# Compresión de Imágenes usando la Transformada de Hadamard

V.G. Ruiz*, I. García
Departamento de Arquitectura de Computadores y Electrónica, Universidad de Almería
04120 Almería. SPAIN

**Resumen**

En este trabajo presentamos la implementación paralela de un algoritmo para la compresión de datos, aplicada a estructuras bidimensionales (imágenes), basado en la Transformada de Hadamard. La ventaja de esta frente a la Transformada Coseno, es la de ser computacionalmente menos costosa, debido a que no requiere cálculos reales. La enorme cantidad de datos que usualmente se manejan en el procesamiento de imágenes, su almacenamiento y transmisión junto al alto grado de paralelismo de este algoritmo, nos ha permitido obtener una implementación paralela que presenta una alta eficiencia. Los resultados que presentamos son los obtenidos de la ejecución del algoritmo sobre un sistema multiprocesador Meiko Computer Surface con 16 procesadores i860.

**Palabras clave:** Transformada de Hadamard, compresión de imágenes irreversible, paralelización, multiprocesadores.

## 1 Introducción

La compresión de imágenes aborda el problema de reducir la cantidad de datos requeridos para almacenar una imagen. Cualquier proceso de compresión se basa en la eliminación de datos redundantes[8, 6, 1]. Desde un punto de vista estadístico, el proceso consiste en transformar las imágenes en un conjunto de datos no correlacionados.

Operacionalmente, antes de procederse a la transmisión o almacenamiento de las imágenes, estas deben someterse a un proceso de compresión. El proceso inverso de descompresión tendrá que aplicarse sobre los datos para obtener la reconstrucción de la imagen original, o bien una buena aproximación de ésta.

Las técnicas de compresión pueden dividirse en dos categorías: "error-free" o preservadoras de la información y "lossy" o no preservadoras de toda la información[8, 6, 1]. Las primeras se utilizan en la compresión de cualquier tipo de información expresada de forma binaria, mientras que la segunda tiene mayor utilidad en la compresión de datos muestreados tales como sonidos, imágenes, etc.. Esto se debe, fundamentalmente, a que parte de la información contenida en una imagen puede ser eliminada sin una perdida apreciable de la calidad de dicha imagen. Por tanto, la compresión de imágenes es un preproceso adecuado previo a la transmisión de señales de TV, videoconferencia, fax, etc..

El término de compresión se refiere al proceso de reducir la cantidad de datos necesarios para representar una determinada información. En este sentido hay una clara distinción entre "dato" e "información"[8]. De hecho, bloques diferentes de datos pueden representar la misma información. En la caracterización del proceso de compresión se define la redundancia relativa de un conjunto de datos S1 respecto de otro conjunto S2, representados por n1 y n2 unidades de información respectivamente, como:

$$Rd = 1 - 1/Cn \tag{1}$$

donde $Cn$ es la razón de compresión:

$$Cn = n1/n2 \tag{2}$$

Valores habituales para la razón de compresión suelen estar alrededor de 10 (10:1), lo que supone una redundancia relativa de 0.9 (90%).

En el caso de compresión de imágenes, aparecen descritos en la bibliografía tres tipos diferentes de redundancia[8], que pueden ser abordados de diferentes formas: redundancia en la codificación, redundancia interpixel, y redundancia psico-visual. Para situar los algoritmos de compresión basados en métodos transformados, veremos rápidamente en que consisten estas redundancias y como pueden ser eliminadas.

*vruiz@gogh.ualm.es

1. **Redundancia en la codificación:** Una imagen presenta este tipo de redundancia si y sólo si, su histograma de colores no es plano. Desde un punto de vista estadístico, este tipo de redundancia puede expresarse por el primer estimador de la entropía de la imagen. Es estos casos, es interesante asignar al color más frecuente el código de mínima longitud, mientras que al color menos frecuente se le asignara el código con un número de bits mayor. Este tipo de compresión se llama codificación de longitud variable (variable-length coding). Un método general para encontrar este código es el propuesto por Huffman en 1952, denominado Código de Huffman[8].

2. **Redundancia interpixel:** Esta forma de redundancia está asociada a los excesos de datos introducidos por la forma en la que los pixeles de una imagen están relacionados entre si. En general, la probabilidad de aparición de uno o más pixeles conociendo el o los que ya han aparecido no es siempre constante. Siempre existe algún tipo de correlación estadística que puede ser calculada por los estimadores superiores de la entropía. La redundancia interpixel engloba a otras muchas, entre las que podemos destacar la redundancia espacial[8, 6] y la redundancia temporal. Existe una amplia variedad de algoritmos que intentan eliminar la redundancia interpixel. Entre ellos podríamos destacar:

   (a) Codificación run-length que codifica las cadenas constantes de pixeles[8].

