Paralelización del Cálculo de Autovalores y Autovectores en un Esquema de Memoria Compartida

Franco Chichizola, Raúl E. Champredonde, Armando E. De Giusti
Universidad Nacional de La Plata, Facultad de Ciencias Exactas
Laboratorio de Investigación y Desarrollo en Informática
La Plata, Argentina, 1900
{francoch, rchampre, degiusti}@lidi.info.unlp.edu.ar

Abstract

Eigenvectors and eigenvalues of a matrix are elements very much used in several areas, specially in image recognition algorithms. An EigenFaces-based face recognition system has been particularly developed in the Laboratory of Research and Development in Computer Sciences of the Faculty of Computer Sciences of the National University of La Plata. This method makes an intensive use of eigenvectors and eigenvalues, as well as other operations over matrixes, whose computation uses up large time quantities relative to the matrixes size.

This work presents the research and experimentation carried out over the eigenvalue and eigenvector computation. Then, it proposes a parallelization method for such computation, in order to fulfill the face recognition system parallelization as a way of upgrading response times. Finally, the obtained results are shown.

Keywords: parallel processing, face recognition, matrices, eigenvalues and eigenvectors.

Resumen

Los autovalores y autovectores de una matriz son elementos muy utilizados en diversas áreas, especialmente cuando se trata de algoritmos de reconocimiento de imágenes. En particular, en el ámbito del Laboratorio de Investigación y Desarrollo en Informática de la Facultad de Informática de la Universidad Nacional de La Plata, se ha desarrollado un sistema de reconocimiento de rostros basado en el método EigenFaces. Este método hace uso intensivo de autovalores y autovectores, así como de otras operaciones sobre matrices, cuyo cómputo consume grandes cantidades de tiempo relativas al tamaño de las matrices.

Este trabajo presenta la investigación y experimentación realizadas sobre el cálculo de autovalores y autovectores, se propone un método de paralelización del mismo, para completar la paralelización del sistema de reconocimiento de rostros, como una forma de mejorar los tiempos de respuesta, y se muestran los resultados obtenidos.

Palabras claves: procesamiento paralelo, reconocimiento de rostros, matrices, autovalores y autovectores.

1 Introducción

En el ámbito del Laboratorio de Investigación y Desarrollo en Informática de la Facultad de Informática de la Universidad Nacional de La Plata, se ha desarrollado un sistema de reconocimiento de rostros [3], que utiliza el método EigenFaces [6][5].

Este método de reconocimiento de rostros utiliza algoritmos que hacen uso intensivo de operaciones sobre matrices, tales como la multiplicación, pero también del cálculo de autovalores y autovectores. Estas operaciones insumen una importante cantidad de tiempo, la cual está directamente relacionada con el tamaño de la matrices.

Para mejorar los tiempos de respuesta del sistema, surge como alternativa natural el uso del procesamiento paralelo.

Con el objetivo de encontrar la justificación suficiente para emprender la paralelización del sistema, y considerando que el tiempo necesario para el cómputo de operaciones entre matrices del tipo de la multiplicación responde a distintos parámetros que el tiempo necesario para el cálculo de autovalores y autovectores, se dividió la experimentación en dos partes.

La primera de ellas se ocupó de los aspectos relacionados con la paralelización de la multiplicación de matrices. Se eligió esta operación entre las demás por ser considerada la más representativa de BLAS nivel 3 [1]. Los resultados de esta parte del trabajo se encuentran publicados en [2] y concluyeron con la paralelización del sistema en lo que hace a las operaciones convencionales entre matrices.

La segunda parte es precisamente el objetivo de este trabajo, y se trata de la paralelización del cálculo de autovalores y autovectores, que forma parte del sistema de reconocimiento de rostros desarrollado.

2 Cálculo de autovalores y autovectores

Por definición, dada una matriz simétrica $A \in R^{nxn}$ existe una matriz ortogonal Q tal que:

$$Q^T AQ = Diag(\lambda_1,...,\lambda_n) \qquad (1)$$

donde λ_i con $i = 1,..,n$, son los autovalores de la matriz A, y Q es la matriz con los autovectores correspondientes.

2.1 Método de Jacobi clásico

Una forma de calcular los autovalores y autovectores de una matriz simétrica cuadrada de números reales, es el método de Jacobi [7].

