DIVISION DE ESTUDIOS DE POSGRADO

FACULTAD DE INGENIERIA

IMPLANTACION DE UN SISTEMA DE COMPRESION DE IMAGENES EN UN MICROPROCESADOR DE SEÑALES

MIGUEL MOCTEZUMA FLORES

VICTOR GARCIA GARDUÑO

TESIS

PRESENTADA A LA DIVISION DE ESTUDIOS DE POSGRADO DE LA

FACULTAD DE INGENIERIA

DE LA

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

COMO REQUISITO PARA OBTENER EL GRADO DE

MAESTRO EN INGENIERIA

(E L E C T R I C A)

CIUDAD UNIVERSITARIA

AGOSTO DE 1990

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

1

INTRODUCCION

Las técnicas de análisis de señales han sido orientadas en las últimas tres décadas hacia un procesamiento digital, en razón de las facilidades que presenta su implantación. El procesamiento digital en la codificación de fuente y de canal es de uso común en la actualidad.

En los centros hospitalarios, las imágenes radiológicas como son : tomografías, radiografías, angiografías e imágenes de resonancia magnética; al ser procesadas mediante técnicas digitales permiten un acceso y almacenamiento más eficiente, así como la posibilidad de un diagnóstico automatizado.

Dentro de los problemas que existen en el procesamiento, se presenta la necesidad de contar con métodos eficientes para almacenar y transmitir imágenes, puesto que estas, en forma digital requieren de gran cantidad de memoria, o bién un ancho de banda grande. La solución a tales problemas radica en incorporar a los sistemas de procesamiento, técnicas de compresión de datos.

En el presente trabajo, se ha implantado un esquema de compresión de imágenes radiológicas, en una tarjeta de desarrollo (SWDS), que emplea al microprocesador para procesamiento de señales TMS320C25 de aritmética entera.

Puesto que las técnicas de compresión reversibles presentan tasas de compresión bajas, se ha preferido emplear una técnica irreversible de codificación por transformada, con la cual se obtienen mejores tasas de compresión.

La transformada que se utiliza es la transformada coseno discreta, puesto que opera en forma muy similar a la transformada optima (Karhunen-Loève), para datos estacionarios aleatorios y sus coeficientes presentan un agrupamiento de acuerdo a su energia, que resulta muy atractivo para la compresión.

Debido a la complejidad de los algoritmos, los cuales requieren un elevado número de operaciones, es necesario reducir el tiempo de procesamiento, conservando una precisión adecuada. Esto se consigue empleando un procesador digital de señales rápido y optimizando los algoritmos de compresión.

Un factor importante cuando se emplea un microprocesador de aritmética de punto fijo, es la deteminación de los formatos adecuados a las variables, para efectos de obtener una convergencia y precisión adecuada.

El esquema de codificacion presentado, previamente a su implantacion fue simulado en lenguaje Pascal, empleando variables enteras de 8, 16 y 32 bits, dividiendo la imagen fuente en sub-imágenes de tamaño 16x16, 32x32 y 64x64. Experimentalmente fué seleccionado el tamaño de bloques de 64x64, de acuerdo a la calidad deseable en la reconstrucción y a las restricciones de implantación.

Inicialmente se presenta, en el segundo capitulo, una revisión de los métodos de codificación clasificados dentro del grupo denominado de la *Primera Generación.*

En el tercer capitulo se presenta un estudio de diferentes algoritmos rápidos para calcular la transformada coseno discreta, directa e inversa, que determina el algoritmo óptimo en función de: el tiempo de ejecución, la precisión requerida y los recursos del microprocesador.

El esquema de codificación empleado se presenta en el capitulo cuarto y el criterio de selección del microprocesador, en el capitulo quinto.

Finalmente, en el capitulo seis se presentan los resultados obtenidos, empleando para su evaluanción, criterios de calidad objetiva.

2

TECNICAS BASICAS DE COMPRESION DE IMAGENES

2.1 Introducción

La meta de la codificación de imagenes es reducir (comprimir), tanto como sea posible, el número de bits necesarios para representar y reconstruir un duplicado fiel de una imagen original.

Dentro de las principales aplicaciones de las diferentes técnicas de compresión podemos mencionar :

- Transmisión de imagenes de : televisión, satélite, teleconferencia etc. Para esta aplicación las técnicas de codificación empleadas estan sujetas, normalmente, a ser en tiempo real.

- Almacenamiento de imagenes médicas o documentos en general. En este caso los diferentes métodos pueden ser efectuados fuera de linea, sin embargo es deseable contar con un sistema que permita la decodificación rápida y eficiente.

-Extracción de caracteristicas aplicable al reconocimiento de formas. El trabajar directamente con la información codificada nos permite el desarrollo de algoritmos rapidos, para la clasificación de caracteristicas locales de una imagen.

La finalidad del presente capítulo es presentar de una manera general algunas de las diferentes técnicas de compresión de imagenes. En cada caso se menciona solamente el principio fundamental de operación. Asi mismo se mencionan los métodos empleados para evaluar la calidad obtenida en las imágenes procesadas.

Técnicas básicas de compresión de imagenes.

Una imagen digitalizada puede ser caractererizada por una secuencia de mensajes, hay varios caminos para seleccionarlos, el único requerimiento es el de obtener un duplicado fiel de esta secuencia de mensajes. El camino particular para seleccionarlos y su codificacion define a los diferentes métodos existentes. Por ejemplo, los mensajes pueden ser niveles de intensidad de cada pixel, o de un grupo de pixeles, o los valores de una función computada de ellos.

Las técnicas de compresión de imagenes pueden ser clasificadas en cuatro grupos segun Jain [4], como :

- Codificación de Pixel.
- Predictiva.
- Por Transformada
- Otras.

Sin embargo no es la única clasificación, Kunt [11] distingue dos generaciones de técnicas de Compresión de Imagenes, agrupa las arriba mencionadas como de la " Primera Generación " , llamando a los métodos que toman en cuenta el mecanismo humano de la visión , como los de la " Segunda Generación ", los cuales describen a la imagen en términos de contornos y texturas.

2.2 Codificación de pixel.

En esta codificación cada pixel es procesado independientemente, ignorando las dependencias interpixel existentes, tampoco las propiedades del sistema visual humano son utilizadas. Supone el conocimiento preciso de las estadisticas globales de la imagen, dentro de esta categoría se encuentran : PCM, Códigos de Huffman, Run-Length, Bit-Plane.

PCM

Conocido tambien como Conversión Analógica Digital(ADC), realiza una discretizacion en el tiempo, muestreando la señal al menos a la frecuencia de Nyquist, mientras que la discretización en amplitud se efectua utilizando un número suficiente de niveles de cuantización. A una tasa de 2.5 : 1 se obtiene una calidad aceptable de imagenes procesadas.

Códigos de Huffman

La idea fundamental es asignar menos bits a aquellos valores de pixel que ocurren más frecuentemente. La construcción del codigo comienza con el cálculo del histograma de la imagen, el cual nos proporciona la probabilidad de ocurrencia de cada valor de pixel. Posteriormente los diferentes niveles se ordenan según un orden de probabilidades decrecientes. Para obtener la palabra correspondiente a cada nivel se asigna un codigo binario creciente hacia la parte menos probable, de este modo se obtiene un diccionario. La codificación y decodificación se realiza por medio de una búsqueda (lookup table).

Esta codificación es sensible a los errores de transmision a través de un canal de tasa constante; lo cual implica un registro temporal de regulación de flujo binario, ya que las palabras codificadas tienen una longitud variable. Es por esto que en la práctica se utilizan algunas variantes de este código, como lo son los códigos de Huffman modificado y truncado.

Run-length

Esta es una codificación libre de error, que se basa en la eliminación de pixeles adyacentes con el mismo valor, ya sea por renglones o por columnas. Para comprender el funcionamiento de esta codificación, considérese por ejemplo, un renglon de una imagen cualquiera que tenga la siguiente secuencia de valores : 20, 30, 41 y en seguida 22 pixeles con un valor de 80. Esta última cadena de pixeles con el mismo valor, puede ser comprimida si se representa como una secuencia de tres números : la

marca, que indica el inicio de una cadena de pixeles contiguos iguales, el tamaño de la cadena y el valor de la cadena. De esta manera la cadena anterior puede ser representada como :
20, 30, 41, 250, 22, 80. Donde el 250 representa la marca. la cual debe ser un valor inexistente en la imagen, o poco probable, para que no sea confundido con un valor de pixel.

Como puede apreciarse la tasa de compresión en este caso depende de la estadística de la imagen.

Bit plane.

Una imagen con 256 niveles de gris por ejemplo, puede ser considerada como un conjunto de 8 planos de un bit, cada uno de los cuales puede ser codificado utilizando el método Run-Length. Este método es muy sensible a los errores en el canal, a menos que las palabras que representan los " bit plane " más significativos sean cuidadosamente protegidos.

2.3 Técnicas predictivas.

DPCM

El principio de esta técnica es remover la redundancia mutua existente entre pixeles sucesivos, codificando solamente la nueva información. Una predicción del pixel actual se hace, por ejemplo, utilizando el valor codificado previamente y sólo el error de predicción (señal diferencial), es cuantizado para su transmisión.

Modulación delta.

Es la codificación predictiva más sencilla, se utiliza una función de retraso de un paso como predictor y la diferencia entre esta predicción y la señal se cuantiza con un bit. Las limitaciones de ésta técnica ocurren cuando la señal presenta grandes cambios o discontinuidades, en este caso el modulador delta no puede seguir a la señal ya que utiliza incrementos

fijos (slop overload) ; cuando la señal es constante la salida del modulador oscila (ruido granular); ambos efectos adversos pueden ser reducidos utilizando un filtro paso-bajas a la entrada y a la salida, el efecto del " slop overload " puede también disminuirse incrementando la frecuencia de muestreo, lo cual reduce la tasa de compresión. El efecto del ruido granular se disminuye también utilizando un modulador de tres estados, en conjunción con códigos de Huffman.

DCPM linea por linea.

En este método cada linea de la imagen es codificada independientemente utilizando la técnica DPCM, generalmente se usa una representación autoregresiva (AR), para diseñar el predictor. Una desventaja de este método, es que no se aprovecha la dependencia estadistica entre lineas y solo se pueden alcanzar tasas de 3 : 1.

DPCM bidimensional.

Es una extensión del método DCPM a dos dimensiones. Se utiliza una ventana de predicción causal bidimensional y la secuencia de pixeles son representados por un modelo causal de varianza minima (MVR), que permite minimizar la varianza del error de predicción en la imagen. Se pueden alcanzar tasas de 4 : 1.

DPCM adaptivo.

El esquema DPCM puede ser hecho adaptivo, si los párametros de la predicción son adaptados a los datos en un camino apropiado. Por ejemplo, se puede definir una medida dela actividad local y los parámetros de la predicción pueden ser actualizados en cada cambio notable de actividad, en este sentido la tasa puede ser aumentada de un 10 a un 20 % .

2.4 Codificación por transformada.

En la codificación por transformada, también llamada de bloques, se

divide generalmente una imagen de tamaño N x N, en sub-imagenes de M x M, donde M < N y es normalmente una potencia de 2 . Posteriormente se aplica una transformación lineal a cada sub-bloque de valores (pixeles), estadisticamente dependientes, transformándolos en otro bloque de coeficientes " mas independientes ", finalmente los coeficientes asi obtenidos son cuantizados y codificados.

Para obtener un buen desempeño en la codificación por transformada, es necesario determinar los siguientes parámetros :

- Tamaño de la sub-imagen.
- Tipo de transformada.
- Selección de los coeficientes a ser cuantizados.
- Asignación de bits.

Diferentes tipos de transformadas.

Para llevar a cabo la compresión de una imagen, sólo son seleccionados los coeficientes de máxima energia para efectuar su cuantización y su codificación , por lo cual es deseable contar con una transformada que compacte la máxima energia de la imagen en el menor número de coeficientes que sea posible y que además sea fácil de implementar. Una transformada que cumple con la primera de estas necesidades es la transformada optima para la compresión de imagenes, llamada transformada de Karhumen - Loève o de la componente principal, ya que nos proporciona coeficientes no correlacionados y simultáneamente la máxima concentración de energia en promedio en un número reducido de coeficientes [4]. Sin embargo, esta transformada es computacionalmente difícil de calcular, puesto que depende de la estadística de la imagen y siendo ésta no estacionaria hay que calcular la matriz de transformación para cada imagen, lo cual no es deseable en el contexto de una codificación en tiempo real. Por tal motivo se utilizan otras transformadas deterministicas, que logran aproximar los resultados óptimos de la transformada K-L. Estas transformadas son sub-óptimas y producen coeficientes que aún estan correlacionados entre sí, sin embargo, esta correlación tiende a disminuir conforme se incrementa el tamaño de la transformada, ver la figura 4.1 . Estas

transformadas redistribuyen así mismo la energía de los pixeles en un número pequeño de coeficientes y además son fáciles de implementar. Entre ellas se encuentran : la transformada de Fourier, Coseno, Seno, Slant, Hadamard. Siendo la transformada Coseno la que proporciona la mejor aproximación con respecto a la transformada Karhunen - Loeve .

Con este método se pueden obtener tasas de compresión de 10 : 1, dependiendo del número de coeficientes retenidos en un área dada. Otra posibilidad consiste en fijar un umbral contra el cual se comparan los coeficientes, haciendo cero aquellos debajo de este umbral, de esta manera se pueden alcanzar tasas de 15 : 1 con una buena calidad.

La codificación por transformada puede hacerse adaptiva, si se igualan los parametros del codificador a la estadística de la sub-imagen a ser codificada, la adaptación puede hacerse ya sea a nivel de la transformada, de la asignación de bits o de los niveles de cuantización. La eficiencia con respecto a la no adaptiva puede ser incrementada de un 25 a un 30 % .

2.5 Otras técnicas de compresión.

Codificación híbrida.

Comúnmente se denomina así a los métodos que combinan a la codificación predictiva con la codificación por transformada. La idea básica es combinar la sencillez en hardware de una, con la alta tasa de compresión y robustez de la otra.

La codificación se realiza aplicando una transformada unidimensional de orden N por renglones, o columnas. Por medio del cual, se obtienen N secuencias vectoriales de orden N, cada una de estas es entonces codificada independientemente por medio de un predictor recursivo vectorial DPCM, con N codificadores DCPM escalares desacoplados. En la práctica el número de codificadores es menor a N, ya que muchos elementos del vector transformado son cero.

Codificación híbrida adaptiva

El esquema de codificación puede ser adaptado a imagenes cuya estadís-

tica espacial varia lentamente. Dos tipos de adaptividad pueden ser introducidos : por un lado, los parámetros del modelo de predicción de cada codificador DPCM, se pueden adaptar según las variaciones de la estadistica de la imagen , utilizando una adaptación de la varianza de los coeficientes. Por otro lado, la distribución de los bits de cuantización se pueden adaptar según una clasificación predeterminada de los vectores.

Codificación intermarcos (Interframe Coding).

Este método explota la redundancia que existe entre los marcos sucesivos, en la transmisión de imagenes con poco movimiento, por ejemplo, la teleconferencia. La diferencia que existe entre los marcos sucesivos es debida al movimiento del objeto mismo, al movimiento de la cámara, o por una ampliación (zooming).

Repetición de marcos.

El método consiste en submuestrear las imagenes y reemplazar cada marco no transmitido por el precedente. Sin embargo, esto no produce una buena reconstrucción del movimiento. Una alternativa es hacer un rellenado selectivo, donde los marcos son transmitidos a una tasa reducida, de acuerdo a un algoritmo predeterminado de actualización. En el receptor cualquier dato no actualizado es refrescado a partir del marco previo, el cual se conserva en una memoria.

Intercambio de resolución.

La respuesta del sistema visual humano es mala, para discriminar detalles en escenas dinámicas con una alta resolución espacial y temporal. Es por esto que las áreas de una escena, que estan cambiando rápidamente, pueden ser representadas con menos bits utilizando una amplitud y una resolución espacial reducida, respecto a otras áreas fijas. Para llevar a cabo esta compresión se segmenta la imagen en zonas estacionarias y de movimiento, de acuerdo a un umbral determinado dado por la señal de diferencia entre

marcos. En zonas estacionarias sólo son transmitidaslas diferencias entre marcos, en el receptor se decodifican y los valores no recibidos son obtenidos del marco anterior. En zonas con movimiento se usa un submuestreo de 2:1 y los valores intermedios son recuperados por interpólación. Usando una codificación de marco diferencial de 5 bits por pixel, se puede obtener una tasa de compresión de 2.5 : 1 .

" Replenishement condicional "

Esta técnica esta basada en la detección y codificación de áreas en movimiento, las cuales son refrescadas de marco en marco. Siempre que la magnitud de la señal de diferencia entre marcos rebase un umbral determinado, esta se cuantiza y codifica con su ubicación para su transmisión. En el receptor un pixel es repetido del marco anterior si pertenece a una área estacionaria, o es decodificado de la señal de diferencia si proviene de una de movimiento.

2.6 Criterios de fidelidad

Los diversos procesamientos a los que son sometidas las señales, les ocasionan a menudo, modificaciónes que las degradan. El simple hecho de digitalizar una señal analógica, trae como consecuencia que sea imposible recuperar exactamente la señal original. El grado de distorsión permisible dependerá de la aplicación a la que se destine la señal procesada.

Despues de aplicar alguna técnica de compresión, es posible observar en la imagen reconstruida cierta degradación, la cual depende en general: de las operaciones aritméticas (errores por truncamiento, redondeo,o falta de precisión), de la técnica empleada, de la tasa de compresión alcanzada y de la estructura de la propia imagen.

La calidad en el procesamiento de imágenes suele evaluarse mediante criterios subjetivos y criterios objetivos, siendo el primero el de más lenta aplicación y también el más importante, puesto que en la mayoria de los procesos la etapa final es la interpretación humana.

2.6.1 Criterios subjetivos de calidad

Método de Apreciación

Es posible evaluar la calidad de las imágenes, mostrándolas a algunos observadores o especialistas; asignando cada persona una puntuacion por imagen, según la calidad observada. En la Tabla 2.1 a) se muestra una Escala de buena calidad global y en la Tabla 2.1 b) una Escala de buena calidad por grupo, en esta las evaluaciónes se realizan buscando la mejor imagen de un grupo, y a partir de ella se juzgan las restantes.

ESCALA DE BUENA CALIDAD GLOBAL

CLASIFICACION	PUNTUACION
EXCELENTE	5
BUENA	4
REGULAR	3
POBRE	2
NO SATISFACTORIA	1

a)

ESCALA DE BUENA CALIDAD POR GRUPO

CLASIFICACION	PUNTUACION
LA MEJOR	7
BUENA SOBRE EL PROMEDIO	6
LIGERAMENTE SOBRE EL PROMEDIO	5
PROMEDIO	4
LIGERAMENTE ABAJO DEL PROMEDIO	3
MUY ABAJO DEL PROMEDIO	2
LA PEOR	1

b)

Tabla 2.1. Métodos de apreciación. a) Escala de buena calidad global, b) Escala de buena calidad por grupo.

Cuando se emplean varios sujetos, los resultados suelen ser evaluados calculando la Puntuación de Opinion Media (POM):

$$POM = \frac{\sum_{i=1}^{k} n_i C_i}{\sum_{i=1}^{k} n_i} \qquad (2.1)$$

donde C_i es la puntuación de la categoria i, n_i es el número de apreciaciones en la categoria i y k es el total de categorias en la escala.

Método de comparación

La escala de deterioro de la tabla 2.2, se emplea comparando una imagen distorsionada con una imagen de referencia, a la que se le va agregando cierto deterioro hasta que ambas parescan tener la misma calidad, a criterio del sujeto.

ESCALA DE DETERIORO

CLASIFICACION	PUNTUACION
NO PERCEPTIBLE	1
APENAS PERCEPTIBLE	2
DEFINITIVAMENTE PERCEPTIBLE PERO SOLO LIGERAMENTE DETERIORADA	3
DETERIORADA PERO NO INCONVENIENTE	4
LIGERAMENTE INCONVENIENTE	5
DEFINITIVAMENTE INCONVENIENTE	6
EXTREMADAMENTE INCONVENIENTE	7

Tabla 2.2. Método de comparación que emplea una escala de deterioro.

Los resultados obtenidos por evaluaciones subjetivas, sin embargo presentan cierto grado de incertidumbre, ya que tienden a depender de la experiencia y motivación de los sujetos, del rango del material usado en las pruebas y de las condiciones bajo las cuales son mostradas las imágenes (por ejemplo la iluminación del medio ambiente, la distancia de observación y el tamaño de las imágenes); por lo que este tipo de pruebas suelen ser complementadas empleando criterios objetivos.

2.6.2 Criterios objetivos de calidad

Como medida de fidelidad entre una imagen original $g(x,y)$ y una imagen degradada $g(x,y)$, ambas de tamaño NxM, es común emplear el error medio cuadrático (MSE), que experimentalmente se estima mediante:

$$MSE = \frac{1}{N\,M} \sum_{x=1}^{N} \sum_{y=1}^{M} \left| g(x,y) - g(x,y) \right|^2 \qquad (2.2)$$

otro parámetro de calidad es el error medio cuadrático normalizado

(NMSE), el cual se expresa como:

$$NMSE = \frac{\sum_{x=1}^{N}\sum_{y=1}^{M}\left| g(x,y) - \hat{g}(x,y) \right|^2}{\sum_{x=1}^{N}\sum_{y=1}^{M}\left[g(x,y) \right]^2} \times 100 \qquad (2.3)$$

e igualmente expresado en el dominio de la transformada, si se emplea una transformada unitaria:

$$NMSE = \frac{\sum_{u=1}^{N}\sum_{v=1}^{M}\left| G(u,v) - \hat{G}(u,v) \right|^2}{\sum_{u=1}^{N}\sum_{v=1}^{M}\left[G(u,v) \right]^2} \times 100 \qquad (2.4)$$

El gran atractivo de emplear el MSE y NMSE es su simplicidad de cálculo, sin embargo, no resultan ser criterios adecuados para medir la calidad visual global. Así, la estadística de las imágenes y el comportamiento no lineal del sistema visual, ocasionan que para ciertas imágenes, un valor grande de NMSE no signifique necesariamente mala calidad. Pero, en general, un valor muy pequeño de error significa buena calidad subjetiva.