   (b) Codificación de áreas constantes (CAC) que identifica área a un nivel constante en bitmaps[8].

   (c) Codificación de trazado de contornos que analiza imágenes bitmap por las fronteras que definen las agrupaciones de pixels[8].

   (d) Codificación predictiva sin error (lossless predictive coding) que codifica información relacionada con el error cometido en cada predicción[8].

   (e) El algoritmo LZW (Lempel-Ziv Welch) que detecta las repeticiones de los pixeles y patrones más frecuentes[11].

   En general, todos los algoritmos que eliminan la redundancia interpixel se conocen como algoritmos no destructivos. Todos ellos recuperan los datos originales después del proceso de descompresión. Esto es deseable si se pretende reducir la cantidad de datos cuando no es posible una reducción en la cantidad de información.

3. **La redundancia psico-visual:** Este tipo de redundancia está relacionado con la sensibilidad del ojo a la información visual. Existe cierta información que puede ser eliminada sin que se produzca un deterioro significativo de la calidad de la imagen. La percepción humana de la información en una imagen no realiza un análisis cuantitativo de cada pixel o del valor de la luminancia en la imagen. En general, un observador intenta fijarse en características tales como las fronteras, aristas o texturas de las regiones, y las combina mentalmente para realizar asociaciones con otros grupos de patrones conocidos. El cerebro correlaciona esta información con el conocimiento anterior y así realiza la interpretación de la imagen[8].

   La redundancia psico-visual está asociada con la información cuantificable o real. Su eliminación es posible sólo debido a que esta no es esencial en el procesamiento visual. Representa una pérdida cuantitativa irreversible de la información, por lo que comúnmente se llama cuantificación. Esta terminología es consistente con el uso que normalmente se hace de la palabra cuantificación, que generalmente significa la proyección de un amplio rango de valores de entrada sobre un número limitado de valores de salida[6].

   Una forma de eliminar la redundancia psico-visual consiste en reducir el número de colores de los que disfruta una imagen. Si estamos trabajando con imágenes en 256 niveles de gris, eliminaremos, por ejemplo, los 4 bits menos significativos de cada byte con lo que generamos una compresión 2:1. Debido a esta cuantificación pueden aparecer contornos perceptibles por el ojo humano. Una forma de eliminar este efecto es introducir un bajo nivel de ruido aditivo a cada pixel de la imagen de forma que se aleatoricen las fronteras de estos contornos. Esta última forma de cuantificar recibe el nombre de cuantificación IGS (Improved Gray-Scale)[8].

   Otra forma más sutil de eliminar este tipo de redundancia la llevan a cabo los algoritmos de codificación predictiva con perdida de información (lossy predictive coding)[8]. La diferencia que existe con respecto a la codificación predictiva sin pérdida de información es que el codificador cuantifica su salida. Este elemento es el responsable de la pérdida de información.

   Por último, otro gran grupo de algoritmos para la compresión de imágenes, que eliminan la redundancia psico-visual, está formado por codificadores basados en transformadas[5, 8]. Se emplea una

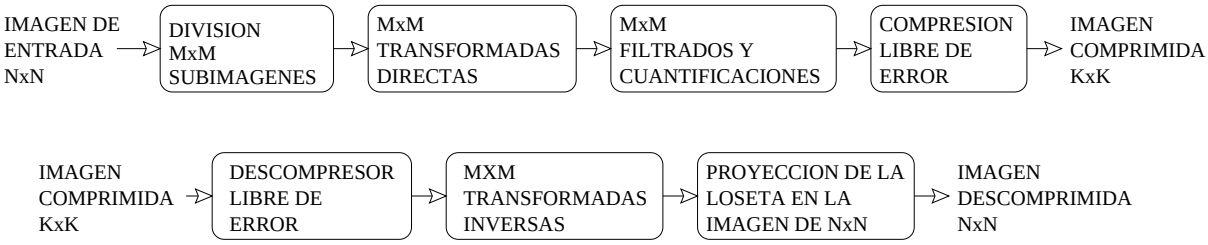


Figura 1: Fases de la compresión/expansión de la imagen.

transformada lineal y reversible (por ejemplo la transformada de Fourier) para proyectar la imagen sobre un conjunto de coeficientes transformados, cuyo orden ya no tiene relación con el dominio espacial al que pertenece la imagen original. La compresión se lleva a cabo cuando los datos en el dominio transformado son filtrados y cuantificados. Posteriormente se codifican mediante otro algoritmo de compresión libre de error. Para algunos de los coeficientes espectrales, la cuantificación es tan dramática, que sencillamente se hacen cero. Un simple filtro paso banda bidimensional es suficiente para realizar la compresión frecuencial, con un deterioro mínimo de la imagen.

