UNIVERSIDAD POLITECNICA DE VALENCIA

SVD para la transmisión progresiva de imágenes y la codificación de vídeo digital

TESIS DOCTORAL

Presentada por: D. José A. Verdoy González

Dirigida por: Dr. D. Rafael J. Villanueva Micó

Dr. D. Javier Villanueva Oller

VALENCIA 2009

Capítulo 1

Introducción

En los tiempos en que vivimos, la tecnología digital nos rodea. Dicha revolución digital, que se extiende a casi todas las ramas de la ciencia, ha llegado también a la Sanidad. Cada vez más se utilizan, en las consultas médicas y hospitales, técnicas de imagen digital como TAC[1], PET[2], RM[3], RX digital, mamografías digitales, etc.

Como ejemplo podemos comentar brevemente la tan utilizada Resonancia Magnética. Ésta es, junto con la Ecografía, una de las pruebas más valoradas últimamente por los médicos, todo ello, gracias a su excelente resolución, sobre todo en tejidos blandos del aparato locomotor y sistema nervioso; además se pueden obtener imágenes de cortes del organismo en cualquier plano y no emplea radiaciones ionizantes.

Todo esto origina una gran cantidad de imágenes médicas así como un aumento, por motivos de resolución y nitidez, del tamaño de éstas. Si a todo esto añadimos el tránsito de estas imágenes médicas no solo por la red interna de un hospital sino incluso por redes externas, dentro de lo que se conoce como telemedicina o medicina a distancia, nos vemos obligados a hacer uso de nuevas técnicas en el proceso, organización, almacenamiento y transmisión de las imágenes médicas digitales.

1.1 La medicina que nos espera

Los avances en las técnicas de diagnóstico por imagen han revolucionado la medicina. En apenas 10 años, la mirada de la medicina digital ha sido capaz de penetrar en los lugares más recónditos del cuerpo humano. Por ejemplo, la penúltima revolución en esta medicina digital es la combinación de las cámaras de PET y TAC en un solo aparato. Mientras que la PET detecta lesiones funcionales y metabólicas que a menudo, en enfermedades como el cáncer, son

[1] Tomografía Axial Computerizada.
[2] Tomografía por Emisión de Positrones.
[3] Resonancia Magnética.

las primeras en aparecer y las más dificiles de ver, el TAC descubre cambios anatómicos y estructurales. Fusionando las imágenes aportadas por ambas cámaras se obtiene un mejor rendimiento en el diagnóstico de la enfermedad a tratar.

Por ejemplo, el hospital La Paz de Madrid incorporó a finales de 2005 al servicio de radiodiagnóstico el escáner más avanzado de la sanidad pública española. Su avanzada tecnología permite obtener hasta 160 imágenes por segundo, que son procesadas inmediatamente para reconstruir en el ordenador una imagen volumétrica en tres dimensiones. Y surge la pregunta ¿cómo guardar y organizar tanta información?

Fotografías de la piel de un paciente que toma y envía el médico de familia para que las evalúe el dermatólogo, el control de apnea del sueño en el propio domicilio del paciente o videoconferencias entre el médico de atención primaria y el especialista, para intercambiar impresiones e información multimedia sobre un paciente, son algunas de las numerosas aplicaciones que está teniendo la telemedicina. Para la OMS la telemedicina es la utilización en consulta de conocimientos médicos cuando la distancia es un factor determinante, utilizando tecnologías de la información y telecomunicación para el intercambio de información válida para el diagnóstico, tratamiento y prevención de enfermedades y lesiones, investigación y evaluación, y formación continua de los profesionales sanitarios.

Diez años lleva ya el hospital central de la Defensa, Gómez Ulla de Madrid, manteniendo una conexión permanente con los médicos pertenecientes a las unidades militares y servicios especiales desplazados en misiones de apoyo, ayuda humanitaria o mantenimiento de la paz. Para ello se utilizan redes de comunicación que combinan tanto redes de circuitos terrestres cerrados como radioenlaces y comunicaciones por satélite. Todas las teleasistencias quedan registradas en un sistema informático, llamado SIMED, para que el especialista pueda recuperarlas en el momento que desee. Otra vez la organización, guardia y custodia, así como la transmisión eficaz y rápida de toda esta ingente cantidad de datos sale a relucir. ¿Cómo almacenar y transmitir de forma eficiente, rápida y económica todos estos datos?

1.2 La transmisión progresiva

1.2.1 Objetivos

Un primer gran objetivo en todo este proceso ya comentado, de codificación, organización, almacenamiento y transmisión de imágenes médicas digitales, podría consistir en reducir el espacio de almacenamiento de éstas, obteniendo altas tasas de compresión. Se lograría entonces un mayor aprovechamiento de los recursos, así como una reducción del tiempo empleado en la elaboración de los diagnósticos. Se buscaría también un compromiso entre tasa de compresión y calidad para evitar la pérdida de información en el diagnóstico, ya que un error puede suponer la pérdida de una vida humana.

En otro nivel, pero muy relacionado con el anterior objetivo, estaría la transmisión de imágenes digitales, el intercambio de información, que se logra con una transmisión rápida y eficaz. Es por eso que se debe escoger un método de compresión eficiente que facilite la transmisión progresiva de los datos que representan la información, como veremos en la siguiente sección.

1.2.2 Transmisión progresiva

Los métodos tradicionales de transmisión de imágenes son secuenciales, es decir, los datos se comprimen (o se empaquetan de una cierta manera) antes de enviarlos, y una vez recibidos son descomprimidos. Este enfoque tiene una desventaja importante: las imágenes están disponibles sólo cuando se ha recibido el conjunto de datos completo. Véase por ejemplo [40] y [43] para un enfoque con wavelets, o [41] y [65] para otras estructuras matemáticas tales como la interpolación.

Una opción alternativa que soluciona esta limitación es la llamada *transmisión progresiva* [64]. Bajo este enfoque el conjunto de datos (imagen digital) se divide, de una cierta manera, en muchos subconjuntos o fragmentos, y éstos se envían uno a uno. El receptor utiliza estos datos entrantes para realizar una reconstrucción progresiva usando, solamente, los datos disponibles en cada paso. Por supuesto esta reconstrucción parcial no será igual que la imagen original hasta que se hayan recibido todos los datos. Pero hay una aproximación que está mejorando continuamente y que se obtiene en el momento que los nuevos datos se añaden a los ya recibidos para mejorar las reconstrucciones que se están llevando a cabo. Este proceso termina una vez que la imagen se ha transmitido totalmente, o el receptor lo decida así, una vez alcanzada una calidad conveniente en la reconstrucción.

Podemos describir entonces el proceso de la transmisión progresiva en los siete pasos siguientes:

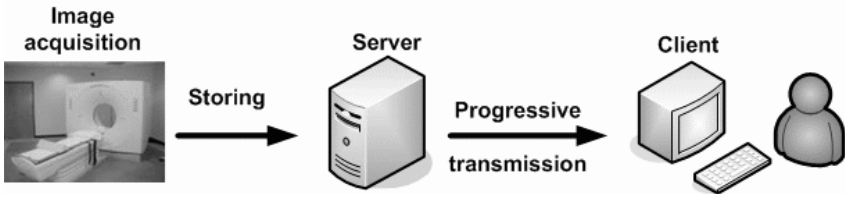


Figura 1.1: Esquema Cliente-Servidor en la transmisión progresiva de datos.

1. Una imagen digital se adquiere (por ejemplo en un escáner) y se almacena en un servidor.

2. En este servidor, la imagen se descompone de cierta manera en *pedazos* o *trozos* llamados regiones. Estas regiones pueden ser parte de la imagen

original, partes de alguna transformación de dicha imagen o incluso partes de una codificación apropiada de la transformada de ésta.

3. Siguiendo un cierto criterio se elige una región de la imagen transformada y se envía al cliente.

4. El cliente entonces utiliza los datos recibidos (regiones) para realizar una aproximación a la imagen original, usando para ello un algoritmo de reconstrucción adecuado.

5. Se elige del servidor una nueva región entre las no transmitidas y se transmite al cliente.

6. Una vez transmitida esta nueva región, el cliente la agrega al conjunto de regiones recibidas y la utiliza para reconstruir la imagen original mediante una nueva aproximación de mejor calidad.

7. Los pasos 5 y 6 se repiten hasta que:

   (a) la transmisión se detiene si la imagen que aparece no es de nuestro interés, o

   (b) la transmisión se detiene si no es necesario mejorar la aproximación, o

   (c) todas las regiones (el conjunto completo de datos) han sido recibidas íntegramente por expreso deseo del cliente (no se desea pérdida en la calidad de la imagen), o

   (d) aparece en la imagen alguna característica de interés para el cliente y entonces el proceso de transmisión debe ser modificado para comenzar a transmitir sólo los subconjuntos de datos que correspondan a dicha característica.

Este modelo de transmisión progresiva descrito tiene sentido cuando existe poco ancho de banda disponible en el canal de comunicación y dichos canales se puelen saturar con una gran cantidad de datos de la imagen, o también cuando el ancho de banda, aún siendo alto, está ocupado con la transmisión de un gran número de imágenes [64]. Excepto en 7(c), en las demás opciones podemos ahorrar consumo del ancho de banda y aligerar una gran cantidad del tráfico diario en la red, y por lo tanto, reducir el tiempo necesario para poder visualizar una imagen. Es en tales situaciones cuando puede ser beneficioso aprovechar los recursos en el nodo receptor y ejecutar un proceso de reconstrucción progresiva cada vez que se reciban nuevos datos, en lugar de esperar el tiempo requerido para recibir la imagen digital entera.

Por otra parte, las opciones 7(a), 7(b) y 7(c) han sido consideradas extensamente desde diferentes puntos de vista, por ejemplo: usando una cierta interpolación de las imágenes [28], [61], [8] y [29], mediante la utilización de wavelets [31], [45], [71] y [40], usando otro tipo de transformadas [72], con técnicas de decorrelación [39] y [28], o introduciendo estrategias adaptativas en la transmisión progresiva [6].

En este trabajo estamos particularmente interesados en 7(d), situación en la que durante la transmisión progresiva de una imagen, el cliente observa un detalle de interés y entonces selecciona para la transmisión una subimagen que contenga dicho detalle (región de interés o ROI) y el proceso continúa con la transmisión progresiva de dicha ROI. Abordaremos detenidamente esta situación en el capítulo 6 y lo aplicaremos preferentemente a imágenes 3D.

1.2.3 Estrategias adaptativas en la transmisión progresiva

En el proceso que hemos comentado sobre la transmisión progresiva, las estrategias desarrolladas habitualmente han sido estáticas, en el sentido que son independientes de detalles internos y por lo tanto el orden para la transmisión de datos es fijo y determinado. Más recientemente se han propuesto *estrategias adaptativas*, en el sentido de que los cambios de datos dentro de la imagen pueden producir secuencias diferentes a la hora de solicitar en diferente orden los subconjuntos de datos para la transmisión. Precisamente en estos casos el principal objetivo es seleccionar los datos más relevantes para enviar en primer lugar esta información. Después, el receptor consigue reconstrucciones de buena calidad con pocos datos transmitidos, pero relevantes [6] y [8]. Así pues, se podría estructurar una estrategia adaptativa de tal forma que se pueda transformar el resultado sobre la representación que utiliza el receptor, si el mecanismo de transmisión cambia. La imagen resultante que el receptor observa podría ser la respuesta al detalle que anda buscando en la imagen.

Esta estrategia adaptativa se basa en diferentes conceptos que veremos y estudiaremos más adelante como son: la descomposición o fragmentación óptima de la imagen para transmitir primero los datos más relevantes, el algoritmo de reconstrucción que utilizará el receptor, así como la estrategia que se debe seguir para elegir o seleccionar la siguiente región de la imagen a transmitir.

1.3 Descripción de este trabajo

El objetivo primordial de este trabajo va a ser la utilización de la factorización de matrices mediante la técnica **SVD para una sencilla y eficaz codificación, transmisión y posterior reconstrucción de manera progresiva de imágenes digitales 2D y 3D**, y más concretamente a su aplicación específica en la transmisión, **sin ningún tipo de redundancia**, y sencilla reconstrucción **de diferentes ROIs a la vez**. Para ello recordaremos las herramientas matemáticas necesarias y relacionadas con dicha factorización.

Comentaremos, también, la utilización tanto de la transformada discreta del coseno (DCT) como de la transformada discreta wavelet (DWT) para la codificación y reconstrucción de imágenes 2D bajo los estándares JPEG y JPEG2000 y realizaremos un estudio comparativo con ambos métodos.

Al igual que con la utilización de la transformada discreta wavelet, el uso de la descomposición SVD nos proporcionará una transformación de la imagen, muy adecuada para procesos posteriores de reconstrucción que en algunos casos

utilizan la interpolación polinómica matricial por trozos de Newton. Esta técnica ha sido empleada para la reconstrucción de imágenes tridimensionales en varios trabajos publicados [27], [28] y [29], y está orientada a la transmisión de imágenes 2D. Nosotros también haremos uso de ella.

Este trabajo está estructurado en los diferentes capítulos que vamos a enumerar a continuación: en el **capítulo 2** los preliminares sobre la descomposición en valores singulares de una matriz, así como las transformadas DCT y DWT y su utilización en la codificación de imágenes. También definiremos las regiones de interés de una imagen. En el **capítulo 3** realizaremos una comparación de nuestro método de compresión (basado en SVD) para regiones de interés de imágenes 2D con el que quizás sea el mejor de los compresores de imágenes digitales, el JPEG2000. Realizaremos también una comparación con el JPEG, comentando cómo influye la transmisión progresiva adaptativa en ambos cotejos. En el **capítulo 4** analizaremos y estudiaremos la codificación de imágenes 3D utilizando SVD y su posterior transmisión progresiva. Para la reconstrucción utilizaremos también la interpolación lineal. En el **capítulo 5** aplicaremos igualmente la transmisión progresiva utilizando SVD pero sin la utilización de la interpolación.

En el **capítulo 6**, **núcleo fundamental** de este trabajo, avanzaremos en la codificación y transmisión progresiva analizando y trabajando principalmente con subimágenes o regiones de interés de imágenes digitales 3D. Expondremos diferentes y novedosos algoritmos de codificación, transmisión y reconstrucción de imágenes 3D y lo más importante, su aplicación sencilla a diferentes regiones de interés (no sólo a una) sin que exista ningún tipo de redundancia en la transmisión de datos.

En el **capítulo 7** intentaremos extender nuestro método basado en SVD a la compresión y visualización de imágenes de vídeo con la ayuda de la transformada wavelet. Las conclusiones y reflexiones finales se describirán en el **capítulo 8**.

Capítulo 2

Preliminares

El objetivo de este capítulo es presentar las herramientas matemáticas y con ellas, sus propiedades más interesantes, que hemos utilizado en todos nuestros experimentos.

Expondremos propiedades sobre la descomposición en valores singulares de una matriz. Describiremos el proceso de la compresión de imágenes digitales y la utilización de las transformadas discretas coseno y wavelet en la compresión JPEG y JPEG2000. Definiremos las regiones de interés de una imagen y estudiaremos cómo son tratadas mediante el método MAXSHIFT.

2.1 Descomposición en Valores Singulares (SVD)

2.1.1 Introducción

Es del todo conocido cómo las descomposiciones de matrices en factores con propiedades especiales, por ejemplo, la diagonalización ortogonal, la descomposición LU, la QR o incluso la descomposición de Schur tienen gran utilidad en variadas y diferentes aplicaciones. Además, una factorización matricial tiene un interés añadido si alguno de los factores son matrices ortogonales. La razón se encuentra en que las transformaciones ortogonales conservan normas y ángulos, en particular, conservan las longitudes de los vectores de error que son inevitables en los cálculos numéricos.

En este capítulo vamos a estudiar una de las factorizaciones más importantes que puede aplicarse a cualquier matriz A de tamaño $m \times n$, la descomposición en valores singulares *Singular Value Decomposition* (SVD). Este método, ya antiguo (se menciona en *Sulle Funzioni Bilineari*, de E. Beltrami, *Giornale di Mathematische II* de 1873) tiene tanto interés teórico como práctico, y merece más atención y crédito del que tiene (en palabras de G. Strang: "descomposición de matrices que no es tan famosa como debiera serlo").

Entre sus muchas aplicaciones está la estimación más fiable del rango de una matriz. La reducción numérica de matrices grandes produce, con frecuencia, un

rango equivocado, debido a la acumulación de errores de redondeo. Los elementos que deberían ser cero podrían estar remplazados por números pequeños. Esto se propaga, se repite y se amplifica durante la reducción, de modo que la eliminación gausiana puede ser inadecuada para el cálculo del rango de una matriz. Por otro lado, cuando una matriz se factoriza mediante SVD puede demostrarse que la mayor parte de los errores de redondeo se presentan en el cálculo de los valores singulares. Así, por lo general, se descartan valores singulares muy pequeños como si fueran cero y se cuentan los restantes para obtener el rango.

SVD se utiliza en una variedad amplia de áreas para el proceso de imágenes tales como restauración, reducción del nivel de ruidos, tomografías computerizadas, imagen *deblurring*[1] y la transmisión oculta de imágenes digitales (*steganography*[2]) [11], [35], [36], [37]. En éstas facetas científicas es muy útil eliminar los valores singulares pequeños y obtener las aproximaciones de la matriz de rango igual al número de valores singulares restantes. SVD también se ha llegado a utilizar como parte del proceso para producir imágenes, pero donde generalmente no se ha utilizado es en la transmisión de dichas imágenes.

Aunque aplicar SVD a una imagen digital al completo genera una gran cantidad de datos (incluso más del doble de los datos originales), a la hora de la reconstruir imágenes mediante SVD, la utilización entre el 19% y el 21% de los valores singulares obtenidos tras la descomposición SVD suele dar una buena reconstrucción de la imagen original, sin que exista mejora visual significativa más allá del 21% [60].

2.1.2 Sobre SVD

En esta sección, recordamos algunos resultados sobre teoría de matrices y SVD. Podemos encontrar más detalles en [33].

Definición 2.1 *Si $A = (A_{ij})$ es una matriz de tamaño $m \times n$, entonces el rango de A es p, con $p \leq \min\{m, n\}$, el número de filas (o columnas) de A linealmente independientes.*

Definición 2.2 *$U \in \mathbb{R}^{n\times n}$ es una matriz ortogonal si $U^T U = I_n$, siendo I_n la matriz identidad de orden n.*

Teorema 2.3 (Descomposición en Valores Singulares) *Si $A \in \mathbb{R}^{m\times n}$ entonces existen matrices ortogonales*

$$U = [u_1, \ldots, u_m] \in \mathbb{R}^{m\times m},$$

y

$$V = [v_1, \ldots, v_n] \in \mathbb{R}^{n\times n},$$

tales que

$$U^T A V = diag\,(\sigma_1, \ldots, \sigma_p), \quad p = \min\{m, n\},$$

[1] enfoque de imágenes borrosas.

[2] esteganografía o el arte de esconder mensajes en fotos, sonidos y demás archivos.

donde

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0.$$

Los σ_i con $i = 1, 2, ..., p$ son los valores singulares de A y los vectores columna u_i y v_i son respectivamente el i-ésimo vector singular izquierdo y el i-ésimo vector singular derecho de A.

Esta descomposición matricial y, con ella, el cálculo numérico de los respectivos valores singulares, está implementada de forma muy eficiente en programas matemático-científicos como *Matlab* [55] y *Mathematica* [69], lenguajes de programación como FORTRAN y C, y en bibliotecas o subprogramas como *Basic Linear Algebra Subprograms* (BLAS) [77].

Recordamos ahora el concepto de norma de una matriz definiendo la norma-2 y la norma Frobenius.

