a large quantity of calculation is feasible and realistic.
Logically, the results obtained cannot be applied to all types of algorithms, though they can be applied to those which can take advantage of shared memory parallel architectures as in the case of Clementina II.
The results obtained show that the use of parallelisms at the level of operation on matrixes will be really beneficiary as regards performance of the face recognition system and of any other application based on computations of the same nature.

6. Further Works

In function of the results obtained, a parallel solution of the face recognition system using Ada will be developed, and the improvement of its performance with respect to the same solution - in this case sequential – will be verified.
On the other hand, it is known that by using the automatic tuning tool ATLAS, Mflops can be achieved 25,5 times greater than the obtained. For this reason, in the face recognition system, once

the division of the resulting matrix is carried out, we will interact with language C in order to take advantage of ATLAS utilities.
It is also possible to apply these experiences to the Real Time research line, in particular as to what computer vision concerns.

7. Acknowledgements

The authors would like to thank the invaluable collaboration of MSc. Fernando Tinetti.

8. References

[1] Anderson E., Bai Z., Bischof C., Demmel J., Dongarra J., DuCroz J., Greenbaum A., Hammarling S., McKenney A., Sorensen D. "LAPACK: A Portable Linear Algebra Library for High-Performance Computers". Proceedings of Supercomputing '90, pages 1-10, IEEE Press, 1990.

[2] Bilmes J., Asanovic K., Chin C., Demmel J. "Optimizing matrix multiply using phipac: a portable, high-performance, ansi c coding methodology". Proceedings of the International Conference on Supercomputing, Vienna, Austria, ACM SIGARC. July 1997

[3] Choi J. "A New Parallel Matrix Multiplication Algorithm on Distributed-Memory Concurrent Computers". Proceedings of the High-Performance Computing on the Information Superhighway, IEEE, HPC-Asia'97. 1997.

[4] Choi J., Dongarra J., Walker D. "PUMMA: Parallel Universal Matrix Multiplication Algorithm on Distributed Memory Concurrent Computers, in Concurrency: Practice and Experience". 6:543-570. 1994.

[5] Correa Martín, Chichizola Franco. "Sistema de reconocimiento de rostros". Trabajo de grado. Facultad de Informática. 2001.

[6] Dekel E., Nassimi D., Sahni S. "Parallel matrix and graph algorithms". SIAM Journal on Computing, 10:657-673. 1981.

[7] Golub G., Van Loan C. "Matrix Computation" Second Edition. The John Hopkins University Press, Baltimore, Maryland. 1989.

[8] Jun Zhang, Young Yan, and Martin Lades. "Face Recognition: Eigenface, Elastic Matching, and Neural Nets." Proceedings of the IEEE. vol. 85. No. 9, pp.1422-1435, 1997.

[9] M. Turk and A. Pentland, "Face recognition using eigenfaces", in Proceedings of International Conference on Pattern Recognition , pp. 586-591,1991

[10] Whaley R., Dongarra J. "Automatically Tuned Linear Algebra Software". PhD Thesis. Dept. of Computer Sciences, Univ of TN, Knoxville. 1998.

[11] www.cab.cnea.gov.ar/difusion/ClementinaIINacion.html

[12] www.setcip.gov.ar

[13] www.sgi.com/origin/2000

[14] www.sgi.com/software/irix6.5