En general, los procesos de compresión y descompresión se descompone en las operaciones descritas en la figura 1. Una imagen se descompone en un conjunto de MxM subimagenes de dimensión PxP, siendo P=N/M. Cada una de estas subimagenes (losetas) pasa a describirse en un dominio transformado del espacio de frecuencias por medio de una transformada. En el dominio frecuencial la información se somete a un filtrado pasa banda, que es el adecuado para la eliminación de la redundancia psico-visual. La etapa de cuantificación terminará el proceso de compresión de la imagen. El proceso inverso de descompresión esta compuesto por las mismas etapas que el de compresión excepto la etapa de filtrado que, evidentemente, es innecesario.

Sistemas de compresión-descompresión[7, 2] basados en las transformadas de Karhunen-Love (KLT)[8], Discreta de Fourier (DFT)[8, 12], Discreta Coseno (DCT)[7, 2, 3, 12], Walsh-Hadamard (WHT) y otras han sido construidos. La elección de una de ellas dependerá del máximo error permisible en el proceso de compresión y del gasto computacional que supone. Aunque el proceso de pérdida de información (y de compresión) se realiza en la etapa de cuantificación, el nivel de correlación alcanzado es diferente para cada una de ellas. Por este orden: KLT, DCT, WHT y DFT, las 4 transformadas tienen la propiedad de aglomerar la máxima cantidad de información en la mínima cantidad de coeficientes. KLT es la óptima, pero presenta el problema de ser una transformación dependiente de los datos de entrada, lo que constituye una tarea computacional no demasiado trivial. DFT padece del fenómeno de Gibbs cuando se produce la cuantificación de los coeficientes de Fourier, y esto provoca aparición de fronteras entre las subimagenes en las que se ha dividido la imagen original[8].

Para elegir entre DCT y WHT estudiaremos una razón de compromiso entre la cantidad de error introducida en la compresión y la velocidad de cálculo. DCT se aproxima más al óptimo KLT. Es menos ruidosa debido a su carácter sinusoidal (y por eso se utiliza en el estándar de compresión JPEG) pero requiere cálculos reales. En contrapartida, WHT únicamente utiliza aritmética entera y ni siquiera presenta productos enteros. Este es un factor a tener en cuenta si se pretende aumentar la velocidad de compresión y descompresión y este es el motivo por el que hemos seleccionado WHT para el desarrollo de nuestro algoritmo.

El resto del trabajo está organizado de la siguiente forma. En la sección 2 presentamos las ideas de los algoritmos FHT y barajamiento perfecto. En la sección 3 describimos el filtrado en el espacio de Hadamard y en la 4 la medida del error. La sección 5 esta dedicada a la descripción del algoritmo paralelo. Finalmente, en la sección 5.1 presentamos un ejemplo de la aplicación de nuestro algoritmo para varios niveles de compresión y en la 5.2 analizamos el rendimiento de su implementación paralela.

# 2 Algoritmo de la transformada rápida de Hadamard n-dimensional

Consideremos una señal unidimensional muestreada sobre un vector $x(i)$, $i = 0, 1, ...N-1$ de $N$ componentes. La transformada de Hadamard[8, 9] de $x(i)$ se define como el vector $X(u)$, $u = 0, 1, ..N-1$

donde:

$$X(u) = \sum_{i=0}^{N-1} x(i) \times (-1)^{b(i,u)} \tag{3}$$

La transformación inversa mediante:

$$x(i) = \frac{1}{N} \sum_{u=0}^{N-1} X(u) \times (-1)^{b(i,u)} \tag{4}$$

donde:

$$b(i,u) = \sum_{h=0}^{\log_2(N)-1} bh(i) \times bh(u) \tag{5}$$

siendo $bh(i)$ = el $h$-ésimo bit del valor $i$.