En esencia, si A es la matriz para la cual se desea calcular los autovalores y autovectores, y V es la matriz identidad, el método de Jacobi realiza una sucesión de modificaciones sobre ellas. Cada modificación hace que la matriz A sea cada vez "más diagonal", hasta llegar a un punto en el cual los elementos que se encuentran fuera de la diagonal principal tienen un valor absoluto lo suficientemente pequeño como para considerarlo cero. En ese momento, termina la secuencia de modificaciones, resultando A en una matriz diagonal cuyos elementos de la diagonal principal son los autovalores y V en la matriz que contiene a los autovectores.

El algoritmo correspondiente es el siguiente:

I- Inicialmente V es la matriz identidad

II- Para cada par de índices (p,q) tal que $1 \le p < q \le n$:

II.a- Se realiza la *Rotación de Jacobi*. Para eso, utilizando la *descomposición de Schur* [4], se obtiene un par de valores (c,s), tal que:

$$\begin{bmatrix} b_{pp} & 0 \\ 0 & b_{qq} \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T * \begin{bmatrix} a_{pp} & a_{pq} \\ a_{qp} & a_{qq} \end{bmatrix} * \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \qquad (2)$$

Se forma $J(p,q)$, matriz de nxn, producto de reemplazar en la matriz identidad, c por los elementos de las posiciones (p,p) y (q,q), s por el elemento de la posición (p,q), y $-s$ por el elemento de la posición (q,p), obteniendo entonces la matriz

$$J(p,q)=\begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \begin{matrix} \\ \\ p \\ \\ q \\ \\ \\ \end{matrix} \qquad (3)$$

$$\qquad\qquad\quad p \qquad\quad q$$

II.b- Reemplazar el valor de la matriz A por el resultado de la expresión

$$J(p,q)^T \, A \, J(p,q) \qquad (4)$$

II.c- Reemplazar el valor de la matriz V por el resultado de la expresión

$$V \, J(p,q) \qquad (5)$$

III- Si los elementos a_{ij} de la matriz A, con $i \neq j$, $i=1,..,n$, $j=1,..,n$, no tienen un valor suficientemente pequeño como para ser considerado cero, entonces volver al paso II.

El algoritmo termina con los autovectores en la matriz V y los autovalores correspondientes en la diagonal principal de A.

2.2 Optimización del método de Jacobi clásico

Este algoritmo podría estar en el orden $O(n^5)$ o mayor aún dependiendo de los valores de la matriz A. Si se pretende aplicarlo a matrices de tamaños relativamente grandes, se hace imprescindible reducir los tiempos de procesamiento necesarios.

Para ello, un primer paso ataca el hecho de que en el método de Jacobi clásico se realizar varias multiplicaciones de matrices para solamente modificar dos filas y dos columnas de una de ellas. Se logra cierta reducción del procesamiento necesario, modificando las filas y columnas mencionadas sin necesidad de trabajar con el resto de los valores, que no afectan a esos datos.

2.3 Método de Jacobi por bloques

Suponiendo que $n = N * r$ y que se particiona la matriz A en NxN bloques de rxr, quedando entonces

$$A=\begin{bmatrix} A_{11} & \cdots & A_{1N} \\ \vdots & & \vdots \\ A_{N1} & \cdots & A_{NN} \end{bmatrix} \qquad (6)$$

donde cada elemento A_{ij} es un bloque de rxr valores reales.

El algoritmo por bloques es semejante al Jacobi clásico optimizado, pero considerando que mientras antes el par de índices (p,q) indicaba la posición de un valor real a_{pq} dentro de la matriz A, ahora hace referencia a la posición de un bloque A_{pq}. Por lo tanto:

a- El rango dentro del cual varían los índices p y q ya no es $[1,..,n]$ sino $[1,..,N]$

b- En la Rotación de Jacobi (paso II.a del algoritmo de Jacobi clásico), en vez de la descomposición de Schur que permite obtener un par de valores (c,s) que cumple con (2), se utiliza la descomposición que permite obtener los bloques V_{pp}, V_{pq}, V_{qp} y V_{qq} tales que

$$\begin{bmatrix} D_{pp} & 0 \\ 0 & D_{qq} \end{bmatrix} = \underbrace{\begin{bmatrix} V_{pp} & V_{pq} \\ V_{qp} & V_{qq} \end{bmatrix}}_{V^{[p,q]}}{}^{T} * \underbrace{\begin{bmatrix} A_{pp} & A_{pq} \\ A_{qp} & A_{qq} \end{bmatrix}}_{A^{[p,q]}} * \begin{bmatrix} V_{pp} & V_{pq} \\ V_{qp} & V_{qq} \end{bmatrix} \qquad (7)$$