Definición 2.4 *Si $A = (A_{ij})$ es una matriz de tamaño $m \times n$ y*

$$U^T A V = diag\,(\sigma_1, \ldots, \sigma_p), \quad p = \min\{m, n\},$$

es la SVD dada en el teorema 2.3, se define la norma-2 de A como

$$\|A\|_2 = \sigma_1. \tag{2.1}$$

Definición 2.5 *Si $A = (A_{ij})$ es una matriz de tamaño $m \times n$, entonces la norma Frobenius de A está dada en [33] como*

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |A_{ij}|^2}. \tag{2.2}$$

Se puede demostrar entonces [33] que si $A \in \mathbb{R}^{m\times n}$ y

$$U^T A V = diag\,(\sigma_1, \ldots, \sigma_p), \quad p = \min\{m, n\},$$

es la SVD proporcionada por el teorema 2.3, entonces

$$\|A\|_F^2 = \sigma_1^2 + \sigma_2^2 + \cdots + \sigma_p^2, \tag{2.3}$$

y además, a partir de (2.1) y (2.3) , se tiene la siguiente relación entre la norma-2 y la norma Frobenius:

$$\|A\|_2 \leq \|A\|_F \leq \sqrt{n}\,\|A\|_2\,. \tag{2.4}$$

La idea implícita que extraemos de la expresion (2.4) es que si dos matrices (imágenes) están cercanas en norma-2, también lo están en norma Frobenius y viceversa.

Corolario 2.6 *[33] Sea la SVD de la matriz $A \in \mathbb{R}^{m\times n}$ dada en el Teorema 2.3. Si $k < r = rango\,(A)$ y*

$$A_k = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_k u_k v_k^T, \tag{2.5}$$

entonces

$$\min_{rank(B)=k} \|A - B\|_2 = \|A - A_k\|_2 = \sigma_{k+1}. \tag{2.6}$$

El corolario anterior nos dice que la mejor aproximación a la matriz A mediante matrices de rango k es la matriz dada por A_k y que el error cometido con dicha aproximación vale σ_{k+1}, el valor singular k-ésimo+1. Por otra parte, cuanto mayor sea el valor de k, mejor será la aproximación. SVD se utiliza generalmente para aproximar matrices de gran tamaño con un número elevado de valores singulares cero o por debajo de un cierto umbral prefijado. Dicho umbral es útil para eliminar ciertos valores singulares irrelevantes y por lo tanto obtener un ahorro en tiempo y cálculos. Todos estos hechos nos conducen a la presentación del siguiente algoritmo que se utilizará para aproximar iterativamente y de forma óptima una matriz mediante su SVD.

Algoritmo 2.7 (Reconstrucion de una matriz con SVD) *Sea la SVD de la matriz $A \in \mathbb{R}^{m\times n}$ dada en el teorema 2.3 y sean $\sigma_1 \geq \cdots \geq \sigma_q \geq 0$ los q primeros valores singulares de A. Sean también $\{u_1, \ldots, u_q\}$ y $\{v_1, \ldots, v_q\}$ sus correspondientes vectores singulares.*

1. *Aproximamos A por*
$$A_1 = \sigma_1 u_1 v_1^T.$$
2. *FOR $i = 2, 3, \ldots, q$ mejoramos la aproximación A_i de A mediante la expresión*
$$A_i = A_{i-1} + \sigma_i u_i v_i^T.$$
3. *END FOR*

Nótese que $\|A - A_q\|_2 = \sigma_{q+1}$ y si $q = p$ entonces $A_p = A$.

2.1.3 Como utilizar SVD en la transmisión progresiva

Hasta aquí, hemos expuesto de una manera resumida la descripción matemática y en forma de algoritmo de la descomposición SVD, pero es importante observar que SVD es un método interesante para organizar la información de una matriz (a partir de ahora, utilizaremos los sustantivos "imagen" y "matriz" alternativamente, como sinónimos, aún cuando el segundo es una descripción numérica del primero, que es una visualización de los datos numéricos). Esta información la organizaremos desde la más relevante, valores singulares más grandes y sus respectivos vectores singulares asociados, a la menos relevante, valores singulares pequeños y sus vectores singulares asociados. En este sentido, podemos elegir los primeros q valores singulares de A

$$\sigma_1 \geq \cdots \geq \sigma_q \geq \epsilon > \sigma_{q+1} \geq \cdots \geq \sigma_p \geq 0,$$

que sean mayores que un umbral $\varepsilon \geq 0$ prefijado y quitar los valores singulares de A más pequeños que el umbral, y sus correspondientes vectores singulares (se considera que éstos no proporcionan una mejora sustancial en la reconstrucción de la imagen). Después, podemos transmitir estos q valores singulares de A junto con sus respectivos vectores singulares y reconstruir la imagen.

Como ya se ha comentado, la SVD se ha utilizado normalmente como parte del proceso para producir imágenes [60], pero no se ha utilizado generalmente en la transmisión. Esto es debido al crecimiento, a casi el doble, de la cantidad de los datos que se manejan en la confección de una imagen utilizando SVD. Pero a pesar de este aumento en la cantidad de los datos, es posible diseñar un algoritmo para la transmisión progresiva de imágenes con pérdida, que conlleva una representación excelente de la imagen a porcentajes bajos en la transmisión de datos. Ésta es la idea central de los experimentos que vamos a desarrollar más adelante. De hecho y como ya hemos comentado, el uso de SVD nos proporcionará un tipo de imagen adecuada para procesos posteriores de reconstrucción de imágenes 2D y 3D.

2.2 Proceso de digitalización y compresión de imágenes digitales

La digitalización de una imagen es el proceso de conversión a formato digital de una señal analógica transmitida por una cámara o sistema óptico. Se realiza mediante un muestreo o digitalización de las coordenadas espaciales. Si este muestreo es suficientemente grande, no afecta a la apreciación del ojo humano. En caso contrario, la degradación de la imagen es notable como se puede ver en la siguiente secuencia de la misma imagen (figura 2.1).

Tras la digitalización de una imagen, lo más usual es comprimirla después y para ello las técnicas más utilizadas para la compresión de imágenes digitales se basan en el hecho de que las representaciones de dichas imágenes contienen información redundante. Esto ocurre porque en general píxeles vecinos tienen valores parecidos. Utilizando entonces ciertas técnicas se puede reducir el tamaño de estas imágenes hasta ciertas tasas de compresión sin que se llegue a notar visualmente en la calidad de la representación. Así muchos compresores (entre los que se encuentra la familia de compresores JPEG) realizan para este fin tres etapas, como son:

- **Transformada de la imagen**
- **Cuantificación**
- **Codificación**

Veáse ejemplo de compresión JPEG2000 en la figura 2.2.

2.3 Transformadas para imágenes digitales

Una transformada aplicada a una imagen trata de obtener y separar la información irrelevante de ésta para que la imagen pueda ser comprimida de una forma mucho más eficaz, reduciendo así el tamaño de su representación sin afectar demasiado a la percepción visual por parte del usuario.

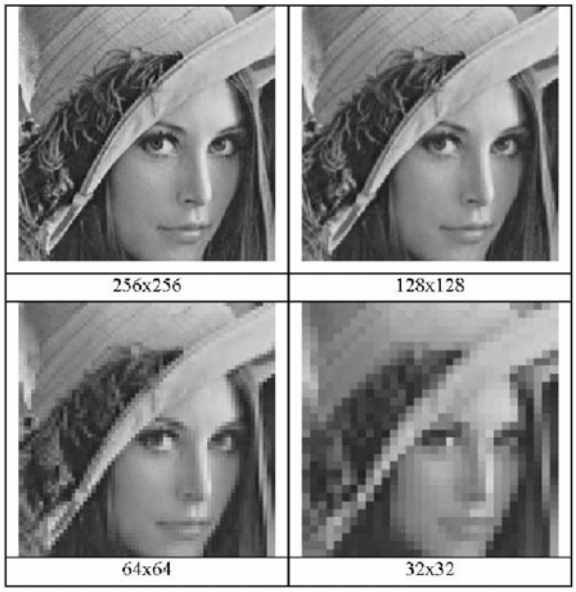


Figura 2.1: En la resolución 256×256 cada píxel equivale a un 1 punto mientras que en la de 128×128 cada píxel equivale a 4 puntos, 64×64 a 16 puntos y 32×32 a 64 puntos.

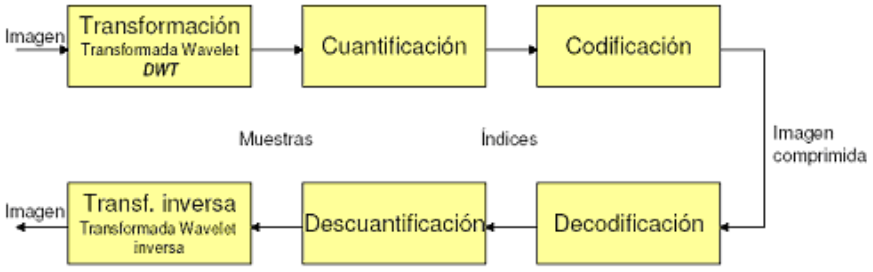


Figura 2.2: Compresión y descompresión JPEG2000.

La aplicación de una transformada a un conjunto de datos originales permite así expresar el mismo en otra "base" que resulte más conveniente para el posterior procesamiento. Un dato muy importante a tener en cuenta en la transformación de imágenes es que el paso de la aplicación de la transformada es reversible. Esto quiere decir que se pueden recuperar los datos originales sin pérdida de información. La transformada inversa aplicada al conjunto de datos "transformados" devuelve los datos originales.

Veamos a continuación un breve resumen de dos de las transformadas más usadas.

2.3.1 Transformada discreta coseno (DCT)

Dispositivos habituales de compresión-descompresión de imágenes digitales como el JPEG, se basan en la más que famosa transformada discreta del coseno (relacionada con la transformada discreta de Fourier o DFT). Veamos en qué consiste.

Definición 2.8 *Dada una imagen o matriz $A(x, y)$ de tamaño $m \times n$ su transformada coseno discreta (DCT) es la función $F(u, v)$ definida como*

$$F(u,v) = \frac{1}{4}C(u)C(v)\left[\sum_{x=0}^{m-1}\sum_{y=0}^{n-1} A(x,y)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}\right]$$

$$\textit{para } x = 0, 1, \cdots, m-1 \quad \textit{e } y = 0, 1, \cdots, n-1 \quad \textit{y donde}$$

$$C(\alpha) = \begin{cases} \sqrt{\frac{1}{N}} & \alpha = 0 \\ \sqrt{\frac{2}{N}} & \textit{en otro caso} \end{cases}$$

Podemos destacar que en el caso de la utilización de esta transformada para la compresión JPEG de imágenes digitales, N toma el valor de 2 y tanto m y n suelen valer 8, convirtiéndose así en la famosa DCT bidimensional (DCT-2D).

2.3.2 Transformada discreta wavelet (DWT)

El concepto de transformada wavelet [43] y [63] actualmente está muy en uso puesto que puede proporcionar una compresión con pérdida con un ratio de compresión muy alto sin una degradación significativa de la imagen. La utilizaremos también con el fin de cuantificar la imagen y eliminar las redundancias psicovisuales, conservando los rasgos distintivos que permiten identificar la imagen (bordes).

Definición 2.9 *La transformada discreta wavelet dos-dimensional [43] de una matriz $A(x, y)$ viene dada por*

$$TW(A(x,y)) = \sum_{(i,j)} a_{M,i,j}\phi_{M,i,j} + \sum_{m=1}^{M}\sum_{(i,j)}\sum_{k=1}^{3} d^k_{m,i,j}\psi^k_{m,i,j} \tag{2.7}$$

donde $a_{M,i,j} = \langle A, \phi_{M,i,j} \rangle$ *y* $d_{M,i,j} = \langle A, \psi_{M,i,j} \rangle$ *son respectivamente los coeficientes scaling y los coeficientes wavelets, donde* $\langle , \rangle$ *representa el producto interior y*

$$\begin{aligned} \psi_{M,i,j} &= 2^M \psi(2^M x - i, 2^M y - j), \\ \phi_{M,i,j} &= 2^M \phi(2^M x - i, 2^M y - j). \end{aligned}$$

La función ψ es la denominada "wavelet madre" y ϕ es la función scaling que se obtiene a partir de ψ. Para que ψ sea una wavelet madre ha de cumplir ciertas condiciones de admisibilidad como el estar bien localizada en el tiempo, ser de medida nula y su transformada ser un filtro pasa-banda de rápido decaimiento. De esta manera, la matriz $A(x, y)$ se muestrea empleando versiones dilatadas y trasladadas del wavelet madre ψ, similar a lo que se hace para el análisis de Fourier.

Lo relevante del análisis wavelet es el hecho de que nos divide la señal en dos partes: una es la dada por el filtro *paso-bajo*, ϕ, y que retiene la energía de la imagen; la denominada parte "scaling" o "coeficientes scaling"; la otra es la obtenida por el filtro *paso-alto*, ψ, es en la que quedan almacenados los perfiles y las variaciones de un píxel de la imagen con los de su entorno; la llamada "parte wavelet" o "coeficientes wavelet".

2.4 Cuantificación

Como ya se ha comentado, una vez que se ha obtenido la transformada de una imagen digital, normalmente se hace necesaria una etapa de *cuantificación*[3]. El efecto de la cuantificación viene dado por la imposibilidad de tener un rango infinito de valores de medida para los datos con los que estamos trabajando. Este rango de valores debe ser limitado y debemos reducir la cantidad de bits necesarios para su almacenamiento. Desgraciadamente no existen criterios fijos que nos permitan decidir el número óptimo de bits con los que guardar datos.

Así, cuantificar significa que a un amplio rango de valores de entrada le corresponden un número limitado de valores de salida, en otras palabras, disminuir la precisión de los coeficientes transformados, reduciendo de esta manera la cantidad de bits necesarios para representar y almacenar la imagen. Esta etapa supone la eliminación de parte de la información de los datos originales.

Esta segunda etapa de cuantificación es optativa en toda compresión y, a grandes rasgos, determina si la compresión será con o sin pérdida. Si se aplica la cuantificación estaremos, de una u otra manera, descartando información de la transformada original por lo que será imposible la reconstrucción perfecta de la misma. El propósito final de la etapa de cuantificación es alcanzar el mayor grado posible de compresión descartando información de manera que se puedan representar los datos con una precisión no mayor a la necesaria, para una calidad final requerida. Puesto que es una operación irreversible, ya que

[3] quantization.

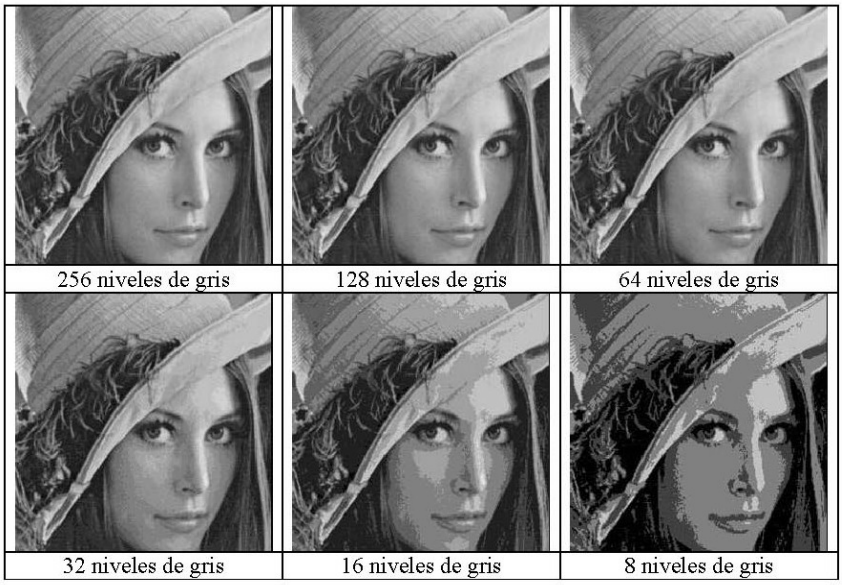


Figura 2.3:

se pierde información visual, la cuantificación conduce a una compresión con pérdida de datos.

Pero afortunadamente la visión humana no se ve tan afectada en esta etapa de la compresión por estas variaciones. Como ejemplo y aplicando la cuantificación al tema de las imágenes, más concretamente a los valores de medida para la intensidad de brillo de los pixeles que conforman una imagen, podremos observar cómo existen determinados valores de umbral para la cuantificación, por encima de los cuales no se aprecia una significativa ganancia en la visualización de la imagen, pero por debajo de ellos sí se aprecia una pérdida importante de resolución en dicha imagen.

Este efecto se puede observar en la siguiente secuencia de imágenes, (figura 2.3) donde se parte de una imagen con 256 niveles de gris y se llega a una imagen con 8 niveles de gris. Se observa que aparecen unos falsos contornos en regiones que eran suaves en la imagen original, pero sólo en caso de una cuantificación extrema.

2.5 Compresores JPEG

2.5.1 Formato JPEG

Uno de los mayores logros en el campo de los estándares de compresión de imágenes estáticas ha sido JPEG. El formato JPEG (Joint Photographic Experts

Group) fue desarrollado a principio de los 90 y estandarizado en 1992 por la International Telecomunication Union de Estados Unidos. El formato es el resultado de años de investigación y define un estándar para la codificación de imágenes, en un formato sumamente eficiente.

El codificador tuvo y tiene una gran aceptación por su sencillez, rapidez y eficacia, lo cual contribuyó en su momento a una amplísima difusión, aunque probablemente Internet y su libre distribución hayan tenido también mucho que ver.

Basicamente, la foma de comprimir que tiene el JPEG es elegir bloques de píxeles contiguos de tamaño 8×8 y aplicarles una transformada de la imagen, la transformada discreta coseno 2D (DCT-2D). Luego, y dado que el ojo humano detecta muy bien pequeños cambios de brillo en áreas relativamente grandes, pero no así cuando éste cambia rápidamente en áreas pequeñas (variación de alta frecuencia), se eliminan las variables de más alta fecuencia, suavizando la imagen y sin perder excesiva calidad visual. Éste es el proceso en el que se pierde la mayor parte de la información (y calidad) cuando una imagen es procesada por este algoritmo.

Y por último, se comprimen dichos bloques con un algoritmo tipo ZIP (RLE, Huffman, LZW, etc.). El compresor más utilizado es el algoritmo de Huffman, que se encarga de transmitir los coeficientes ordenados. Una razón para utilizarlo es que es fácil de implementar.

El formato JPEG está totalmente indicado para la compresión de imágenes de tono continuo, como las fotografías. Una fotografía contiene una cantidad considerable de información que el ojo humano no puede descubrir realmente y que puede ser desechada sin que uno perciba visiblemente dicha pérdida. Por el contrario, esta misma característica convierte al formato JPEG en un pésimo compresor para imágenes que no presenten esta característica, como imágenes de archivos de texto. En archivos de este tipo, no sólo la compresión sería baja, sino que la pérdida de datos resultaría visible. Incluso para compresiones altas se puede llegar a obtener una nueva imagen, completamente diferente a la original, sin que el descompresor detecte el error.

2.5.2 El estándar JPEG2000

El avance de las nuevas tecnologías multimedia, a la par que el uso tan multitudinario de Internet, supuso que en 1997 se realizase un llamamiento al Comité JPEG para aportar nuevas contribuciones en la creación de un reciente y novedoso estándar de compresión de imágenes, el JPEG2000. Este grupo de desarrollo está formado por unas 100 empresas e instituciones públicas de más de 20 países (como representante español se encuentra el Instituto de Óptica, perteneciente al CSIC).

A pesar de tener grandes bazas a su favor (eficiente uso de memoria, baja complejidad y compresión eficiente) con el tiempo fueron surgiendo nuevas necesidades a las que JPEG no respondía adecuadamente: resolución y calidad única, no se puede seleccionar tasa de bits (número de bits por píxel), no tiene regiones de interés, no es bueno para imágenes generadas por ordenador

(fue creado para fotografías), le afectan mucho los errores de transmisión: el error en 1 bit provoca grandes pérdidas.

Así pues, el JPEG2000 es un estándar de compresión de imágenes creado por el mismo comité que creó JPEG, y estaba destinado a supervisar el estándar JPEG. Esta alternativa al JPEG nació con la intención de utilizar los nuevos avances tecnológicos y de investigación para mejorar la calidad de las imágenes tratadas y aumentar sus posibles aplicaciones: compresión de imágenes médicas, bibliotecas digitales, multimedia, Internet o teléfonos móviles, todo ello a cambio de un mayor tiempo de cálculo en la compresión. Aunque realmente hay que comentar que el JPEG2000 no llega a ser una alternativa al JPEG, sino que se convierte en un formato complementario para ser empleado en aquellos casos en que JPEG no es suficiente.