Podemos ver que las transformadas directa e inversa son idénticas, excepto que la inversa debe de ser ponderada dividiendo cada uno de los componentes del vector resultado, $x(i)$, entre el número de componentes del vector ($N$). Además, si precalculamos los valores:

$$(-1)^b(i,j) \quad i,j = 0,1,2,4 \tag{6}$$

la transformada se compone solamente de sumas y restas de valores enteros. A pesar de tener estos valores precalculados para sucesivas transformaciones, la complejidad del algoritmo descrito es de $N^2$. La transformada unidimensional puede calcularse multiplicando el vector $x(i)$ por una matriz cuadrada. Se trata, por lo tanto, de resolver el problema de multiplicar una matriz cuadrada por un vector. La matriz característica de la Transformada de Hadamard se construye recursivamente de la siguiente forma:

1. El caso $N = 1$, es trivial: $x(0) = X(0)$.

2. El caso $N = 2$, la matriz de transformación se define:

$$H_0 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{7}$$

3. El caso $N = 4$, la matriz característica se construye como:

$$H_1 = \begin{pmatrix} H_0 & H_0 \\ H_0 & -H_0 \end{pmatrix} \tag{8}$$

   donde $H_0$ es la expresión 7. Además se cumple que:

$$H_1 = (-1)^{b(i,j)} \quad i,j = 0,1,2,3 \tag{9}$$

Como ya hemos comentado esta solución presenta una complejidad computacional de $N = 2^n$. La idea que subyace detrás del algoritmo de la Transformada rápida de Hadamard (FHT) es la misma que aparece en la Transformada rápida de Fourier (FFT) y consiste en dividir la secuencia original de N muestras en dos secuencias más pequeñas[2, 10]. La Transformada de Hadamard de estas subsecuencias puede combinarse para obtener la Transformada de la secuencia original. La aplicación recursiva de este procedimiento, cuando $N$ es una potencia de 2 ($N = 2^n$) necesita realizar $N \times \log_2(N)$ operaciones. El algoritmo que implementamos en este trabajo para la Transformada rápida de Hadamard se realiza "in place" y con "radix" dos[2].

Otro aspecto que es necesario considerar es la aplicación de un barajamiento perfecto a los datos de entrada para de esa forma obtener una secuencia de salida ordenada[2].

Por último, teniendo en cuenta la propiedad de separabilidad de la transformada multidimensional, podremos implementar el algoritmo multidimensional aplicando el par de operaciones barajamiento perfecto 1D-FHT a cada una de las dimensiones del espacio.

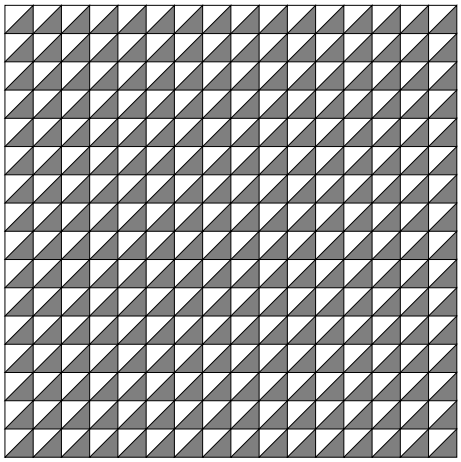

Figura 2: Filtro paso banda 2D.

# 3 El filtrado de los coeficientes: la compresión

Existe una gran variedad de filtros bidimensionales paso banda utilizables en la compresión de imágenes[7, 8, 6, 3], pero en general, se pueden dividir entre los que necesitan que los coeficientes espectrales se encuentren ordenados y en los que no lo necesitan. El buen orden al que nos referimos viene dado por el algoritmo de ordenación Perfect Shuffle[2]. Si este no se aplica, un filtrado paso baja bidimensional carece de sentido, siendo en este caso aconsejable conservar aquellos coeficientes que tengan máxima amplitud espectral. Esto puede hacerse de dos formas distintas, o bien, se eliminan los coeficientes que no superen un umbral (que controla en nivel de compresión), o bien se conservan aquellos coeficientes que presentan una varianza máxima. El problema del primer método es evidente: deben de almacenarse con la imagen comprimida[8] un array con tantos bits como coeficientes espectrales existen, donde un bit indica la presencia o ausencia de ese coeficiente espectral. Sin embargo, en el método de la maximación de la varianza, el array tendrá tantos bits como coeficientes existen en una loseta.