La submatriz $V^{[p,q]}$ son los autovectores de $A^{[p,q]}$, y $Diag(D_{pp}) \cup Diag(D_{qq})$ son los autovalores, con D_{pp} y D_{qq} matrices diagonales. Así, para obtener $V^{[p,q]}$, en vez de utilizar la descomposición de Schur se usa el método de Jacobi clásico optimizado. Con los bloques V_{pp}, V_{pq}, V_{qp} y V_{qq} obtenidos, se forma la matriz $J(p,q)$.

2.4 Método de Jacobi por bloques paralelo

Este método es la contribución básica de este trabajo. Se encontró que el método de Jacobi por bloques resulta ser muy adecuado para el cálculo de autovalores y autovectores en paralelo, debido a que en cada rotación de Jacobi, los bloques de las filas y las columnas p y q se modifican recién en los pasos II.b y II.c.

Por lo tanto, para cada par de índices (p_1, q_1) y (p_2, q_2), es posible realizar simultáneamente la secuencia de pasos II.a y II.c de la rotación de Jacobi sin que se produzca ningún conflicto entre ellos, con la única condición de que los valores de p_1, p_2, q_1 y q_2 sean distintos entre sí.

Sólo en el paso II.b aparece la necesidad de exclusión mutua sobre la matriz A, dado que es el momento en el cual la misma es modificada.

Por inducción se puede demostrar que esta propiedad es extensible a $N/2$ pares de índices. Es decir que también se cumple para:

$$(p_1, q_1),..,(p_{N/2}, q_{N/2}) \tag{8}$$

siempre que se cumplan las siguientes condiciones:

$$p_i \neq p_j, \forall i, j = 1..N/2, i \neq j$$
$$p_i \neq q_j, \forall i, j = 1..N/2$$
$$q_i \neq q_j, \forall i, j = 1..N/2, i \neq j$$

Esta propiedad indica que pueden ejecutarse simultáneamente $N/2$ rotaciones de Jacobi (pasos II.a y II.c), y luego se realizan las modificaciones de la matriz A (paso II.b) en forma secuencial para cada par de índices.

2.5 Optimización con BLAS

En los algoritmos por bloque, tanto los secuenciales como el paralelo, las modificaciones correspondientes a los pasos II.b y II.c que se realizan sobre las filas y las columnas p y q de las matrices A y V, involucran una sucesión de multiplicaciones entre bloques de matrices, que no son necesarias en los métodos que no utilizan bloques.

El uso de librerías existentes de optimización permiten lograr el mejor rendimiento de una arquitectura en particular para las operaciones convencionales sobre matrices. En particular, en este trabajo se utilizó la función cblas_sgemm para reducir el tiempo de procesamiento que insumen las multiplicaciones de los bloques de las matrices, obteniendo de esta manera el mejor rendimiento posible para la multiplicación de matrices sobre la arquitectura utilizada.

3 Experimentación

La fase de experimentación de este trabajo incluyó la prueba de los siguientes casos:

(a) Método de Jacobi clásico optimizado

(b) Método de Jacobi por bloques sin Blas

(c) Método de Jacobi por bloques con Blas

(d) Método de Jacobi por bloques paralelo con Blas

Los cuatro algoritmos fueron probados con matrices cuadradas de tamaño *nxn* con $n = 100,200,300,400,500,600,700,800,900,1000$.

En los algoritmos por bloques secuenciales se hizo variar el tamaño de los bloques entre 5 y 500 con pequeñas variaciones entre uno y otro.

Se debe considerar que el tiempo de procesamiento necesario para obtener los autovectores a través del método Jacobi, no depende solamente del tamaño de la matriz, sino también de los datos que hay en ella. Se probaron todos los casos con los mismos datos para evitar las posibles influencias de la distribución de los datos. Claramente el resultado de las pruebas realizadas puede variar si se utilizan distintas distribuciones de datos.