Otro índice, más adecuado de estimación, es la relación señal-ruido (SNR), la cual se expresa como:

$$SNR = 10 \cdot \log_{10} \frac{\sigma^2}{\sigma_e^2} \quad [dB] \qquad (2.5)$$

donde σ^2 es la varianza de la imagen original y σ_e^2 puede calcularse con (2.2).

Otra expresión para calcular la relación señal-ruido es:

$$SNR = 10 \log_{10} \frac{(\text{VALOR PICO DE LA IMAGEN ORIGINAL})^2}{\sigma_e^2} \quad [dB] \qquad (2.6)$$

donde el valor pico de la imagen originan es igual a $2^n - 1$, y n es el número de bits por pixel. Al emplear la expresión (2.6), debe tomarse en cuenta que esta proporciona un valor de alrededor 12 a 15 decibeles sobre el calculado con la expresión (2.5).

3

ALGORITMOS RAPIDOS PARA EL CALCULO DE UNA DCT

3.1 Introducción

Existen varios algoritmos que realizan la computación de una DCT, estos pueden ser clasificados en 3 categorias : Computación Directa, Indirecta y Recursiva. En este capitulo se revisa un método de cada categoria, teniendo como finalidad determinar el mejor algoritmo sobre la base de los parámetros siguientes :

- tiempo de cálculo
- cantidad de memoria requerida para su implantación
- error número acumulado durante su ejecución en aritmética entera

3.2 Método de Computación Directa (MCD).

Definición del algoritmo

Se denota a la Transformada Coseno Discreta (DCT), unidimensional de una secuencia de datos x(k), k =0,1 ... N-1, por X(n), n = 0,1 ... N-1 , definida inicialmente por [12], como :

$$X(n) = \frac{2}{N}\, e(n) \sum_{k=0}^{N-1} x(k) \cos\left[\frac{\pi\,(2k+1)n}{2N}\right] ; \quad n = 0,1,..,N-1 \qquad (3.1)$$

la Transformada Coseno Discreta Inversa (IDCT) correspondiente, se define por :

$$x(k) = \sum_{n=0}^{N-1} e(n)\, X(n) \cos\left[\frac{\pi\,(2k+1)n}{2N}\right] ; \quad k = 0,1,..,N-1 \qquad (3.2)$$

donde :

$$e(n) = \begin{cases} 1/\sqrt{2} & \text{si } n=0 \\ 1 & \text{c.c.} \end{cases} \qquad (3.3)$$

Para fines de simplicidad de nomenclatura se definen :

$$C_{2N}^{(2k+1)n} = \cos [\ \pi\ (2k+1)n/2N\ ]$$

$$\hat{X}(n) = e(n)\ X(n) \qquad (3.4)$$

podemos entonces reescribir la IDCT mediante

$$x(k) = \sum_{n=0}^{N-1} \hat{X}(n)\ C_{2N}^{(2k+1)n} \quad ;\quad k = 0,1,..,\ N-1 \qquad (3.5)$$

La expresión anterior puede ser factorizada en dos transformadas IDCT's de orden N/2, arreglando sus términos pares e impares como :

$$x(k) = g(k) + h'(k)$$

$$x(N-1-k) = g(k) - h'(k) \quad \text{para} \quad k = 0,1,\ ..,\ N/2\ -1 \qquad (3.6)$$

donde :

$$g(k) = \sum_{n=0}^{N/2-1} \hat{X}(2n)\ C_{2N}^{(2k+1)2n} \qquad (3.7a)$$

$$h'(k) = \sum_{n=0}^{N/2-1} \hat{X}(2n+1)\ C_{2N}^{(2k+1)(2n+1)} \qquad (3.7b)$$

De acuerdo a la identidad trigonométrica cosA * cosB = 1/2 { cos(A-B) + cos(A+B) }, se puede comprobar que :

$$2C_{2N}^{(2k+1)}\ C_{2N}^{(2k+1)(2n+1)} = C_{2N}^{(2k+1)2n} + C_{2N}^{(2k+1)2(n+1)} \qquad (3.8)$$

luego a partir de (3.8) podemos escribir (3.7b) como :

$$2C_{2N}^{(2k+1)}\ h'(k) = \sum_{n=0}^{N/2-1} \hat{X}(2n+1)C_{2N}^{(2k+1)2n} + \sum_{n=0}^{N/2-1} \hat{X}(2n+1)C_{2N}^{(2k+1)2(n+1)} \qquad (3.9)$$

Definiendo, $\hat{X}(2n-1)\Big|_{n=0} = 0$

el segundo término de la expresión (3.9), se puede expresar como :

$$\sum_{n=0}^{N/2-1} \hat{X}(2n+1)\, C_{2N}^{(2k+1)2(n+1)} = \sum_{n=0}^{N/2-1} \hat{X}(2n-1)\, C_{2N}^{(2k+1)2n} \qquad (3.10)$$

entonces a partir de la expresión (3.9), se tiene :

$$h'(k) = 1 / 2C_{2N}^{2k+1} \sum_{n=0}^{N/2-1} (\hat{X}(2n+1) + \hat{X}(2n-1))\, C_{2N}^{(2k+1)2n} \qquad (3.11)$$

De las expresiones (3.6,3.7,3.11) se obtiene el algoritmo para el cálculo de una IDCT dado por LEE [2], que consíste en separar primero los datos de entrada $\hat{X}(k)$ en dos partes, una par y otra impar :

$$G(n) = \hat{X}(2n) \;\; ; \;\; H(n) = \hat{X}(2n+1) + \hat{X}(2n-1) \;\; ; \; n = 0,1,..,N/2-1 \qquad (3.12)$$

posteriormente, se evaluan las sumatorias de productos :

$$g(k) = \sum_{n=0}^{N/2-1} G(n)\, C_{2(N/2)}^{(2k+1)n} \quad ; \quad h(k) = \sum_{n=0}^{N/2-1} H(n)\, C_{2(N/2)}^{(2k+1)n}$$

para $n = 0,1,..,N/2-1$ (3.13)

y finalmente calculando (3.12) y (3.13) se obtiene la IDCT de $\hat{X}(k)$ como :

$$x(k) = g(k) + (1/ 2C_{2N}^{2k+1})\, h(k)$$

$$x(N-1-k) = g(k) - (1/ 2C_{2N}^{2k+1})\, h(k) \quad ; \quad k = 0,1,.., N/2 -1 \qquad (3.14)$$

La evaluación de las expresiones anteriores se resume en la gráfica de flujo de la fig. 3.1, la cual tiene una estructura parecida a la descomposición tradicional radix-2 de una FFT.

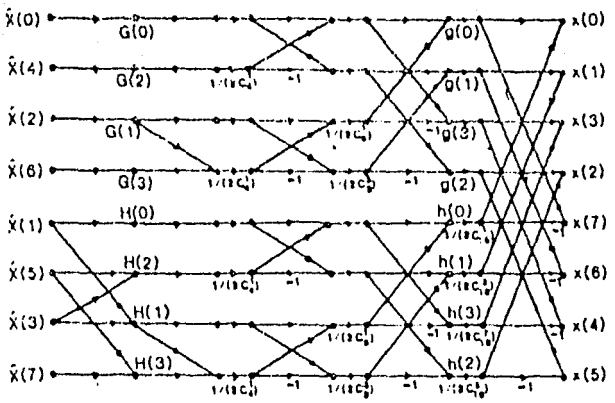


Fig. 3.1 Representación de una IDCT de 8 puntos por el método MCD.

Evaluación del algoritmo

Este algoritmo fué simulado en lenguaje Pascal para una transformada de orden 64, el número de operaciones necesarias para su ejecución es de 513 sumas y 192 multiplicaciones. Sin embargo su implantación en lenguaje ensamblador no es aconsejable , ya que por un lado para evaluar las expresiones (3.13) con aritmética entera, es necesario contar con una tabla de cosenos entre cero y uno, con una determinada precisión ; pero por (3.14) se tendria una tabla con valores mayores a uno, lo que ocasionaria errores de redondeo considerables.

3.3 Método de Computación Indirecta (MCI).

Una transformada rápida de Fourier (FFT), de N puntos puede ser utilizada para calcular una DCT de n puntos, por medio de un simple reordenamiento de los datos a la entrada, más N/2-1 multiplicaciones complejas como fué propuesto inicialmente por Narashima [9].

Definicion del algoritmo

Considerando la evaluación de una DCT sin tomar en cuenta (por facilidad), a 2 e(n) / N, de (3.1) se tiene :

$$F(r) = \sum_{k=0}^{N-1} x(k) \cos[\ \pi(2k+1)r/2N\ ] \quad ; \quad r = 0,1.. \ N-1 \qquad (3.15)$$

considerando ahora una N par y definiendo una nueva secuencia de entrada y(k), dada por :

$$\begin{aligned} y(k) &= x(2k) \\ y(N-1-k) &= x(2k+1) \qquad \text{para } k = 0,1, \ldots, N/2 -1 \end{aligned} \qquad (3.16)$$

Haciendo la substitución de (3.16) en (3.15), se puede calcular F(r) como la suma de dos transformadas de datos pares e impares, de orden N/2 :

$$F(r) = \sum_{k=0}^{N/2-1} y(k) \cos[\ \pi(4k+1)r/2N\ ] + \sum_{k=0}^{N/2-1} y(N-1-k) \cos[\pi(4k+3)r/2N\ ]$$

$$\text{para } r = 0,1,\ldots, N-1 \qquad (3.17)$$

substituyendo k' = N-1-k, en el segundo término de la expresión anterior y dado que $\cos(2\pi - \alpha) = \cos(\alpha)$, (3.17) se puede expresar mediante :

$$F(r) = \sum_{k=0}^{N-1} y(k) \cos[\ \pi(4k+1)r/2N\ ] \quad ; \ r = 0,1\ldots \ N-1 \qquad (3.18)$$

Esta expresión puede ser evaluada como :

$$F(r) = \mathrm{Re}\ \{\ H(r)\ \} \quad ; \quad r = 0,1,\ldots, N-1 \qquad (3.19)$$

donde H(r) resulta de transformar (3.18) en una forma compleja :

$$H(r) = e^{j\pi r/2N} \sum_{k=0}^{N-1} y(k)\ e^{j2\pi kr/N} \quad ; \quad r = 0,1,..,\ N-1 \qquad (3.20a)$$

Se puede observar que esta expresión consta del cálculo de una IFFT de la secuencia y(k) y de N-1 multiplicaciones complejas. Tomando la parte real como lo indica (3.19) se puede obtener (3.18). Entonces dado que solo hay que tomar la parte real de (3.20a), esta expresión puede ser reemplazada por:

$$H(r) = e^{-j\pi r/2N} \sum_{k=0}^{N-1} y(k)\ e^{-J2\pi k/2N} \qquad (3.20b)$$

La cual implica el cálculo de una FFT de la secuencia y(k). Es fácil verificar que la multiplicación por $e^{-j\pi r/2N}$ en esta última ecuación , tiene la ventaja de que el

$$\cos[\pi r/2N\ ] = \text{sen}[\ \pi(N-r)/2N\ ],\ \text{para}\ \ r = 1,2,\ ..,\ N/2\ -1 \qquad (3.21)$$

de tal manera que F(r) puede ser evaluada a partir de la expresión (3.20b), como :

$$F(r) \quad = \quad \text{Re}\ [H(r)] \qquad ; \ r = 0,1\ ..\ N/2 \qquad (3.22a)$$

$$F(\ N-r\ ) \quad = \quad -\ \text{Imag}\ [\ H(r)^{*}\ ] \qquad ; \ r = 1,2 \quad N/2\ -1 \qquad (3.22b)$$

En resumen, una DCT de N puntos de una secuencia x(k), puede ser evaluada a partir de una FFT de N puntos , aplicada sobre un nuevo orden de los datos de entrada, más N/2 - 1 multiplicaciones complejas, como lo indican las expresiones (3.22a) y (3.22b).

De manera similar, una IDCT puede ser evaluada a partir de la siguiente expresión :

$$P(k) = \text{Re}\{\ \sum_{r=0}^{N-1} [\ e(r)\ X\ (r)\ e^{j\pi r/2N}\ ]\ e^{j2\pi rk/N}\ \}$$

$$\text{para}\ \ k = 0,1,..,N-1 \qquad (3.23)$$

donde la IDCT de la secuencia X(k) es calculada entonces como :

$$x(2k) = P(2k) \quad ; \quad x(2k+1) = P(N-1-k) \qquad \text{para } k = 0,1,\ldots,N/2-1 \qquad (3.24)$$

La expresión (3.23) nos indica el cálculo de una IFFT de una secuencia compleja , tomando de esta solo la parte real en un orden determinado por (3.24).

Entonces para poder evaluar tanto una DCT como una IDCT con este metodo, es necesario encontrar un algoritmo que realice el cálculo de una FFT. A continuación se presenta el algoritmo emplado.

Algoritmo split-radix para el calculo de una FFT.

El algoritmo split-radix introducido por Duhamel [14], tiene varias ventajas sobre algunos otros algoritmos, de las cuales podemos mencionar :

- no necesita un reordenamiento de datos dentro del algoritmo
- su ejecución es "en el lugar"
- es aplicable a secuencias complejas
- posee una reducción de redundancia aritmética en secuencias reales

Definición del algoritmo

Sea el cálculo de una FFT de una secuencia y(n), dada por Y(k) donde :

$$Y(K) = \sum_{n=0}^{N-1} y(n)\, W_N^{nk} \qquad (3.25)$$

y

$$W_N = \cos(2\pi/N) - j\,\mathrm{sen}(2\pi/N) \qquad (3.26)$$

El algoritmo se basa en la siguiente descomposición para los término pares :

$$Y(2k) = \sum_{n=0}^{N/2-1} [\, y(n) + y(n + N/2) \,]\, W_N^{2nk} \qquad (3.27a)$$

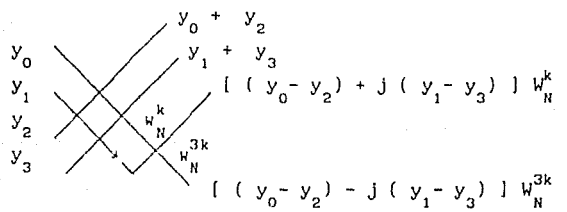
y para los términos impares :

$$Y(4k+1) = \sum_{n=0}^{N/4-1} \{ [\, y(n) + y(n + N/2) \,] - j [\, y(n + N/4) - y(n + 3N/4) \,] \}\; W_N^{n}\; W_N^{4nk} \qquad (3.27b)$$

$$Y(4k+3) = \sum_{n=0}^{N/4-1} \{ [\, y(n) - y(n + N/2) \,] + j [\, y(n + N/4) - y(n + 3N/4) \,] \}\; W_N^{3n}\; W_N^{4nk} \qquad (3.27c)$$

La idea básica del split-radix FFT (SRFFT), es la aplicación de un mapa indexado radix-2 sobre los términos pares y un mapa indexado radix-4 sobre los términos impares ; Se repite el proceso a cada una de las tres etapas resultantes, según las ecuaciones (3.27), de una manera similar a la decimación en frecuencia radix-2 del algoritmo Cooley-Tukey FFT.

La " mariposa básica " del SRFFT tiene la siguiente estructura :



En la fig. 3.2, se muestra una SRFFT de orden 32 .

Se puede observar, a partir de las expresiones (3.27a),(3.27b),(3.27c) y de la fig. 3.2, que el algoritmo avanza en la parte superior estado por estado, para cada una de las M etapas ($N = 2^M$); mientras que en la parte inferior por ser radix-4, calcula dos etapas a un mismo tiempo, es

decir no avanza estado por estado, o en términos del indexado, no completa cada suma anidada en orden. Un esquema de indexado alternativo fué propuesto por Sorensen [17], el cual permite el cálculo de una SRFFT etapa por etapa.

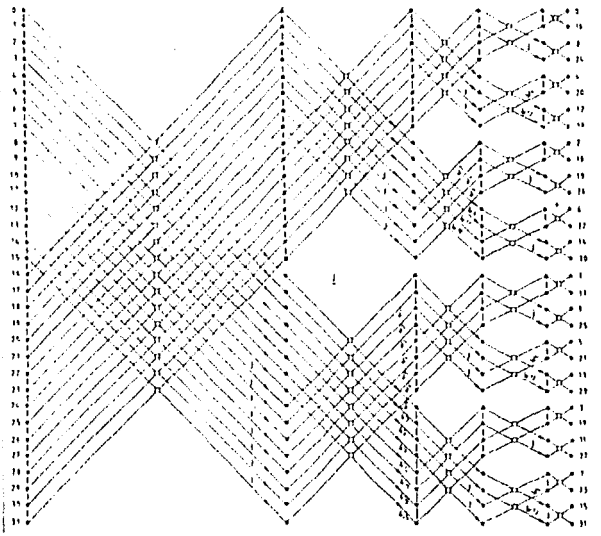


Fig. 3.2 Algoritmo split-radix FFT para una longitud de 32.

El número de operaciones necesarias para el cálculo de una SRFFT para N=64, con una secuencia de entrada compleja, es de 964 sumas y 196 multiplicaciones.

Split-radix FFT valuada real (SRFFT-RV)

Si la secuencia de entrada y(n) es real, existe una redundancia de operaciones, ya que Y(k) e Y(N-k) son complejos congujados. Esto significa que a partir de las ecuaciones (3.27b) y (3.27c) es inútil calcular ambos coeficientes : Y(4k+1) e Y(4k+3), dado que :

$$Y(4k+3) = Y(N-(4k+1)) = Y^{*}(4k+1) \qquad (3.28)$$

Para entender mejor el funcionamiento del algoritmo en cada etapa, se puede considerar la fig. 3.3 para una SRFFT y una SRFFT-RV de orden 64.

En la primera etapa se calcula un bloque de longitud N/2 de términos pares : Y(2k) para k = 0,1..7 ; en la segunda etapa se calculan dos bloques de N/4 : Y(4k+3) e Y(4k+1), para k = 0,1 ..3 . Este proceso se aplica recursivamente a todos los bloques siguientes.

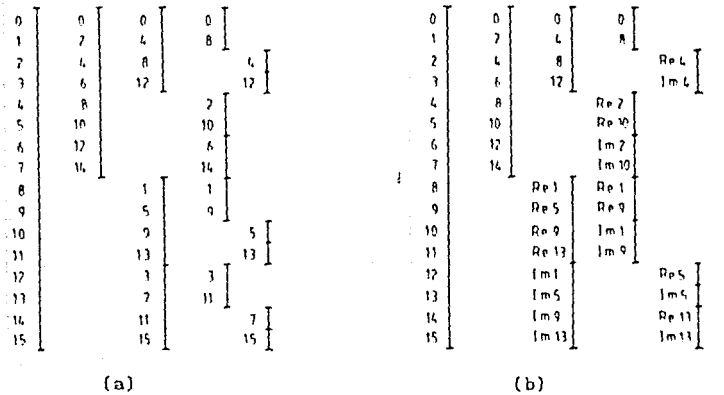


Fig 3.3 Representación esquematica de la progresión del algoritmo SR de 16 puntos: (a) datos complejos, (b) datos reales

Cuando el algoritmo es utilizado con una secuencia real fig 3.3b, la primera etapa será una secuencia real, pero los dos bloques en la segunda etapa son complejos y como lo indica la ecuación (3.28), Y(4k+3) no necesita ser calculado, sin embargo su localización correspondiente en el diagrama puede ser usada para guardar la parte imaginaria del bloque Y(4k+1), este proceso es repetido hasta obtener la transformada correspondiente. Se debe notar que el algoritmo para este caso, debe distinguir entre " mariposas " reales y complejas. Sorensen [17] presenta un programa para calcular una SRFFT-RV el cual tiene como entrada un vector $\bar{y}$ = [y(0),..,y(N-1)] y en el mismo lugar se obtiene la salida con el orden siguiente :

Y = [Re(0),Re(1)..Re(N/2),Im(N/2 -1),.. Im(1)].

El numero de múltiplicaciones para este caso se reduce a 98 y el de

sumas a 420 para un tamaño de N = 64.

Evaluación del algoritmo

De acuerdo a la expresion (3.20b) una DCT implica el cálculo de una FFT de una secuencia real, pero como lo indica la ecuación (3.22) no hace falta calcular toda la FFT, basta con conocer solo la mitad de esta, para tal caso se utilizó la SRFFT-RV. Las simulaciones en lenguaje de alto nivel indicarón un buen desempeño para su implantación en lenguaje ensamblador, al final de este capitulo se proporcionan los resultados correspondientes. En lo que respecta al cálculo de una IDCT, no se puede evitar el hecho de obtenerla a partir de una secuencia compleja, ver la ecuación (3.23), lo cual nos obliga a utilizar el método SRFFT.

3.4 Método de Computación Recursiva (MCR).

Este método dado por Hou [3], permite generar una DCT de orden N a partir de dos DCT de orden N/2, en este caso la matriz de orden superior es generada directamente a partir de las matrices de orden inferior; aunque el cálculo se efectúa con el mismo número de operaciones que con el método MCD, este no requiere una inversión de los coeficientes.

Definición del algoritmo.

La expresión (3.1) anterior, puede ser representada en forma vectorial de la siguiente manera :

$$\bar{X} = (2/N)\, T(N)\, \bar{x} \qquad (3.29)$$

donde $\bar{X}$ y $\bar{x}$ son los vectores columna que denotan la salida y la entrada en orden natural y T(N) es la matriz de transformación DCT de orden N, la cual cuenta con propiedades intrínsecas importantes .

Propiedades de la DCT.