Por todo ello, el mismo comite JPEG creó un nuevo estándar capaz de tratar con diferentes tipo de imágenes (fotografías, imágenes médicas, científicas, texto) con diferentes características (blanco y negro, escala de grises, multicomponente) permitiendo diferentes modelos de computación (cliente/servidor, transmisión en tiempo real) y a ser posible con un sistema unificado. Este nuevo proyecto figura como norma ISO (ISO/IEC 15444-1).

Una de las características destacables del estándar JPEG2000 es la utilización de la transformada discreta wavelet (DWT) en sustitución de la DCT (figura 2.2). En 1997 se propusieron diferentes algoritmos pero fue la descomposición wavelet la que se adoptó como base del nuevo estándar. Actualmente la transformada wavelet está ampliamente extendida a innumerables disciplinas como la dinámica molecular, la astrofísica, el estudio de las turbulencias y la mecánica cuántica, así como en otros insospechados campos como la biometría, los análisis de sangre, el análisis de electrocardiogramas, el estudio del ADN o el análisis de proteínas.

2.5.3 Fotos en Alta Definición

Al igual que ha ocurrido con los televisores y reproductores de vídeo, el formato de Alta Definición (HD) ha llegado a la industria fotográfica. HD Photo (Foto de Alta Definición) o JPEG-XR es un nuevo formato gráfico creado y adoptado por Microsoft. Este nuevo formato de codificación de imágenes fijas ofrece mayor calidad en menor tamaño que el formato JPEG.

El formato HD Photo permite crear imágenes con la misma calidad que JPEG con solo la mitad del tamaño del fichero, o el doble de calidad en la imagen con el mismo tamaño que las actuales JPEG. Una fotografía podrá contener hasta 68,6 Terapíxeles y ocupar 32 Gb. Además, este nuevo formato ofrece la posibilidad de guardar las imágenes con, o sin pérdida. En la segunda opción se mantienen todas las características originales de la imagen, y consecuentemente el archivo tendrá un mayor tamaño, aún así la ventaja del nuevo formato está en que una vez guardada la imagen sin pérdida, el tamaño es menor en comparación con JPEG. Entre las firmas que ya han anunciado su apoyo al HD Photo figuran, además de Microsoft, el primer fabricante de herramientas gráficas Adobe.

Mientras que el formato HD Photo ya está incorporado en Windows Vista,

el gigante informático Microsoft intenta y espera próximamente que HD Photo sea certificado como estándar por el grupo JPEG bajo el nombre JPEG-XR. La certificación contribuiría a que Microsoft motivara a los fabricantes de cámaras digitales a incorporar el formato en sus productos facilitando la implementación del nuevo formato, y ofreciendo una versión libre de costes por los derechos de uso de sus patentes. Lo que a fecha de hoy aún no tenemos es una comparación entre JPEG-XR y JPEG2000.

2.6 Regiones de interés de una imagen (ROIs)

2.6.1 Definición

Las regiones de interés (ROIs) son zonas o detalles de una imagen (subimágenes) que llaman la atención del observador o a las que éste da una mayor importancia respecto al resto de la imagen. Esta mayor atención o tratamiento preferencial del observador sobre esa zona de la imagen puede considerarse como: solicitar más calidad o resolución de esa región o pedir prioridad en la transmisión progresiva de la imagen, es decir, que si se interrumpe la transmisión, nos aseguramos que lo primero que se ha enviado haya sido esa región.

Expresado ahora en un término más "informático" podríamos decir que una ROI es una región de la imagen que va a recibir un mejor trato por parte del codificador. Pero no todos los compresores o codificadores de imágenes permiten dar un tratamiento diferencial a estas zonas de interés, e incluso entre los que lo consiguen, muy pocos permiten la codificación de más de una de estas regiones de interés.

Esta necesidad de priorizar una o varias zonas de interés sobre el resto de la imagen sí que puede ser recogida y efectuada cuando descompongamos la imagen utilizando SVD dando lugar, como veremos en capítulos siguientes, a un eficaz y sobre todo, sencillo método de codificación y transmisión progresiva de dichas regiones de interés. Pero ahora comentemos cómo son tratadas las ROIs por el JPEG y el JPEG2000.

2.6.2 Regiones de interés con el estándar JPEG

JPEG no incluye un régimen específico para la codificación y descodificación de ROIs. Por supuesto que existen y se han desarrollado diferentes planteamientos a la hora de encauzar el tema [42] y [66]. Algunos de ellos [51] seleccionan una ROI de la imagen original, utilizando más bits para la ROI y desechándolos para las secciones fuera de la ROI. Así puede lograrse un gran aumento del índice de compresión, pero con la desventaja de que la ROI debe ser conocida a priori y que no es posible elegir de forma arbitraria regiones de la imagen una vez que ésta haya sido codificada. Otros enfoques [42] dividen la imagen en bloques (por ejemplo: 32x32 píxeles) y codifican cada bloque varias veces, con codificaciones separadas para cada nivel de detalle. En el primer paso de la transmisión son enviados los bloques correspondientes al primer nivel de detalle,

y cuando el observador selecciona una ROI, se envían los bloques con más detalle correspondientes a dicha ROI. Este proceso es costoso en cuanto a tiempo se refiere y propenso a la redundancia de datos, pero tiene la ventaja de que esto debe hacerse sólo una vez en el codificador.

2.6.3 Regiones de interés con el estándar JPEG2000

Como ya hemos comentado las regiones de interés son regiones de la imagen que van a recibir un mejor trato por parte del codicador. Básicamente y utilizando el estándar JPEG2000 esto consiste en un realce de los coeficientes de dichas regiones (cuadradas o circulares) de manera que el codificador no pueda eliminarlos cuando tenga que cuantificar. Realzar una muestra es en definitiva multiplicarla por 2^u, es decir, una operación de desplazamiento de dicha muestra [79].

Procedimiento o método MAXSHIFT

El procedimiento MAXSHIFT es el método que el estándar JPEG2000 utiliza por defecto para el tratamiento de ROIs. Se llama así debido a que los píxeles de la ROI se desplazan[4] antes de aplicar la DWT de manera tal que en la transmisión estén separados la ROI y el segundo plano o fondo[5] de la imagen, y consiste en elegir un determinado realce de manera que el coficiente de menor valor de la ROI sea mayor que cualquiera del resto de los coeficientes que no estén en la ROI (véase figura 2.4).

Este tratamiento diferente entre la ROI y el fondo de imagen, permite una diferente compresión para cada una de las partes. Además, con este método el decodificador no necesita conocer las coordenadas de la ROI puesto que basta con saber el umbral a partir del cual el coeficiente pertenece a la región.

Dado que el realce de las regiones de interés se realiza antes de transformar la imagen mediante la transformada wavelet, hay que tener en cuenta ciertas consideraciones. Una es la forma que tendrán las ROIs en cada nivel de codificación. Éstas serán diferentes en función del filtro wavelet utilizado (Daubechies, CRF, SWE, etc.). También hay que tener muy en cuenta el caso en que se definan varias regiones de interés. Lo más probable es que las ROIs se solapen en los niveles de baja resolución (figura 2.5).

Podemos observar entonces cómo la elección de dos o más ROIs perjudica enormemente a la hora de utilizar la transformada wavelet, por lo que lo normal es reorganizar las dimensiones de las ROIs para convertir dichas ROIs en una sola que cubra o abarque todas las demás. De esta condición se desprende que JPEG2000 soporte una única ROI.

Restricciones del método MAXSHIFT

El enfoque o método MAXSHIFT posee algunas limitaciones [2] y [4]:

[4] shifted.
[5] background.

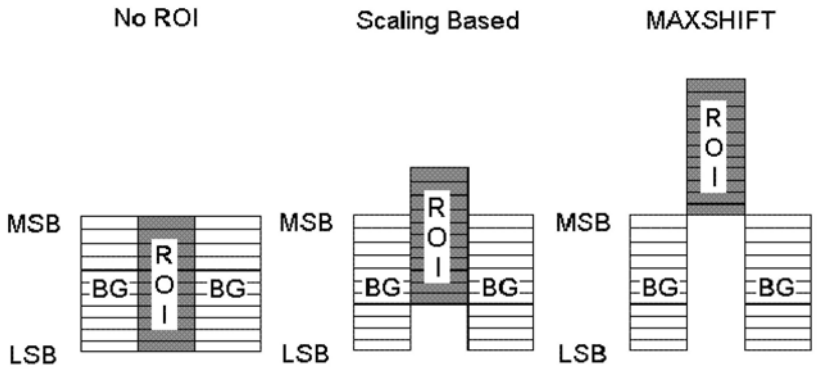


Figura 2.4: Esquema sobre el procedimiento o método MAXSHIFT.

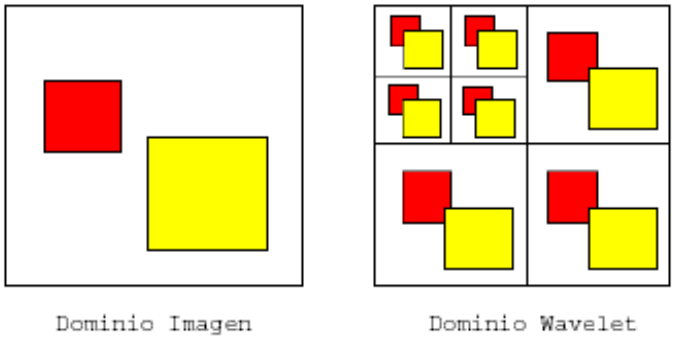


Figura 2.5: Esquema sobre el solapamiento de ROIs en el dominio wavelet.

- Como ya hemos comentado, el escalamiento MAXSHIFT sólo admite la elección de una única ROI para toda la imagen. Si surgiese algún otro interés por cualquier otra región de la imagen, deberíamos definir entonces una nueva ROI que contuviese a todas estas regiones al mismo tiempo.

- El codificador debe saber dónde está la ROI antes de iniciar la codificación para escalarla, por lo que la elección de la ROI no es posible hacerla "sobre la marcha". Si la ROI cambia, aparecen nuevos datos que codificar y la transmisión de datos debe reanudarse desde el principio, produciéndose así una redundancia en la transmisión de datos.

- Lo usual es utilizarlo para niveles de color de 8 bpp. Para ratios más altos, por ejemplo imágenes con 16 bpp en nivel de gris, el desplazamiento de la ROI puede saturar la imagen.

2.7 JPEG2000 frente al JPEG

La utilización de la transformada discreta wavelet (DWT) en sustitución de la DCT a la hora de desarrollar el JPEG2000 ha producido una serie de cambios importantes en el momento de transformar una imagen digital antes de poder comprimirla. Podemos entonces comentar las principales diferencias entre la DWT y la DCT (JPEG2000 versus JPEG) a la hora de tratar la información, que son:

- Previo a la codificación, la imagen se divide en las llamadas *baldosas*[6]. Cada baldosa es una región rectangular de la imagen, las cuales no se solapan entre ellas y se codifican por separado. Para la DWT las imágenes pueden estar compuestas por una sola baldosa (toda la imagen en sí) o muchas de ellas, pero todas deben tener el mismo tamaño (excepto, quizás, las baldosas situadas en los bordes de la imagen). Baldosas son, por ejemplo, los bloques de 8×8 píxeles del JPEG. Por un lado, las baldosas de pequeño tamaño aceleran el proceso de codificación, con la ventaja de que es posible codificarlas con diferente calidad, o decodificar sólo algunas, pero por otro lado la elección de una baldosa demasiado pequeña puede dar lugar al conocido efecto *píxelado en bloques*[7] padecido por el JPEG.

- La DWT genera una serie de niveles de descomposición, donde cada nivel contiene un número de sub-bandas, y cada sub-banda contiene los coeficientes que describen las características horizontal y vertical de las baldosas originales. Dado que DWT se aplica a dichas baldosas y éstas cubren la imagen completa, se evita así el problema del píxelado en bloques del JPEG, que ocurre cuando sólo se utilizaban unos pocos coeficientes de la DCT.

[6] tiles.
[7] blocking.

Figura 2.6: Ejemplo de ROI codificada con JPEG2000.

- La gestión o el manejo de las ROIs se realiza de forma directa con el JPEG2000. Las ROIs se codifican de tal manera que se colocan en primer lugar en el flujo de datos y con un mayor nivel de detalle, por lo que las ROIs son decodificadas antes que el resto de la imagen y, además, se ven mejor. Con el JPEG esto no era posible.

Y todo ello nos lleva a poder resumir las principales y más novedosas características del estándar JPEG2000 como son [81]:

Compresión de imágenes de tono continuo y binarias: JPEG2000 es capaz de comprimir imágenes de tono continuo[8] y binarias[9]. Permite además una compresión de imágenes con un rango dinámico de profundidad de bits para cada componente, desde 1 bit hasta los 32 bits.

Robustez ante errores en los datos: JPEG2000 define una serie de mecanismos para permitir la detección de errores de los datos. Por ejemplo, las aplicaciones que hacen uso de canales de comunicación Wireless harían uso de esta característica.

Mejor rendimiento a ratios bajos: JPEG2000 ofrece un rendimiento superior a bajos ratios con respecto al resto de los estándares. Las aplicaciones de transmisión de imágenes a través de la red se benefician de esta característica (40-60% más de compresión que JPEG al mismo ratio).

Posibilidad de definir regiones de interés (ROI): Al contrario que su predecesor, el estándar JPEG2000 permite definir una ROI en la imagen con el

[8] continuous-tone.
[9] bi-levels.

fin de que sea codificada y transmitida con mayor prioridad, mejor calidad y menor distorsión que el resto de la imagen (véase figura 2.6).

Capítulo 3

Comparación entre algunos métodos de transmisión progresiva de ROIs en imágenes digitales 2D

3.1 Introducción

El uso de imágenes digitales ha llegado de forma generalizada a la mayoría de los aspectos de nuestra vida cotidiana. Tomamos fotografías con nuestro teléfono móvil o cámara digital, y si sobrepasamos el límite de velocidad en la carretera, el radar de la policía parpadea y toma una imagen digital de nuestro coche; el médico realiza una tomografía computerizada de nuestro brazo roto, o la imágenes de satélite utilizadas para predecir el tiempo, todas ellas, se han tomado en formato digital. Por otra parte, estas imágenes deben ser procesadas, almacenadas y, finalmente, recuperadas para verlas.

Pero nos centraremos ahora en las imágenes de gran tamaño y en sus aspectos más importantes. Un buen ejemplo es el diagnóstico médico por imagen. Una tomografía computerizada (TC) estaba formada, en los años 70, por un par de imágenes de unos 100 kilobytes (Kb), mientras que hoy en día una imagen por ultrasonido 3D, compuesta de 500 cortes de 512×512 píxeles cada uno y "pesando" más de 500 megabytes (Mb), es una práctica común. El uso intensivo de TC, Resonancias Magnéticas (RM) y Ecografías produce una gran cantidad de terabytes (Tb) en información, que debe ser tratada y almacenada, y más tarde recuperada para poder mostrarla a los médicos cuando éstos hacen su diagnóstico. Por ejemplo, la red local de un departamento de radiología de un hospital sufre, de promedio, un tráfico de datos de más de 1 Gb/s, y mucho más que eso en momentos "pico" de uso de estas imágenes, lo que significa que una imagen promedio de unos 20 Mb requiere, en el mejor de los casos, unos

Figura 3.1: Ejemplo de imagen de gran tamaño.

segundos para transmitirla, pero podría llegar a varios minutos en momentos críticos de saturación de la red [39]. La situación es aún peor si en lugar de una red local, utilizamos Internet, una red de conexión dial-up o de línea conmutada (las que utilizan módem, en contraposición a las redes de tipo DSL, también llamada internet flash), o cualquier otro canal de banda estrecha. Podemos pensar también en grandes imágenes aéreas, por satélite o de vigilancia, imágenes que se manejan habitualmente y que cubren al mismo tiempo grandes superficies con un gran nivel de detalle.

En la figura 3.1, se puede encontrar un buen ejemplo con nuestra imagen de referencia. Esta imagen tiene 5000×3120 píxeles con 16 bits por píxel de niveles de gris en origen, con un total de 31200000 bytes (cada píxel cubre aproximadamente un centímetro). Enviar esta imagen usando un módem puede tardar varias horas, incluso si se ha comprimido (por ejemplo, la alta calidad de compresión JPEG todavía requiere de 5 a 7 Mb para asignar la imagen de referencia).

3.1.1 Transmisión progresiva y ROIs

La manera de resolver el ya comentado problema sobre la transmisión de imágenes digitales de gran tamaño es la utilización de métodos de transmisión progresiva, capaces de comprimir durante la fase de almacenamiento y transmisión, y eficientes en la reconstrucción o visualización de la imagen transmitida.

Por un lado, se pueden usar métodos convencionales de transmisión secuencial, donde la imagen no puede ser visualizada hasta completar la transmisión

Figura 3.2: Reconstrucción de la imagen de referencia usando un 33% de los datos transmistidos (transmisión secuencial).

de toda la imagen [40] y [41]. Incluso en los casos en que no es necesaria una transmisión completa, por lo general, la imagen obtenida es inservible hasta que la mayoría de los datos se han recibido, como podemos ver en la figura 3.2.

Por otro lado están los sistemas de transmisión progresiva [64] que organizan los datos de la imagen de tal forma que, desde el comienzo mismo de la transmisión puede verse una reconstrucción aproximada de la imagen original (veáse figura 3.3) y la calidad de esta reconstrucción va mejorando según se van recibiendo un mayor número de datos.

Otro aspecto interesante de la transmisión progresiva es el gran nivel de compresión efectiva que puede lograrse de **forma indirecta**, ya que el cliente puede detener la transmisión de datos cuando observe que la imagen haya alcanzado suficiente nivel de detalle o que la imagen ya no resulte de su interés [39] y [64]. Además, si el cliente es capaz de interactuar con el proceso de transmisión y seleccionar ciertas regiones de interés en el **preciso instante** en que algunas regiones de la imagen le parezcan relevantes, entonces podrá reconstruir la secuencia de transmisión y enviar en primer lugar sólo la información relativa a dichas ROIs (figura 3.4).

3.1.2 Transmisión con pérdida y sin pérdida

Después de la recepción completa de la serie de datos que configuran una imagen y de su posterior reconstrucción, la imagen reconstruida puede ser idéntica al original. En este caso al proceso se le llama *transmisión sin pérdida*[1], es decir, no se pierde ningún tipo de información durante el tratamiento de los datos y la reconstrucción. Pero por otra parte, la imagen reconstruida podría no ser idéntica a la imagen original, es decir, parte de la información original se ha perdido y no se puede recuperar. En este caso llamamos al proceso *transmisión con pérdida*[2].

Podría pensarse que la transmisión con pérdida no tiene sentido, y que deberíamos tener siempre el 100% de la imagen original. Sin embargo las estrategias con pérdida tienen una gran ventaja: se necesita muchísima menos información para reconstruir una imagen de forma bastante aceptable.

[1] lossless transmission.
[2] lossy transmission.

Figura 3.3: Reconstrucción de la imagen de referencia usando un 2.5% de los datos transmitidos de forma progresiva.

Figura 3.4: Reconstrucción de la imagen de referencia usando un 2.5% de los datos transmitidos de forma progresiva con un mayor nivel de detalle en la ROI seleccionada.

Cada método es adecuado para un tipo diferente de datos. Por un lado, una estrategia sin pérdida sería la única aceptable para datos como los de un libro de texto o incluso para algunas imágenes médicas. Por otra parte, una estrategia con pérdida sería la adecuada para tipos de datos como fotografías, vídeo o sonido, donde el ruído (o falta de precisión) es importante.

3.2 La transformada discreta del coseno (DCT) y el JPEG

Como ya se ha comentado en el capítulo 2, el JPEG es un formato de compresión para imágenes digitales bastante conocido y ha sido utilizado desde principios de los años 90. JPEG define básicamente dos métodos de compresión, el método de referencia basado en un sistema con pérdida DCT, y un método de predicción por compresión sin pérdidas. Nos centraremos ahora en el método de referencia, ya que es el más utilizado y el que ofrece mejor compresión.