Los algoritmos correspondientes a los distintos métodos probados fueron desarrollados con el lenguaje de programación Ada, y se los ejecutó sobre la supercomputadora SGI Origin 2000 [10] de cuarenta procesadores, conocida bajo el nombre de Clementina 2 [8][9], la cual cuenta con cuarenta procesadores MIPS RISC R12000 de 300 MHz., un cache secundario de 4 MB, una memoria principal de 10 GB compartida por todos los procesadores y 360 GB de almacenamiento en disco, y su sistema operativo es el Irix 6.5 [11].

El tiempo utilizado por los algoritmos secuenciales, se calculó en base al tiempo de uso efectivo de procesadores, el cual es informado por el sistema operativo. Este método de medición no sirve para el caso de los algoritmos paralelos debido a que el tiempo de respuesta depende de la carga de trabajo que tenga cada uno de los procesadores del sistema.

3.1 Resultados obtenidos

La Tabla 1 muestra los tiempos de respuesta obtenidos con el método de Jacobi clásico optimizado, caso de prueba (a), para cada uno de los tamaños de matriz utilizados.

Tabla 1: Método de Jacobi Clásico Optimizado

Tamaño de la matriz	Tiempo de respuesta (seg.)
100	1,281
200	11,480
300	41,808
400	111,066
500	232,260
600	428,852
700	746,798
800	1094,764
900	1634,949
1000	2281,703

Para el caso de prueba (b), los tiempos obtenidos son los que se muestran en la Tabla 2. No fue necesario realizar las mediciones de todas las combinaciones posibles de tamaños de matriz y de bloques, resultando suficiente la muestra tomada. En ella puede verse que el tiempo de respuesta es inversamente proporcional al tamaño de los bloques utilizados. Por lo tanto el mejor tiempo de respuesta se obtiene cuando se utiliza el tamaño de bloque máximo. Es decir que para una matriz de nxn, el mejor tamaño de bloque es rxr con $n = 2 * r$.

Tabla 2: Método de Jacobi por bloques sin Blas

Tam. Matriz / Tam. Bloque	100	200	300	400	500	600	700	800	900	1000
10	5,959									
25	4,455	45,353								
50	1,447	36,384	137,869	365,230	789,346		2359,443			
75			123,875							
100		12,625		293,904		1212,386		3227,975		
125					639,436					6380,442
150			45,173			1117,207			4204,082	
175							1781,717			
200				119,455				2754,785		
225									3891,822	
250					251,921					5516,127
300						464,495				
350							781,436			
400								1199,372		
450									1749,360	
500										2488,214

La Tabla 3 muestra los resultados obtenidos para el método de Jacobi por bloques utilizando Blas. En este caso fue necesario un mayor número de pruebas que en el caso anterior para precisar el tamaño de bloque óptimo para cada tamaño de matriz.

Tabla 3: Método de Jacobi por bloques con Blas

Tam. Matriz / Tam. Bloque	100	200	300	400	500	600	700	800	900	1000
5	1,865	14,316	48,710	123,674	249,648	423,307	721,985	1086,669	1531,781	2112,167
10	1,886	11,111	34,380	80,299	156,531	277,886	431,312	674,344	935,106	1284,376
25	2,367	14,793	41,677	85,646	145,031	245,786	352,080	489,168	695,117	909,138
50	1,304	19,205	54,747	121,438	209,375	332,174	497,340	685,709	846,915	1169,318
75			64,514							
100		11,458		155,695		466,817		988,419		1822,365
125					326,577					1991,821
150			41,865			588,439			1627,207	
175							925,107			
200				110,845				1469,498		
225									2069,692	
250					230,299					2999,736
300						434,975				
350							724,982			
400								1104,597		
450									1617,041	
500										2307,555

Se encuentra que el tamaño de bloque para el cual se logra el mejor rendimiento depende del tamaño de la matriz, pudiendo identificarse tres rangos. Para matrices de tamaño 100, el mejor rendimiento se obtiene con bloques del mayor tamaño posible, es decir 50. Para matrices de tamaño 200, 300 y 400, el mejor tamaño de bloque es 10, mientras que para los tamaños 500 a 1000 el mejor rendimiento se obtiene con bloques de tamaño 25.