Proposición 1. Una DCT puede ser convertida en una FFT, agregándole a

ésta un ángulo de fase variable, tal como se hizo en el método MCI anterior. Es decir la ecuación (3.18), puede ser expresada de la siguiente manera :

$$X_r = \sum_{k=0}^{N-1} y_k \cos[\Theta_r + 2\pi kr/N] \quad ; \quad r = 0,1,..,N-1 \qquad (3.30a)$$

donde el ángulo de fase es : $\Theta_r = \pi r/2N$ (3.30b)

En otras palabras la secuencia de entrada x_n ha sido reordenada de acuerdo a (3.16); en forma matricial esto puede ser representado como $\bar{y} = P\,\bar{x}$, donde P es la matriz de permutación que efectúa el reordenamiento de la secuencia de entrada y tiene la siguiente forma:

$$P = \begin{bmatrix} 1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,1\,0\,0\,\ldots\ldots\,0\,0\,0 \\ \ldots\ldots\ldots\ldots\,1\,0 \\ 0\,0\,..\,0\,0\,0\,0\,0\,0\,1 \\ 0\,0\,\ldots\,0\,0\,0\,0\,1\,0\,0 \\ \ldots\ldots\ldots\ldots\ldots \\ 0\,1\,\ldots\,0\,0\,0\,0\,0\,0\,0 \end{bmatrix} \qquad (3.30c)$$

Proposición 2. La matriz T(N) puede ser dividida en cuatro cuadrantes cambiando el orden de los datos de entrada.
Considerando el kernel de la ecuación (3.30a), dado por :

$$\tilde{t}_{rk} = \cos(\Theta_r + 2\pi kr/N)$$
$$= \cos\Theta_r \cos(2\pi kr/N) - \operatorname{sen}\Theta_r \operatorname{sen}(2\pi kr/N) \qquad (3.31)$$

donde k, r toman valores de 0,1,..,N-1. Dividiendo k en dos regiones iguales, como por ejemplo, k' : (0, N/2-1) y k'' =k' + N/2 : (N/2, N-1); de la ecuación anterior, se tiene para k'en el primer intervalo :

$$\tilde{t}_{rk'} = \cos\Theta_r \cos(2\pi rk'/N) - \operatorname{sen}\Theta_r \operatorname{sen}(2\pi rk'/N) \qquad (3.32a)$$

y para k'' en el segundo intervalo , en función de k':

$$\tilde{t}_{rk''} = \cos\Theta_r \cos(\pi r + 2\pi rk'/N) - \operatorname{sen}\Theta_r \operatorname{sen}(\pi r + 2\pi rk'/N) \qquad (3.32b)$$

Entonces, de la expresión anterior se puede comprobar que para r par :

$$\tilde{t}_{rk'',} = \cos \Theta_r \cos (2\pi rk'/N) - \operatorname{sen} \Theta_r \operatorname{sen} (2\pi rk'/N) \qquad (3.33a)$$

y para r impar :

$$\tilde{t}_{rk'',} = - \cos \Theta_r \cos (2\pi rk'/N) + \operatorname{sen} \Theta_r \operatorname{sen} (2\pi rk'/N) \qquad (3.33b)$$

Luego, a partir de las ecuaciones (3.32a), (3.33a) y (3.33b), los resultados anteriores se pueden resumir como :

$$\tilde{t}_{rk'',} = \tilde{t}_{rk'} \quad \text{para los renglones pares} \quad \text{y} \quad r : (0, N-2) \qquad (3.34a)$$

$$\tilde{t}_{rk'',} = - \tilde{t}_{rk'} \quad \text{para los renglones impares} \quad \text{y} \quad r : (1, N-1) \qquad (3.34b)$$

Como lo demuestran estos resultados se puede dividir la matriz T(N) en cuatro cuadrantes, donde la mitad superior representa a los renglones pares y la mitad inferior a los renglones impares :

$$\tilde{T}(N) = \begin{bmatrix} \tilde{E}(N/2) & \tilde{E}(N/2) \\ \tilde{D}(N/2) & - \tilde{D}(N/2) \end{bmatrix} \qquad (3.35)$$

donde $\tilde{E}(N/2)$ y $\tilde{D}(N/2)$ son matrices cuadradas de tamaño N/2 x N/2. En éste caso la secuencia de salida es :

$\bar{X} = [\, X_0, X_2 .. X_{N-2}, X_1, X_3 .. X_{N-1} \,]$

Proposición 3. La mitad superior de (3.35) corresponde a dos matrices DCT de orden N/2. Se puede comparar la matriz superior izquierda de esta ecuación, con una matriz DCT de orden N/2.

Haciendo $r = 2m$, para $m = 0, 1, .., M-1$, donde $M = N/2$ y $\Theta_m = m\pi/2M$.

Entonces el cuadrante superior izquierdo de una matriz DCT de tamaño N x

N, esta dado por :

$\cos\theta_r \cos(2\pi rk/N) - \text{sen}\theta_r \,\text{sen}(2\pi rk/N)$

$= \cos\theta_m \cos(2\pi mk/M) - \text{sen}\theta_m \,\text{sen}(2\pi mk/M)$

lo cual corresponde a elementos de una matriz DCT de M x M .

Definición del algoritmo recursivo

Se definirá ahora la relación existente entre las sub-matrices de (3.35). El kernel de la ecuación (3.31) se puede escribir como :

$$\tilde{t}_{rk} = \cos r\,\phi_k \quad ; \quad \text{donde } \phi_k = (k + 1/4)(2\pi/N) \quad ; \quad \text{para } k = 0,1..\ N-1$$

entonces de (3.35) para M = N/2 :

$$\tilde{D}(M) = \begin{bmatrix} \cos\phi_0 & \cos\phi_1 & \cos\phi_{M-1} \\ \cos 3\phi_0 & \cos 3\phi_1 & \cos 3\phi_{M-1} \\ \vdots & \vdots & \\ \cos(N-1)\phi_0 & \cos(N-1)\phi_1 & \cos(N-1)\phi_{M-1} \end{bmatrix}$$

De la misma manera para $\tilde{E}(M)$:

$$\tilde{E}(M) = \begin{bmatrix} 1 & 1 & 1 \\ \cos 2\phi_0 & \cos 2\phi_1 & \cos 2\phi_{M-1} \\ \vdots & \vdots & \\ \cos(N-2)\phi_0 & \cos(N-2)\phi_1 & \cos(N-2)\phi_{M-1} \end{bmatrix}$$

Haciendo uso de la siguiente identidad :

$\cos(2r+1)\phi_m = 2\cos(2r\phi_m)\cos\phi_m - \cos(2r-1)\phi_m$, se puede obtener la siguiente relación :

$$\tilde{D}(M) = L\,\tilde{E}(M)\,\text{diag}\,[\cos\phi_m\,] \text{ , para } m = 0,1,..,M-1 \qquad (3.36)$$

donde L es una matriz triangular inferior, dada por :

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ -1 & 2 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ -1 & -2 & 2 & 0 & 0 & 0 & 0 & \dots & 0 \\ -1 & 2 & -2 & 2 & 0 & 0 & 0 & \dots & 0 \\ -1 & 2 & -2 & 2 & -2 & 2 & -2 & \dots & 2 \end{bmatrix} \qquad (3.37)$$

Resumiendo de (3.35) se tiene :

$$\begin{bmatrix} X_{par} \\ X_{imp} \end{bmatrix} = \begin{bmatrix} \tilde{E}(M) & \tilde{E}(M) \\ \tilde{D}(M) & -\tilde{D}(M) \end{bmatrix} \begin{bmatrix} \tilde{x}_p \\ \tilde{x}_i \end{bmatrix} \qquad (3.38)$$

donde las X_{par} y X_{imp} son respectivamente las partes par e impar de la secuencia de la DCT y las $\tilde{x}$ son ordenadas de acuerdo a la matriz de permutación definida en la ecuación (3.30c).

Haciendo R la matriz de permutación, que realiza el reordenamiento de la salida en " bit-reverse ". De tal manera que : $\tilde{X}_{par} = R\, X_{par}$; $\tilde{X}_{imp} = R\, X_{imp}$; por este reordenamiento y por el uso de la proposición 3, de (3.38) obtenemos :

$$\begin{bmatrix} \tilde{X}_{par} \\ \tilde{X}_{imp} \end{bmatrix} = \begin{bmatrix} \tilde{T}(M) & \tilde{T}(M) \\ \tilde{D}(M) & -\tilde{D}(M) \end{bmatrix} \begin{bmatrix} \tilde{x}_p \\ \tilde{x}_i \end{bmatrix} \qquad (3.39)$$

donde $\tilde{T}(M) = R\, \tilde{E}(M)$. Y a partir de (3.36) :

$\tilde{D}(M) = R\, L\, \tilde{E}(M)\, \mathrm{diag}\,[\cos\phi_m]$; para $m = 0,1,..,M-1$

Finalmente se obtiene la decimación en frecuencia del algoritmo recursivo para el calculo de una DCT, de (3.39) se tiene :

$$\begin{bmatrix} \tilde{X}_{par} \\ \tilde{X}_{imp} \end{bmatrix} = \frac{2}{N} \begin{bmatrix} \tilde{T}(N/2) & \tilde{T}(N/2) \\ K\,\tilde{T}(N/2)\,Q & -K\,\tilde{T}(N/2)\,Q \end{bmatrix} \begin{bmatrix} \tilde{x}_p \\ \tilde{x}_i \end{bmatrix} \qquad (3.40)$$

donde $K = R\,L\,R'$. Pero como R es simétrica, entonces :

$$K = R\,L\,R \qquad (3.41)$$

$$Q = \mathrm{diag}[\cos \varnothing_m\,] \;;\quad \text{para}\quad m = 0,1\,..\,n/2\,-1 \qquad (3.42)$$

La decimación en tiempo puede ser obtenida a partir de (3.40), invirtiendo el indexado entre la entrada y la salida y utilizando la transpuesta de $\tilde{T}(N)$, se tiene :

$$\begin{bmatrix} \tilde{X}_{par} \\ \tilde{X}_{imp} \end{bmatrix} = \frac{2}{N} \begin{bmatrix} \tilde{T}'(N/2) & Q\,T'(N/2)K' \\ \tilde{T}'(N/2) & -Q\,T'(N/2)K' \end{bmatrix} \begin{bmatrix} \hat{x}_p \\ \hat{x}_i \end{bmatrix} \qquad (3.43)$$

donde, $\hat{x}_p = R\,x_p$ y $\hat{x}_i = R\,x_i$

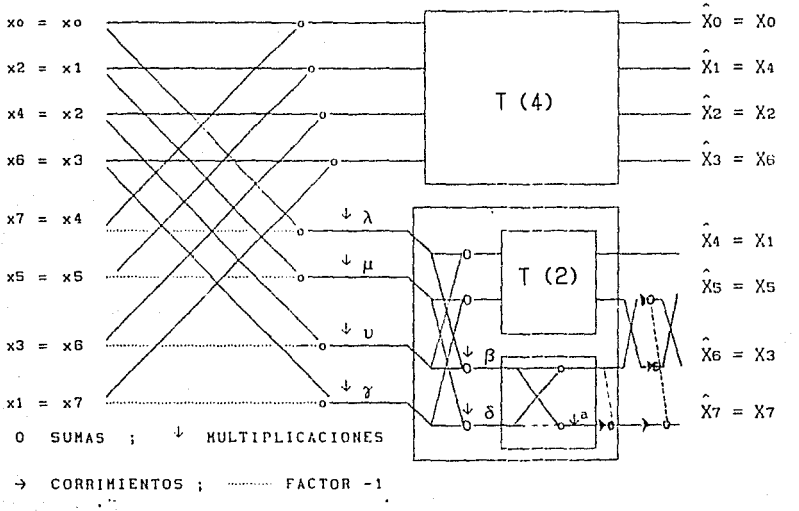


Fig. 3.4 Transformada DCT de orden 16, utilizando el algoritmo MCR.

Evaluación del algoritmo

El número de operaciones necesarias para calcular una DCT de 64 puntos es de 192 múltiplicaciones y 513 sumas, esta cantidad es la misma que la que se requiere para el método MCD.

El problema que se presenta con este algoritmo, es la necesidad de reordenar los datos en cada etapa, ver la fig. 3.4. Observando la expresión (3.40) para el cálculo de una DCT, se necesita múltiplicar una matriz de coeficientes por la secuencia de entrada ordenada, lo cual corresponde a la parte superior de la matriz DCT; para la parte inferior de esta matriz de coeficientes, antes de ser multiplicada debe de calcularse un nuevo indexado, aunque también es un " bit-reverse " hay que agregar corrimientos sumas y restas como lo indica (3.41).

Aprovechando la ventaja de su recursividad se puede calcular una transformada DCT de orden 64, tomando como estructura básica una transformada DCT de orden 8 . Sin embargo aunque se tiene una tabla de coeficientes menor se deben de ordenar los datos en cada etapa, esta opción fué finalmente la utilizada en las simulaciones en Pascal.

3.5 Implantación de los algoritmos MCI y MCR en lenguaje ensamblador

Para efectuar la DCT de una imagen digital de tamaño N x N, generalmente se divide esta en bloques más pequeños, esta operación de subdivisión se hace para reducir los cálculos, ya que el número de operaciones se incrementa conforme aumenta el tamaño de la transformada. Para la parte experimental de este trabajo, se utilizaron imagenes de 256 x 256, las cuales fuerón subdivididas en bloques de 64 x 64, después de lo cual se les aplicó una transformación DCT bidimensional, que se puede obtener a partir de la utilización de una DCT unidimensional, aplicando esta sobre los renglones y enseguida sobre las columnas de los coeficientes transformados; al tomar la transpuesta de los resultados se obtiene la DCT bidimensional.

Como se mencionó al inicio del capítulo, el primer parámetro de comparación entre los algoritmos MCI y MCR, es el número de operaciones necesarias para el cálculo de una DCT de orden 64, estas operaciones se

resumen en la tabla 3.1, como puede observarse el mejor caso ocurre para el método MCI . Estos algoritmos fuerón simulados en lenguaje de alto nivel, Pascal, con operaciones aritméticas enteras, para poder determinar el mejor formato. Los datos de entrada fuerón multiplicados por 64 (formato Q6) y las constantes por 4096 (formato Q12). Otros parámetros determinados a partir de las simulación, fuerón : la longitud de las operaciones (16 o 32 bits) y los requerimientos de memoria.

	Método MCI		Método MCR	
	DCT	IDCT	DCT	IDCT
Número de sumas	484	964	513	513
Número de multiplicaciones	226	324	192	192

Tabla 3.1 Número de operaciones para el cálculo de las transformaciones DCT e IDCT unidimensionales para N = 64.

La tabla 3.2, resume los resultados obtenidos a partir de la implantación de los algoritmos en lenguaje ensamblador del microprocesador TMS320C25, los cuales fuerón aplicados a una imagen de 256 x 256, con 8 bits por pixel. La precisión de los algoritmos se obtuvo comparando su resultado, en aritmética de punto flotante, haciendo un redondeado a enteros; con respecto al resultado obtenido en lenguaje ensamblador.

La longitud necesaria de las operaciones, para poder obtener una misma relación señal a ruido, entre la versión en lenguaje de alto nivel y la versión en lenguaje ensamblador, requirió la implantación de operaciones en 32 bits, con excepción del cálculo de la transformación IDCT por el método MCI, esto repercute directamente en el tiempo de ejecución del programa.

En lo que respecta a la memoria de programa, hubo una marcada diferencia en favor del método de computación indirecta (MCI), aunque en este se realizan operaciones en 32 bits dentro del algoritmo, los resultados

intermedios (transformada de renglones), quedan en 16 bits. En contraparte, en este mismo método se necesitan 4096 localidades adicionales, para almacenar sus resultados intermedios.

	Método MCI		Método MCR	
	DCT	IDCT	DCT	IDCT
Longitud de las operaciones (bits)	32	16	32	32
Memoria de programa (palabras)	595	640	608	608
Memoria de datos (palabras)	272	224	4287	4287
Tiempo de ejecución por imagen (seg)	3.69	3.8	9.17	9.17
SNR (db)	51	51	51	51

TABLA 3.2 Comparación del los métodos MCI y MCR en lenguaje ensamblador

Finalmente, todos estos aspectos de operaciones en 32 bits y manejo de datos, se ven reflejados en el tiempo de ejecución de los programas. Este favorece considerablemente al método MCI que requiere 3.69 seg. para el cálculo de una DCT y 3.8 seg. para una IDCT. Para calcular estos tiempos se tomó en cuenta la lectura de datos y la escritura de los resultados, considerando un tiempo de 100 nanosegundos por instrucción del microprocesador TMS320C25 . En resumen, en el esquema final de compresión fuerón utilizados los algoritmos del método MCI .

4

ESQUEMA DE CODIFICACION

En este capitulo se presentan inicialmente los criterios de selección para el tamaño de las sub-imagenes, posteriormente se describe el cuantizador que se utiliza para la codificación de los coeficientes de la transformada y finalmente se proprocionan los esquemas seguidos para los procesos de codificación y decodificación.

4.1 Tamaño de los bloques

Un aspecto importante en la codificación, es el tamaño de los bloques en los que se subdivide a la imagen original, para que mediante una transformada unitaria, se efectúe la transformación de esta imagen en un conjunto de coeficientes idealmente independientes, los cuales son codificados posteriormente.

Para secuencias de Markov estacionarias, de primer orden, con una correlación ρ cercana a 0.9, la transformada coseno discreta presenta caracteristicas de compactación de energia muy similares a las que se pueden obtener con la transformada óptima, Karhunen-Loeve (K-L).

Netravali [13] ha demostrado experimentalmente, con imágenes estacionarias de correlación exponencial en la dirección horizontal $\rho_h = 0.95$ y en la dirección vertical $\rho_v = 0.95$, que el error medio cuadrático de codificación a una tasa fija, varía según la transformada y el tamaño del bloque empleado. En la figura 4.1, se observa que para bloques pequeños el error MSE es similar en todas las transformadas al de la transformada K-L, pero para bloques más grandes, solamente la transformada coseno discreta presenta un comportamiento muy similar al de esa transformada ideal.

Gran parte de los trabajos de codificación de imágenes que aplican transformadas unitarias, han sido realizados empleando bloques pequeños,

del orden de 4x4, 8x8, o tipicamente de 16x16. Inclusive la DCT-2D de dimensiones 8x8 y 16x16, ha sido implantada exitosamente en circuitos VLSI [10], para propósitos de codificación de imágenes y señales de video. Si el tamaño de los bloques es nxn, para $n \leq 16$, la DCT-2D es fácil de implantar; pero para obtener una codificación eficiente, es necesario que los coeficientes de la transformada estén tan descorrelacionados como sea posible. Los coeficientes en la frontera entre los bloques adyacentes están también correlacionados, tal efecto es ocasionado por la correlación existente entre los pixeles de la frontera común. Entonces para disminuir la correlación, una solución radica en incrementar el tamaño de los bloques, como se muestra en la Fig. 4.1.

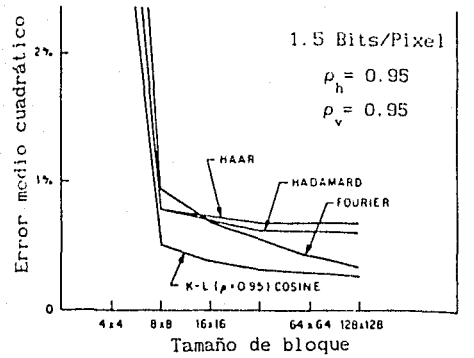


Fig. 4.1 Evaluación del MSE a una tasa fija de compresión, para diferentes transformadas, empleando una imagen Markoviana bidimensional con factor de correlación horizontal y vertical $\rho_h = \rho_v = 0.95$.

Acercándose al límite del tamaño de los sub-bloques, se puede codificar la imagen completa como lo establece la técnica conocida como "full frame bit-allocation", empleada por Lo y Huang [8]. Tal técnica ha sido utilizada en imágenes de diagnóstico médico de dimensiones 512x512,

1024x1024 y 2048x2048, aplicando una sola DCT-2D, del mismo orden.

En el presente trabajo se emplearon bloques de dimensiones 16x16, 32x32 y 64x64; experimentalmente se demostró que a medida que se incrementa el tamaño de los bloques es posible obtener una mejor calidad en la reconstrucción. En consecuencia, la implantación final de los algoritmos fue realizada sobre bloques de 64x64.

4.2 Cuantización de los coeficientes de la DCT

Haciendo $g(x,y)$, para $x,y= 0,1,\ldots,N-1$, los pixeles de un bloque fuente de tamaño NxN; $G(u,v)$ para $u,v= 0,1,\ldots,N-1$, su transformada coseno discreta; $\hat{G}(u,v)$ el bloque decodificado de coeficientes de la transformada y $\hat{g}(x,y)$ los pixeles reconstruidos del bloque cuantizado. Se tiene el error medio cuadrático promedio entre un bloque fuente y su reconstrucción, expresado mediante:

$$e = \frac{1}{N^2} E \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} [\, g(x,y) - \hat{g}(x,y) \,]^2 \qquad (4.1)$$

Puesto que cada coeficiente de la transformada es una combinación lineal de los N^2 pixeles, cuando N tiende a ser grande la distribución de $G(u,v)$, para $u,v= 0,1,..N-1$, tiende a ser Gaussiana; el término $G(0,0)$ es mejor modelado mediante una función de densidad de Rayleigh; a partir de estas propiedades, el cuantizador empleado es no-uniforme, ya que para el caso de una distribución Gaussiana, un cuantizador de este tipo puede ser entre 20 y 30%, más eficiente que un cuantizador uniforme [20].

Para realizar la codificación es necesario determinar el número de bits con que se debe cuantizar $G(u,v)$, para $u,v= 0,1,..N-1$. Un primer paso es conocer la cota mínima con que se puede representar cada uno de los coeficientes. Sabiendo que se tiene una variable aleatoria Gaussiana de varianza σ^2, no es posible representarla con menos de $1/2 \log_2(\sigma^2/e)$ bits y si se desea que la cuantización sea proporcional al contenido de la información de cada coeficiente, entonces la asignación de bits debe realizarse proporcionalmente a $\log_2 \sigma^2$.