Tras exponer anteriormente la definición de la DCT (definición 2.8) vamos a observar detenidamente los pasos (transformada, cuantificación y codificación) que se aplican a una imagen en general para poder codificarla según el estándar JPEG, basados en la transformada DCT. Unos pequeños cambios a estos pasos que ya comentaremos, como por ejemplo, el tratamiento y transmisión de ROIs que no incluye el JPEG, nos permitirá elaborar un esquema especial para poder utilizar la transmisión progresiva de ROIs, en torno a la DCT, y poder establecer así una comparativa con el mismo método pero utilizando SVD.

3.2.1 La transformada DCT

Al utilizar la DCT para codificar una imagen de entrada, ésta se divide en bloques de 8×8 píxeles y a continuación los valores de los píxeles se desplazan: de enteros sin signo en un rango $\left[0, 2^{P-1}\right]$ a enteros con signo en un rango $\left[-2^{P-1}, 2^{P-2}\right]$. Aplicamos, entonces, la transformada DCT a cada bloque.

El dividir la imagen en bloques de tamaño 8×8 se realiza por razones de rendimiento de la DCT, dado que la transformación de la imagen completa mediante DCT requiere una enorme cantidad de tiempo. Entonces, si a la transformada de la imagen le llamamos F, y $F(u, v)$ es el valor del píxel (u, v) de la nueva imagen, la transformada DCT se calcula según la fórmula de la definición 2.8 con $N = 2$ y $m = n = 8$, es decir, la DCT-2D.

Después de aplicar esta transformación, cada bloque 8×8 de F aparece ordenado por frecuencias (bajas arriba-izquierda y altas abajo-derecha) y está formado por los llamados *coeficientes DCT* [68]. El coeficiente situado en la posición $(0, 0)$ se llama *coeficiente DC* (también llamado *coeficiente de continua*) y los restantes 63 son llamados *coeficientes AC*. El coeficiente de continua es el más importante del bloque. En él se compacta la mayor parte de la energía. Para un bloque 8×8 de una imagen en general, la mayoría de los coeficientes AC están cerca del valor cero y pueden ser descartados, de modo que sólo los

más significativos se almacenarán y como es lógico, a más coeficientes descartados se alcanzarán mayores tasas de compresión, pero también se pierde más información de la imagen original.

3.2.2 Cuantificación DCT

Después de obtener los coeficientes DCT, éstos serán cuantificados de manera uniforme usando una matriz de cuantificación de tamaño 8×8. Esta matriz es un conjunto de enteros en el rango $[1, 255]$, que especifica el tamaño del paso para cuantificar su correspondiente coeficiente DCT, por lo que cada coeficiente se puede representar sin más precisión que la realmente necesaria y aumentar así la relación de compresión. La matriz de cuantificación no es única (es el estándar JPEG quien las define) y depende de cuánto queramos comprimir la imagen y de cuánta calidad queramos perder. Este paso es la principal fuente de pérdida en los codificadores basados en la DCT [68].

La cuantificación se define como la división de cada coeficiente DCT por su correspondiente coeficiente de la matriz de cuantificación $Q(u, v)$, redondeado al entero más cercano:

$$F^Q(u, v) = INT\left(\frac{F(u, v)}{Q(u, v)}\right)$$

3.2.3 Codificación de coeficientes

La codificación de los nuevos coeficientes obtenidos tras la cuantificación se realiza de la siguiente manera: el coeficiente DC se almacena como la diferencia del término DC del bloque anterior en el orden de codificación. A continuación, los 64 coeficientes se ordenan siguiendo una secuencia de zigzag (como se muestra en la figura 3.5), de modo que los coeficientes de baja frecuencia se almacenan antes que los de alta frecuencia.

Una vez que los coeficientes han sido ordenados, se empaquetan o codifican utilizando la codificación aritmética de Huffman [68].

3.2.4 Reconstrucción

El proceso de reconstrucción es sencillo: los coeficientes son desempaquetados o decodificados (de la codificación Huffman) y reorganizados en una matriz 8×8. Deshacemos después la cuantificación (multiplicando por los respectivos valores de la matriz de cuantificación), y por último aplicamos la DCT-2D inversa a cada bloque 8×8 para intentar recuperar los valores originales, ya que los valores que recuperemos estarán más o menos cerca de los valores originales en función del número de coeficientes descartados en cada paso de la codificación.

Hay que destacar que los cálculos para dicha reconstrucción se realizan, como ya se ha comentado, con bloques de 8×8 píxeles debido al elevado tiempo de computación que se requiere para este método. La DCT-2D requiere un total de T^2 pasos por cada píxel procesado, siendo T el tamaño del bloque.

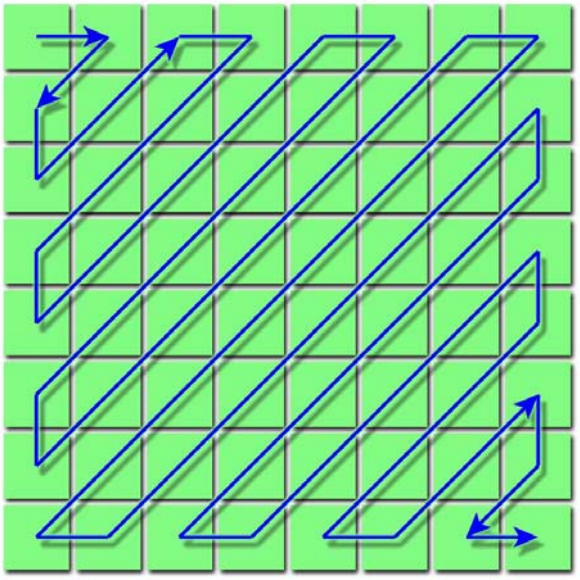


Figura 3.5: Ordenación de coeficientes en zigzag.

3.3 Transmisión progresiva utilizando la DCT

Para una transmisión progresiva de imágenes digitales basada en la DCT, los bloques de tamaño 8×8 deberán estar codificados en el mismo orden, pero a través de múltiples exploraciones de la imagen, con la incorporación en cada exploración de información adicional. Esto se puede hacer: empezando por el bit más significativo (MSB[3]) de cada coeficiente en la primera exploración y después añadir, de uno en uno o más bits en las siguientes exploraciones, hasta llegar al bit menos significativo (LSB); o codificando un subconjunto inicial de los 64 coeficientes, por ejemplo los diez primeros, y añadiendo más coeficientes en las próximas exploraciones.

3.3.1 Manejo de ROIs y su transmisión progresiva con la DCT

Como ya se comentó en la sección 2.6.2, el estándar JPEG no incluye un régimen específico para la codificación y decodificación de ROIs. Es por ello que proponemos entonces un nuevo enfoque para el manejo y posterior transmisión

[3]Se llama bit más significativo (More Significant Bit) al bit que tiene mayor valor dentro del conjunto; análogamente, se llama bit menos significativo (LSB: Less Significant Bit) al bit que tiene menor valor dentro del conjunto. En un byte, el bit más significativo es el de la posición 7, y el menos significativo es el de la posición 0.

de ROIs basado en la transmisión progresiva mediante la DCT y que será con el que llevaremos a cabo nuestro experimento según un estudio comparativo.

Después de la transmisión de un primer subconjunto de coeficientes DCT de cada bloque 8×8, hay que esperar hasta que el receptor, tras observar una primera reconstrucción (o varias), solicite más detalle de una determinada ROI. A continuación, se determinan todos aquellos bloques de tamaño 8×8 que incluyan o abarquen dicha ROI y entonces se envían, sólo para estos bloques, los coeficientes adicionales. Esto tiene varias ventajas: no hay necesidad de definir la ROI antes de la codificación de la imagen, es posible elegir cualquier ROI arbitrariamente o seleccionar varias ROIs para ser enviadas en un solo paso, y no hay necesidad de ningún tipo de cálculo adicional en el lado del servidor.

Por el contrario, la gran desventaja que tenemos es un aumento importante en el tiempo de los cálculos necesarios para realizar la reconstrucción por parte del cliente, ya que las DCTs inversas sobre los bloques de tamaño 8×8 deben ser calculadas una y otra vez mientras vaya llegando información adicional de la imagen o de la ROI.

3.4 Transmisión progresiva con JPEG2000

Vista la transformada wavelet (capítulo 2) y su aplicación en el estándar de compresión de imágenes JPEG2000, podemos añadir que además de lograr un mejor índice de compresión que el JPEG, el principal objetivo del JPEG2000 es la posibilidad de almacenar diferentes partes de la misma imagen con diferentes grados de calidad.

Esta mejor compresión proporciona logicamente una mejora, respecto del JPEG, en el proceso de la transmisión progresiva (tanto en la resolución de los píxeles, así como en su precisión) gracias a la organización de un código-flujo más eficiente y, permite gestionar mejor las ROIs debido a su código aleatorio en el acceso y capacidad de procesamiento de datos.

3.5 El SVD en la codificación adaptativa y la transmisión de ROIs

En esta sección presentamos una codificación para imágenes 2D utilizando SVD que permite la transmisión de varias ROIs en el mismo momento en que el receptor las selecciona.

En primer lugar, proponemos un algoritmo de codificación con pérdida mediante SVD.

3.5.1 Algoritmo para la codificación SVD con pérdida.

El algoritmo que presentamos a continuación, que utiliza SVD y permite la transmisión de ROIs de forma sencilla, se puede dividir en los siguientes pasos:

Algoritmo 3.1 (Codificación SVD con pérdida) *Los pasos a seguir para codificar una imagen 2D con pérdida y utilizando SVD son:*

1. *Aplica SVD a la imagen.*
2. *Define las regiones a transmitir como* $\{u, \sigma, v\}$ *donde* σ *es un valor singular y* $\{u, v\}$ *sus correspondientes vectores singulares.*
3. *Encuentra el primer valor singular* σ^* *tal que* $\sqrt{\frac{\sum_{\sigma_i > \sigma^*} \sigma_i^2}{\sum_{\sigma_i} \sigma_i^2}} \leq \epsilon$*, donde* ε *es un umbral prefijado con anterioridad [52].*
4. *Rechaza las regiones con valores singulares más pequeños que* σ^**.*
5. *Cuantifica las regiones no rechazadas, ordenándolas en un archivo codificado.*

Es importante destacar que la utilización de la *distancia relativa* entre dos imágenes [52] y la elección adecuada de un umbral ε (punto 3 del algoritmo) nos permite eliminar los valores singulares pequeños, ya que se supone que no proporcionan una mejora sustancial en la reconstrucción de imágenes.

Como regla práctica, para imágenes 2D podemos elegir $\varepsilon = 0.05$ y entonces la reconstrucción que obtenemos después de codificar la imagen original mediante SVD y con este umbral, será una aproximación aceptable (desde el punto de vista de nuestra percepción) de la imagen original [67].

3.5.2 Transmisión progresiva de ROIs

Una vez que la imagen está codificada, la transmisión de la ROI se basa en la siguiente propiedad de SVD: si

$$\{\sigma, (u_1, \ldots, u_m), (v_1, \ldots, v_n)\}$$

es un valor singular y los vectores singulares asociados de una imagen 2D de tamaño $m \times n$, el producto

$$\sigma (u_1, \ldots, u_m)^T (v_1, \ldots, v_n)$$

es una aproximación a la imagen 2D y entonces

$$\sigma (u_i, \ldots, u_{i+k})^T (v_j, \ldots, v_{j+l})$$

pasa a ser una aproximación de la ROI de tamaño $(k+1) \times (l+1)$ cuyas coordenadas de la esquina superior izquierda son (i, j) (véase figura 3.6).

Cuando el cliente detenga la transmisión progresiva para seleccionar las ROIs, se creará una matriz de control tanto en el servidor como en el cliente. Esta matriz de control tendrá en cuenta los valores y vectores singulares ya transmitidos para evitar así la redundancia en la transmisión de datos.

Cada agrupación de valores y vectores singulares $\{\sigma, u, v\}$ se transmite de forma ordenada según el valor singular (de mayor a menor) y la imagen va mejorándose cada vez que un nuevo grupo se recibe.

Por un lado, este sistema posee importantes ventajas:

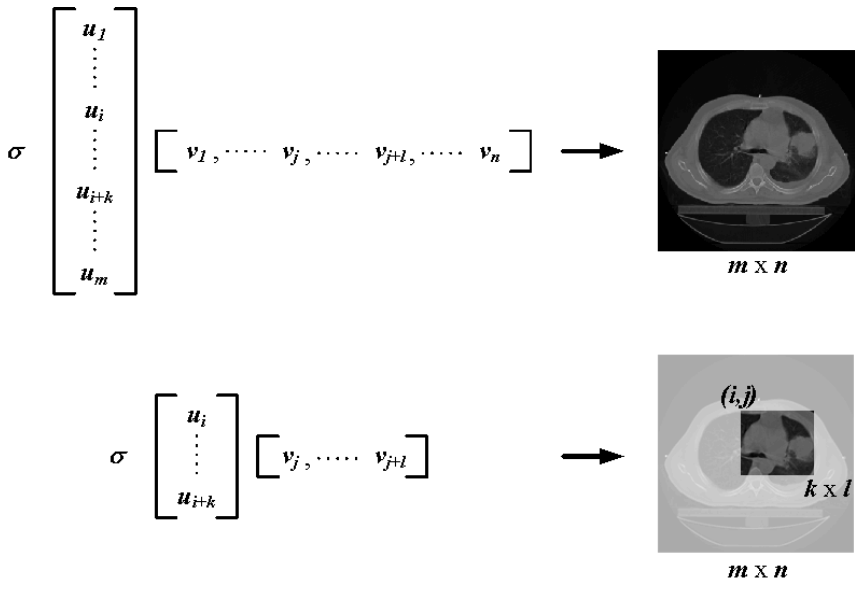


Figura 3.6: Reconstrucción de la ROI usando SVD.

- El algoritmo de reconstrucción es muy rápido y tiene una implementación muy sencilla $\{\sigma, u^T, v\}$.

- No hay redundancia en la transmisión de datos.

- Si es necesario, pueden ser seleccionadas varias ROIs (rectangulares) al mismo tiempo.

- Una vez que las ROIs son seleccionadas y sus coordenadas transmitidas al servidor, no es necesario volver a codificar la imagen original para transmitir progresivamente las citadas ROIs.

- Inmediatamente después de la transmisión de datos diferentes en cada una de las nuevas transmisiones, se mejora la calidad de las ROIs, y siempre sin repetir ningún dato.

Pero por otra parte hay que tener en cuenta una desventaja importante y es que la imagen codificada mediante SVD aumenta su tamaño respecto al tamaño original (entre un 150% y un 190%), por lo que se hace imprescindible la eliminación de los valores singulares pequeños y sus respectivos vectores singulares, así como cuantificar adecuadamente los restantes.

Figura 3.7: Selección de las ROIs *Mochila* y *Antena* de la imagen de referencia.

3.6 Experimentos

La figura 3.1 es una imagen en blanco y negro de 5000×3120 píxeles con 16 bits de niveles de gris y vamos a utilizarla como imagen de referencia.

3.6.1 Experimento I. SVD versus JPEG2000

Una vez tenemos la imagen, se eligen las siguientes subimágenes o ROIs (detalladas en la figura 3.7). Las ROIs elegidas las hemos llamado ROI *Antena* y ROI *Mochila*. Esta situación no es la habitual para un sistema de transmisión progresiva, ya que exige el conocimiento a priori de las ROIs elegidas, pero se ha adoptado así ya que es la única forma que admite el JPEG2000.

Preparativos del JPEG2000

El primer paso es la definición de una ROI (veáse figura 3.8) que contenga tanto a la ROI Antena como a la ROI Mochila y una vez que esta nueva ROI ha sido seleccionada, la imagen se codifica según el estándar JPEG2000 y utilizando el método MAXSHIFT (sección 2.6.3) para la ROI.

Preparativos del SVD

El segundo paso es la aplicación del algoritmo 3.5.1 (codificación SVD con pérdida) a toda la imagen.

Transmisión progresiva y reconstrucción de la imagen

Para ello comenzamos la transmisión progresiva de las ROIs utilizando ambos métodos, codificación SVD y codificación MAXSHIFT comenzando con un ratio de compresión de 0.01 bits/píxel (en total 19547 bytes) hasta el ratio

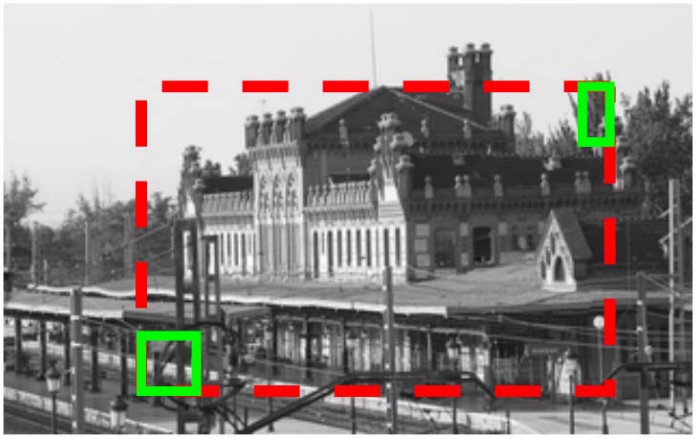


Figura 3.8: Selección de la ROI en la imagen de referencia para codificarla mediante el método MAXSHIFT.

0.78 (1521063 bytes) con incrementos en cada paso de la transmisión de 0.01 bits/píxel.

La calidad de la imagen se mide en cada reconstrucción con la medida PSNR[4] definida como:

$$PSNR = 10 \log_{10} \left(\frac{MAX_I^2}{MSE} \right)$$

Es necesario comentar que la codificación MAXSHIFT y la reconstrucción en cada etapa de la transmisión se han llevado a cabo con el software Kakadu 6.0 [76] y que reconstrucciones con un PSNR superior a 30 dB son consideradas como "de buena calidad" [56].

Resultados

Como podemos ver en la figura 3.9, SVD y JPEG2000 son comparables a ratios bajos (valores entre 0.1 y 1.0). Con una pequeña ventaja de JPEG2000 al principio, es un poco antes del ratio 0.4 cuando el método SVD supera al JPEG2000. Esto es debido a que en los primeros pasos de la transmisión el método SVD, con pocos valores singulares recibidos no consigue unas buenas reconstrucciones, aunque podemos observar como su mejora es significativa y su crecimiento lineal supera rapidamente al JPEG llegando a obtener unas reconstrucciones de "muy buena calidad".

Por otro lado a medida que se van recibiendo más datos, el JPEG2000 se ve superado por el SVD debido a que la ROI a reconstruir por JPEG2000 es bastante más grande que las de SVD (figura 3.10).

[4] Peak Signal-to-Noise Ratio.

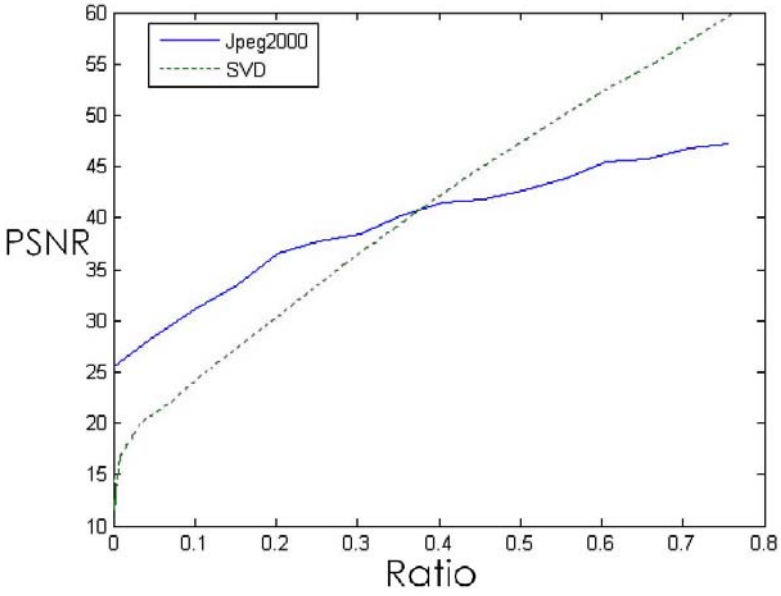


Figura 3.9:

(a) ROI JPEG 2000 (b) ROIs SVD

Figura 3.10: ROI del JPEG2000 y ROIs de la SVD, todas ellas a un ratio de 0.78 bits/píxel.