Por otra parte, si hemos reordenado los coeficientes espectrales, técnicas de filtrado bidimensionales son aplicables, con lo que el filtro no debe de almacenarse con la imagen comprimida[7, 8]. La energía espectral se concentra en los coeficientes asociados a las frecuencias más bajas, por lo que un filtrado como el que se presenta en la figura 2, parece más razonable. Se trata de un filtro bidimensional de 256 filas por 256 columnas. La imagen sobre el que se aplica debe de haber sido descompuesta en losetas de 16x16 pixels. Las zonas blancas corresponden a coeficientes en el espacio de Hadamard que se van a conservar. Como puede observarse, la mitad de los puntos son blancos y la otra mitad negros, por lo que si aplicamos este filtro a una imagen con 256x256 pixels, eliminaremos la mitad de los coeficientes. Se trata de un filtro paso bajo que elimina las bajas frecuencias.

# 4 Cuantificación del error cometido en el proceso de compresión

Existen dos criterios de fidelidad o de parecido entre imágenes: el subjetivo y el objetivo[8]. El primero se basa en la apreciación subjetiva de un observador experto que valorará la calidad de la aproximación entre la imagen original y la procesada. El segundo criterio se basa en aplicar métodos de calculo de error o diferencia entre dos señales.

La mayoría los algoritmos conocidos para el cálculo de error, se basan en obtener la diferencia punto a punto entre la imagen original y la imagen reconstruida, acumular todas estas diferencias y finalmente aplicar alguna transformación sencilla al valor obtenido. De esta forma, obtenemos un número real que utilizamos para decir qué parámetros del algoritmo de compresión obtienen mejores resultados, es decir, el mínimo error. Definimos el error entre dos imágenes $f$ (original) y $f'$ (comprimida) de $N \times N$ pixels:

$$e(x,y) = f(x,y) - f'(x,y); x = 0, 1, \cdots, N-1; y = 0, 1, \cdots, N-1 \tag{10}$$

En la evaluación del error hemos utilizado el valor de la relación señal-ruido (Signal- Noise Relation)[4][7], que viene dado por la siguiente expresión:

Figura 3: Resultados visuales de la compresión: imagen original (izquierda), imagen comprimida al 12.5% (centro) e imagen comprimida al 0.7% (derecha).

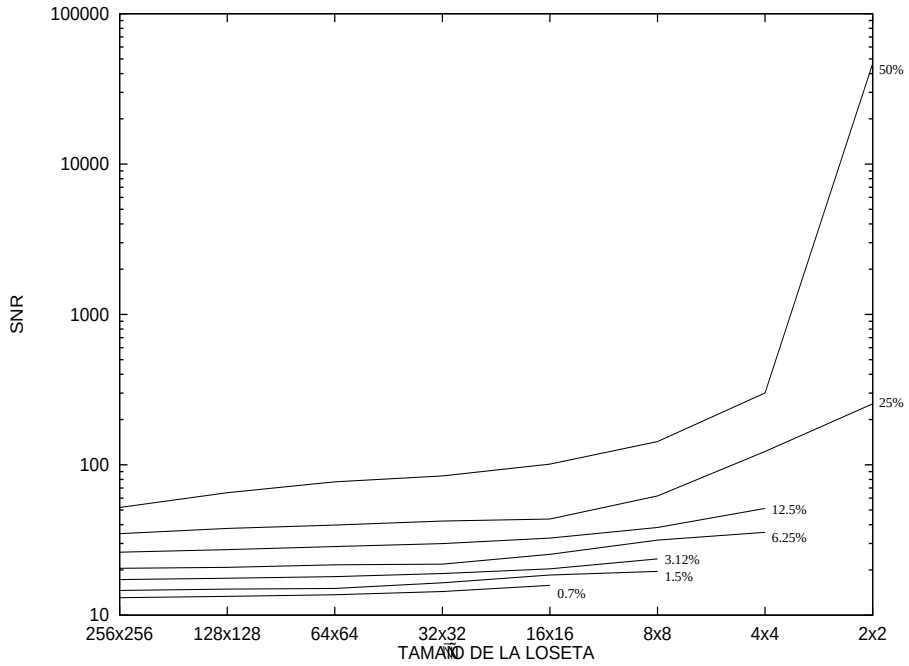


Figura 4: Errores de compresión.

$$SNR = \frac{\sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f'(x,y)^2}{\sum_{x=0}^{N-1} \sum_{y=0}^{N-1} e(x,y)^2} \tag{11}$$