Mediante la normalización de cada coeficiente de la transformada con respecto a la desviación estandar respectiva, esto es $G(u,v)/\sigma_{u,v}$, es

posible obtener variables aleatorias de varianza unitaria. Cuando el cuantizador está diseñado para operar con variables de este tipo y considerando $d_{u,v}$ el error de cuantización medio cuadratico, el promedio del error de cuantización total D para todo un bloque se expresa como:

$$D = \frac{1}{N^2} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \sigma^2_{u,v} \cdot d_{u,v} \cdot b_{u,v} \qquad (4.2)$$

donde $b_{u,v}$ (el número de bits con que se debe de cuantizar al coeficiente u,v respectivo), debe ser tal que minimice a D, sujeto a:

$$\sum_{u=0}^{N-1} \sum_{v=0}^{N-1} b_{u,v} = N^2 \cdot b_{prom} \qquad (4.3)$$

donde b_{prom} es el valor promedio en bits que necesitaria cada coeficiente. Asi, una expresión obtenida por Wintz [12], que minimiza la ecuación (4.2), sujeta a la restricción (4.3) es:

$$B(u,v) = b_{prom} + \frac{2}{\log_{10}} \left[\ln\sigma^2_{u,v} - \frac{1}{N.N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \ln\sigma^2_{u,v} \right] \qquad (4.4)$$

4.3 Codificación

Siendo M=256, N=64 y g(x,y) la imagen original de tamaño MxM, para x,y= 0,1,...,M-1. Primeramente debe aplicarse sobre g(x,y) un preprocesamiento, consistente en dividir la imagen en sub-bloques de tamaño NxN, obteniendose los bloques $g_i'(x,y)$ con x,y= 0,1,...,N-1, donde i es el número de bloque e i= $0,1,\ldots,(M/N)^2-1$.

Enseguida, a partir de las transformadas coseno discretas unidimensionales de longitud N, a cada bloque $g_i'(x,y)$ le es aplicada una transformada DCT-2D, utilizando el algoritmo MCI, presentado en la sección 3.2.

Antes de efectuar la cuantización de los coeficientes $G_i(u,v)$ de la transformada, estos deben ser normalizados empleando una matriz de coeficientes máximos M(u,v), obtenida mediante:

$$M(u,v) = \max \left\{ | G_i(u,v) | \right\} \qquad (4.5)$$

para $u,v = 0,1,\ldots,N-1$
$i = 0,1,\ldots,(M/N)^2-1$

y para la normalización:

$$G_{ni}(u,v) = \frac{G_i(u,v)}{M(u,v)} \qquad (4.6)$$

de esta manera $G_{ni}(u,v)$, toma valores entre +1 y -1.

Luego mediante la expresión (4.4), se calcula la tabla de asignación de bits B(u,v), asignando iterativamente valores a b_{prom}, hasta que la expresión:

$$\frac{\sum_{u=0}^{N-1} \sum_{v=0}^{N-1} B(u,v)}{N^2} \qquad (4.7)$$

proporcione la tasa de compresión deseada. La representación de B(u,v) se realiza empleando 8 bits por elemento.

Las varianzas de la expresión (4.4) son calculadas mediante:

$$\sigma^2_{u,v} = E\left\{ |G_i(u,v)|^2 \right\} = \frac{1}{(M/N)^2} \sum_{i=0}^{(M/N)^2-1} \left[G_i(u,v) \right]^2 \qquad (4.8)$$

donde i es el número de bloque procesado.

La cuantización se efectua con un máximo de 16 bits, según la magnitud y signo de los coeficientes $G_{ni}(u,v)$, utilizando las expresiones:

$$C_i(u,v) = \begin{cases} G_{ni}(u,v) \cdot (2^{B(u,v)-1} - 1) & \text{si } G_{ni}(u,v) \geq 0 \\ G_{ni}(u,v) \cdot (2^{B(u,v)-1}) & \text{si } G_{ni}(u,v) < 0 \end{cases} \qquad (4.9)$$

donde $C_i(u,v)$ son los coeficientes codificados del bloque i.

En la figura 4.2 se esquematizan los procesos completos para la codificación y decodificación.

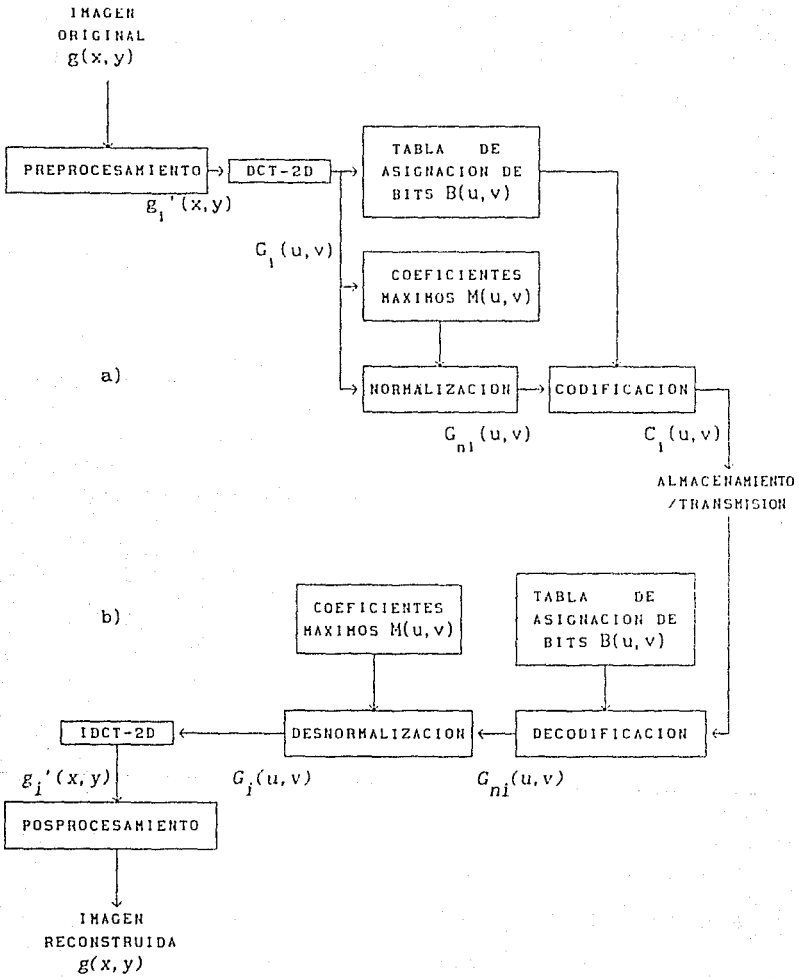


Fig. 4.2 Esquemas que resumen los procesos de: a) codificación y b) decodificación

4.4 Decodificación

Para la reconstrucción de la imagen codificada es necesario contar con la tabla de asignación de bits B(u,v) y la matriz de máximos M(u,v). Inicialmente cada coeficiente C(u,v) es expandido de acuerdo al número de bits empleados durante su codificación, por medio de las expresiones:

$$G_{ni}(u,v)= \begin{cases} C_i(u,v)/\ (2^{B(u,v)-1}-1) & \text{si } C_i(u,v) \geq 0 \\ C_i(u,v)/\ (2^{B(u,v)-1}) & \text{si } C_i(u,v) < 0 \end{cases} \qquad (4.10)$$

Enseguida se efectúa la desnormalización con respecto a la matriz de máximos, por medio de:

$$G_i(u,v)= G_{ni}(u,v) \cdot M(u,v) \qquad (4.11)$$

Despues, para calcular la transformada coseno discreta inversa bidimensional, se aplica el algoritmo MCI visto en la sección 3.3, sobre cada uno de los bloques $G_i(x,y)$ de coeficientes reconstruidos. Las matrices resultantes $g_i(x,y)$ representan la imagen reconstruida particionada en sub-imágenes, entonces debe aplicarse un posprocesamiento que posicione los bloques $g_i(x,y)$ en el orden natural.

Finalmente, la calidad obtenida puede ser evaluada calculando el NMSE, la SNR, o bién aplicando otro criterio de fidelidad, al comparar la imagen reconstruida $g(x,y)$ con la imagen original $g(x,y)$.

5

MICROPROCESADORES PARA EL PROCESAMIENTO DIGITAL DE SENALES

Aun y cuando el procesamiento numérico intensivo no es nuevo, en los últimos años ha surgido como factor determinante para el desarrollo de los sistemas de simulación y en general del procesamiento de señales, el contar con equipo de cómputo que funcione a muy alta velocidad y preferentemente en *tiempo real*.

Asi, actualmente en pocos minutos, es posible realizar en estaciones de trabajo, simulaciones de choques para el diseño de automóviles, que hasta hace no mucho tiempo tomaban varias horas. Tipicamente las estaciones de trabajo comerciales realizan modelados tridimensionales, desarrollando alrededor de 3.5 millones de operaciones por segundo en aritmética de punto flotante , gracias al empleo de procesadores matemáticos rápidos.

A continuación se presenta un resumen con las caracteristicas más significativas de los procesadores, por medio de los cuales es posible ejecutar operaciones aritméticas a muy alta velocidad.

Posteriormente, se describe la estrategia seguida para la implantación del esquema de codificación en el microprocesador TMS320C25, así como el procedimiento para efectuar las operaciones aritméticas necesarias para la codificación y no contenidas en el conjunto de instrucciones del dispositivo (raiz cuadráda y logaritmos).

5.1 Procesadores aritméticos rápidos.

En general, el procesamiento aritmético puede acelerarse empleando arreglos de procesadores de propósito general, en parejas del tipo *procesadores-coprocesadores*, o bién empleando procesadores de propósito específico, para procesamiento digital de señales (DSP). Ambos casos se describen a continuación.

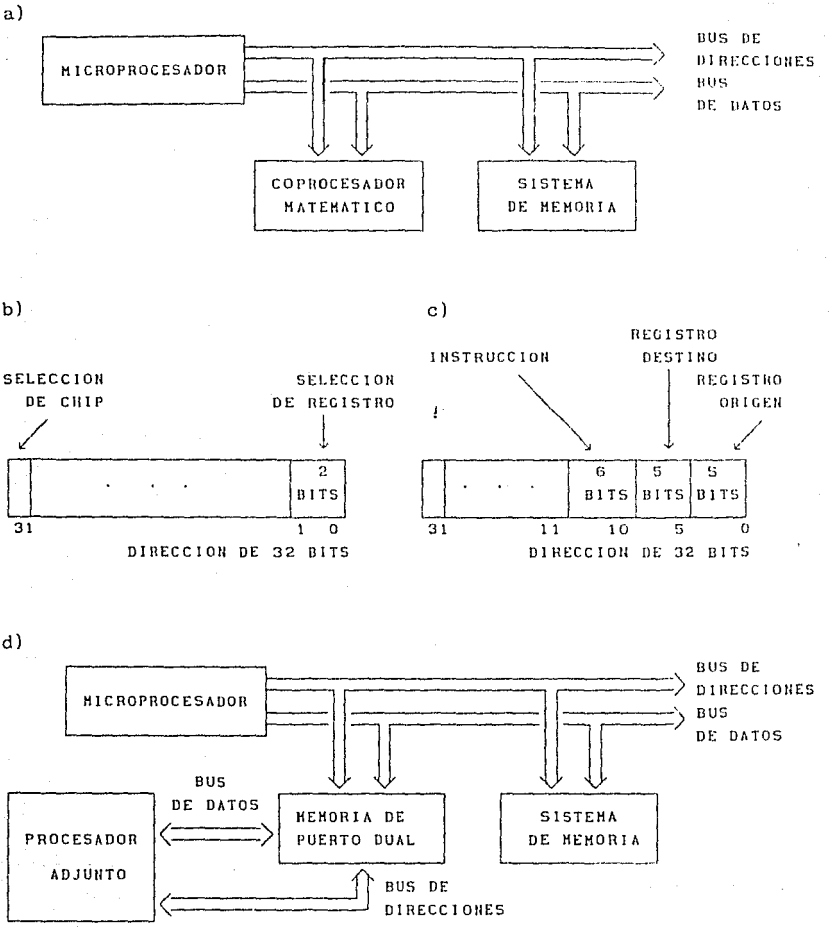


Fig. 5.1 Comunicación entre la unidad central de procesamiento (CPU) y el coprocesador matemático.
a) Un coprocesador estandar recibe instrucciones y datos por el bus de datos. b) Las palabras de direcciones cuentan con 1 bit solamente como señal de aviso al coprocesador y un registro de dirección de 2 bits. c) El *coprocesador de memoria mapeada* recibe instrucciones y direcciones por el bus de direcciones, al al mismo tiempo que los datos son enviados por el bus de datos. d) Los *procesadores adjuntos* reciben instrucciones y datos directamente del CPU, via una *memoria de puerto dual*. Teniendo acceso a esta siempre que sea necesario.

5.1.1 Coprocesadores

Este es el medio más fácil y barato de incrementar la velocidad de un sistema pequeño de cálculo, por ejemplo, una microcomputadora personal.

Para el procesador anfitrión, el coprocesador representa una extensión de su arquitectura, que ejecuta cierto tipo de operaciones aritméticas más rápida y eficientemente.

5.1.2 Coprocesadores de memoria mapeada

Existe una segunda clase de coprocesadores con un mejor desempeño sobre los coprocesadores estandar, ya que emplean ventajosamente los buses de la unidad central de procesamiento (CPU), ver Figura 5.1. Normalmente son ofrecidos por los fabricantes de estaciones de trabajo, como tarjetas opcionales, aceleradoras de operaciones en aritmética de punto flotante.

Los *coprocesadores de memoria mapeada* son tomados por la unidad central de procesamiento como un segmento de la memoria principal. Aún y cuando el formato y contenido de sus instrucciones no son semejantes a las del CPU anfitrión, es posible contar con los beneficios de la programación en lenguajes de alto nivel

5.1.3 Coprocesadores adjuntos de propósito general.

Estos coprocesadores son ampliamente usados en la graficacion y la simulación de circuitos. A diferencia de los coprocesadores que ejecutan solo determinadas instrucciones, los *procesadores adjuntos* ejecutan programas completos o secciones de código compilado, predominando las operaciones aritméticas y excluyendo las tareas de acceso a los recursos del sistema.

Un *procesador adjunto*, debe ser diseñado para combinar una arquitectura óptima en el cálculo de operaciones aritméticas (con un alto grado de paralelismo, por ejemplo), con una estructura que permita emplear elementos de programación de alto nivel (compiladores y debuggers).

En la Tabla 5.1 se muestran los coprocesadores más representativos de la clasificación anterior.

FABRICANTE	UNIDAD •	VELOCIDAD EN MIPS †	VELOCIDAD EN MFLOPS ‡
COPROCESADORES			
INTEL	8087 (8086,8088)	0.5	< 0.1
	80287 (80286)	1.5	< 0.1
	80387 (80386)	4.0	0.2- 0.24
MOTOROLA	68881 (68020,68030)	4.0, 6.0	0.11-0.13
	68882 (68020,68030)	4.0, 6.0	0.22-0.26 (ESTIMADO)
COPROCESADOR DE MEMORIA MAPEADA			
WEITEK CORP.	WTL1167 (INTEL 80386)	4.0	0.41-0.67
PROCESADORES ADJUNTOS DE PROPOSITO GENERAL			
ADVANCED MICRO DEVICES	Am29027 @ (Am 29000)	12 (ESTIMADO)	2 (ESTIMADO)
FAIRCHILD	CLIPPER	5	0.6- 1.0
INMOS LTD.	TRANSPUTER T800	8	NO DISPONIBLE
MIPS COMPUTER INC.	R2010 (R2000)	6	0.6- 0.9
WEITEK	XL-8064	7	1.7- 1.8

Tabla 5.1 Alternativas en el empleo de coprocesadores matemáticos.

• Entre paréntesis se indica el procesador para el cual operan.

† 1 MIPS es el equivalente a 1 millon de instrucciones por segundo, ejecutadas por una computadora VAX 11/780.

‡ La velocidad en MFLOPS (millones de operaciones en punto flotante por segundo), fue medida empleando el programa Linpack.

@ A excepción de este dispositivo; para el resto de los coprocesadores se cuenta con un alto grado de soporte para la programación a partir de los lenguajes de alto nivel.

5.1.4 Procesadores especializados

En el comienzo de la década de los 70's, aparecieron los primeros arreglos de microprocesadores de punto fijo y punto flotante, para el procesamiento digital de señales (DSP). El costo de estos dispositivos oscilaba entre 10,000 y 50,000 dolares, con un peso de aproximadamente 100

lbs y un consumo de potencia de alrededor de un kilowatt.

La evolución tecnológica en circuitos integrados, ha propiciado la aparición de procesadores de señales que han excedido considerablemente los logros alcanzados por los arreglos con coprocesadores, puesto que estos nuevos dispositivos son capaces de ajecutar varios millones de instrucciones por segundo (MIPS). Para obtener tales logros, se ha hecho uso intensivo de la arquitectura Harvard y de las técnicas de pipelining, a fin de ejecutar instrucciones en paralelo y con etapas superpuestas, lo cual reduce significativamente el ciclo de instrucción. Así, en el microprocesador TMS320C25, se emplea una estructura pipeline de tres niveles y otra de cuatro niveles en el microprocesador TMS320C30 [5].

En general las técnicas de pipelining se clasifican en: *entrelazado, codificación de tiempo estacionario y codificación de datos estacionarios*[7].

Los microprocesadores de procesamiento digital de señales del fabricante Texas Instrument, han utilizado ampliamente el pipelining de tipo *entrelazado*. Por medio del cual el usuario tiene la impresión de que, cada instruccion es ejecutada antes de que comience la siguiente. La temporización y el paralelismo son dados por el propio procesador, siendo desconocido para el usuario el número de sub-ciclos que consume cada instrucción, puesto que depende de las instrucciones vecinas.

Mediante la *codificación de tiempo estacionario*, utilizada por Motorola, y por AT&T en el procesador DSP16/16A, el usuario tiene control sobre los estados del pipeline, puesto que se puede especificar en las propias instrucciones, las etapas de las mismas que deben ejecutarse paralelamente. Se cuenta adicionalmente con un mayor control sobre las interrupciones, pudiendose ejecutarse estas más rapidamente.

El fabricante AT&T también ha aplicado la técnica de pipelinig por medio de la *codificación de datos estacionarios*, en el procesador DSP32/32C. A diferencia la técnica de pipelinig anterior, se tiene menor control sobre el hardware en determinado tiempo, y en las instrucciones se especifica unicamente las operaciones a ejecutar sobre algún dato. Otra diferencia que no necesariamente significa menor eficiencia, es que los resultados de la instrucción pueden no estar disponibles durante la siguiente instrucción.

FABRICANTE	PROCESADOR	FECHA	MAC (nseg.)	BITS PTO.FIJO ‡	BITS PTO. FLOTANTE ‡
AMI	S2811	78	300	12/16	
ANALOG DEVICES	ADSP-2100	86	125	16/40	
	ADSP-2100A	88	80-100	16/40	
AT&T	DSP32	84	160	16	32/40
	DSP32C	88	80	16/24	32/40
	DSP16	87	55	16/36	
	DSP16A	88	33	16/36	
FUJITSU	MB8764	83	100	16/26	
	MB86232	87	150		32
	MB86220	89	160		24
HITACHI	HD61810	82	250		12/16
	DSP1	88	50		
IBM	HERMES	81	100	NO COMERCIAL	
MOTOROLA	DSP56001	87	74.1	24/56	
	DSP96002	89	75	32/64	44/96
NATIONAL	LM32900	87	100	16/32	
NEC	µPD7720	80	250		
	µPD77230	85	150		32
	µPD77220	86	100	24/48	
OKI	6992	86	100		22
THOMSON/ MOSTEK	68930	87	360	16/32	
TOSHIBA	6386/7	83	250	16/31	
ZORAN	34161	86	100		16
	35325	89			32
TEXAS INSTR.	TMS32010	82	400	16/32	
	TMS32020	85	200	16/32	
	TMS320C25	87	100	16/32	
	TMS320C30	88	60	24/32	32/40

Tabla 5.2 Principales microprocesadores de procesamiento digital de señales (DSP), de punto fijo y punto de flotante [6]. Se emplea como parámetro comparativo el tiempo de procesamiento utilizado en la instrucción MAC.

‡ Bits en palabra de datos/bits en la ALU.

A diferencia de los procesadores de propósito general, en los procesadores DSP, se emplea una lógica cableada; es decir, sin microprogramas, integrandose además un harware para el cálculo de ciertas operaciones aritméticas.

En la tabla 5.2 se proporciona una lista de los principales procesadores tipo DSP, tomándose como medida de calidad, el tiempo necesario para ejecutar una instruccion MAC, esto es:

$$\begin{aligned} ACC &\longleftarrow reg\ P + ACC \\ reg\ P &\longleftarrow dato\ X * dato\ Y \end{aligned}$$

El gran atractivo de los procesadores DSP de punto flotante, radica en que no es necesario escalar o normalizar la información, puesto que se puede obtener un mayor rango dinámico al emplear más bits en la palabra de datos (con 8 bits para el exponente y 24 bits para la mantisa, en el procesador TMS320C30 se alcanzan alrededor de 1500 dB de rango dinámico); con lo cual se reducen los riesgos de saturación. Este tipo de procesadores son los de más alta velocidad, sin embargo, el tener que manejar 3 campos (signo, mantisa y exponente) por dato numérico, ocasiona que se disminuya considerablemente la velocidad de procesamiento.

Los desventajas anteriores, son eliminadas al emplearse un procesador con aritmética de punto fijo.

Para reducir los errores por falta de precisión, estos deben programarse tomando en cuenta la estadistica de la señal, ya que necesariamente deben escalarse los datos y resultados intermedios. Al emplear datos de longitud de N bits, es necesario que la ALU y los acumuladores sean de almenos 2N bits, para evitar desbordamientos cuando se calculan operaciones sucesivas.

Es recomendable tambien, contar en la ALU con recursos para prevenir errores por saturación o por truncamiento. Asi por ejemplo, el procesador DSP96002 de Motorola, cuenta con cuatro modos de redondeo por hardware.

Resulta importante la longitud de la palabra de datos, puesto que el rango dinámico es incrementado en 6 dB con cada bit empleado [6]. Por ejemplo, el procesador ADSP-2100 presenta un rango dinámico de 96 dB (16 bits) y el DSP56001 otro de 144 dB (24 bits).