Después de esto y a medida que en la reconstrucción se utilizan muchos más datos (ratios mayores de 1.0) JPEG2000 vuelve a ofrecer un mejor PSNR que SVD.

3.6.2 Experimento II. SVD versus DCT

Este experimento describe una situación más habitual de lo que es realmente la transmisión progresiva. En este caso no tenemos un conocimiento previo de las ROIs que deben enviarse.

Punto de inicio

En un primer paso y después de utilizar la transformada DCT-2D enviamos el primer coeficiente o coeficiente DC, de cada bloque 8×8 en los que se ha dividido la imagen una vez codificada. Reconstruimos la totalidad de la imagen utilizando la DCT-2D inversa para cada bloque 8×8 (compuesto por un sólo coeficiente). La imagen así reconstruida tiene un PSNR del 22.9 dB (véase figura 3.11 (a)).

(a) PSNR = 22.9 dB (b) PSNR = 21.72 dB

Figura 3.11: (a) Reconstrucción DCT en el primer paso. (b) Reconstrucción SVD en el primer paso.

Para reproducir una situación similar con SVD, llevaremos a cabo una reconstrucción completa de la imagen utilizando el mismo número de bits que con el método DCT. La imagen así reconstruida tiene un PSNR del 21.72 dB (véase figura 3.11 (b)).

Siguientes pasos en la transmisión

Después de la primera reconstrucción, el cliente detiene la transmisión y la examina. Durante la observación puede que aparezca alguna región de interés para el cliente, entonces éste selecciona la ROI (en nuestro ejemplo ROI *Farol*) en la que está interesado y ambos métodos de transmisión se reanudan, comenzando el envío de los datos pertenecientes sólo a las ROIs.

- **Método DCT**: Se transmite un coeficiente de cada bloque de tamaño 8×8 de los que contienen o cubren a la ROI seleccionada. Después de

cada transmisión se mejora la ROI, se reconstruye mediante la DCT-2D inversa y se calcula el PSNR.

- **Metodo SVD**: Se transmiten, al mismo número de bits que cada transmisión realizada con el método DCT, grupos de valores singulares con sus respectivos vectores singulares $\{\sigma, u, v\}$. Se realiza la reconstrucción de la ROI mejorada, se calcula el PSNR y comparamos.

Mejora de la ROI

Para el método DCT repetimos la transmisión coeficiente a coeficiente (uno por bloque) hasta que los 64 coeficientes de cada bloque que conforman la ROI se hayan enviado (véase la figura 3.12). Al mismo tiempo y para mejorar la reconstrucción de la ROI mediante SVD se transmiten la misma cantidad de grupos de datos (vectores y valores singulares) al mismo número de bits de información que fueron enviados para el método DCT (figura 3.12).

Resultados

Como puede verse claramente en la figura 3.13, el método DCT tiene un mejor rendimiento que SVD. Sin embargo, debe tenerse en cuenta el efecto de píxelado en las reconstrucciones y que ocurre normalmente cuando se han transmitido pocos coeficientes DCT, y el conocido alto coste computacional y de tiempo que requiere el cálculo de la DCT-2D inversa para la reconstrucción de la imagen en cada paso.

3.7 Conclusiones

En este capítulo se ha intentado comparar los tres algoritmos propuestos sobre codificación adaptativa con pérdida de imágenes digitales 2D basado en SVD, DCT y JPEG2000. La codificación basada en SVD será utilizada más adelante para la transmisión progresiva de imágenes 3D, mediante diferentes algoritmos que iremos desarrollando en capítulos sucesivos.

Además, los métodos de codificación desarrollados nos han permitido la transmisión progresiva de una o varias ROIs a la vez, mediante simples cambios en los algoritmos de transmisión y reconstrucción, evitando así la re-codificación y redundancia en la transmisión de datos. Además, el cliente puede seleccionar las ROIs, en cualquier etapa durante el proceso de transmisión.

Hemos realizado también una comparación de SVD con JPEG y JPEG2000 con diferentes resultados, y las conclusiones obtenidas deben analizarse adecuadamente en el ámbito correspondiente y según las limitaciones de cada una de las estrategias expuestas aquí.

En la primera comparación, **JPEG2000 frente a SVD**, por limitaciones del método MAXSHIFT sólo ha sido posible elegir una ROI y ésta ha tenido que ser definida a priori, dado que el método MAXSHIFT no posee la posibilidad de seleccionar ROIs en el momento. Bajo esta circunstancia, hay que resaltar

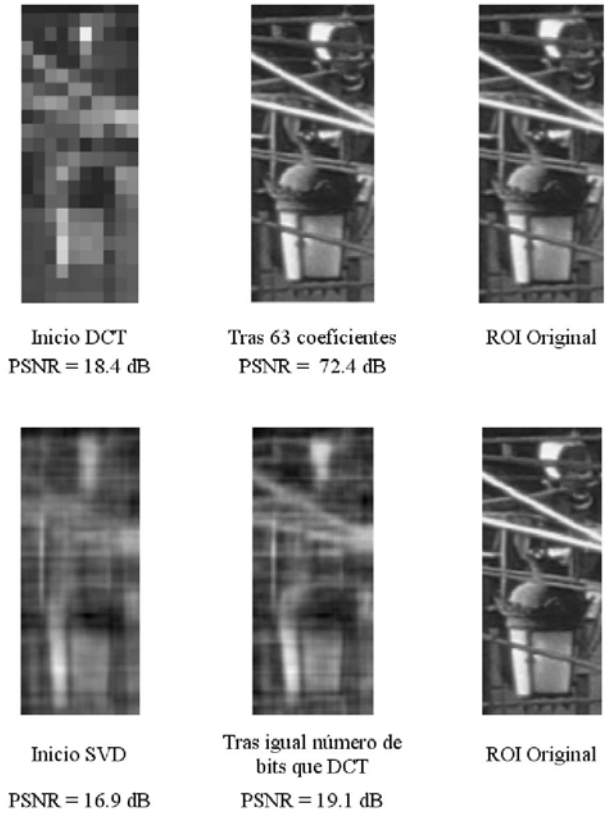


Figura 3.12: Comparación entre la ROI *Farol* de la imagen original, el primer paso de reconstrucción mediante DCT y SVD, y el final de la reconstrucción (con DCT y 63 coeficientes; mediante SVD al mismo número de bits que DCT).

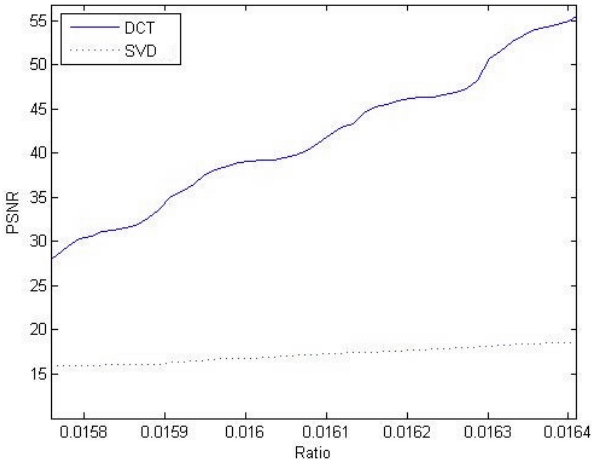


Figura 3.13: DCT versus SVD.

que JPEG2000 es el mejor método conocido de codificación para una imagen completa, pero por otro lado hay que comentar que si el cliente cambia de opinión y elige otra ROI, la imagen debe ser codificada de nuevo y la transmisión tiene que reiniciarse desde el principio. Redundancia en la transmisión y datos entrelazados son los principales inconvenientes del método MAXSHIFT en casos como éste.

Por otro lado, la codificación SVD incrementa notablemente la cantidad de datos. Una cuantificación adecuada de éstos podría salvar este inconveniente, pero a costa de perder calidad en la reconstrucción de la imagen. Y como ventajas del método SVD tenemos que se puede trabajar muy fácilmente con varias ROIs al mismo tiempo, los datos nunca son redundantes y la codificación de la imagen se realiza solamente una vez. Todo esto permite un ahorro importante en el tiempo de cálculo.

Por lo tanto, se puede concluir que ambos métodos son comparables especialmente en bajos ratios de compresión y transmisión, y esto es así porque con SVD hemos tomado dos ROIs bastantes separadas, obligando al JPEG2000 a tomar una ROI "muy grande". Luego, el método propuesto mediante SVD funciona mejor cuanto más pequeñas y separadas estén las ROIs que hayan sido seleccionadas y con él evitamos también la recodificación y la transmisión redundante de datos.

En la segunda comparación, **SVD versus JPEG**, hemos elegido un esquema especial que gira en torno a la transformada DCT-2D a la hora de utilizar el JPEG, ya que este estándar no admite la manipulación de ROIs. Usando el sistema propuesto, hemos descubierto que DCT es un buen método para la transmisión progresiva de ROIs. En comparación con el método original

JPEG, la codificación de la imagen utilizando solamente la DCT-2D aumenta el tamaño final de la imagen codificada. Además el efecto píxelado se observa en los primeros pasos de la transmisión y como principal inconveniente tenemos que, dado que la reconstrucción se lleva a cabo con la DCT-2D inversa en cada paso, existe un alto coste computacional y de tiempo de cálculo.

Por otra parte, con la SVD y en los primeros pasos, las reconstrucciones de las ROIs son "más suaves" que las que corresponden a la DCT, aunque el método DCT mejora considerablemente cuando los datos de la imagen aumentan. Podemos concluir este capítulo afirmando que el método DCT es mejor que el método SVD.

Capítulo 4

Estructura general de la transmisión progresiva de imágenes digitales 3D

4.1 Introducción

Nuestro objetivo en este capítulo es exponer el proceso general sobre la transmisión progresiva de imágenes digitales 3D, usando SVD y la interpolación matricial lineal por trozos de Newton. En general, el escenario en el cual nos encontramos normalmente en una transmisión progresiva (capítulo 1) es: la existencia de un remitente que posee los datos almacenados de una manera conveniente y está listo para enviarlos, un receptor que solicita una imagen dada y entonces realiza una petición al remitente, un canal de transmisión y la imagen que es enviada a través del canal de transmisión.

Así pues, y en primer lugar, esta imagen debe ser descompuesta en distintas regiones o trozos para su posterior envío o transmisión. Dicha descomposición puede ser directa −en el sentido de tomar regiones de datos (subimágenes) disjuntos−, o puede estar basada en transformadas de la imagen. Este primer paso que consideraremos en el proceso general a seguir sobre la transmisión progresiva se llama **descomposición** o **segmentación**.

También hay que elegir un esquema de **reconstrucción** como parte del proceso de la transmisión progresiva. Tal esquema debe ser apropiado para el receptor, sobre todo, en términos de calidad de imágenes y tiempo de cálculo.

Para medir la *diferencia* entre la imagen original y la reconstruida, es decir, la calidad de reconstrucción, utilizaremos algunos métodos comparativos basados en diferentes normas matriciales y que comentaremos de forma más detallada más adelante. Es decir, será necesario definir una medida que nos permita saber la calidad de reconstrucción de la imagen. La elección de una u otra medida nos determinará la forma de cómo transmitir la información en cada etapa. Elegir

qué región es más relevante en lo que a datos se refiere, para que sea enviada en primer lugar por el emisor y que el receptor pueda realizar con ella una reconstrucción de la mayor calidad posible, e iniciar así una secuencia óptima de transmisión es lo que podríamos llamar una **estrategia adaptativa** a la hora de seleccionar la siguiente región a transmitir.

4.2 Una transmisión progresiva basada en SVD y en la interpolación matricial lineal por trozos de Newton para imágenes 3D

De acuerdo con los pasos comentados en la sección anterior, vamos a desarrollar ahora un proceso para la transmisión progresiva de imágenes digitales 3D usando SVD.

4.2.1 Descomposición

El algoritmo para la descomposición que presentamos a continuación está basado en la siguiente idea: si O es una imagen 3D de tamaño $r \times s \times t$, ésta puede ser descompuesta en r cortes paralelos 2D, formando el conjunto de datos 3D siguiente:

$$\{O_1, O_2, \ldots, O_r\}$$

donde O_i es la imagen 2D correspondiente al corte de orden i. Entonces se puede aplicar SVD a cada corte, con un umbral prefijado de antemano, de tal modo que podamos obtener una descomposición en regiones de cada corte.

Algoritmo 4.1 (Descomposición de una imagen 3D) *Sea O una imagen 3D de tamaño $r \times s \times t$ y sea $\varepsilon \geq 0$ un umbral prefijado.*

1. *Dividimos O en r cortes paralelos 2D (matrices) O_i, $i = 1, 2, \ldots, r$, de tamaño $s \times t$.*

2. *Aplicamos SVD según lo visto en el teorema 2.3 a cada corte O_i, con $i = 1, 2, \ldots, r$. Obtenidos todos los valores singulares, elegimos el índice q_i de tal forma que*

$$\sigma_1^i \geq \cdots \geq \sigma_{q_i}^i \geq \varepsilon > \sigma_{q_i+1}^i \geq \cdots \geq \sigma_p^i \geq 0, \qquad p = \min\{s, t\},$$

   y consideramos los correspondientes vectores singulares

$$\{u_1^i, \ldots, u_{q_i}^i\}, \{v_1^i, \ldots, v_{q_i}^i\}.$$

3. *Definimos la **región** (i, j) como*

$$reg(i, j) = \{u_j^i, \sigma_j^i, v_j^i\}, \ i = 1, 2, \ldots, r, \ j = 1, \ldots, q_i.$$

Como se puede observar, según lo indicado en la definición de región, las regiones definidas en el algoritmo anterior son muy diferentes del concepto más general que tenemos de subconjunto de una imagen dada.

Ahora bien, tengamos en cuenta dos consideraciones: primero, que comenzando con una imagen 3D de tamaño $r \times s \times t$, el algoritmo de descomposición genera $\sum_{i=1}^{r} q_i$ regiones con $s + t + 1$ datos cada una, es decir, un total de

$$\sum_{i=1}^{r} q_i(s + t + 1), \tag{4.1}$$

datos, y en el caso más desfavorable donde $\varepsilon = 0$ ($q_i = p$ para todo i) y $p = s = t$ (cuando las matrices son cuadradas), la expresión (4.1) se transforma en $2(r \times s \times t) + s \times t$, cantidad ésta, que es más del doble que datos originales tenemos.

En segundo lugar, la elección del umbral dependerá del uso de SVD o del tipo de matrices. Un umbral más o menos pequeño puede implicar la eliminación (o no) de una gran cantidad de valores singulares. Por lo tanto, dependiendo del número de valores singulares eliminados, el número de regiones se reducirá, y así también la cantidad de datos generados por el algoritmo 4.1. Esto nos llevaría a considerar un método de transmisión con pérdida donde la información eliminada no es transmitida y por tanto, la imagen reconstruida por el receptor no será identica a la imagen original.

4.2.2 Reconstrucción

El algoritmo de reconstrucción para la transmisión progresiva $\mathcal{R}$ seleccionado aquí se basa en el uso de SVD y en la interpolación matricial lineal por trozos de Newton, tal y como se desarrolló en [29].

Algoritmo 4.2 (Algoritmo de reconstrucción $\mathcal{R}$) *Sea O una imagen 3D de tamaño $r \times s \times t$ y $reg(i,j) = \{u_j^i, \sigma_j^i, v_j^i\}$, $i = 1, 2, \ldots, r$, $j = 1, \ldots, q_i$ las regiones de O obtenidas del algoritmo* 4.1 *con un umbral dado $\varepsilon \geq 0$. Y supongamos que la región $reg(i^*, j^*)$ acaba de ser transmitida.*

1. *Calculamos una aproximación mejor a la anterior aproximación O_i^* de la siguiente forma*

$$\left[\sigma_1^i u_1^i (v_1^i)^T + \cdots + \sigma_{j^*-1}^i u_{j^*-1}^i (v_{j^*-1}^i)^T\right] + \sigma_{j^*}^i u_{j^*}^i (v_{j^*}^i)^T,$$

   donde las primeras $j^* - 1$ regiones fueron ya transmitidas en etapas anteriores (veáse el algoritmo 2.7*).*

2. *Sean $O_{i_1}^*, O_{i_2}^*, \ldots, O_{i_k}^*$, $i_1 < \ldots < i_k$, las aproximaciones a los cortes obtenidas en etapas anteriores y en el PASO* 1*.*

3. *Si $\{i_1, \ldots, i_k\} = \{1, \ldots, r\}$ la interpolación no es necesaria. IR AL PASO* 6*.*

4. *Se construye un polinomio matricial lineal interpolador de Newton de la siguiente manera:*

$$\begin{aligned} P_j(x) &= O^*_{i_j} + \frac{O^*_{i_{j+1}} - O^*_{i_j}}{i_{j+1} - i_j}(x - i_j), \\ j &= 1, 2, \ldots, k-1, \end{aligned} \tag{4.2}$$

y construimos también el polinomio matricial por trozos de Newton en este paso de la transmisión y para el intervalo j-ésimo como:

$$P(x) = P_j(x) \quad si \quad x \in [i_j, i_{j+1}].$$

5. *Calculamos $P(i), i = 1, \ldots, r$, y se obtiene un sistema completo de r cortes, k de ellos serán los datos originales $O^*_{i_1}, O^*_{i_2}, \ldots, O^*_{i_k}$, y el resto, $r-k$, están generados por el polinomio matricial $P(x)$.*

6. *FIN*

Está claro que el polinomio matricial por trozos de Newton $P(x)$ (paso 4) cambia en cada nueva iteración, pero solamente en los "trozos" donde se inserta una región nueva, tomada de la última transmisión. Así pues, al trabajar sólo con los cortes donde ha habido cambios respecto de la transmisión anterior, los pasos 1, 2, 4 y 5 del algoritmo anterior se verán reducidos en lo que respecta a tiempos de cálculo.

4.2.3 Estrategia adaptativa para seleccionar la siguiente región a transmitir

La estrategia para seleccionar la siguiente región a transmitir es una decisión muy importante para asegurar la eficacia en la transmisión y en el algoritmo de reconstrucción. Un diseño eficaz de esta estrategia dependerá de qué se esté buscando. En el algoritmo que propondremos más adelante, vamos a exponer un proceso para la transmisión progresiva basada en la descomposición de las imágenes 3D según lo visto en el algoritmo 4.1 de descomposición (cortes 2D y SVD) y el algoritmo 4.2 de reconstrucción (SVD e interpolación matricial lineal por trozos de Newton), donde se incluye en unos pasos claves, una estrategia adaptativa para seleccionar la siguiente región a transmitir.

La idea principal de esta estrategia adaptativa es transmitir las regiones relacionadas con los cortes peor aproximados para así mejorar las siguientes reconstrucciones. Obsérvese que la orden de transmisión de cada región por parte del servidor no está prefijada, y se construye en cada paso, dependiendo de los detalles internos de cada imagen transmitida progresivamente y sobre todo, de la diferencia entre la imagen original y la reconstruida.

Las diferentes opciones que se han utilizado en la literatura para medir esta diferencia incluyen desde medidas tan distintas como: señal de la entropía o el PSNR [34], hasta variables estadísticas (mediana, desviación típica, varianza o

el error cuadrático medio) o incluso algunas de las diferentes normas que existen para vectores o matrices [33].

Nosotros en principio vamos a utilizar la norma Frobenius (véase la expresión 2.2). Esta elección puede parecer extraña al principio (elegir norma Frobenius en vez de norma-2) considerando sobre todo que SVD y la norma-2 tienen en común propiedades interesantes (véase el capítulo 2), pero la norma Frobenius es más fácil y más rápida de calcular y la expresión (2.4) nos dice que ambas se comportan del mismo modo. Una de sus principales características es que no depende del tamaño de la imagen, y además la norma Frobenius se ha utilizado satisfactoriamente en artículos recientes [28], [29] donde la interpolación matricial de Newton forma parte del proceso de reconstrucción. Según esta consideración, a la hora de medir la bondad de la reconstrucción, podemos encontrar cortes 2D reconstruidos integramente por interpolación, sin el envío de ninguna región (y por tanto ningún valor singular) de dicho corte. En ese momento, calcular la diferencia entre la imagen original y la reconstruida utilizando el valor singular asociado a la región envíada (es decir calcular la norma-2) no tiene sentido (véase la expresión 2.6).