Las pruebas del método de Jacobi por bloques paralelo con Blas fueron realizadas utilizando los tamaños de bloque óptimos para cada tamaño de matriz, obtenidos en las pruebas del método anterior. Los tiempos de respuesta medidos así como el tamaño de bloque y la cantidad de tareas utilizados se muestran en la Tabla 4.

Tabla 4: Método de Jacobi por bloques paralelo con Blas

Tamaño de la matriz	**Tiempo de respuesta (seg.)**	**Tamaño de bloque óptimo**	**Cantidad tareas**
100	1,278	50	1
200	8,304	10	10
300	26,531	10	15
400	64,793	10	20
500	82,033	25	10
600	130,225	25	12
700	193,719	25	14
800	252,635	25	16
900	382,657	25	18
1000	499,879	25	20

Finalmente en la Tabla 5 se resumen los mejores tiempos de respuesta obtenidos por los distintos métodos para cada tamaño de matriz. Para los métodos por bloque se utilizan los tiempos de respuesta obtenidos con los mejores tamaños de bloque.

En el gráfico de la Figura 1 se comparan las curvas correspondientes a los tiempos obtenidos por los cuatro métodos.

Tabla 5: Mejores tiempos de respuesta (seg.)

Método / Tam. Matriz	Secuencial	Bloques sin Blas	Bloques con Blas	Bloques paralelo
100	1,281	1,447	1,304	1,278
200	11,480	12,625	11,111	8,304
300	41,808	45,173	34,380	26,531
400	111,066	119,455	80,299	64,793
500	232,260	251,921	145,031	82,033
600	428,852	464,495	245,786	130,225
700	746,798	781,436	352,080	193,719
800	1094,764	1199,372	489,168	252,635
900	1634,949	1749,360	695,117	382,657
1000	2281,703	2488,214	909,138	499,879

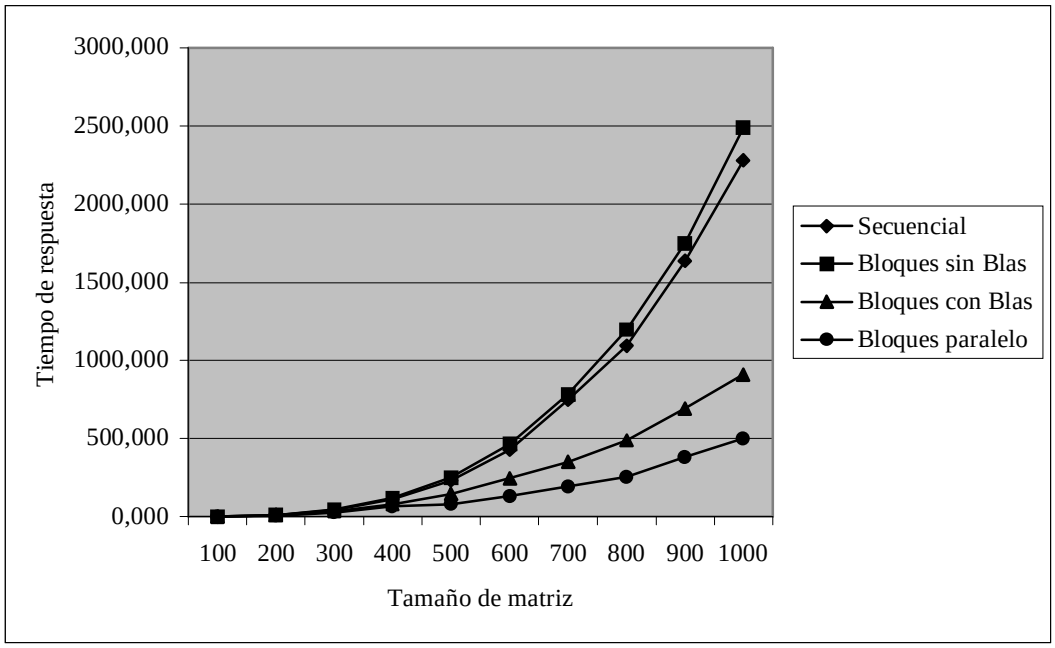

Figura 1: Tiempos de los distintos métodos comparados

De la experimentación realizada se desprende que:

- El método secuencial muestra una forma del tipo polinomial. El tiempo de respuesta crece más rápidamente que el tamaño de la matriz.
- El método por bloques sin Blas no sólo no mejora los tiempos de respuesta obtenidos por el método clásico optimizado sino que los empeora. El tamaño de bloque óptimo encontrado para el método por bloques sin Blas resultó ser el mayor posible, es decir de la mitad del tamaño de la matriz. En este caso, el método por bloques resulta algorítmicamente igual al clásico optimizado, y por esta razón obtiene un rendimiento similar más un pequeño costo adicional (overhead).
- El método por bloques con Blas obtiene un mejora sustancial en el rendimiento sacando provecho de la cantidad de multiplicaciones de matrices que se deben realizar y de la optimización de las mismas, encontrando el mejor tamaño de bloque en el compromiso (trade off) entre el tamaño de las matrices a multiplicar y la cantidad de multiplicaciones necesarias.
- El método por bloque paralelo logra mejor rendimiento que los demás métodos. Es interesante destacar que para este método los tiempos medidos son tiempos máximos, es decir que podrían mejorarse sensiblemente si se tuviera acceso exclusivo a la arquitectura.

4 Aplicación al sistema de reconocimiento de rostros

Habiendo probado la conveniencia de paralelizar el cálculo de autovalores y autovectores, se probó el sistema de reconocimiento de rostros, incorporando en el proceso de aprendizaje los métodos por bloques con Blas y por bloques paralelo en reemplazo del clásico optimizado. Se utilizaron para ello matrices de $100x100$ y de $400x400$.

Con matrices de $100x100$ los tiempos de aprendizaje fueron:

Tabla 6: Matrices de 100x100

Método clásico optimizado	1,538 seg.
Método por bloques con Blas	3,046 seg.
Método por bloques paralelo	1,546 seg.

Con matrices de $400x400$ los tiempos de aprendizaje fueron:

Tabla 7: Matrices de 400x400

Método clásico optimizado	109,894 seg.
Método por bloques con Blas	105,139 seg.
Método por bloques paralelo	98,047 seg.

Tal como lo probado en la experimentación anterior, con un tamaño de $100x100$ no se logra ninguna mejora. Por otro lado, con un tamaño de $400x400$ se observan mejoras en el rendimiento aunque no en las proporciones que se podrían prever, debido a la dependencia de la distribución de los valores de la matriz.

5 Conclusiones

Se probaron distintos métodos de cálculo de autovalores y autovectores y se compararon con el método propuesto, método por bloques paralelo. El método propuesto aventaja considerablemente a los demás e insinúa que esta ventaja se acentúa con el crecimiento del tamaño de la matriz.

Sin embargo no es posible determinar exactamente la mejora en el rendimiento debido a dos razones fundamentales. No se dispone de acceso exclusivo a la supercomputadora; por lo tanto, los tiempos medidos son tiempos máximos. Esto significa que se pueden obtener mejores rendimientos que los medidos. Por otro lado, la cantidad de cómputo necesario con estos métodos depende de la distribución de los valores dentro de la matriz. Con respecto a esto no hay mucho que se pueda hacer.

Por otro lado, con este trabajo se completó la versión paralela del sistema de reconocimiento de rostros quedando así actualmente operativa sobre la arquitectura mencionada.

Como trabajo futuro se está evaluando la migración del sistema de reconocimientos de rostros paralelo al modelo de memoria distribuida de un cluster de microprocesadores sobre el cual se dispone acceso exclusivo.

6 Referencias bibliográficas

[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] Champredonde R., Chichizola F. “Parallelizing Algorithms in Ada on Clementina II. Face Recognition System”. Proceeding del VII Congreso Argentino de Ciencias de la Computación (CACIC 2001). Calafate, Santa Cruz. Octubre de 2001.

[3] Correa Martín, Chichizola Franco. “Sistema de reconocimiento de rostros”. Trabajo de grado. Facultad de Informática. 2001.

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

[5] 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.

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

[7] Rutishauser H. “The Jacobi Method for Real Symmetric Matrices”. Numer. Math. 9, 1-10. 1966,

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

[9] www.setcip.gov.ar

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

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