Buscando optimizar el acceso a memoria, se han incorporado instrucciones especializadas de direccionamiento. Asi mismo, la arquitectura Harvard ha sido modificada para emplear bancos paralelos de memoria, memorias multi-puerto, instrucciones *cache*, etc.

En general los procesadores de tipo DSP, han sido diseñados para accesar eficientemente y a la velocidad máxima, una reducida cantidad de datos; como lo es la información entre registros y la contenida en la memoria interna del circuito integrado.

Asi por ejemplo, el microprocesador TMS320C25 trabaja a la velocidad pico anunciada por el fabricante (10 MIPS), solamente cuando los datos son internos al circuito. Mientras que al emplear memorias externas (aún y cuando estas sean lo suficientemente rápidas para no necesitar ciclos de espera), la velocidad puede ser hasta 5 veces menor a la anunciada.

De entre los procesadores de punto flotante, el DSP96002 de Motorola es el que presenta el mayor número de recursos asignados. Sus principales caracteristicas se resumen en la tabla 5.3.

BITS PTO. FLOTANTE †	INTERUP /SEG.	MEMORIA ‡		MFLOPS	NORMA ARITHETICA
		INTERNA	EXTERNA		
32/64	2M	512 RAM-Prog. (2)512 RAM-Dat. (2)512 ROM-Dat.	4G-Prog. (2)4G-Dat.	40.5	IEEE 754-1985

Tabla 5.3 Principales caracteristicas del microprocesador DSP96002.

† Bits en palabra de datos/bits en la ALU.
‡ Se indica la capacidad en palabras.

Se muestra en la tabla 5.4, un análisis comparativo entre los mejores microprocesadores DSP de punto fijo. Se observa que el DSP56000/1 presenta la mejor asignación de recursos.

El desempeño medido en la ejecución de diversos programas (filtrado IIR y FIR, multiplicación matricial, tranformación FFT, convolución

RECURSO	ANALOG DEVICES ADSP-2101/2	NEC 77C25	AT&T DSP16A	TI TMS320C25	MOTOROLA DSP56000/1
CICLO INSTRUCCIÓN(nseg.)	80	122	33	100	74.1
PALABRA DE DATOS (bits)	16	16	16	16	24
PALABRA DE PROGRAMA(bits)	24	24	16	16	24
RAM DATOS	1K	256	2048	544	2X256
ROM DATOS	-	1024	4096	NINGUNO	2X256
MEMORIA DE PROGRAMA	2K	2048 ROM	NO HARVARD	4096 ROM	3840 ROM ó 512 RAM
PILA(STACK)	16	4	NINGUNO	8	2X15
MULTIPLICADOR	16X16	16X16	16X16	16X16	24X24
TAMAÑO DE ALU (bits)	16	16	36	32	56
ACUMULADORES	2X40	2X16	2X36	1X32	2X56
REGISTROS AUX.	24	2	6	8	24
EXPANSIÓN DE MEMORIA	64K X8	NINGUNA	64K	128K	192K
INTERRUPCIONES	4	1	5	7	18
PINS	68	28	84	68	88
TECNOLOGIA	1.0u CMOS	CMOS	CMOS	1.8u CMOS	1.2u CMOS
DISIPACIÓN MAX. DE POTENCIA	0.8 W	?	0.450W	1.5W	0.5W (tipico)

Tabla 5.4 Comparación de recursos de los microprocesadores DSP de punto fijo.

bidimensional, etc), ha demostrado que en la actualidad, el DSP56000/1 es el procesador más adecuado para la implantación de algoritmos de procesamiento digital de señales.

El microprocesador empleado en este trabajo, el TMS320C25, aún y cuando no incorpora todas las caracteristicas del DSP56000/1, como lo son: extensa memoria RAM/ROM, registos auxiliares adicionales y sobre todo una ALU y una palabra de datos de mayor longitud. Sin embargo, si presenta la versatilidad necesaria en el conjunto de instrucciónes y en los recursos fisicos, para implantar el esquema propuesto de codificación de imágenes, motivo de este estudio.

5.2 Metodología de implantación del esquema de codificación en aritmética de punto fijo.

Aún y cuando comienzan a aparecer herramientas semejantes a las empleadas en los procesadores de propósito general (compiladores, debuggers, etc.), la implantación de algoritmos en el lenguaje ensamblador de los microprocesadores DSP, no es una tarea fácil de realizar.

Por ejemplo, el paquete SIMPAC (sistema de implantación en procesadores digitales de señales asistido por computadora) [17], efectúa en forma transparente para el usuario, la traducción de programas creados en lenguaje Pascal, al lenguaje de los microprocesadores TMS32010 y TMS320C25. Sin embargo, el código generado por el sistema y los formatos asignados a las variables no siempre són óptimos, recomendándose entonces su empleo para programas de complejidad razonables o cuando las exigencias de tiempo no constituyen una limitante importante.

En caso contrario, como en el presente trabajo, cuando los algoritmos son altamente complejos y se desea que estos sean ejecutados eficientemente y en el menor tiempo posible, es necesario efectuar la implantación manualmente.

La metodología seguida para la implantación se compone de las siguientes etapas*:

- Desarrollo de algoritmos en aritmética de punto flotante
- Simulación de algoritmos en aritmética entera
- Implantación en lenguaje ensamblador

Las dos primeras etapas fueron desarrolladas en lenguaje Pascal, ya que a partir de la versión 4 de Turbo Pascal, se cuenta con representaciónes enteras de 8, 16 y 32 bits; útiles para la representación de variables y registros del TMS320C25.

* Esencialmente se emplea la misma metodología desarrollada empíricamente por Alcantara [1].

5.2.1 Desarrollo de algoritmos en aritmética de punto flotante

Esta etapa es relativamente sencilla, ya que se emplean todos los recursos de los lenguajes de alto nivel (operaciones aritméticas, arreglos vectoriales, archivos, etc.), para el desarrollo algoritmico. Y puesto que se trabaja con punto flotante, pueden emplearse, incluso variables reales de doble precisión para conocer los resultados "ideales" a obtener, estableciendo los limites algoritmicos.

El punto final de esta etapa, es la determinación de la dinámica de las variables, es decir, conocer su acotamiento, para lo cual pueden elaborarse histogramas.

5.2.2 Simulación de algoritmos en aritmética entera

Después de obtener el mejor desempeño algoritmico en la etapa anterior, la versión final obtenida debe ser implantada en aritmética entera.

Esta etapa debe ser desarrollada tomando en cuentalos recursos fisicos y aritméticos del microprocesador empleado, como son: longitud de la palabra de datos y de la ALU, tamaño de los acumuladores y de los registros auxiliares, y capacidad aritmética y de memoria RAM/ROM. Es decir, se deben desechar o modificar, por ejemplo, los algoritmos que presenten grandes requerimientos de memoria, o los procesos aritméticos que conduzcan a inestabilidades numéricas debido a su representación entera, (la cual emplea un número finito de bits), generando en concecuencia errores por truncamiento, por redondeo, por una convergencia inadecuada, etc.

Para tener mayor versatilidad en la programación, es recomendable efectuar esta etapa en lenguaje de alto nivel, puesto que la dinámica de las variables exige que las operaciones aritméticas se lleven a cabo mediante escalamientos o normalizaciones de datos y resultados intermedios.

En realidad, la parte de la implantación que demanda mayor tiempo y atención, para efectos de conservar una precisión adecuada, es la asignación de formatos; es decir, la asignación del número de bits para representar la parte entera y la parte fraccionaria de las variables.

5.2.3 Implantación en lenguaje ensamblador

Conocer la dinámica de las variables y los formatos adecuados para su representación, facilita enormemente la labor de traducir los algoritmos al lenguaje ensamblador.

En esta etapa, la experiencia e intuición del programador para representar las estructuras empleadas en lenguaje de alto nivel, por ejemplo, y para efectuar el cambio de las variables y el direccionamiento de datos, son importantes para generar un código de programa óptimo. Esto es, un código óptimo en el sentido de una administración adecuada de los recursos y de la rapidez de ejecución; manteniendo una precisión adecuada.

En el presente trabajo, por tratarse de un sistema de alta complejidad, la validación de los resultados fué realizada evaluando inicialmente los resultados parciales (por medio de la relación señal-ruido y el error NMSE) y finalmente, evaluando la calidad obtenida en las imágenes reconstruidas.

Por ejemplo, los resultados de la DCT en lenguaje ensamblador, fueron comparados con los obtenidos en la simulación con aritmética entera. Después, el resultado del proceso DCT-IDCT fué evaluado comparándolo con la imagen fuente.

El esquema de cuantización fué incorporado al sistema, posteriormente a la depuración de los programas DCT-IDCT.

5.2.4 Implantación de operaciones aritméticas adicionales

En general los procesadores de señales digitales rápidos, presentan la desventaja de poder efectuar solamente las operaciónes aritméticas básicas.

Fue entonces necesario implantar las operaciones de tipo raiz cuadrada y logaritmo, dado que no se encontraban contenidas en el conjunto de instrucciones del microprocesador TMS320C25.

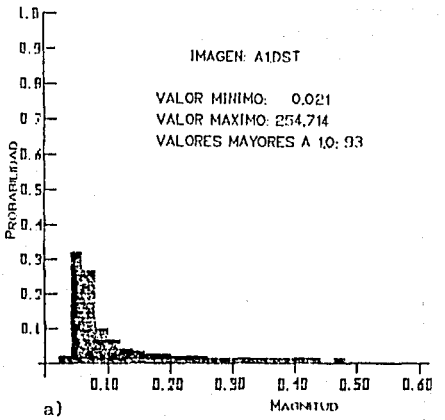
La raiz cuadrada fué implantada a partir del "principio de docotomia" [15] (el cual presenta importantes ventajas sobre el habitual método de Newton-Rapson), empleando 15 multiplicaciones y 60 adiciones.

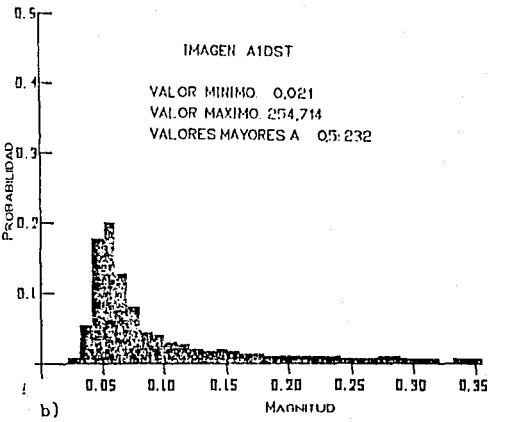
Con el algoritmo CORDIC (coordinate rotation digital computer) [18]-[19], es posible calcular funciones trascendentales del tipo: seno, coseno, tan, $\tan^{-1}$, senh, cosh, tanh, $\tanh^{-1}$, e^{x}, $\log_e$, etc. Básicamente, este algoritmo consiste en rotar un vector P, de coordenadas x,y, radio R y ángulo θ; en pasos angulares predeterminados. Mediante las condiciones iniciales de x, y; de la sumatoria de las variaciones angulares z y del sistema de coordenadas empleado (polar, lineal o hiperbólico), se determina el tipo de operación a realizar.

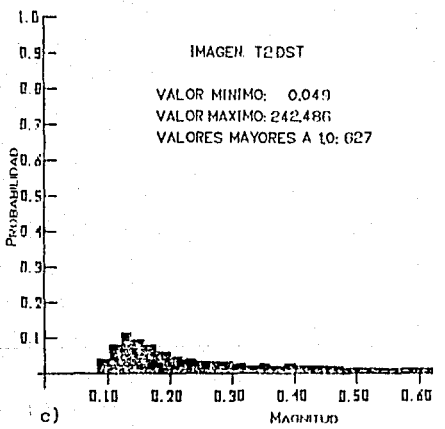
En el cálculo de la tabla de asignación de bits, es necesario calcular $\ln \sigma^2$, tal operación fué realizada mediante $\ln \sigma^2 = 2 \ln \sigma$.

Con la forma de Newton, a partir de "métodos de interpolación polinomial", es posible calcular logaritmos con 2n sumas y n multiplicaciones, pero para obtener una precisión aceptable, el orden n del polinomio debe ser ≥ 50. En este trabajo, para el cálculo de los logaritmos, con el algoritmo CORDIC, fué necesario emplear solamente 48 sumas, 32 multiplicaciones y una tabla de 32 elementos.

En la figura 5.2 se proporcionan dos histogramas tipicos de las imágenes procesadas, mostrando la matriz de desviaciones estandar (σ) de los coeficientes de la DCT-2D. Puede observarse que buena parte de los coeficientes presentan $\sigma < 1$. En consecuencia, una segunda forma más rápida aún de evaluar los logaritmos, fué por medio de una tabla de 128 elementos, conteniendo valores precalculados de $\ln \sigma$, en el rango $0 < \sigma \leq 0.5$.







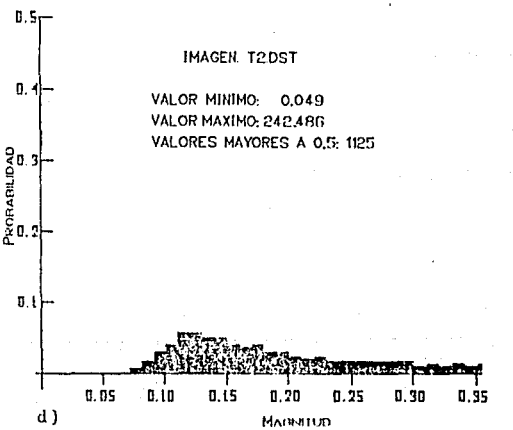


Fig. 5.2 Histogramas de las desviaciones estandar de dos imágenes tipicas.
a) Imagen A1, b) Histograma magnificado de la imagen A1.
c) Imagen T2, d) Histograma magnificado de la imagen T2.

6

RESULTADOS EXPERIMENTALES

El esquema de compresión descrito en el presente trabajo, fué aplicado sobre imágenes de tipo radiológico y microscópico, utilizadas en el diagnóstico médico, como són: angiografías (A1, A2 y CR), imágenes de resonancia magnética del craneo (I2 e I3), un tejido (T1) y una radiografía de las vértebras (V).

En este capítulo, inicialmente se presentan los resultados obtenidos al procesar las imágenes, empleando bloques de tamaño 16x16, 32x32 y 64x64. Enseguida, se muestran los resultados de la implantación final, realizada sobre el microprocesador de procesamiento de señales TMS320C25, utilizando bloques de tamaño 64x64.

Como se ha mencionado anteriormente, la calidad obtenida en las imágenes reconstruidas fué evaluada calculando el error NMSE y la relación señal-ruido.

6.1 Codificación realizada empleando bloques de tamaño 16x16, 32x32 y 64x64.

En el conjunto de figuras 6.1.1-6.1.5, se muestran los resultados del procesamiento de la imagen **T1**. Dentro de este conjunto, en la Fig. 6.1.1 se proporciona la imagen original; en la Fig. 6.1.2, el esquema de compresión fué aplicado segmentando la imagen en bloques de 16x16; en la Fig. 6.1.3 se emplearon bloques de 32x32 y finalmente de 64x64 en la Fig. 6.1.4. En los incisos a) correspondientes, la tasa de compresión fué de 8:1 y en los incisos b), esta fué de 10:1.

Las gráficas respectivas de error NMSE vs. tasa de compresión y de la relación señal-ruido vs. tasa de compresión, se proporcionan en la Fig. 6.1.5. Los resultados del procesamiento de la imagen V, son

mostrados en forma similar en el conjunto de figuras 6.2.1-6.2.5.

En la tabla 6.1, se proporciona un índice de referencia de las fotografías mostradas, para las imágenes T1, V y A1.

a)

Imagen T1

FIGURA	TASA DE COMPRESIÓN	TAMAÑO DE BLOQUE
6.1.1	1:1	
6.1.2.a	8:1	16x16
6.1.2.b	10:1	
6.1.3.a	8:1	32x32
6.1.3.b	10:1	
6.1.4.a	8:1	64x64
6.1.4.b	10:1	

FIGURA	
6.1.5.a	error NMSE vs. tasa de compresión
6.1.5.b	relación SNR vs. tasa de compresión

b)

Imagen V

FIGURA	TASA DE COMPRESIÓN	TAMAÑO DE BLOQUE
6.2.1	1:1	
6.2.2.a	10:1	16x16
6.2.2.b	14:1	
6.2.3.a	10:1	32x32
6.2.3.b	14:1	
6.2.4.a	10:1	64x64
6.2.4.b	14:1	

FIGURA	
6.2.5.a	error NMSE vs. tasa de compresión
6.2.5.b	relación SNR vs. tasa de compresión

c)

Imagen A1

FIGURA	TASA DE COMPRESIÓN	TAMAÑO DE BLOQUE
6.3.1.a	1:1	
6.3.1.b	14:1	16x16
6.3.1.c	14:1	32x32
6.3.1.d	14:1	64x64

FIGURA	
6.3.2.a	error NMSE vs. tasa de compresión
6.3.2.b	relación SNR vs. tasa de compresión

Tabla 6.1. Indice de referencia de las fotografías mostradas, para las imágenes: a) T1, b) V y c) A1.

El propósito de segmentar las imágenes en sub-bloques uniformes, de dimensiones pequeñas, es de disminuir la complejidad de los algoritmos de compresión. Sin embargo, el codificar cada sub-bloque en forma independiente, sin considerar que existe una correlación entre bloques adyacentes, ocasiona a menudo que en las imágenes reconstruidas sean perceptibles las fronteras entre los bloques.

Tal fenómeno es conocido como "efecto de bloques" y es visible en las Figs. 6.3.1.c y 6.3.1.d.

Una imagen decodificada, puede considerarse como el resultado de la superposición de una cierta cantidad de ruido sobre la imagen original. En general, la percepción de tal ruido depende de las caracteristicas de las imágenes y de la sensitividad del ojo humano.

Asi por ejemplo, el ruido que presénta una cierta energia dentro del ancho de banda de la imagen, o aquel en el cual las formas espectrales de ambas señales sean similares, resulta ser menos perceptible, es decir, queda *enmascarado* por la propia imagen.

Comunmente, la energia del ruido es alta en las regiones próximas a los bordes, provocando una degradación a lo largo de estos. A tal degradación se le conoce como "efecto escalera". En las imágenes de las Figs. 6.1.2.a y 6.1.2.b, es visible este tipo de degradación.

Una tercera variedad de distorsión, acontece cuando se trabaja con imágenes en donde existen grandes areas de tonalidad casi-uniforme, es decir, los cambios de intensidad ocurren gradualmente. Este tipo de distorsión se conoce como "efecto rejilla" y consiste en que la continuidad de los niveles de gris se preserva dentro de cada bloque durante la codificación, pero tal caracteristica no es precervada entre los bloques vecinos.