# 5 Paralelización del algoritmo

Como hemos visto en la sección 3, el filtrado paso banda en el espacio de Hadamard se obtiene de la aplicación de un filtro paso baja bidimensional sobre los coeficientes obtenidos al aplicar la transformada a cada una de las subimagenes (losetas) que componen la imagen completa. Bajo estas condiciones el proceso de compresión-descompresión puede describirse algorítmicamente de la siguiente forma:

```
/* Compresion */
DO i=1,...,M
     DO j=1,...,M   /* Losetas de MxM */
          Perfect Shuffle (loseta[i][j]);
```

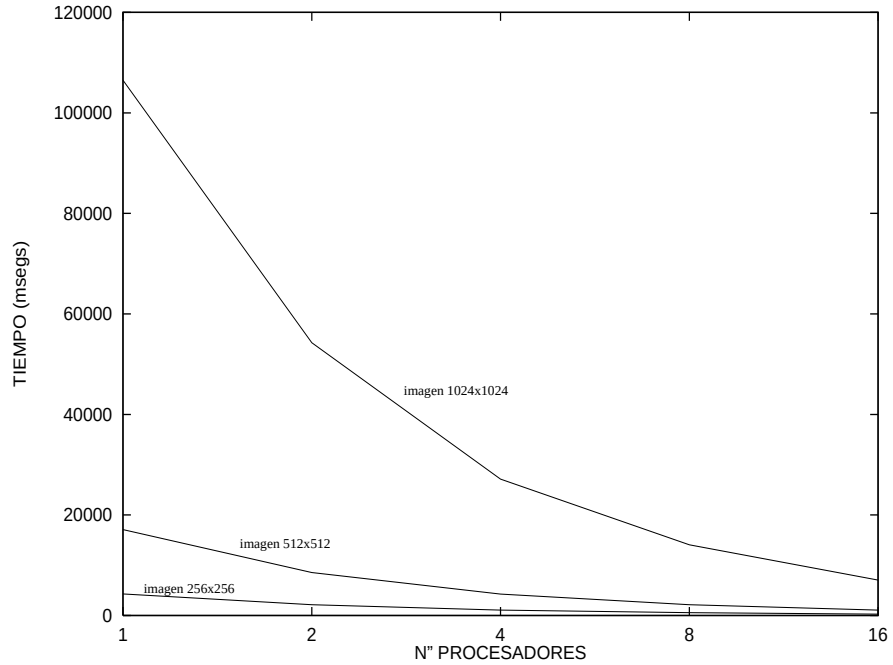


Figura 5: Tiempos de cómputo.

```
          FHT Directa(loseta[i][j]);
          Filtrado 2D(loseta[i][j]);
          Normalizacion&Cuantizacion(loseta[i][j]);
     ENDDO
ENDDO

/* Descompresion */
DO i=1,...,M
     DO j=1,...,M   /* Losetas de MxM */
          Perfect Shuffle (loseta[i][j]);
          FHT Inversa(loseta[i][j]);
          Desnormalizacion(loseta[i][j]);
     ENDDO
ENDDO
```

La paralelización de los 2 lazos DO que aparecen en el algoritmo es trivial ya que no existe ninguna dependencia de datos. Por tanto, partiendo de un sistema con Np procesadores, el reparto de la carga consiste simplemente en el tratamiento de $M^2/Np$ losetas en cada procesador. Por otro lado, hay que destacar que no son necesarias las comunicaciones interprocesador, por lo que como comentaremos en el apartado de evaluación del algoritmo, se obtienen unos rendimientos óptimos. No obstante, la escalabilidad[4] del problema tendría una única limitación, que es la referente al número máximo de procesadores sobre los que el problema puede ser resuelto sin pérdida de eficiencia ni desbalanceo de la carga. Sin embargo, para un tamaño del problema pequeño (imágenes de $256 \times 256$ pixels), todavía la eficiencia continua siendo óptima cuando utilizamos multiprocesadores masivamente paralelos con hasta 1024 procesadores.

La complejidad del algoritmo paralelo es proporcional a la dimensión del problema $N^2$ debido a las funciones Filtrado 2D y Normalización/Cuantificación, es proporcional a $\log_2(N)$ debido a la función Perfect Shuffle y a $2N^2 \times \log_2(N)$ para la FHT, como queda reflejado en la siguiente expresión:

$$O(\frac{(N^2 + \log_2(N) + N^2 \times \log_2(N))}{Np}) \tag{12}$$

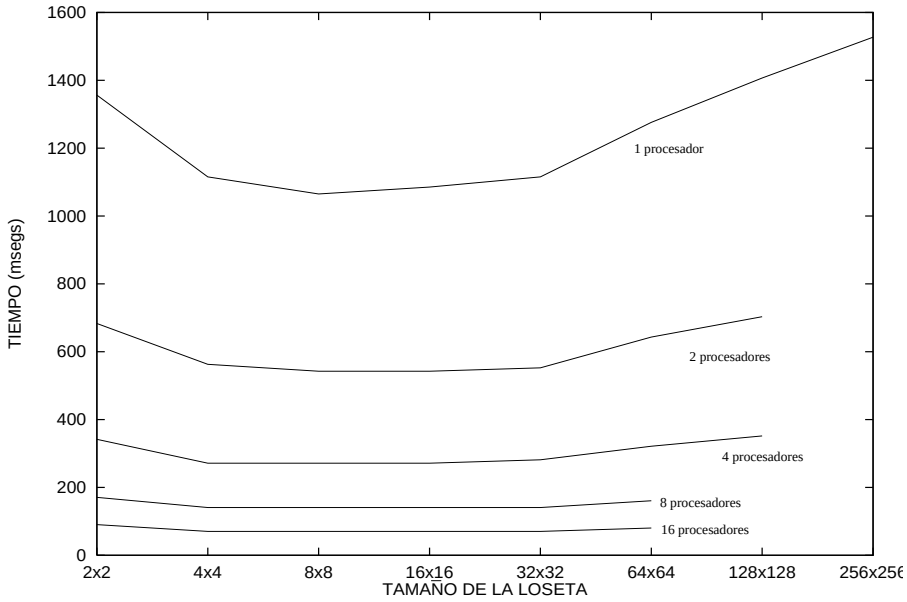


Figura 6: Tiempos de cálculo.

## 5.1 Resultados experimentales

Presentamos en esta sección los resultados obtenidos por aplicación del algoritmo de compresión sobre la imagen representada en la imagen 3 (izquierda). Se trata de un mapa de $256 \times 256$ pixeles, representados cada uno de ellos en 8 bits (256 niveles de grises). La figura 3 muestra además el resultado de la compresión-descompresión de la imagen original utilizando un filtro que conserva el 12.5% y el 0.7% de los coeficientes espectrales respectivamente. En el proceso de compresión hemos dividido la imagen en $16 \times 16$ losetas con $16 \times 16$ pixeles cada una. Los coeficientes se han cuantificado a 8 bits.

La información comprimida, además de los coeficientes de la transformada de Hadamard, incluye los valores máximo y mínimo de cada loseta. Esto significa que tenemos $16 \times 16$ parejas de valores máximo, mínimo. Lo mismo que ocurre en el algoritmo de compresión JPEG, deberemos aplicar un compresor libre de error, con lo que obtendremos el tamaño real de la imagen comprimida en disco.

Una valoración de los errores debidos a la compresión utilizando como parámetro de medida la relación señal-ruido, son los que se muestran en la figura 4. Esta es una representación de la SNR en función del tamaño de las losetas. Hemos representado el error para distintas relaciones de compresión. Como era de esperar, el error disminuye con el valor de la relación de compresión. En esta gráfica puede observarse que para una relación de compresión constante, el error aumenta cuando el tamaño de la loseta aumenta. Esto se debe a que en la fase de normalización el rango de valores con los que trabaja es mayor para tamaños de loseta más grandes y, por tanto, al cuantificar a 8 bits, se pierde más información que cuando se trabaja con un margen más estrecho de valores. Algunos de los coeficientes espectrales no superan el umbral mínimo y por tanto se hacen 0. Esto implica que realmente se está aplicando una relación de compresión mayor que la que en principio se había previsto. Este problema puede eliminarse parcialmente, si trabajamos con imágenes de valor medio 0, o bien con imágenes cuyo rango de valores se encuentre entre [-128, 127] que es el que hemos aplicado para hacerlo independiente de las características de la imagen.

## 5.2 Evaluación

Como se ha comentado anteriormente, el proceso de paralelización del algoritmo no presenta ninguna complejidad conceptual debido a las magníficas características del algoritmo de compresión respecto a la falta de dependencia de datos. En la evaluación del algoritmo hemos trabajado con un multiprocesador Meiko Computer Surface con 16 i860 de Intel. Cada nodo es un procesador vectorial de 64 bits que alcanza los 80 MFLOPS y dispone de 4 MBytes de memoria. La arquitectura responde al modelo MIMD con memoria distribuida y comunicaciones por paso de mensajes. No obstante, nuestra aplicación utiliza un modelo de programación SCMD puesto que todos los procesadores ejecutan la misma tarea sobre datos diferentes[4].