Definimos entonces una medida μ que permite analizar la bondad de las aproximaciones como:

Definición 4.3 *Sea O una imagen 3D de tamaño $r \times s \times t$, descompuesta en r cortes paralelos O_i, $i = 1, 2, \ldots, r$, de tamaño $s \times t$, entonces*

$$\mu(O) = \sum_{i=1}^{r} \|O_i\|_F . \tag{4.3}$$

Utilizando entonces esta medida μ basada en la norma Frobenius, el algoritmo de reconstrucción $\mathcal{R}$ calcula nuevas aproximaciones de la imagen original a partir de los datos de las regiones 3D transmitidas. Para ello será necesario extraer y comparar los cortes de esta aproximación 3D y es entonces cuando el operador matricial determinado simplemente a través de la multiplicación de matrices, elemento a elemento, desempeña un papel básico en dicha comparación.

Definición 4.4 *Sean $A = (a_{ijk})$ y $B = (b_{ijk})$ dos matrices de tamaño $r \times s \times t$, entonces $A \otimes B$ es la matriz de tamaño $r \times s \times t$ definida como*

$$A \otimes B = (a_{ijk} * b_{ijk}) .$$

Definimos ahora una nueva matriz de tamaño $r \times s \times t$, llamada matriz compañera. La matriz compañera nos proporcionará la herramienta para determinar las estrategias óptimas en la transmisión progresiva.

Definición 4.5 *Si $A = (a_{ijk})$ es una matriz de tamaño $r \times s \times t$, entonces la matriz compañera de A, $\delta(A)$, es la matriz $B = (b_{ijk})$ de tamaño $r \times s \times t$ cumpliendo que*

$$b_{ijk} = \begin{cases} 0, & \text{si } a_{ijk} = 0 \\ 1, & \text{en otro caso} \end{cases} \tag{4.4}$$

para todo i, j, k.

Es importante observar entonces que si P es una reconstrucción de O (es decir, una cierta matriz que aproxima a la matriz O) entonces

$$\mu\left(\delta\left(O_a\right)\otimes O-\delta\left(O_a\right)\otimes P\right),$$

nos da estimaciones de cómo de bien P aproxima a O en el corte identificado como $\delta\left(O_a\right)$. Es decir, esta expresión proporciona una estimación numérica de cuán cercano está el corte de la reconstrucción $\delta\left(O_a\right)\otimes P$, del corte correspondiente $\delta\left(O_a\right)\otimes O$, de la imagen original O.

4.2.4 Algoritmo sobre la transmisión progresiva con una estrategia adaptativa

En primer lugar, hay que comentar que el algoritmo comienza enviando primero la primera región del primer corte $reg\left(1,1\right)$ y la primera región del último corte $reg\left(r,1\right)$, de ahí que para comenzar utilizamos información de dos de las regiones de los "cortes límite" primero y último. Esto es debido al uso de la interpolación a la hora de reconstruir, necesitando de los dos "cortes límite" para iniciar dicha interpolación. Hay que destacar también la pobre aproximación o reconstrucción que obtendremos inicialmente con el uso de la interpolación sobre los datos que corresponden a los dos cortes iniciales.

Algoritmo 4.6 (Transmisión progresiva con estrategia adaptativa) *Sea O el conjunto de datos 3D de tamaño $r\times s\times t$, $S=\{O_i\}_{i=1}^{r}$, con r cortes 2D paralelos de tamaño $s\times t$, y sea la descomposición en regiones*

$$\left\{reg\left(i,j\right),\quad i=1,2,\ldots,r,\quad j=1,2,\ldots,q_i\right\},$$

obtenida por el algoritmo 4.1. *Sea $\mathcal{R}$ la reconstruction según el algoritmo* 4.2*, y considerando la medida μ definida por la expresión* (4.3).

1. *Sea $\tilde{N}=\sum_{i=1}^{r}q_i$ el máximo número posible de transmisiones.*
2. *Sea $T_1=\left\{reg\left(1,1\right),reg\left(r,1\right)\right\}$ (las 2 primeras regiones transmitidas) y*

   $$U_1=\left\{reg\left(i,j\right),\quad i=1,2,\ldots,r,\quad j=1,2,\ldots,q_i\right\}\sim T_1,$$

   (las regiones no transmitidas).
3. *Sea $V_O=\left(\left(1,1\right),\left(r,1\right)\right)$ (el vector donde se guarda el orden de transmisión de las regiones).*
4. *FOR $k=1,2,3,\ldots,\tilde{N}$ DO*

   (a) *Encuentra un corte O_{i^*} de S para el cual*

   $$\max_{O_a\in S}\mu\left(\delta\left(O_a\right)\otimes\mathcal{R}\left(T_k\right)-\delta\left(O_a\right)\otimes O\right),$$

   se alcance.

(b) *Selecciona para la transmisión la región* $reg\,(i^*, j^*)$, *la primera región no transmitida del corte* O_{i^*}.

(c) *Haz* $T_{k+1} = T_k \cup \{reg\,(i^*, j^*)\}$ (*se transmite la* $reg\,(i^*, j^*)$).

(d) *Haz* $U_{k+1} = U_k \sim \{reg\,(i^*, j^*)\}$ (*eliminar la región transmitida del conjunto de regiones no transmitidas*).

(e) *Añade* (i^*, j^*) *al final de* V_O.

(f) *Calcula* $\mathcal{R}_O^k = \mathcal{R}\,(T_{k+1})$ *como la reconstrucción aproximada de* O *en la etapa* k.

5. *FIN DO*

Aunque en el anterior paso 1 consideremos $\tilde{N}$ como el máximo número posible de transmisiones, no es necesariamente ésta una cantidad prefijada ni tiene porque ser alcanzada, dado que el proceso de la transmisión progresiva puede ser interrumpido antes de que se hayan recibido todos los datos. La estrategia adaptativa se define en los pasos 4(a) y 4(b) del algoritmo 4.6. Obsérvese que esta estrategia adaptativa implica que la región $reg\,(i, j)$ no se puede transmitir si la región $reg\,(i, j-1)$ no se ha transmitido ya. Esta restricción permite un diseño sencillo de la estrategia y nos evita una gran cantidad de tiempo de cálculo. La reconstrucción progresiva de la imagen se hace en el paso 4(f).

4.3 Experimento y resultados

Después de desarrollar de forma teórica la descomposición, reconstrucción y transmisión adaptativa para imágenes digitales 3D, nuestra intención ahora es ilustrar y discutir este proceso para la transmisión progresiva de imágenes 3D usando como objeto de estudio y patrón de referencia, un conjunto de datos obtenidos del TAC de una cabeza humana.

Vamos a considerar el conjunto O de datos 3D de la figura 4.1, la cual es una reconstrucción, usando VTK[1] [57], [58] y [78] del TAC de una cabeza humana y que consiste en un conjunto de 93 cortes paralelos. Cada corte es una imagen 2D (de tamaño 256×256 píxeles) del TAC, en blanco y negro; más exactamente en niveles de gris y con una profundidad de gris en cada píxel de 16 bits.

Nuestro experimento consistirá, en primer lugar en descomponer el conjunto O de datos 3D en diferentes regiones utilizando SVD, según se ha descrito en el algoritmo 4.1. Una vez segmentado este conjunto de imágenes 3D, comenzaremos la transmisión de las primeras regiones del primer y último corte 2D para reconstruir, utilizando la interpolación, una primera aproximación al TAC. Tras esto y después de calcular la bondad de dicha aproximación, iremos transmitiendo las diferentes regiones elegidas según el paso 4 del algoritmo 4.6 y que según la medida utilizada son las que reconstruyen mejor nuestro TAC una vez

[1] Visualización Toolkit. Herramienta de visualización, representación 3D y procesamiento de imágenes implementada en C++ y de código o núcleo abierto.

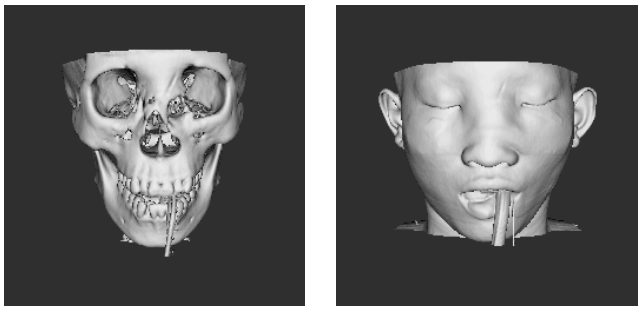


Figura 4.1: Resultados a nivel de hueso y de piel usando VTK, de todo el conjunto de datos de los 93 cortes del TAC.

transmitidas. Y así procederemos hasta que el nivel de reconstrucción sea el satisfactorio para el receptor.

El experimento ha sido ejecutado en código de MATLAB 6.5.1, en un PC Pentium IV, a 2.4 Ghz, 512 Mb de RAM, y bajo Windows XP.

4.3.1 Descomposición del TAC en regiones y número de datos

Según lo comentado, para la descomposición de nuestro TAC, aplicaremos el algoritmo 4.1 a O, con un umbral $\epsilon = 10^{-9}$. En primer lugar obtenemos de la imagen 3D cada uno de sus 93 cortes 2D $\{O_i,\ i = 1, 2, \ldots, 93\}$, siendo cada uno de ellos una matriz de tamaño 256×256. A cada una de ellas se le aplica entonces el SVD (paso 2 del algoritmo 4.1). El tiempo de cálculo empleado por la CPU para la descomposición de estos 93 cortes utilizando SVD es de unos 49 segundos. Después del paso 3, las regiones obtenidas son:

$$\begin{array}{|c|c|c|c|} \hline O_1 & O_2 & \cdots & O_{93} \\ \hline reg\,(1,1) & reg\,(2,1) & & reg\,(93,1) \\ reg\,(1,2) & reg\,(2,2) & & reg\,(93,2) \\ \vdots & \vdots & & \vdots \\ reg\,(1,q_1) & reg\,(2,q_2) & & reg\,(93,q_{93}) \\ \hline \end{array} \tag{4.5}$$

donde en este experimento, $q_1 = q_2 = \ldots = q_{93} = 240$.

Esto significa que mientras los datos originales consistían en

$$93 \times 256 \times 256 = 6094848 \text{ valores},$$

después del algoritmo 4.1, y dado que cada región posee $256 + 1 + 256 = 513$ valores, el número de datos a transmitir pasa a ser de

$$93 \times 240 \times (256 + 1 + 256) = 11450160 \text{ valores},$$

o lo que es lo mismo

$$93 \times 240 = 22320 \text{ regiones a transmitir.}$$

Este gran número de datos o valores[2] representan un incremento del 187.86% sobre el tamaño original de la imagen 3D. Por de pronto, el umbral $\epsilon = 10^{-9}$ nos está ahorrando en la transmisión 16 regiones por cada corte, que expresado en forma de valores hacen un total de

$$93 \times 16 \times 513 = 763344 \text{ valores.}$$

Para los cálculos y gráficos presentados en todo el capítulo, consideraremos la cantidad anterior de datos originales (6094848) como el 100% de los datos (aproximadamente 11881 regiones a transmitir). Por lo tanto, en los gráficos, el eje del porcentaje (eje de abscisas) mostrará el intervalo $[0, 187.86]$ y así será más fácil comparar la bondad de la reconstrucción y la cantidad de datos transmitidos.

4.3.2 Transmisión, reconstrucción, medida del error y selección de la siguiente región a transmitir

La transmisión progresiva de regiones comienza como indica el algoritmo 4.6, mediante la transmisión de las primeras regiones del primer y último corte, $reg(1, 1)$ y $reg(93, 1)$. Tras esta transmisión construimos la primera aproximación a la imagen 3D, reconstruyendo los demás cortes intermedios (del 2 al 92) interpolando. Una vez construida esta primera aproximación, hallamos el peor corte reconstruido (mediante nuestra medida μ introducida anteriormente) y de este corte, seleccionamos la siguiente región no transmitida para la nueva transmisión.

La idea principal de esta estrategia adaptativa es transmitir las regiones relacionadas con los cortes peor aproximados para mejorarlas. Y este proceso continúa: transmitiendo regiones, reconstruyendo por interpolación, midiendo la calidad de la imagen reconstruida y seleccionando la siguiente región a transmitir, todo ello con dos importantes advertencias.

La primera es que mientras que hay interpolación, ésta se realiza sólo en la *horquilla* entre los dos cortes más próximos, uno por exceso y otro por defecto, ya transmitidos; y segundo, que esta interpolación termina en el momento en que se hayan transmitido al menos una región de todos los cortes, es decir, que mientras de algún corte no se haya envíado la primera región, el proceso de interpolación no habrá terminado definitivamente.

Para ilustrar los resultados obtenidos, presentamos las reproducciones 3D usando VTK a nivel de hueso (figura 4.2) y a nivel de piel (figura 4.3).

Podemos observar cómo a pesar del aumento de los datos debido a SVD, conseguimos reconstrucciones de buena calidad para porcentajes bajos en la

[2] En este primer experimento con imágenes 3D vamos a considerar como valor a todo aquel dato individual que se transmita, independientemente del tamaño en bits de éste. Será en otros capítulos donde cuantificaremos en bits cada valor transmitido.

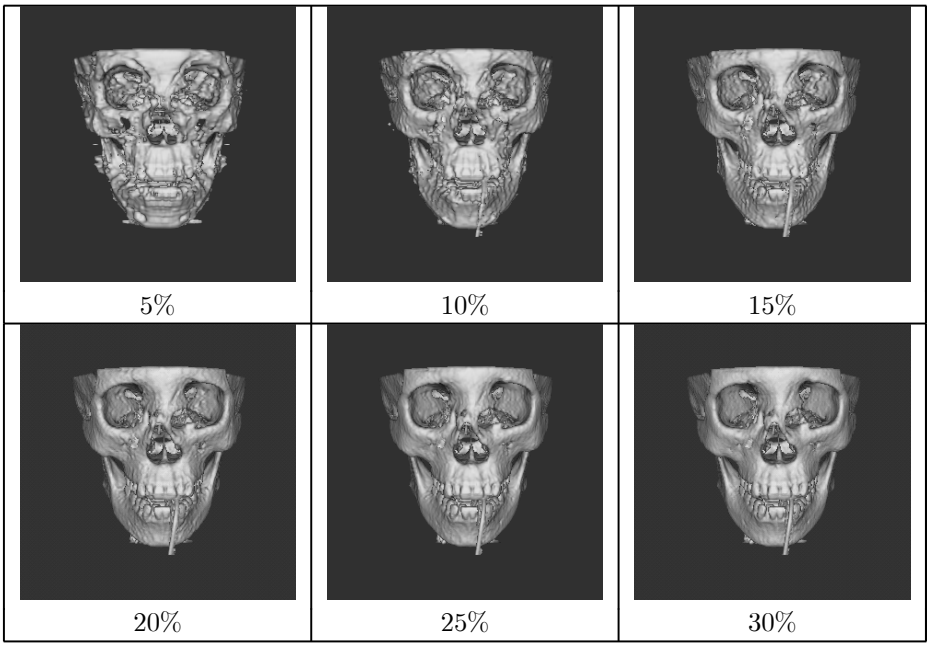


Figura 4.2: Representaciones 3D usando VTK a nivel de hueso y según diferentes porcentajes de datos transmitidos.

transmisión de datos. Por ejemplo, la imagen al "15% de los datos" de la figura 4.2 se parece visualmente, a la representación original a nivel del hueso de la figura 4.1.

Las gráficas sobre el error cometido en las reconstrucciones usando la medida μ se presentan en las figuras 4.4 y 4.5. En la figura 4.4 encontramos la evolución del error cuando se transmiten todos los datos. Es interesante observar la velocidad con que disminuye el error entre el principio de la transmisión y el 30% de datos transmitidos.

En la figura 4.5 (A), podemos observar un zoom de la gráfica sobre la evolución del error al principio (del 0% al 25% de datos transmitidos), y en la figura 4.5 (B) un zoom del final (del 100% a 187.86% de datos transmitidos). Podemos apreciar también en la figura 4.5 (B) cómo la velocidad de decrecimiento de la magnitud del error es pequeña si la comparamos con la correspondiente al intervalo del 0% hasta el 75% de datos transmitidos.

Esto nos permite afirmar que mientras las sucesivas reconstrucciones del principio mejoran notablemente las anteriores visualizaciones del TAC (el error disminuye de forma muy rápida entre envío y envío), llega un momento que por más transmisiones que realicemos, la mejora de la imagen (se observa visualmente y se confirma con la evolución del error) es prácticamente nula, muy pequeña. Es decir, llega un momento en que tenemos una reproducción, con pérdida, del TAC original bastante aceptable y aunque tratemos de mejorarla

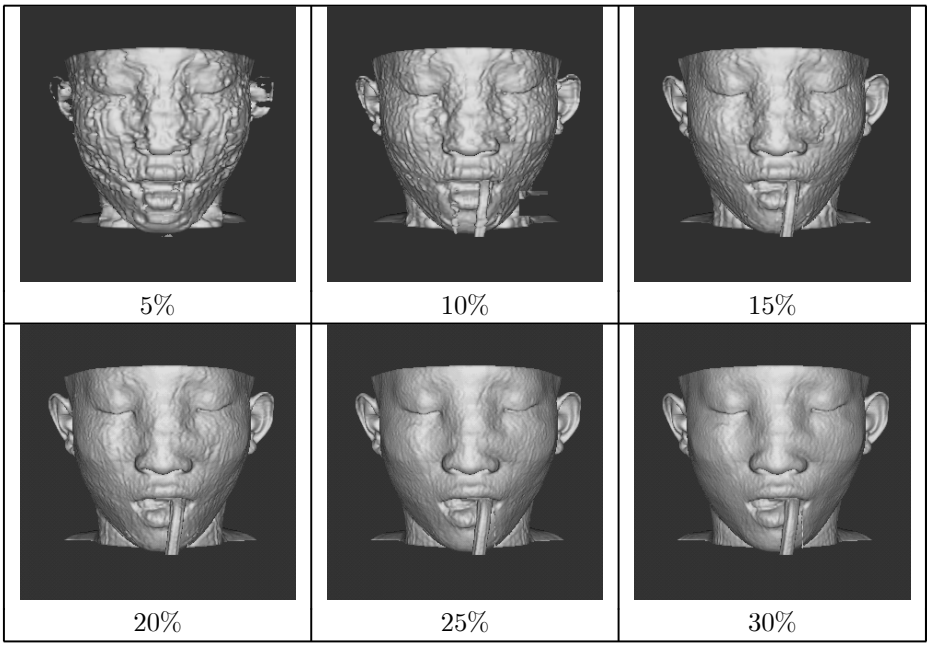


Figura 4.3: Representaciones 3D usando VTK a nivel de piel y según diferentes porcentajes de datos transmitidos.

y *acercarla* al original, es superior el coste de la transmisión que la mejora que se alcanza.

4.3.3 Tiempo de cálculo e información más relevante

La evolución del tiempo acumulado de la CPU se muestra en el figura 4.6. Obsérvese que el aumento de éste es de orden lineal y aunque el tiempo total es aproximadamente de 4 horas y media, hay que observar que el algoritmo de reconstrucción hace una nueva reconstrucción cada vez que se recibe una nueva región de las 22319 que tenemos.

Con la transmisión del 30% de los datos (la reconstrucción es muy parecida al original visualmente) el tiempo de CPU invertido en el proceso es de 2722.44 segundos (aproximadamente 45 minutos). Esto es debido principalmente a que es necesario realizar una nueva reconstrucción en cada etapa de la transmisión para conocer el corte peor reconstruido. Cada reconstrucción se realiza en 0.77 segundos aproximadamente.