Asi por ejemplo, por haberse empleado una tasa de compresión demasiado alta, en las imágenes procesadas con bloques de tamaño 16x16, se observa una tendencia entre los bloques adyacentes a cambiar bruscamente sus tonalidades de gris. Este efecto se observa más marcadamente en la Fig. 6.3.1.b

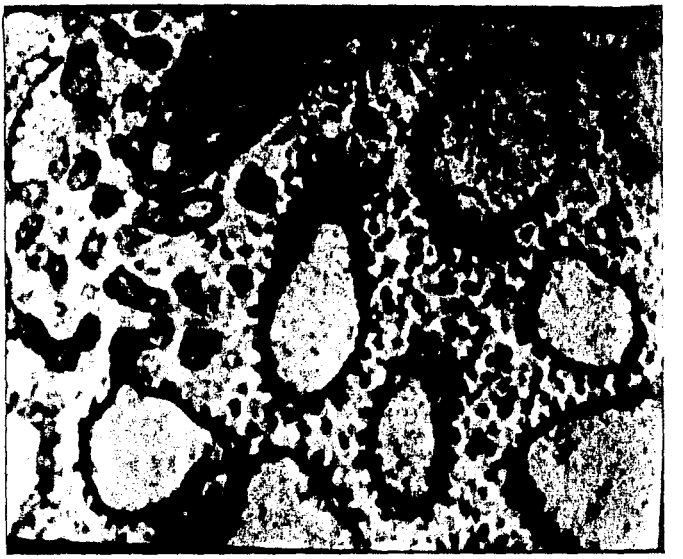


Fig. 6.1.1 Imagen T1 original

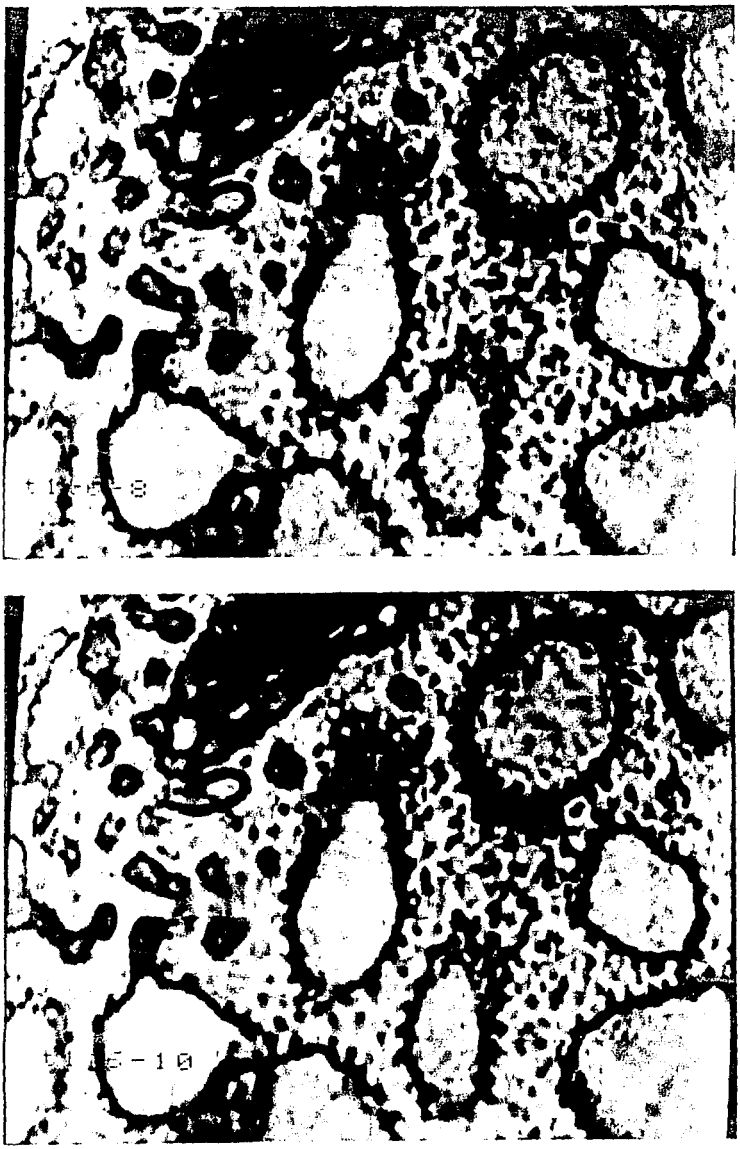


Fig. 6.1.2 Imagen T1 procesada con sub-bloques de tamaño 16x16

a) Tasa de compresión 8:1
b) Tasa de compresion 10:1

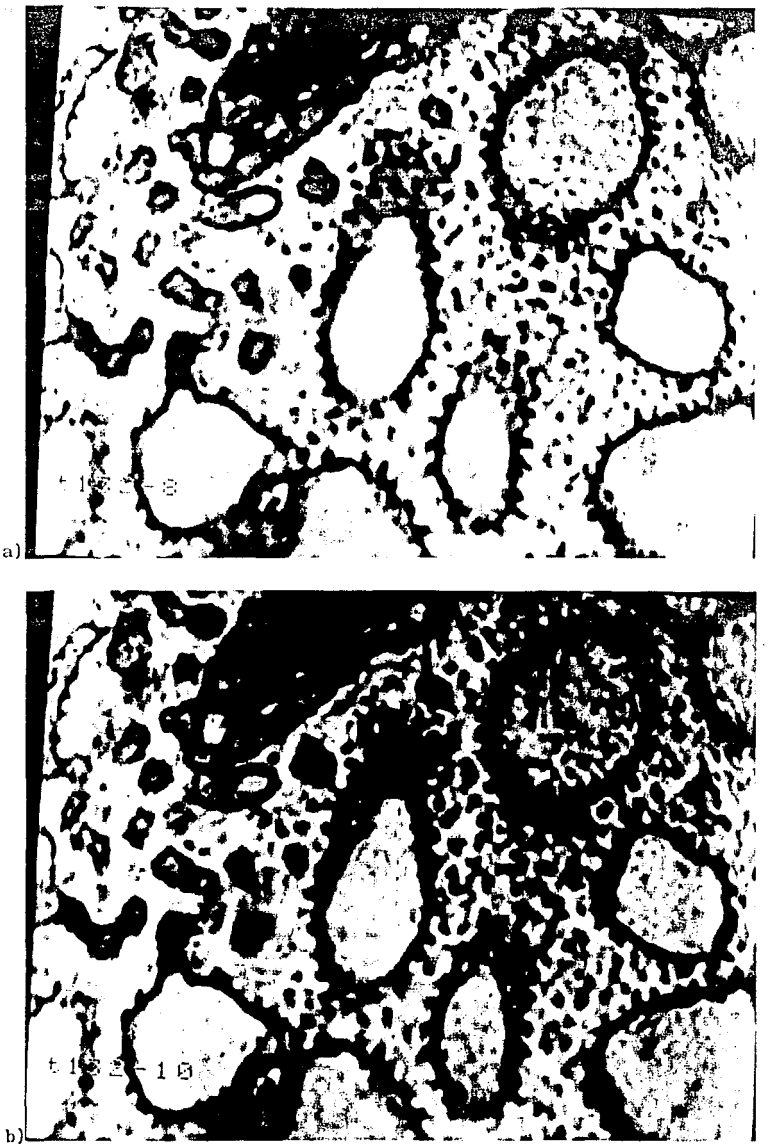


Fig. 6.1.3 Imagen T1 procesada con sub-bloques de tamaño 32x32

a) Tasa de compresión 8:1
b) Tasa de compresión 10:1

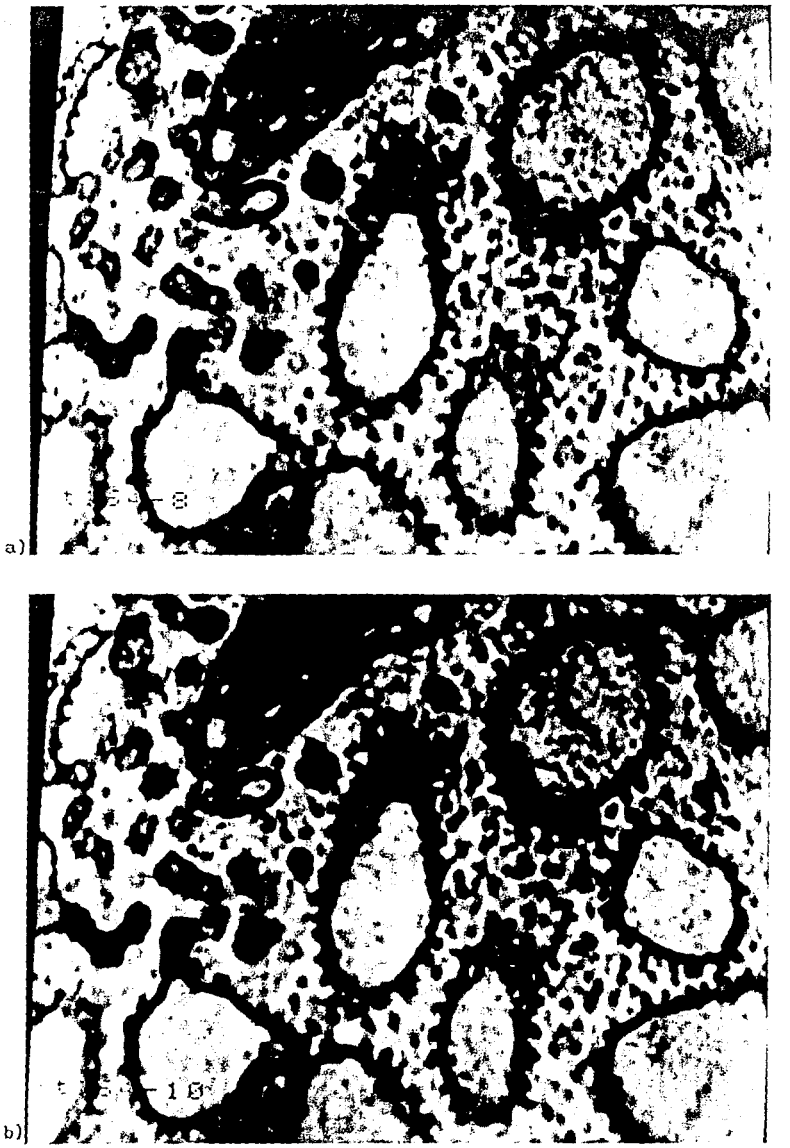
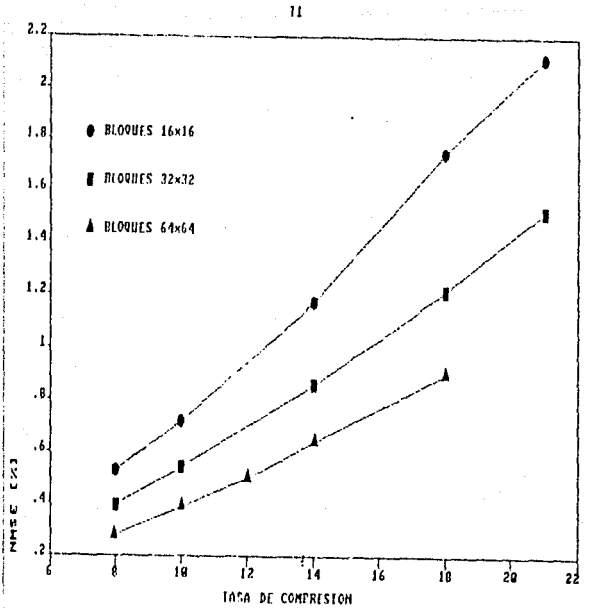


Fig. 6.1.4 Imagen T1 procesada con sub-bloques de tamaño 64x64

a) Tasa de compresión 8:1
b) Tasa de compresión 10:1

a)



b)

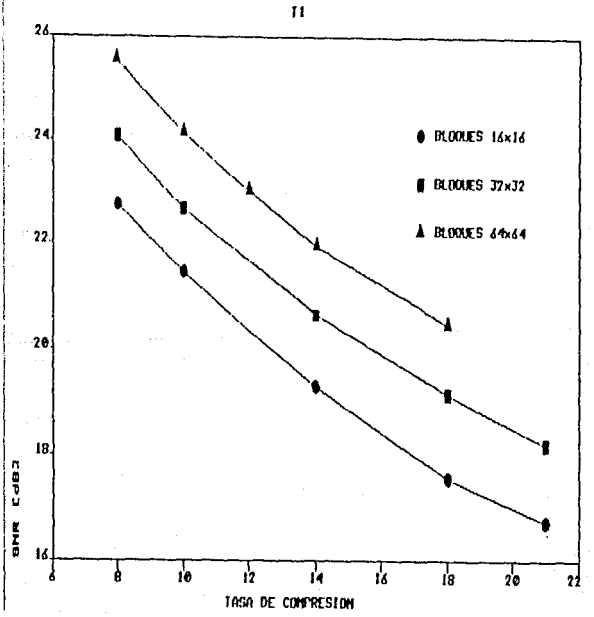


Fig. 6.1.5 Evaluación de la calidad obtenida en la reconstrucción de la imagen T1, empleando tres diferentes tamaños de sub-bloques.
a) Error NMSE [%] vs. tasa de compresión
b) Relación SNR [dB] vs. tasa de compresión

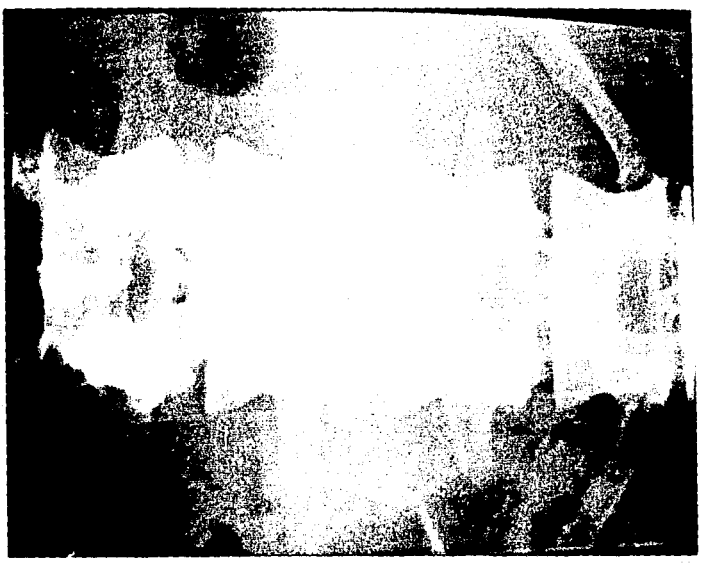
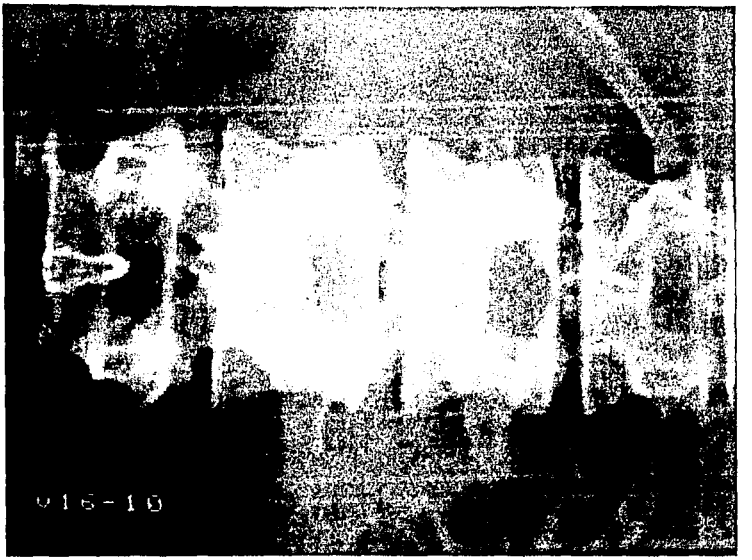


Fig. 6.2.1 Imagen V original

a)



b)

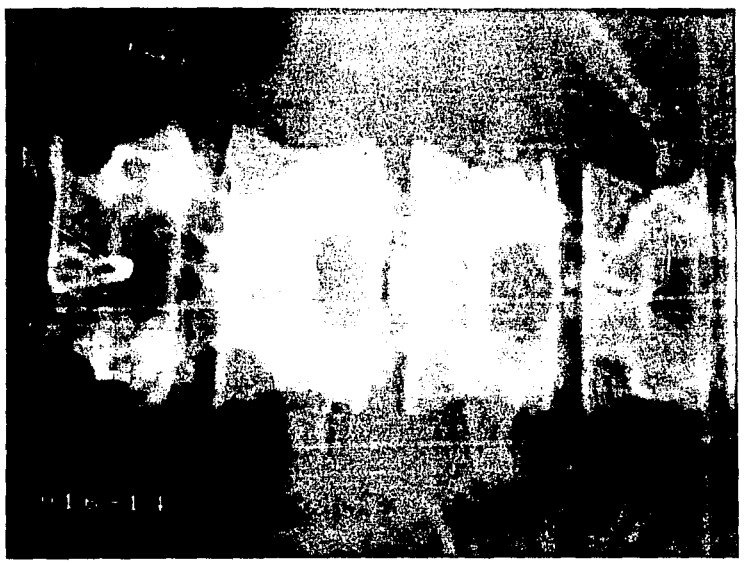
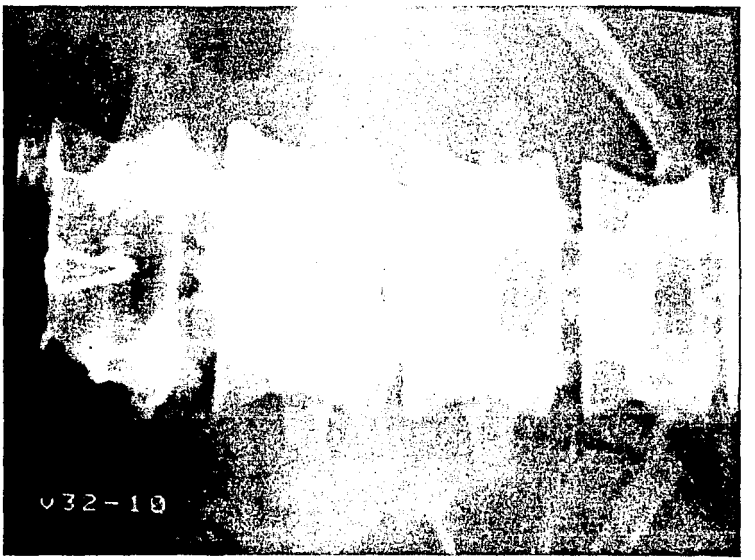
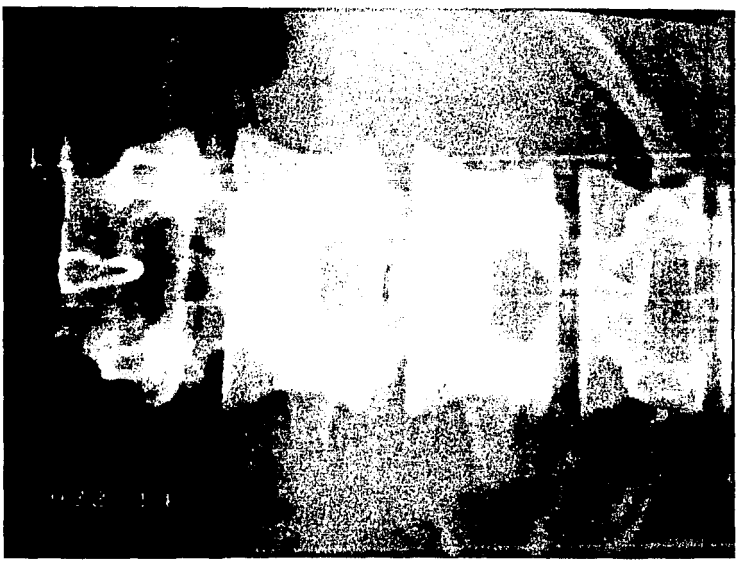


Fig. 6.2.2 Imagen V procesada con sub-bloques de tamaño 16x16

a) Tasa de compresión 10:1
b) Tasa de compresión 14:1



a)

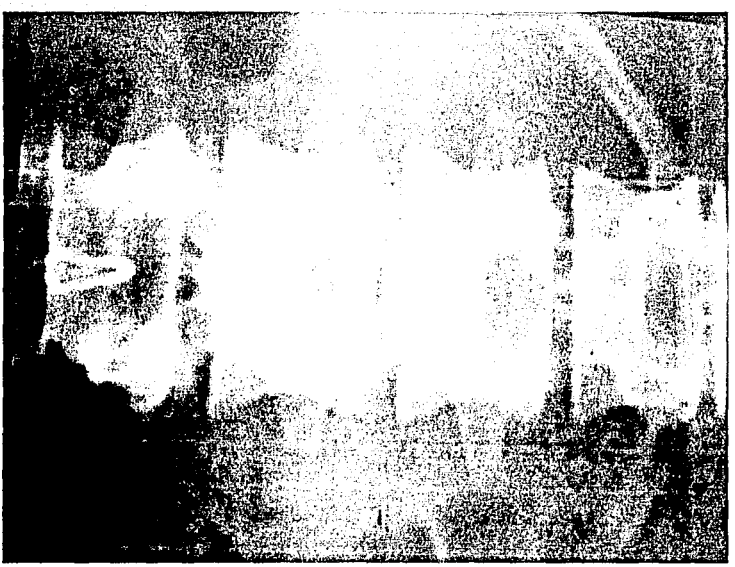


b)

Fig. 6.2.3 Imagen V procesada con sub-bloques de tamaño 32×32

a) Tasa de compresión 10:1
b) Tasa de compresión 14:1

a)



b)

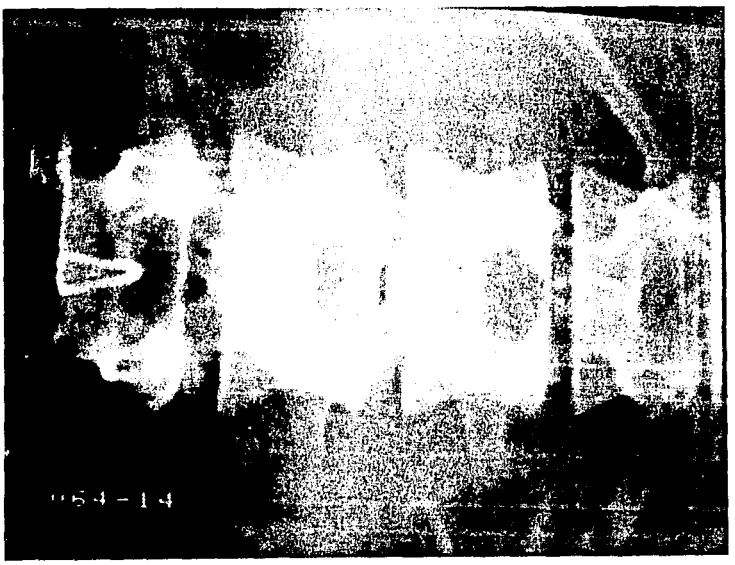
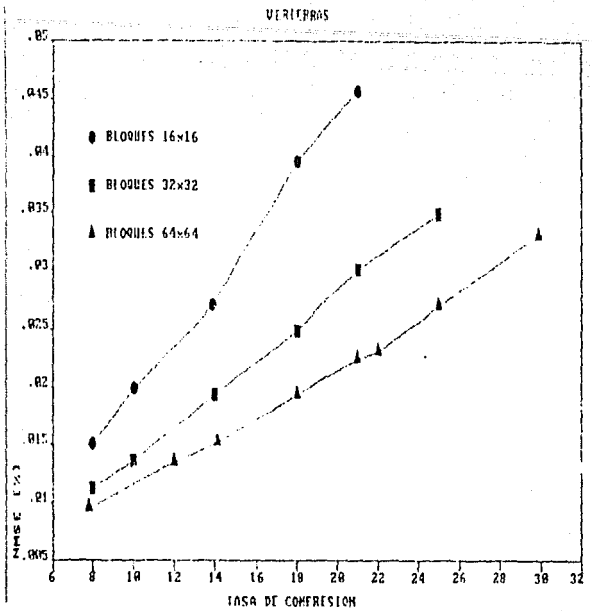
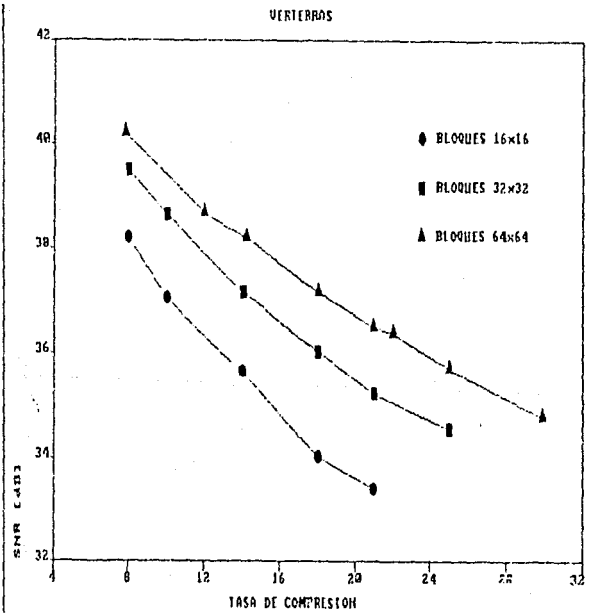