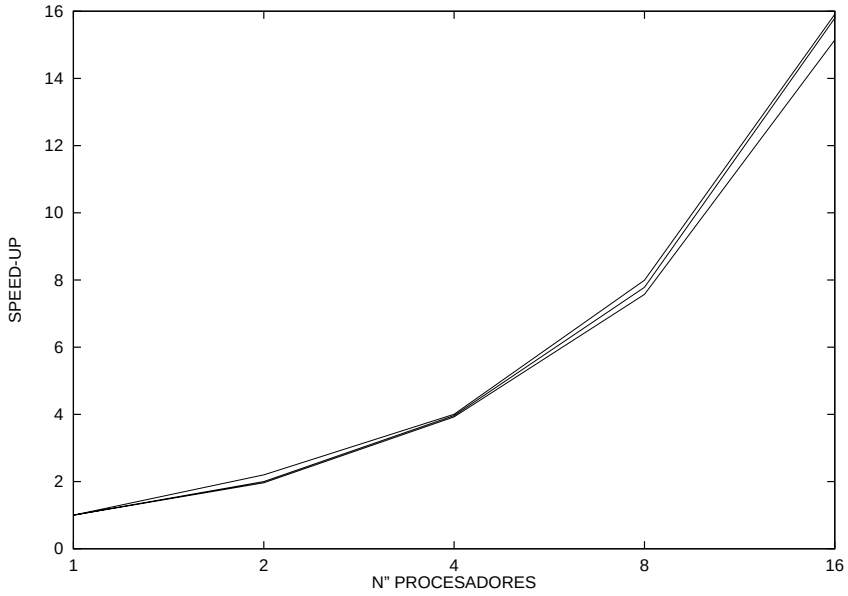


Figura 7: Speed-Up.

En la figura 5 se han representado los tiempos de ejecución en función del número de procesadores utilizados, para varios tamaños de loseta (desde $2 \times 2$ hasta $256 \times 256$). Como puede observarse, para cualquier valor de $Np$, el tiempo de cálculo es mínimo para un tamaño de loseta de $8 \times 8$. La obtención de este valor mínimo está relacionada con la complejidad del algoritmo, que como hemos comentado anteriormente no sigue una relación lineal, sino que aparece un término que es proporcional a $\log_2(N^2)$. De la figura 6 puede comprobarse que la eficiencia de la implementación paralela es siempre superior a 0.94[4].

En la figura 5 hemos representado el tiempo de ejecución en función del número de procesadores para 3 tamaños de problema diferentes ($N = 256$, $N = 512$ y $N = 1024$). Los tiempos fueron tomados para un tamaño de loseta de $8 \times 8$. Como puede observarse el tiempo de ejecución crece proporcionalmente a $N$.

Por último, en la figura 7 hemos representado el speed-up[4] para 3 tamaños de problema. En esta se pone de manifiesto que la eficiencia se acerca, en todos los casos, al valor máximo (1) ya que existe una relación lineal entre el speed-up y el número de procesadores $Np$.

# Referencias

[1] A.C. Kak A. Rosenfeld. *Digital Picture Processing.* Academic Press, 1982.

[2] R.W. Schafer A.V. Oppenheim. *Discrete-Time Signal Processing.* Prentice-Hall Signal Processing Series, 1991.

[3] L.A. Roew B.C. Smith. Algorithms for manipulating compressed images. *IEEE Computer Graphics*, pages 34–42, 1993.

[4] T. Braunl. *Parallel Programming.* Prentice-Hall, 1993.

[5] P.M. Farrelle. *Recursive Block Coding for Image Data Compresion.* Springer-Verlag, 1990.

[6] A.K. Jain. *Fundamentals of Digital Image Processing.* Prentice Hall, 1989.

[7] B. Kimble P.M. Embree. *C language Algorithms for Digital Signal Processing.* Prentice-Hall, 1991.

[8] P. Wintz R.C. Gonzalez. *Digital Image Processing.* Addison-Wesley, 1988.

[9] J.C. Muzio S.L. Hurst, D.M. Miller. *Spectral Techniques in Digital Logic.* Academic Press, 1985.

[10] M. Soumekh. *Fourier Array Imaging.* Prentice-Hall, 1994.

[11] T.A. Welch. A technique for high-performance data compression. *IEEE Computer*, pages 8–19, June 1984.

[12] W.H. Press W.T. Vetterling, S.A. Teukolsky and B.P. Flannery. *Numerical Reciples in C*. Cambridge Univ. Press, 1988.