Otro análisis interesante es conocer dónde se localiza la información más relevante de la imagen, es decir, la primera información que se transmite. Para descubrirlo, se presenta en las figuras 4.7, 4.8 y 4.9 la frecuencia de las regiones transmitidas pertenecientes a cada corte en el 1%, el 5% y el 10% de datos transmitidos. Los gráficos demuestran que la información más relevante está

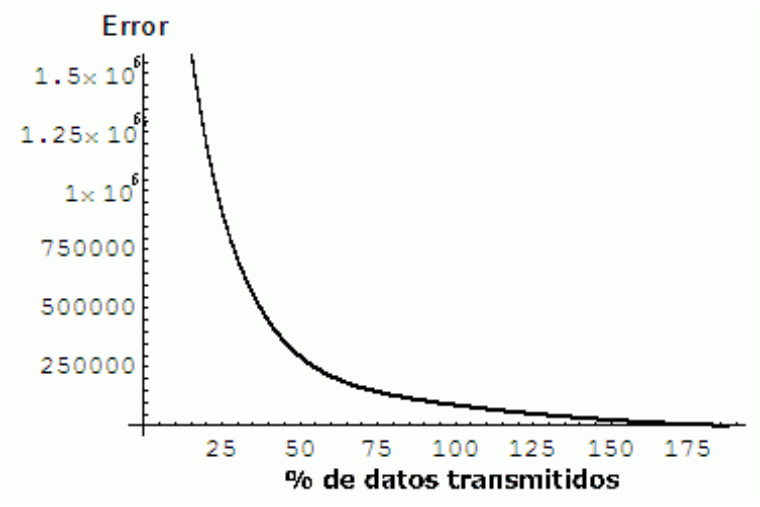


Figura 4.4: Evolución gráfica del error. Éste se calcula usando la medida μ definida en la expresión (4.3) .

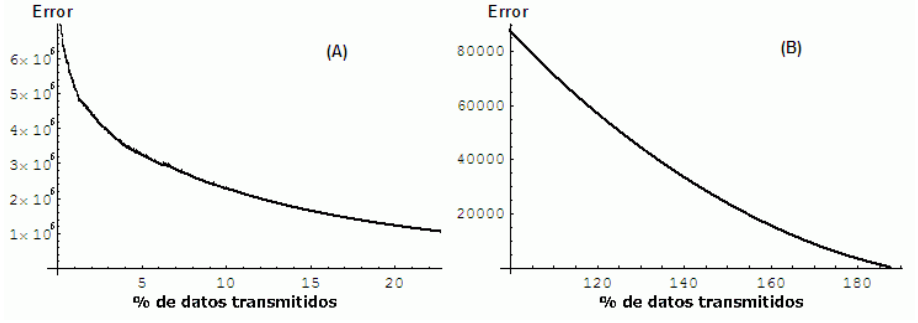


Figura 4.5: Estos gráficos representan el principio y la parte final de la figura 4.4. Puede observarse el límite superior del error de la aproximación después de enviar $reg\,(1,1)$ y $reg\,(93,1)$, y el error cometido después de la transmisión del 100% de los datos.

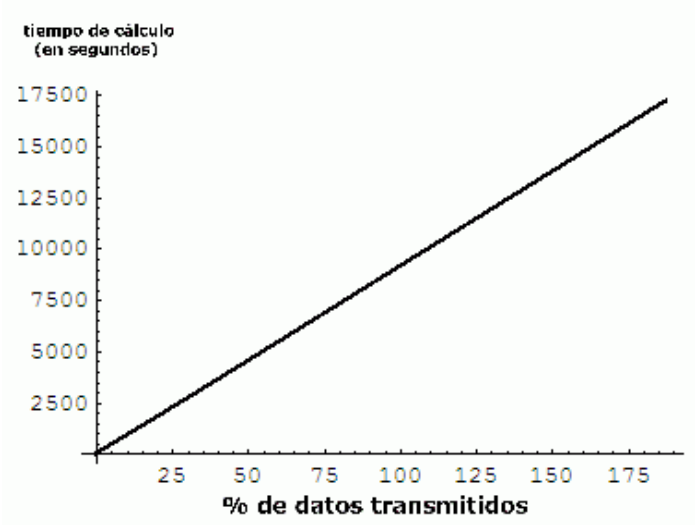


Figura 4.6: Evolución del tiempo acumulado de CPU (en segundos), aproximadamente 4 horas y media, para el proceso completo de reconstrucción. Obsérvese que el algoritmo de reconstrucción realiza una nueva aproximación en cada una de las 22319 transmisiones (la primera transmisión es de dos regiones).

situada entre los cortes 7 y 31, que corresponde a la la zona *antifaz* de la cabeza, entre las cejas y la nariz. En la figura 4.9 se observa cómo ninguna región del corte 75 ha sido aún transmitida. Esto significa que el corte 75 es probablemente el corte con la información menos relevante, y la interpolación puede aproximar muy bien dicho corte con sólo la información proporcionada por cortes vecinos. No es hasta el envío 1189 cuando se realiza el primer envío de dicho corte, y son entonces 5 consecutivas el número de regiones que se transmiten, desde la región $reg(75, 1)$ hasta la región $reg(75, 5)$.

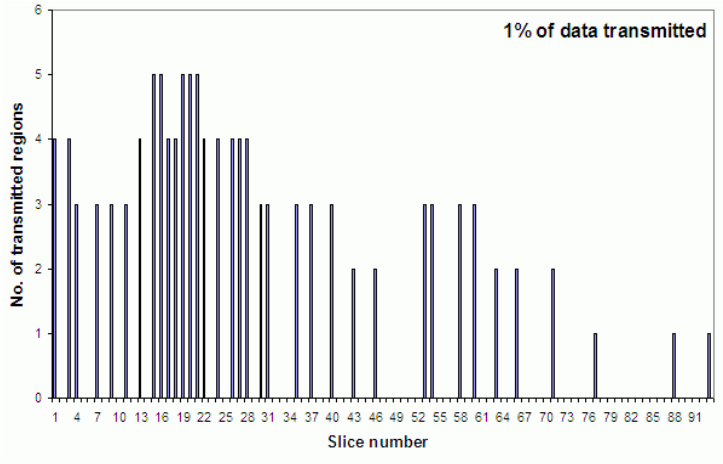


Figura 4.7: Frecuencia de las regiones transmitidas al 1% de los datos totales, respecto de los cortes utilizados. Obsérvese que los cortes con más regiones transmitidas están en el intervalo $[15, 28]$. Podemos decir, por lo tanto, que la información más relevante de los 93 cortes de nuestro TAC, utilizado el 1% de los datos transmitidos, se encuentra entre los cortes 15 y 28.

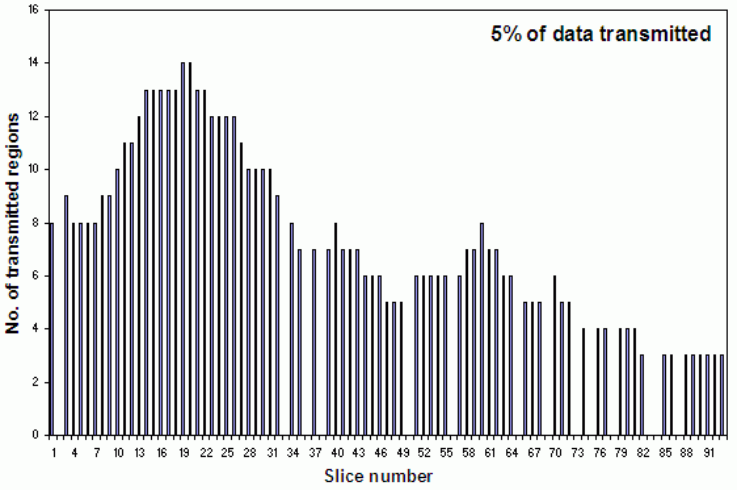


Figura 4.8: Frecuencia de las regiones transmitidas al 5% de los datos totales, respecto de los cortes utilizados. Obsérvese ahora que los cortes con más regiones transmitidas están en el intervalo $[7, 31]$.

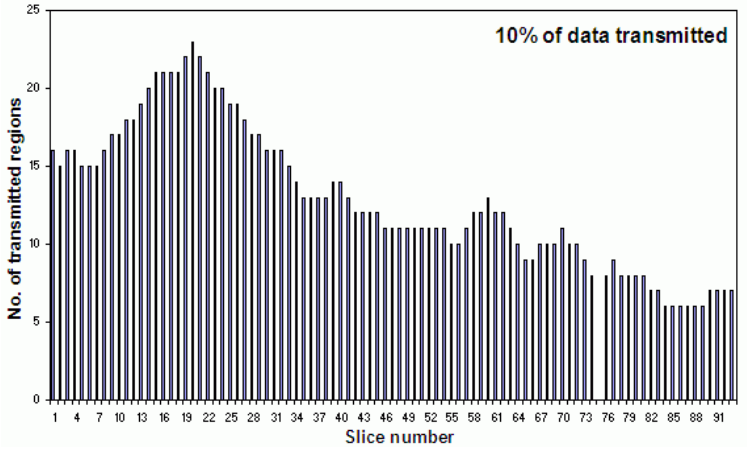


Figura 4.9: Consolidación del hecho de que la información más relevante se encuentra entre los cortes 7 y 31. Observamos también que del corte 75 no se ha enviado ninguna región. Es en el envío 1189 donde aparece la primera región del corte 75, y a partir de este envío cuando la interpolación ya no es necesaria.

Capítulo 5

Transmisión progresiva de imágenes 3D sin interpolación

Como hemos visto en el capítulo anterior, la transmisión de cada región y su reconstrucción mediante interpolación consumían (en tiempos de CPU) unos 0.77 segundos. En este tiempo está incluido el tiempo de cálculo del error con la norma Frobenius en cada iteración o envío, necesario para elegir la siguiente región a transmitir, perteneciente al corte peor reconstruido. Nuestro objetivo ahora es encontrar la forma de mejorar el tiempo de cálculo en todo el proceso descrito en el capítulo 4.

Para ello procederemos teniendo en cuenta dos ideas. Primero eliminaremos completamente la interpolación lineal de Newton de todo el proceso, transmitiendo inicialmente todas las primeras regiones de cada corte, es decir, desde la primera region del primer corte $reg(1,1)$, hasta la primera región del último corte $reg(93,1)$. Hasta ahora ocurría que mientras no se hubiese transmitido ninguna región de algún corte, dicho corte se reconstruía mediante interpolación de los cortes más cercanos ya transmitidos. Y aunque realmente la interpolación terminaba pronto (en el envío 1189, alrededor del 11% de la transmisión ya efectuada), si tenemos en cuenta que con aproximadamente el 30% de los datos transmitidos se conseguían unas reconstrucciones bastante aceptables (visualmente y con un error que a partir de entonces iba a disminuir muy poco), estamos hablando de que la interpolación aparecía en aproximadamente un tercio del proceso descrito.

Así pues y como ya hemos comentado, la eliminación de la interpolación lineal es fácil. Hacemos que la primera transmisión del servidor al cliente sea el conjunto

$$T_1 = \{reg(1,1), reg(2,1), reg(3,1), \ldots, reg(93,1)\}$$

o lo que es lo mismo, que el vector V_O (vector donde se guarda el orden de

transmisión de las regiones) sea inicialmente (ver algoritmo 4.6)

$$V_O = \{(1,1),(2,1),(3,1),\ldots,(93,1)\}$$

y calculamos entonces una primera reconstrucción sin necesidad de utilizar en ningún momento la interpolación.

La segunda idea que quisiéramos comentar es la utilización de la norma-2 (veáse expresión (2.1)) como nueva medida para expresar la diferencia entre la imagen original y la reconstruida.

5.1 La norma-2

El corolario 2.6 nos da la clave para esta nueva idea sobre la transmisión progresiva sin interpolación. Observemos que este corolario viene a decir que: $A_1 = \sigma_1 u_1 v_1^T$ es una aproximación de A cuyo error en norma-2 es σ_2 y que dicha aproximación puede mejorarse con $A_2 = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T$ cuyo error es σ_3 que es menor que σ_2. En general, $A_k = A_{k-1} + \sigma_k u_k v_k^T$ es una aproximación de A mejor que A_{k-1} porque su error es σ_{k+1} menor que σ_k. Y todo ello hasta que calculemos $A_p = A_{p-1} + \sigma_p u_p v_p^T = A$, donde el error es nulo (con $p = rango(A)$).

Este razonamiento nos lleva, después de segmentar mediante SVD nuestra imagen 3D, a **ordenar** para todos los cortes y todas las regiones, salvo las primeras ya transmitidas, los valores singulares obtenidos en la fase de descomposición. Con dicha ordenación, de mayor a menor, tenemos a su vez ordenadas las diferentes regiones de forma óptima y dispuestas para ser transmitidas.

Con todo ello obtenemos un método progresivo de reconstrucción de una imagen 3D a partir de sus valores singulares y sus vectores asociados y con un coste en el tiempo de cálculo de las reconstrucciones muy pequeño, el de un producto de vectores para crear una matriz más el producto de esa matriz por un escalar.

La aplicación de todo lo comentado da origen a un pequeño cambio en el algoritmo 4.6 para obtener un nuevo algoritmo de reconstrucción progresiva donde no se utiliza la interpolación y la norma-2 se utiliza para expresar la diferencia entre la imagen original y la reconstruida.

5.2 Reconstrucción sin interpolación. Algoritmo

Según el algoritmo 4.6, después de cada transmisión de una región, el cliente tenía que encontrar la siguiente región a transmitir del peor corte aproximado, utilizando para ello la norma Frobenius (paso 4(a) del algoritmo 4.6) y entonces solicitar (la región) al servidor que la envíase. Si en vez de utilizar la norma Frobenius utilizamos la norma-2, el servidor sabrá ya de antemano la siguiente región a transmitir. De hecho, si el servidor después de aplicar SVD a los cortes 2D, ordena de mayor a menor valor singular todas las regiones (salvo el conjunto T_1) independientemente del corte al que corresponda dicha región

(véase definición de $reg(i,j)$ en el paso 3 del algoritmo 4.1) lo que se consigue es asegurar que cuando se realiza una nueva transmisión de una región y se reconstruye la imagen 3D, esta reconstrucción será la mejor de todas las posibles en norma-2. Y todo ello sin tener que calcular ninguna medida de error. La relación entre la norma Frobenius, la norma-2 (2.4), los valores singulares de una matriz y la reconstrucción mediante SVD de ésta, nos permite eliminar de todo el proceso el cálculo del error que realizábamos utilizando la norma Frobenius.

Tras todos estos comentarios, nuestro proceso de transmisión progresiva se simplifica bastante y queda de la siguiente forma:

Algoritmo 5.1 (Reconstrucción progresiva sin interpolación) *Sea O el conjunto de datos $3D$ de tamaño $r \times s \times t$, $S = \{O_i\}_{i=1}^{r}$, con r cortes $2D$ paralelos de tamaño $s \times t$, y sea la descomposición en regiones*

$$U = \left\{ reg\,(i,j) = \{u_j^i, \sigma_j^i, v_j^i\}, \quad i = 1,2,\ldots,r, \quad j = 1,2,\ldots,q_i \right\},$$

obtenida por el algoritmo 4.1.

1. *Sea $\tilde{N} = \sum_{i=1}^{r} q_i$ el máximo número posible de transmisiones.*
2. *Sea $T_1 = \{reg(1,1), reg(2,1), reg(3,1), \ldots, reg(r,1)\}$ las primeras r regiones transmitidas y $U_1 = U \sim T_1$ las regiones no transmitidas.*
3. *Calcula $\mathcal{R}_O^1 = \{O_i^*\}_{i=1}^{r}$ con $O_i^* = \sigma_1^i u_1^i (v_1^i)^T$ para todo $i = 1,2,\ldots,r$ como la primera reconstrucción aproximada de O.*
4. *Ordena U_1 de forma **decreciente** según el valor numérico de los valores singulares σ_j^i y renombra cada elemento de U_1 como*

$$reg(k), k = 1,2,3,\ldots,(\tilde{N} - r) \text{ donde } reg(k) = \{u_k, \sigma_k, v_k\}.$$

5. *FOR $k = 1,2,3,\ldots,(\tilde{N} - r)$ DO calcula $\mathcal{R}_O^{k+1} = \mathcal{R}_O^k + \sigma_k u_k (v_k)^T$ como la reconstrucción aproximada de O en la etapa $k+1$.*
6. *FIN DO*

5.3 Resultados del nuevo experimento

Nuestro nuevo experimento ha consistido, en primer lugar y al igual que en el experimento del capítulo anterior, en descomponer el conjunto O de datos 3D en diferentes regiones utilizando SVD. Una vez segmentado este conjunto de imágenes 3D comenzamos con la transmisión de todas las primeras regiones, desde el primer hasta el último corte para poder reconstruir, si es necesario, una primera aproximación a la imagen del TAC sin necesidad de interpolar. Tras esto iremos transmitiendo las restantes regiones, previamente ordenadas de mayor a menor valor singular y que según la norma-2 son las que mejor reconstruirán nuestra imagen 3D una vez transmitidas. Procederemos así hasta que el nivel

de reconstrucción sea satisfactorio para el receptor. Podemos añadir, como ejemplo, que el proceso completo de reconstruir, una a una, 11881[1] regiones nos ha costado un total de **46.9** segundos.

Según los resultados vistos en el capítulo anterior y teniendo en cuenta que con aproximadamente un 30% de datos transmitidos las reconstrucciones eran bastante aceptables, estamos en condiciones de poder reconstruir en el receptor y de paso observar **aceptablemente** una imagen de un TAC de estas características en tan sólo **14.22** segundos, tiempo muchísimo más breve que el utilizado en la reconstrucción con interpolación, vista en el capítulo anterior.

[1] 11881 regiones son el 100% de los datos (igual número de datos que el TAC original), pero no el 100% de las regiones. Recordemos que las regiones obtenidas después de la descomposición SVD ascendían a un total de 22320.

Capítulo 6

Codificación adaptativa y transmisión progresiva para regiones de interés

Es en este capítulo donde vamos a proponer un **nuevo algoritmo para la codificación adaptativa con pérdida de imágenes digitales 3D** basado, como hasta ahora, en la descomposición en valores singulares de matrices y **aplicado sobre todo a ciertas regiones de interés** seleccionadas de las imágenes 3D. Esta nueva codificación mediante SVD nos permitirá diseñar algoritmos para la transmisión y reconstrucción de forma progresiva de una o varias ROIs, pertenecientes a una imagen 3D, que seleccione el cliente y evitando principalmente la redundancia en la transmisión de datos.

La principal característica de los algoritmos que propondremos es que las ROIs pueden ser **seleccionadas durante el proceso de transmisión** y que **no es necesario codificar la imagen de nuevo** para transmitir los datos correspondientes de la ROI seleccionada.

6.1 Introducción a la codificación y transmisión de ROIs

Como ya comentamos en el capítulo 1, el proceso de la transmisión progresiva se describe en siete pasos (véase sección 1.2.2).

Pero en este capítulo estamos particularmente interesados en el paso **7(d):** durante la transmisión progresiva de una imagen, el cliente observa un detalle de interés (o varios) en la reconstrucción de dicha imagen y entonces, selecciona para la transmisión una subimagen que contenga dicho detalle. El proceso continúa con la transmisión progresiva de dicha subimagen o ROI y su posterior reconstrucción.

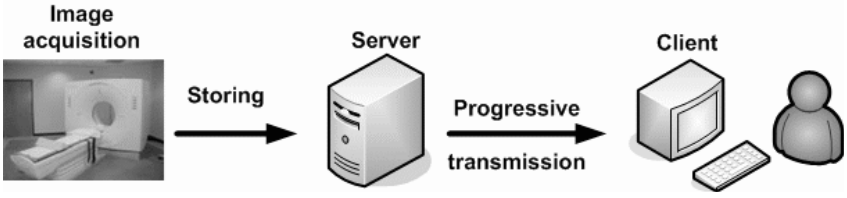


Figura 6.1: Esquema Servidor-Cliente en la transmisión progresiva de datos.

La situación descrita, es decir, cuando el cliente selecciona la ROI puede abordarse de dos maneras:

(a) el servidor recibe las coordenadas de la ROI, codifica dicha región de la imagen original y la transmite progresivamente, o

(b) el servidor posee de antemano la imagen codificada de cierta manera y esto le permite transmitir de forma progresiva y sin consumo adicional alguno en el tiempo de cálculo, cualquiera de las posibles ROIs que el cliente pueda o quiera observar.

En la opción (a) habrá que tener en cuenta el tiempo de cálculo adicional necesario para la codificación de la ROI en el momento de la petición de ésta, lo que conllevaría retraso en la transmisión y visualización de la imagen y donde además, es difícil evitar la redundancia de los datos transmitidos. Por otro lado la opción (b) requiere una relación fuerte entre los datos codificados de la imagen entera y los datos de cualquiera de sus subimágenes, hecho que no tiene porque ser así necesariamente.