Fig. 6.2.4 Imagen V procesada con sub-bloques de tamaño 64x64

a) Tasa de compresión 10:1
b) Tasa de compresión 14:1



a)

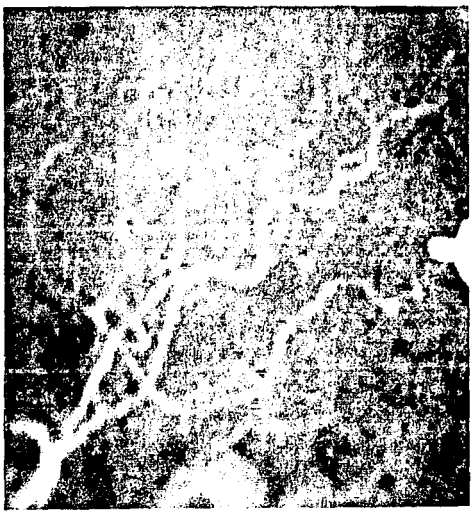


b)

Fig. 6.2.5 Evaluación de la calidad obtenida en la reconstrucción de la imagen V, empleando tres diferentes tamaños de sub-bloques.
a) Error NMSE [%] vs. tasa de compresión
b) Relación SNR [dB] vs. tasa de compresión

a)

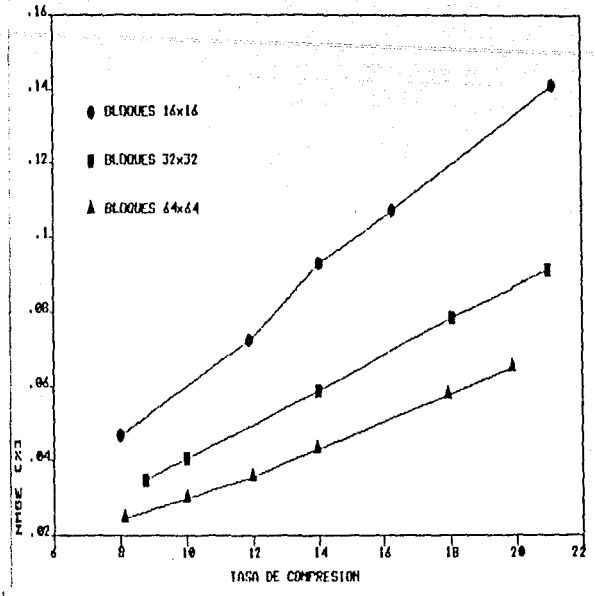
b)



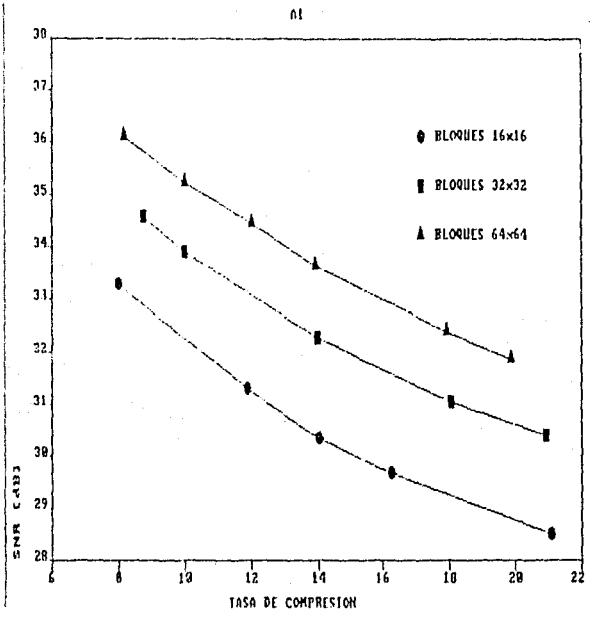
d)

c)

Fig. 6.3.1 Calidad obtenida en la imagen A1, a una tasa de compresión de 14:1, empleando tres diferentes tamaños de sub-bloques.
a) Imagen original b) Sub-bloques de 16x16
c) Sub-bloques de 32x32 d) Sub-bloques de 64x64



a)



b)

Fig. 6.3.2 Evaluación de la calidad obtenida en la reconstrucción de la imagen A1, empleando tres diferentes tamaños de sub-bloques.
a) Error NMSE [%] vs. tasa de compresión
b) Relación SNR [dB] vs. tasa de compresión

6.2 Implantación final

En esta étapa, la implantación de los algoritmos fué realizada segmentando las imágenes fuente en bloques de tamaño 64x64.

En la tabla 6.2, se proporciona la evaluación de los métodos de computación indirecta y recursiva, descritos en el capitulo 3, para el cálculo de la transformada discreta coseno. Tal evaluación fué realizada en función del número de sumas y de multipliaciónes necesarias para su cálculo.

	Método MCI		Método MCR	
	DCT	IDCT	DCT	IDCT
Número de sumas	1484	964	513	513
Número de múltiplicaciones	226	324	192	192

Tabla 6.2 Número de operaciones para el cálculo de las transformaciones DCT e IDCT unidimensionales de orden 64, empleando los métodos de computación indirecta y recursiva (MCI Y MCR).

	Método MCI		Método MCR	
	DCT	IDCT	DCT	IDCT
Longitud de las operaciones (bits)	32	16	32	32
Memoria de programa (palabras)	595	640	608	608
Memoria de datos (palabras)	272	224	4287	4287
Tiempo de ejecución por imagen (seg)	3.69	3.8	9.17	9.17
SNR* (db)	51	51	51	51

TABLA 6.3 Desempeño de los métodos MCI y MCR, en función de los recursos necesarios para su implantación en lenguaje ensablador, del tiempo de ejecución y de la relación señal-ruido.

* Relación SNR calculada con la expresión 2.5

Información complementaria de los algoritmos se presenta en la tabla 6.3. Puede observarse que aún y cuando el número de operaciones aritméticas es muy semejante, existe una notable diferencia en el tiempo de procesamiento.

Los tiempos medidos, en promedio, para realizar los procesos de compresión-expansión, fueron los siguientes:

COMPRESIÓN

	TIEMPO (SEG.)
DCT-2D	3.69
CODIFICACIÓN	0.6105
TOTAL	4.3005

Expansión

	TIEMPO (SEG.)
IDCT-2D	3.8082
DECODIFICACIÓN	0.2127
TOTAL	4.0209

Tabla 6.4 Tiempos totales de cálculo, para efectuar la codificación.

IMAGEN	FIGURA	MAXIMA TASA DE COMPRESIÓN	ERROR NMSE [%]	RELACION* SNR [dB]
I2	6.4.1	1:1		
	6.4.2	8:1	1.1225	19.497
I3	6.5.1	1:1		
	6.5.2	4:1	0.4158	23.81
T1	6.6.1	1:1		
	6.6.2	8:1	0.2786	25.55
CR	6.7.1	1:1		
	6.7.2	10:1	0.014	38.31
A1	6.8.1	1:1		
	6.8.2	12:1	0.0356	34.477
A2	6.9.1	1:1		
	6.9.2	8:1	0.0349	34.57
V	6.10.1	1:1		
	6.10.2	22:1	0.023	36.37

Tabla 6.5 Máximas tasas de compresión alcanzadas. Se emplean criterios objetivos de evaluación.

* Relación SNR calculada con la expresión 2.5.

La tabla 6.5 muestra un resumen de los resultados obtenidos.

Se indican las tasas máximas de compresión alcanzadas, que permiten conservar una buena calidad durante la reconstrucción.

A partir de la fig. 6.4.1, se muestran los resultados indicados en la tabla 6.5. Para una mejor referencia, se incluye la fotografía de la imagen original y la de la imagen reconstruida.

Con respecto a estos resultados es conveniente señalar, que la tasa máxima de compresión alcanzada depende directamente de la estructura de las imágenes. Por un lado, aquellas con un alto contraste o con altas frecuencias espaciales, no pueden ser comprimidas eficientemente. Tal es el caso de las imagenes de resonancia magnética I2 e I3, para las cuales es aconsejable incluir un preprocesamiento adicional [8].

Por otro lado, aquellas imagenes de caracteristicas contrarias a las mencionadas pueden alcanzar mayores tasas de compresión, como por ejemplo, la imagen V.

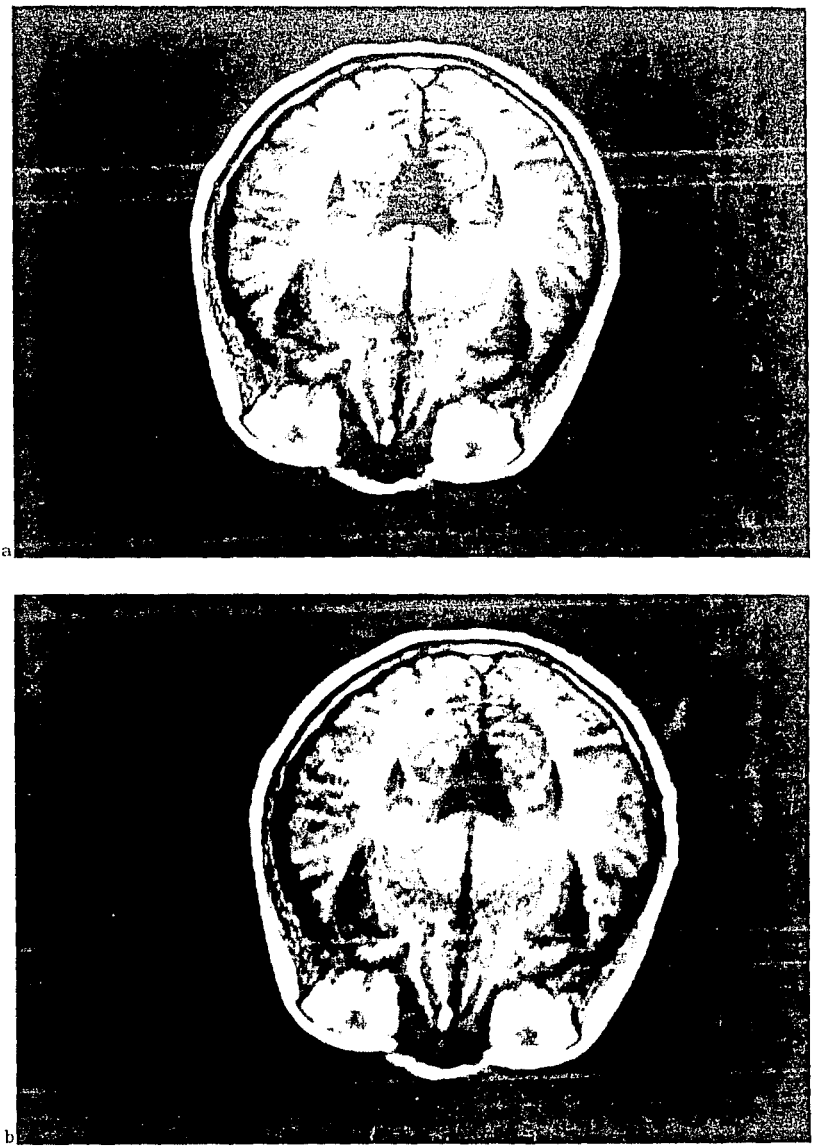


Fig. 6.4.1 Imagen 12

a) Imagen original
b) Imagen procesada.
Tasa de compresion 8:1

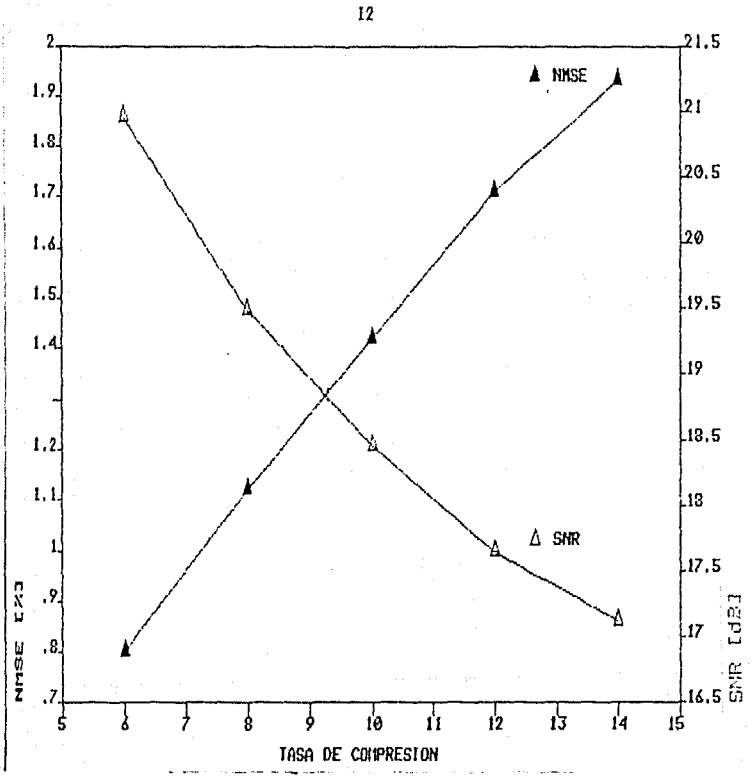
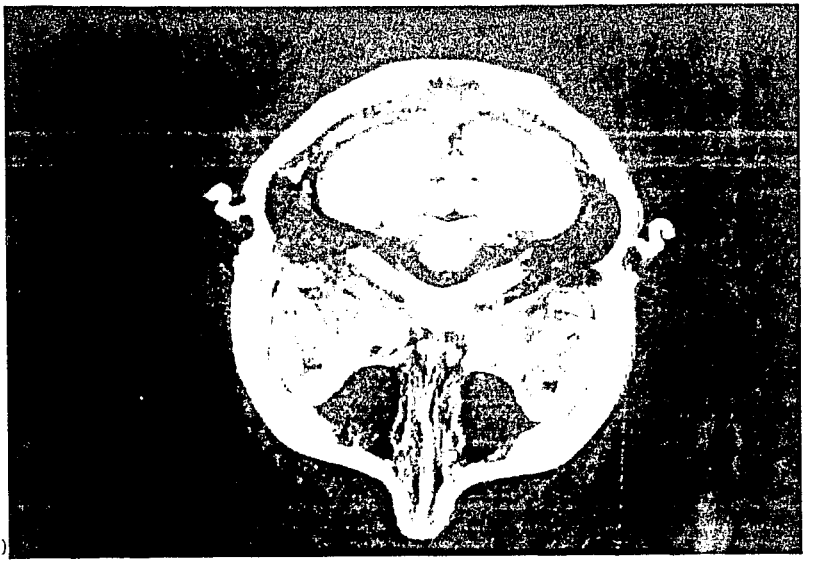


Fig. 6.4.2 Evaluación de la calidad obtenida, en la compresión de la imagen I2.

a)



b)

Fig. 6.5.1 Imagen 13

a) Imagen original
b) Imagen procesada.
Tasa de compresión 4:1

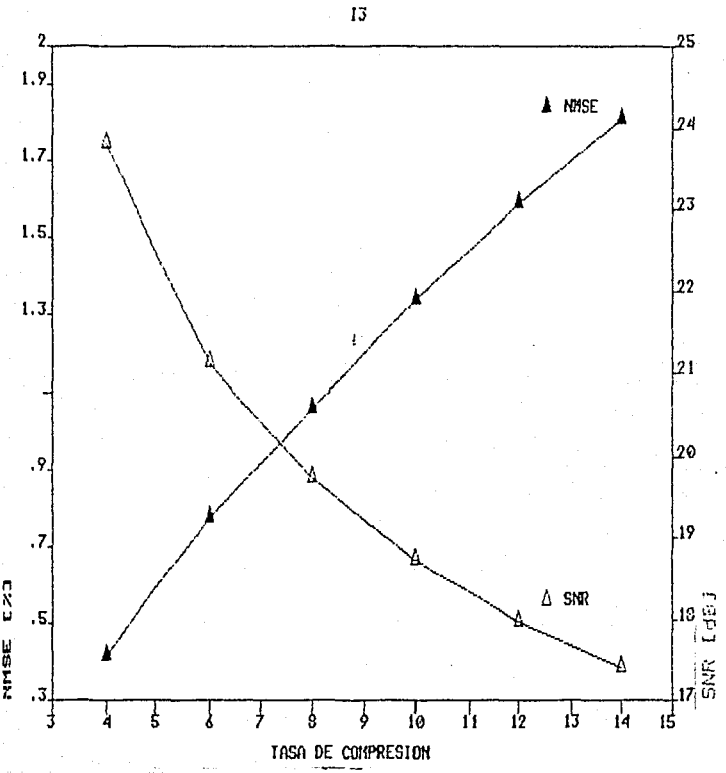


Fig. 6.5.2 Evaluación de la calidad obtenida, en la compresión de la imagen I3.

a)

b)

Fig. 6.6.1 Imagen T1

a) Imagen original
b) Imagen procesada.
Tasa de compresión 8:1

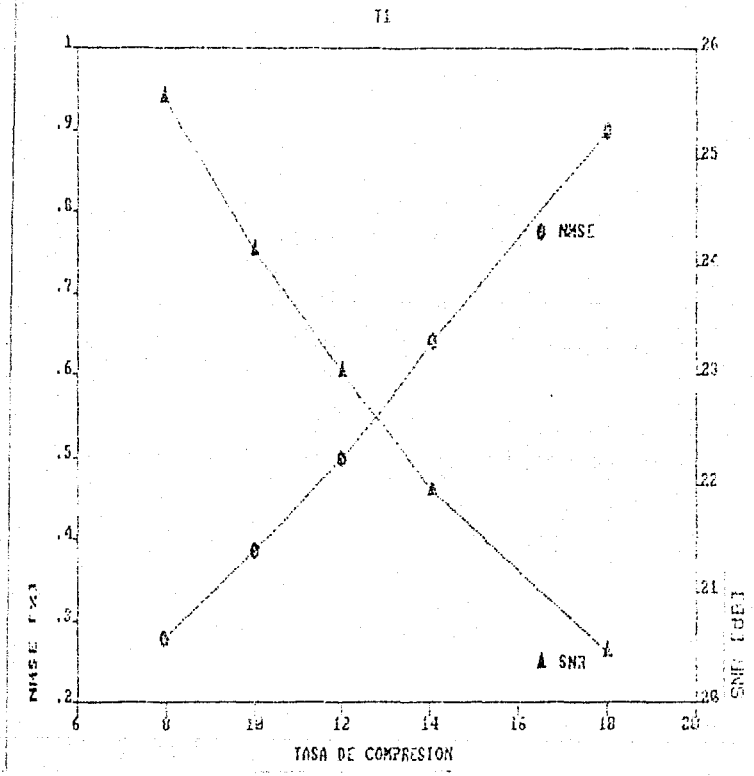
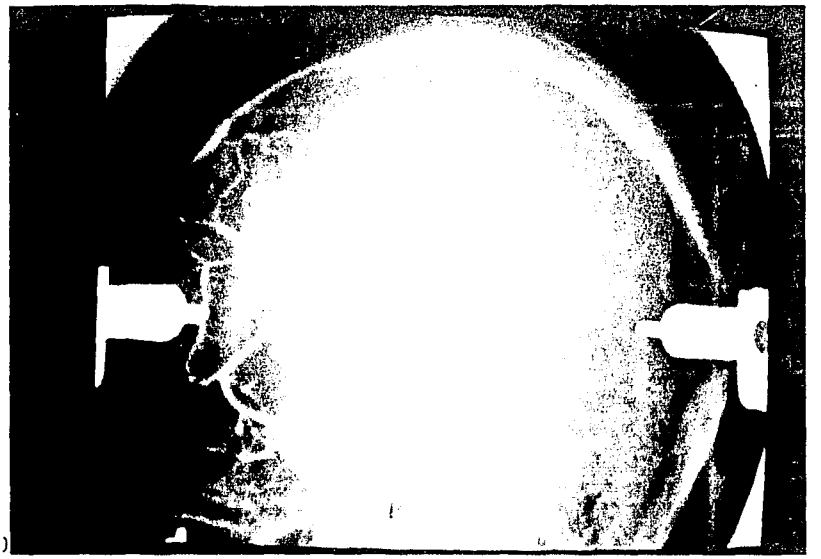


Fig. 6.6.2 Evaluación de la calidad obtenida, en la compresión de la imagen T1.

a)



b)

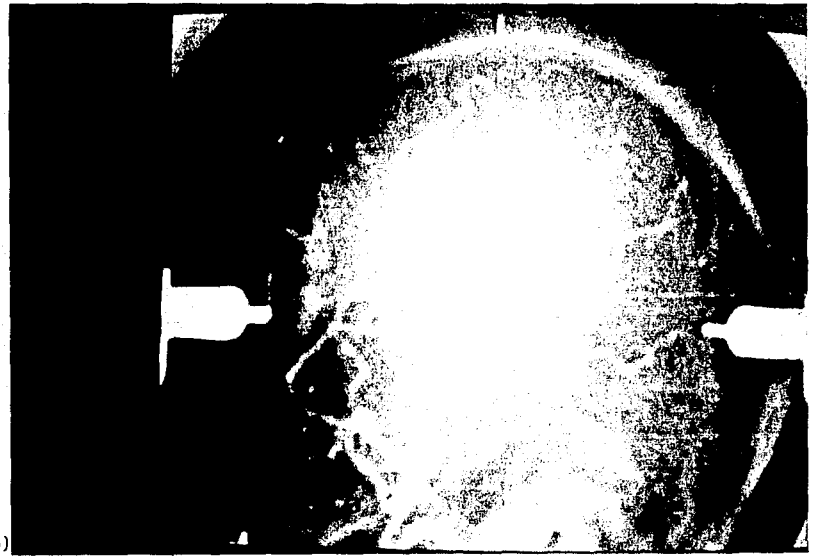


Fig. 6.7.1 Imagen CR

a) Imagen original
b) Imagen procesada.
Tasa de compresión 10:1

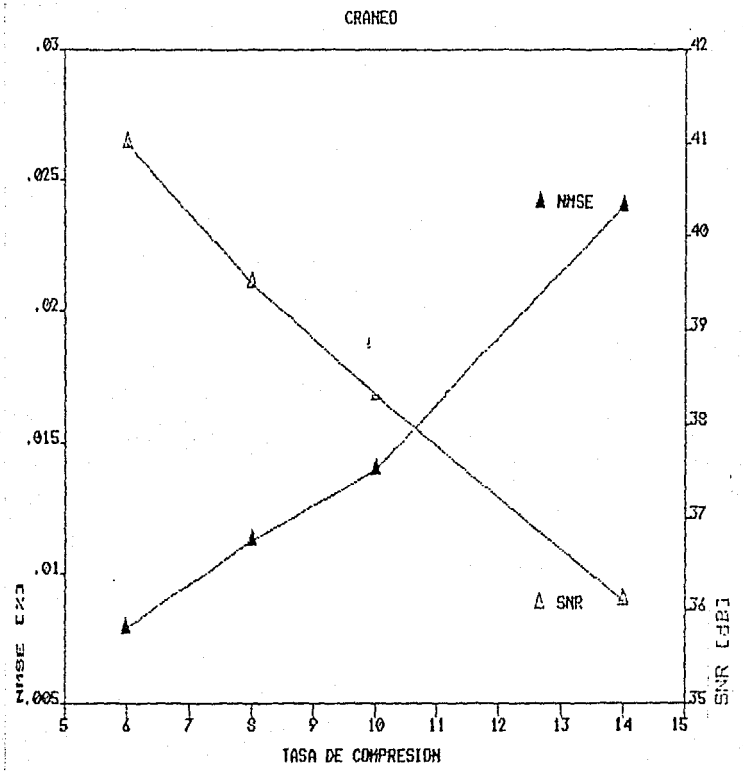
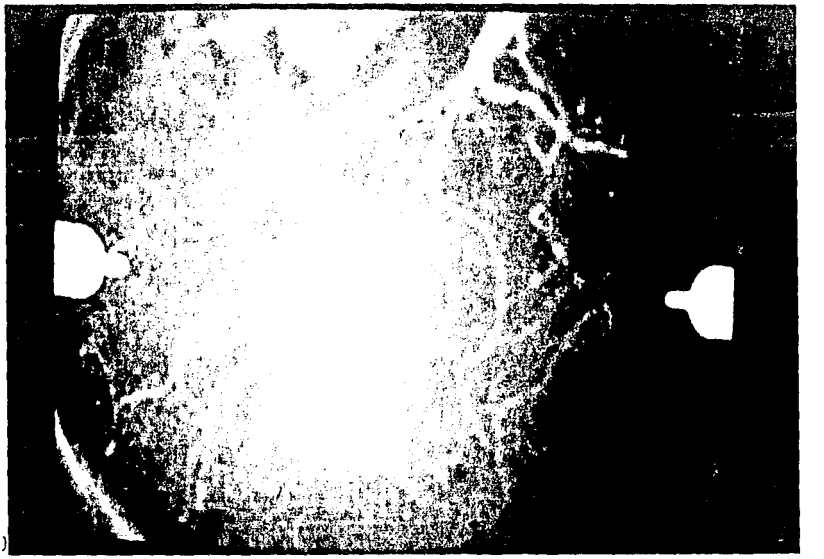


Fig. 6.7.2 Evaluación de la calidad obtenida, en la compresión de la imagen CR.

a)



b)

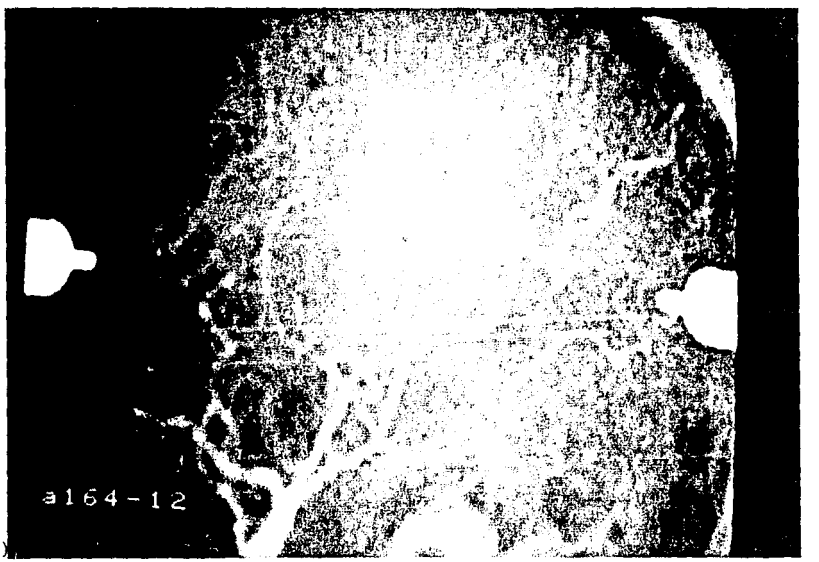


Fig. 6.8.1 Imagen A1

a) Imagen original
b) Imagen procesada.
Tasa de compresión 12:1

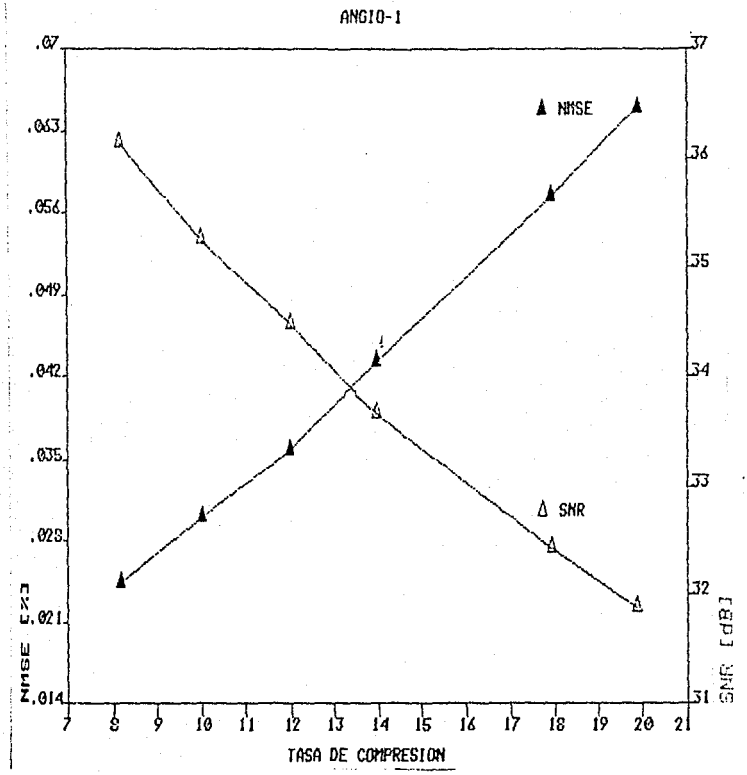


Fig. 6.8.2 Evaluación de la calidad obtenida, en la compresión de la imagen A1.

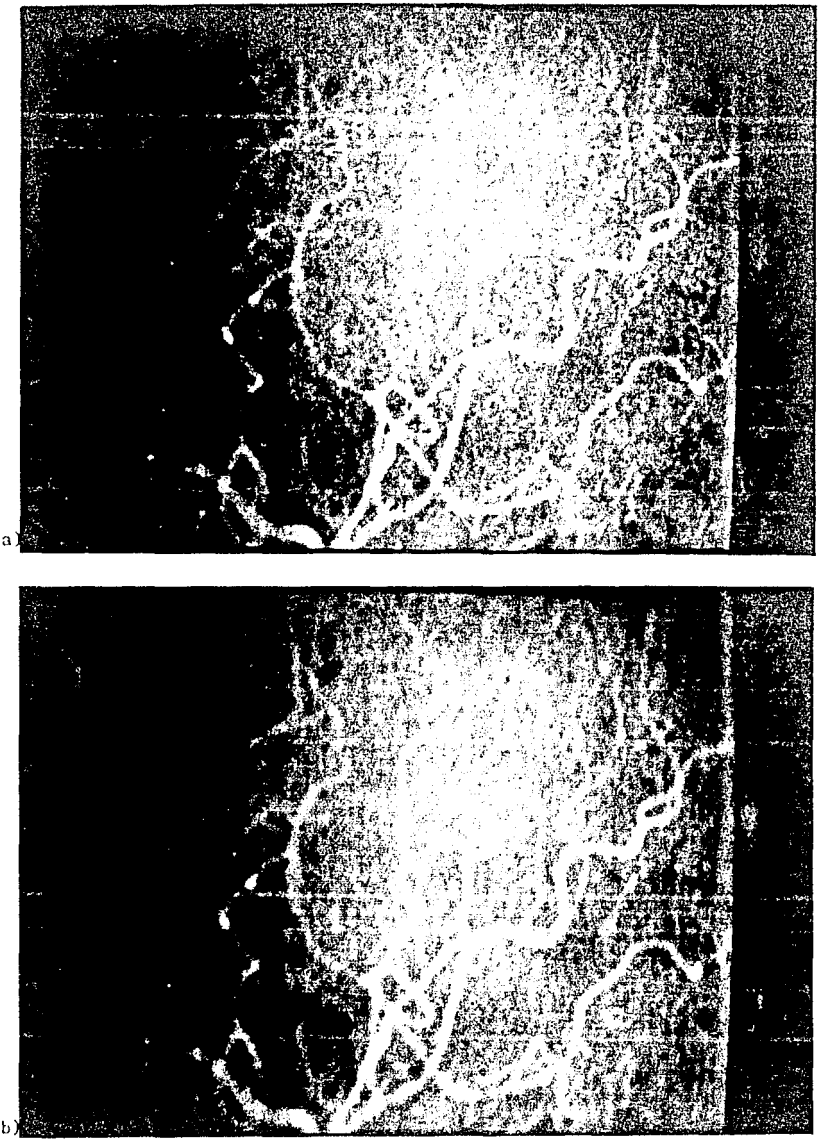


Fig. 6.9.1 Imagen A2

a) Imagen original
b) Imagen procesada.
Tasa de compresion 8:1

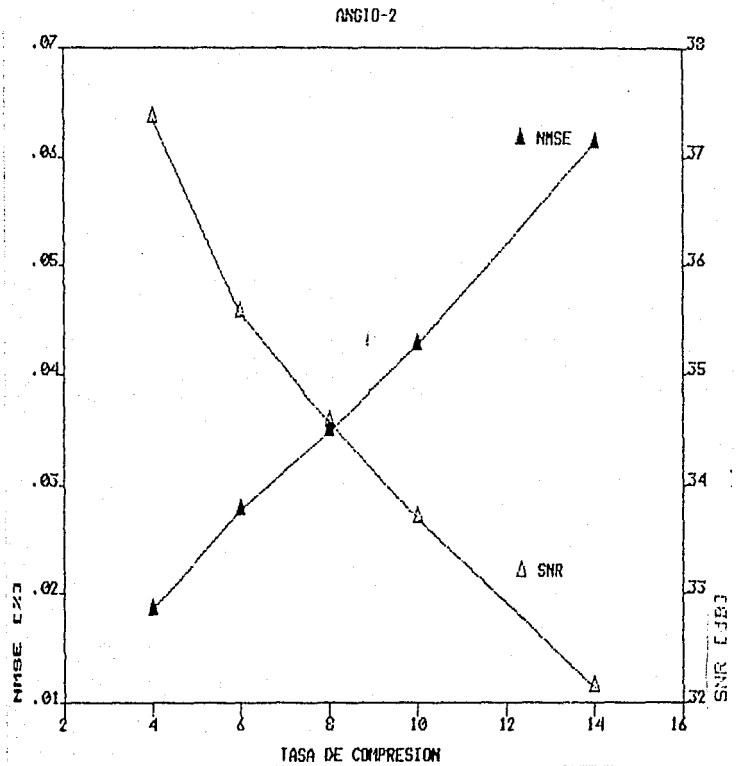
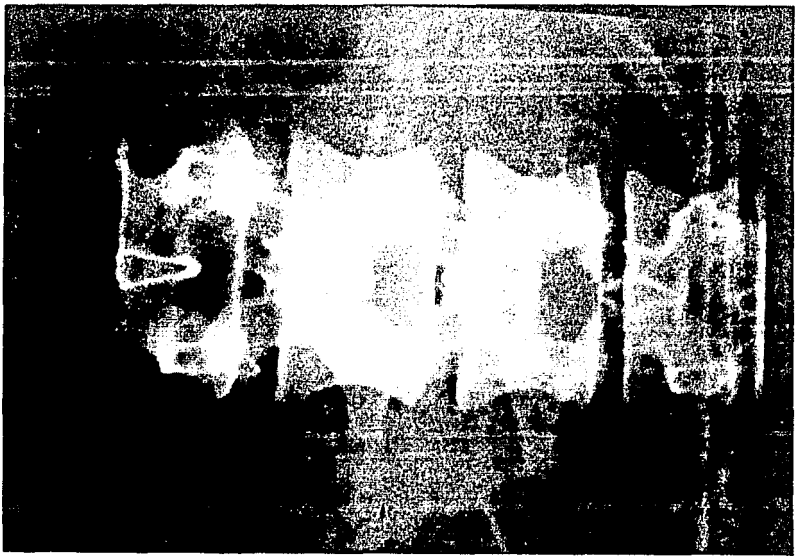


Fig. 6.9.2 Evaluación de la calidad obtenida, en la compresión de la imagen A2.

a)



b)

Fig. 6.10.1 Imagen V

a) Imagen original
b) Imagen procesada.
Tasa de compresion 22:1

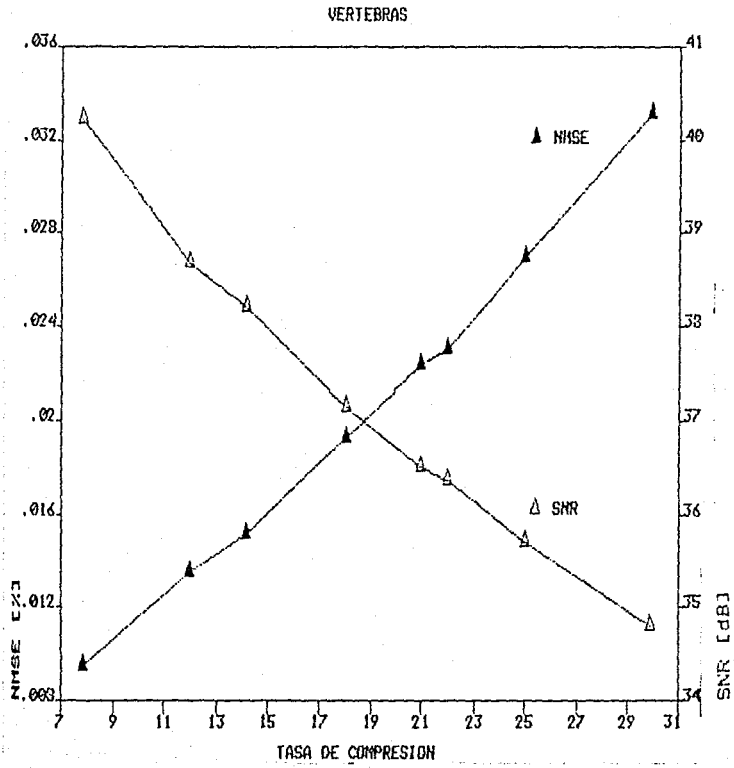


Fig. 6.10.2 Evaluación de la calidad obtenida, en la compresión de la imagen V.

7

CONCLUSIONES

Dada la gran cantidad de operaciones y velocidad demandadas para efectuar la compresión de imágenes, las tendencias actuales de investigación estan orientadas a abatir en la mejor medida los tiempos de cómputo necesarios. En este sentido, las dos posibles alternativas de solución son :

- optimizar tanto como sea posible, los diferentes tipos de algoritmos de codificación
- utilización de un hardware que permita su procesamiento en tiempo real.

Inicialmente en este trabajo, se revisaron diferentes algoritmos rápidos que calculan la transformada discreta coseno. La selección del algoritmo óptimo, en el sentido de minimizar su tiempo de ejecución, manteniendo una buena precisión; estuvo ligada a su implantación en el microprocesador TMS320C25. Es decir, considerando la arquitectura y los recursos del mismo.

De la misma manera, se demostró experimentalmente que por medio de la codificación por transformada discreta coseno, dependiendo de las características propias de cada imagen, es posible alcanzar altas tasas de compresión. Así, una tasa de hasta 22 : 1 ,en la imagen V, con una relación señal-ruido de 36.37 dB, se pudo obtener. El tiempo total promedio de procesamiento fué de 8.32 seg., del cual 7.49 seg. corresponden al cálculo de la DCT-IDCT.

Dadas las características propias del esquema de codificación empleado, el alcanzar una mayor tasa de compresión ameritaría la utilización de otro tipo de codificación, como por ejemplo uno adaptivo con respecto al tipo de imagen, o uno híbrido (DPCM-Transformada).

En lo que concierne al tiempo total de ejecución, los algoritmos

implantados fueron cuidadosamente programados y depurados, aprovechando al máximo los recursos del sistema empleado. Los programas de traducción automática de codigo, de lenguaje de alto nivel a lenguaje ensamblador, como por ejemplo el paquete SIMPAC [16], no proporcionan aún traducciones óptimas.

No obstante los resultados alcanzados en este trabajo [22], los tiempos de ejecución obtenidos no pueden ser considerados como los correspondientes a un sistema que opera en tiempo real.

Para disminuir el tiempo de ejecución del algoritmo de codificación, dada su complejidad, se hace necesario un procesamiento en paralelo con microprocesadores tipo DSP. Por ejemplo, se puede dedicar un microprocesador a cada sub-bloque de la imagen a procesar.

Dado que los microprocesadores tipo DSP no presentan grandes facilidades para trabajar concurrentemente, otra opción es el uso de arreglos de "transputers", o circuitos integrados de aplicación especifica (ASIC).

Dado que la mayor parte del tiempo de ejecución del algoritmo de codificación (90%), es consumido en el cálculo de la transformada discreta coseno, han surgido dispositivos VLSI que la calculan para un tamaño de 8x8, a una mayor velocidad que la obtenida con un microprocesador tipo DSP.

Es de esperarse que debido al continuo avance tecnológico surjan, en un futuro próximo, dispositivos que calculen transformadas de mayor tamaño en menos tiempo, permitiendo asi una implantación más simple y en tiempo real.

BIBLIOGRAFIA

[1] Alcántara S. Rogelio,"Implantation d'algorithmes rapides sur des processeurs de traitement du signal", tesis de doctorado, ENST Paris, Francia, septiembre 1986.

[2] Byeong G. Lee, " A new algorithm to compute the discrete cosine transform ", IEEE ASSP-32, No 6 decembre 1986.

[3] Hsieh S. Hou, " A fast recursive algorithm for computing the discrete cosine transform ", ASSP -35, No 10 , october 1987.

[4] Jain Anil K., " Fundamentals of digital image processing ", Prentice Hall Inc., 1989.

[5] Kun-Shan Lin , Gene A. Frantz and Ray Simar Jr., " The TMS 320 family of digital signal processors ", Proc. of the IEEE , vol. 75, No 9, pp 1143-1159, septembre 1987.

[6] Lee A. Edward, " Programable DSP architectures : Part I ", IEEE ASSP magazine, pp 4-19, octobre 1988.

[7] Lee A. Edward, " Programable DSP architectures : Part II ", IEEE ASSP magazine, pp 4-14, january 1989.

[8] Lo S. C.,Huang H. K., " Radiological image compression : Full Frame Bit Allocation Technique ", Radiology, vol. 155, No 3, pp 811-817, june 1985.

[9] M. J. Narashima and Allen M. Peterson, " On the computation of the discrete cosine transform ", Trans. on Com., vol. Com-26, No 6, june 1987.

[10] Manual INMOS, Digital signal processing databook, first edition july 1989.

[11] Murat Kunt, A. Ikonomopoulos and M. Kocher, " Second-Generation Image-Coding techniques ",Proc. of the IEEE, vol 75, No 4 april 1985.

[12] N. Ahmed, T. Natarayan, and K. R. Rao, " Discrete Cosine Transform " IEEE Trans. on Comp., vol C-23, pp 90-93, January 1974.

[13] Netravali A. N., Limb J. O., " Picture Coding. A Review ", Proc. of the IEEE , vol. 68, No 3, pp 366-406, march 1980.

[14] P. Duhamel, " Implementation of Split-Radix FFT algorithms for complex, real and real-symetric data ", ASSP-34, No 2, april 1986.

[15] Prado J., Alcántara R., " A fast square-rooting algorithm using a digital signal processor ", Proc. of the IEEE, vol. 75, No 2, pp 262-264, february 1987.

[16] Sánchez V. J, "Sistema de Implantación en Procesadores digitales de señlaes Asistido Por Computadora", tesis Maestria, DEPFI-UNAM, México D.F., agosto 1989.

[17] Sorensen H. V., M. T. Heideman and C. S. Burrus, " On computing the Split-Radix FFT ", ASSP-34, No 1, february 1986.

[18] Volder J. E., " The Cordic trigonometric computing technique ", IRE Trans.on Electronic Computers, vol. EC-8, No 3, pp 330-334, september 1959.

[19] Walther J. S., " A unified algorithm for elementary functions ", Spring Voint Computer Conference, pp 379-385, 1971.

[20] Wintz P. A. " Transform picture coding ", Proc. of the IEEE, vol. 60, No 7, pp 809-820, july 1972.

[21] Wintz P. A., Kurtenbach A. J., " Waveform error control in PCM telemetry ", IEEE Trans. in Inf. Theory, vol. IT-14, pp 650-651, 1968.

[22] F. J. Garcia Ugalde, V. García Garduño et M. Moctezuma Flores, "Implantation d'un schéma de compression d'images radiologiques sur le microprocesseur des signaux TMS320C25 (TI)", EUROCODE 90, Udine, Italy, November 5-9, 1990.