Por ejemplo, cuestiones semejantes se comentan en [31], [45] y [71] donde usando JPEG200, la codificación basada en la ROI es realizada por el método MAXSHIFT descrito en [14] y ya comentado en el capítulo 2, y donde se observó cómo este método requiere la elección, a priori, de la ROI y a continuación, la codificación de la imagen junto con la ROI seleccionada anteriormente. Mientras que como ya hemos comentado, con nuestro método las ROIs podrán ser seleccionadas durante el proceso de transmisión y sin la necesidad de codificar la imagen de nuevo para transmitir los datos correspondientes de la ROI seleccionada.

6.2 El SVD en la codificación de las ROIs

Proponemos por tanto un método para la codificación de imágenes 3D usando la descomposición en valores singulares (SVD) donde los datos correspondientes a una ROI se puedan extraer de la imagen codificada entera, evitando así redundancia en los datos a transmitir. La idea principal se basa en la característica siguiente de SVD para imágenes 2D vista en la sección 3.5 y que repetimos aquí:

si

$$\{\sigma, (u_1, \ldots, u_m), (v_1, \ldots, v_n)\}$$

es un valor singular y los vectores singulares asociados de una imagen 2D de tamaño $m \times n$, el producto

$$\sigma\,(u_1, \ldots, u_m)^T\,(v_1, \ldots, v_n)$$

es una aproximación a la imagen 2D y entonces

$$\sigma\,(u_i, \ldots, u_{i+k})^T\,(v_j, \ldots, v_{j+l})$$

pasa a ser una aproximación de la ROI de tamaño $(k+1) \times (l+1)$ cuyas coordenadas de la esquina superior izquierda son (i, j) (véase figura 6.2).

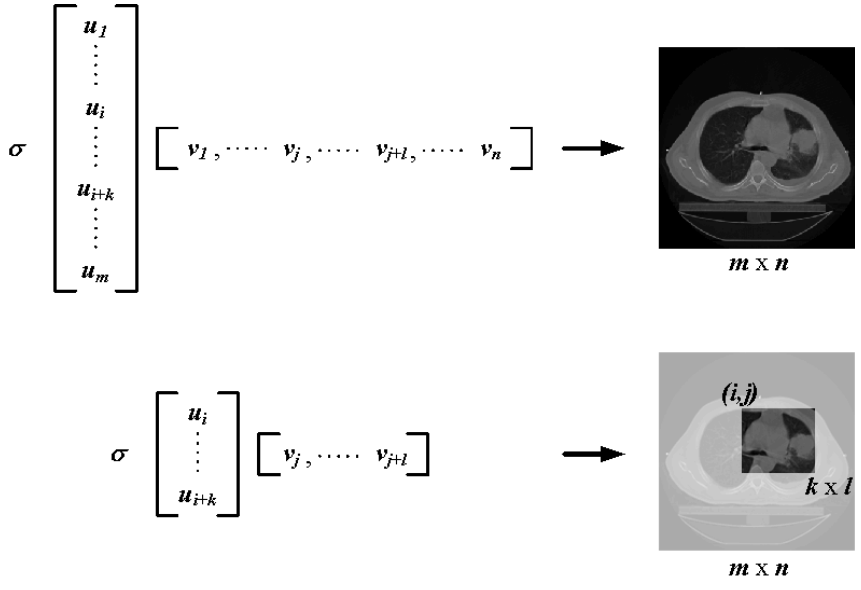


Figura 6.2: Reconstrucción de la ROI 2D usando SVD.

Este método para codificar imágenes tridimensionales usando SVD se enfocará principalmente desde un punto de vista algorítmico, porque uno de nuestros objetivos principales es presentar los algoritmos de manera que nos permitan diseñar los procesos para la transmisión progresiva de las imágenes digitales de una forma automática, transparente y sobre todo sencilla para el usuario final.

6.3 Diferencia relativa 3D

Ya se ha comentado (capítulo 2) que es usual durante la SVD eliminar los valores singulares más pequeños porque se asume que no proporcionan una mejora substancial en la reconstrucción de la imagen.

Entonces, cuanto menor es el valor del umbral ϵ mayor número de regiones vamos a obtener en la descomposición de la imagen. En experimentos anteriores hemos llegado a utilizar umbrales prefijados del orden de 10^{-9}. Ahora, la nueva pregunta que se presenta es si podemos aumentar el valor de ese umbral, y disminuir entonces el número de regiones, sin que esto afecte a la calidad visual de nuestras reconstrucciones, así como dar un método, no arbitrario, para la elección adecuada de ese umbral. Nuestra nueva propuesta se basa en una extensión a las imágenes tridimensionales de la llamada diferencia relativa introducida en [52].

Definición 6.1 *Sea $O = (O_{ijk})$ una imagen 3D de tamaño $r \times s \times t$ y $O^* = \left(O^*_{ijk}\right)$ una aproximación de O. La diferencia relativa 3D se define como*

$$D_3\left(O, O^*\right) = \sqrt{\frac{\sum_{i=1}^{r}\sum_{j=1}^{s}\sum_{k=1}^{t}\left(O_{ijk} - O^*_{ijk}\right)^2}{\sum_{i=1}^{r}\sum_{j=1}^{s}\sum_{k=1}^{t}O^2_{ijk}}}. \tag{6.1}$$

Después de lo razonado en el teorema 2 de [52], si calculamos el SVD de los cortes paralelos $O_i = (O_{ijk})$, $i = 1, 2, \ldots, r$; las aproximaciones a cada corte O^*_i, $i = 1, 2, \ldots, r$ se construyen utilizando la expresión (2.5) y definimos los conjuntos de valores singulares Δ y Γ como

$$\begin{aligned} \Delta &= \{\sigma \text{ tales que son valores singulares de algún corte } O_i,\ i = 1, \ldots, r\}, \\ \Gamma &= \{\sigma \text{ tales que no se usan en ninguna reconstrucción } O^*_i,\ i = 1, \ldots, r\}. \end{aligned}$$

Entonces es fácil demostrar [52] que:

$$D_3\left(O, O^*\right) = \sqrt{\frac{\sum_{\sigma\in\Gamma}\sigma^2}{\sum_{\sigma\in\Delta}\sigma^2}}. \tag{6.2}$$

La expresión (6.2) nos permitirá entonces determinar las regiones que no proporcionan una mejora substancial en la reconstrucción de la imagen y así poder rechazarlas.

Como regla práctica para las imágenes 2D podemos decir que si $D(A, B) \leq 0.05$ entonces la matriz B es una aproximación aceptable (desde el punto de vista de nuestra percepción) de la imagen almacenada en A [67].

Pero esta condición no podemos aplicarla para imágenes tridimensionales. Por ejemplo: si consideramos el mismo conjunto O de datos 3D del capítulo 4 que consistía en un conjunto de 93 cortes 2D del TAC de una cabeza humana y en niveles de grises, de 16 bits por píxel (figura 6.3), y consideramos por otro lado O^* como la reconstrucción aproximada de O con $D_3(O, O^*) = 0.05$ (podemos observar una visualización de O^* en la figura 6.4), la cara arrugada que aparece en la figura 6.4 nos da a entender que una cota superior del 5% para la diferencia relativa 3D no es suficiente para imágenes tridimensionales. Después de algunos experimentos y tras observar las diferentes reconstrucciones y sus errores cometidos, sugerimos como cotas superiores los valores 0.01 ó incluso el valor 0.005. ($D_3\left(O, O^*\right) \leq 0.01$ ó $D_3\left(O, O^*\right) \leq 0.005$).

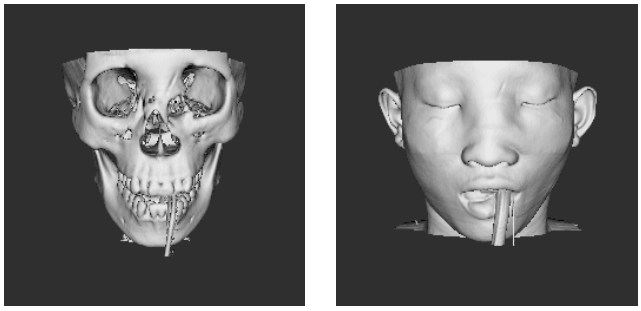


Figura 6.3: Resultados a nivel de hueso y de piel de todo el conjunto de datos de los 93 cortes del TAC.

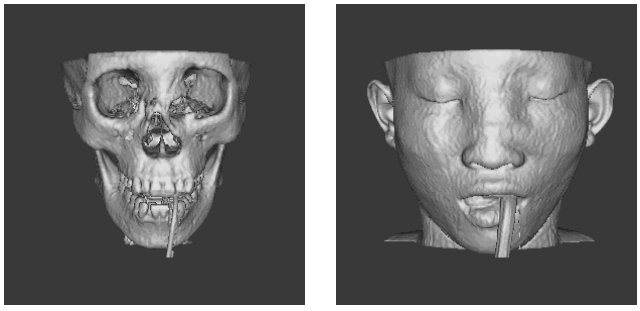


Figura 6.4: Resultados a nivel de hueso y de piel de la imagen tridimensional aproximada O^* con $D_3(O, O^*) = 0.05$.

6.4 Algoritmo de codificación utilizando cuantificación

La idea principal en la creación de un nuevo algoritmo para codificar imágenes digitales tridimensionales utilizando SVD y la cuantificación consistirá en:

1. dividir la imagen tridimensional en cortes 2D,
2. aplicar SVD a cada corte,
3. definir las regiones correspondientes de cada corte,
4. ordenar las regiones y con ellas los valores singulares, de la siguiente forma:
   (a) primero, las regiones con el primer valor singular (más alto) de cada corte,
   (b) en segundo lugar y de forma decreciente, el resto de regiones, según el valor singular,
5. después de la ordenación anterior, calcular el primer valor singular σ^* tal que $D_3(O, O^*) \leq \epsilon$, con $\epsilon = 0.01$ u otros umbrales prefijados,
6. rechazar las regiones con valores singulares más pequeños que σ^*,
7. cuantificar las regiones no rechazadas, ordenándolas en un archivo codificado.

En principio, la cuantificación sugerida para los valores singulares es de $b_{val} = 32$ bits y $b_{vec} = 16$ bits para cada una de las componentes de los vectores singulares, aunque podrían considerarse otras posibilidades para la elección del número de bits. Nótese que a menor número de bits para representar los datos, se produce una mayor cuantificación y como consecuencia, una codificación con pérdida mayor. En cualquier caso, es importante observar que la fórmula (6.2) deducida a partir de la expresión (6.1) sigue siendo cierta si la cuantificación del valor singular es baja, es decir, una cuantificación alta (pocos bits) para representar los valores singulares puede conducir a la inaplicabilidad de la fórmula (6.2) y entonces, lo que obtendríamos sería una *mala* aproximación de la imagen tridimensional O.

Veamos, pues, y con detenimiento, la exposición precisa del algoritmo de codificación con cuantificación.

Algoritmo 6.2 (Codificación) *Sea O una imagen tridimensional de tamaño $r \times s \times t$, sea $\epsilon \geq 0$ un umbral dado, y b_{vec} y b_{val} el número de bits en los que son cuantificados respectivamente las componentes de los vectores singulares y los valores singulares.*

1. *Divide O en r cortes paralelos 2D O_i, $i = 1, 2, \ldots, r$, de tamaño $s \times t$.*

2. *Aplica SVD a cada corte O_i, es decir,*

$$\sigma_1^i \geq \cdots \geq \sigma_p^i \geq 0, \quad p = \min\{s, t\},$$

siendo $\left\{u_1^i, \ldots, u_p^i\right\}, \left\{v_1^i, \ldots, v_p^i\right\}$ los vectores singulares correspondientes para $i = 1, 2, \ldots, r$.

3. *Define las diferentes regiones de cada corte como*

$$reg\,(i, j) = \left\{u_j^i, \sigma_j^i, v_j^i\right\}, \quad i = 1, 2, \ldots, r, \quad j = 1, 2, \ldots, p.$$

4. *Ordena las regiones y con ellas los valores singulares, como sigue:*

   (a) *la primera región de cada corte $reg(1,1)$, $reg(2,1), \ldots, reg(r,1)$ en primer lugar (con el valor singular más alto de cada corte),*

   (b) *en segundo lugar y de forma decreciente, el resto de regiones, según el valor singular,*

5. *Encuentra el primer valor singular σ^* tal que*

$$\sqrt{\frac{\sum_{\sigma > \sigma^*} \sigma^2}{\sum_{k=1}^{rp} \sigma_k^2}} \leq \epsilon.$$

6. *Rechaza las regiones cuyo valor singular sea más pequeño que σ^*.*

7. *Define $q\,(O, \epsilon)$ como el número de regiones no rechazadas.*

8. *Archiva y guarda en un fichero las siguientes codificaciones:*

   (a) *Escribe r, s y t, codificados a 16 bits.*

   (b) *Escribe el b_{vec} y b_{val} codificados a 5 bits.*

   (c) *Escribe $q\,(O, \epsilon)$ codificado a 16 bits.*

   (d) *Cuantifica la primera región de cada corte $reg(1,1), reg(2,1), \ldots, reg(r,1)$ de la forma siguiente: en b_{vec} bits cada componente de los vectores singulares y en b_{val} bits los valores singulares y escríbelos como sigue:*

$$\left[\underbrace{\sigma_1^1}_{b_{val}\ bits} + \underbrace{u_1^1}_{sb_{vec}\ bits} + \underbrace{v_1^1}_{tb_{vec}\ bits}\right] + \cdots + \left[\underbrace{\sigma_1^r}_{b_{val}\ bits} + \underbrace{u_1^r}_{sb_{vec}\ bits} + \underbrace{v_1^r}_{tb_{vec}\ bits}\right].$$

   (e) *Calcula $b_s = Int\,(\log_2 r) + 1$, que es el número de bits necesarios para codificar el valor de r, (número de cortes o número de imágenes 2D). $Int(x)$ es la parte entera de x.*

(f) Cuantifica el resto de regiones en: b_{vec} bits las componentes de los vectores singulares y en b_{val} bits los valores singulares y escríbelos en el archivo como sigue:

$$\left[\underbrace{N^o\ de\ corte}_{b_s\ bits} + \underbrace{\sigma}_{b_{val}\ bits} + \underbrace{u}_{sb_{vec}\ bits} + \underbrace{v}_{tb_{vec}\ bits}\right] + \cdots +$$
$$+ \cdots + \left[\underbrace{N^o\ de\ corte}_{b_s\ bits} + \underbrace{\sigma}_{b_{val}\ bits} + \underbrace{u}_{sb_{vec}\ bits} + \underbrace{v}_{tb_{vec}\ bits}\right].$$

El nuevo algoritmo 6.2 que utiliza SVD, proporciona una codificación de una imagen tridimensional donde los datos relevantes (regiones con los valores singulares mayores que el umbral) están almacenados en casi su totalidad y con un número de bits que permite alcanzar los objetivos deseados de:

- buena calidad de la reconstrucción en los primeros pasos de la transmisión progresiva,
- reconstrucción final (casi) idéntica a la imagen original,
- el algoritmo de codificación de la imagen no genera muchos más datos que la imagen original (cuantificados en bits).

Obsérvese que la ordenación realizada en el paso 4 del algoritmo 6.2 depende inicialmente de la descomposición SVD y, finalmente, de la imagen. Así pues, podemos decir que la forma de la codificación final depende de la imagen. En este sentido, el algoritmo anterior es una codificación adaptativa de imágenes tridimensionales y por lo tanto conducirá a una transmisión progresiva adaptativa óptima.

Y por supuesto, esta transmisión progresiva es con pérdida. Esto es debido principalmente a la utilización de un umbral para desechar regiones y a la cuantificación de los valores y vectores singulares.

6.4.1 Tamaño del archivo de la imagen 3D codificada

El tamaño original de una imagen tridimensional O de tamaño $r \times s \times t$ es

$$b \times r \times s \times t \ \text{ bits}, \tag{6.3}$$

donde b es el número de bits con que los datos se almacenan y se graban.

Si aplicamos el algoritmo 6.2 a O, la imagen codificada O tendrá el tamaño

$$\underbrace{16+16+16+5+5+16}_{74} + r\left(sb_{vec} + b_{val} + tb_{vec}\right) \tag{6.4}$$
$$+ \left(q\left(O,\epsilon\right) - r\right)\left(b_s + sb_{vec} + b_{val} + tb_{vec}\right) \ \text{ bits}.$$

Veamos ahora un ejemplo del tamaño del archivo que nos quedaría después de codificar el conjunto de datos tridimensionales O de la figura 6.3 que consiste en 93 cortes paralelos, de tamaño 256×256 y donde cada píxel está codificado en 16 bits.

El tamaño original de O es:

$$16 \text{ bits} \times 93 \times 256 \times 256 = 97517568 \text{ bits} = 11.625 \text{ Mb}.$$

Y si aplicamos el algoritmo 6.2 a este conjunto de datos tridimensionales O, obtenemos después de aplicar SVD a cada corte

$$93 \times 256 = 23808 \text{ regiones},$$

que ordenamos según comentamos en el paso 4 del algoritmo 6.2. Entonces el σ^* tal que

$$\sqrt{\frac{\sum_{\sigma>\sigma^*} \sigma^2}{\sum_{k=1}^{rp} \sigma_k^2}} \leq 0.01,$$

es $\sigma^* = 414.256$, que corresponde a la región número 7680, es decir, hay $q(O, \epsilon) = 7679$ regiones no rechazadas (32.25% del total). Por lo tanto y tomando como $b_{vec} = 16$ bits, $b_{val} = 32$ bits y

$$b_s = Int\left[\log_2(93)\right] + 1 = 7,$$

podemos codificar la imagen tridimensional O en un archivo con (véase la expresión (6.4))

$$\begin{gathered} 74 + 93\,(256 \times 16 + 32 + 256 \times 16) \\ + (7679 - 93)\,(7 + 256 \times 16 + 32 + 256 \times 16) = \\ 63205272 \text{ bits} = 7.534 \text{ Mb}, \end{gathered}$$

lo que corresponde al 64.81% del tamaño original de O.

6.5 Diseño de los algoritmos para la transmisión y reconstrucción de imágenes 3D y de sus ROIs

En esta nueva sección y tras la segmentación y codificación de imágenes digitales desarrolladas en la sección anterior, vamos a estudiar ahora los algoritmos apropiados para resolver la siguiente situación: el cliente desea visualizar una imagen tridimensional. Una vez identificada la imagen, el servidor lee datos de la imagen, más exactamente de los archivos codificados por el algoritmo 6.2 y los transmite. El cliente recibe estos datos y comienza entonces a realizar las reconstrucciones progresivas mientras van llegando más datos.

Para poder realizar todo lo anterior debemos comenzar por exponer un nuevo algoritmo de reconstrucción de una imagen 3D completa, basado en la fórmula

(2.5) y en el hecho de que cada vez que el servidor transmite o envía una nueva región y el cliente la recibe, éste lee el número del corte al que corresponde dicho envío y lo mejora.

Imaginemos también que durante la reconstrucción realizada con este algoritmo, el cliente desea observar con más detalle alguna zona en particular de la imagen. Propondremos entonces, dos nuevos algoritmos (uno de transmisión y otro de reconstrucción) para mejorar exclusivamente dichas zonas o regiones de interés.

Tras la exposición de estos algoritmos, veremos su aplicación en un ejemplo, al final del capítulo.

6.5.1 Algoritmo de reconstrucción de una imagen 3D

Algoritmo 6.3 (Reconstrucción de la imagen) *Supongamos que el servidor está listo para transmitir progresivamente la imagen deseada. Éste comienza a leer del archivo codificado y transmite secuencialmente los datos. Entonces, el cliente:*

1. *Lee los primeros* $16+16+16+5+5+16$ *bits transmitidos y almacénalos en las variables* $r, s, t, b_{vec}, b_{val}$, *y* q *respectivamente.*
2. *PARA* $i = 1, \ldots, r$
   (a) *lee los siguientes* b_{val} *bits y almacénalos en la variable* σ,
   (b) *lee los siguientes* $s \times b_{vec}$ *bits y almacénalos en el vector variable* u,
   (c) *lee los siguientes* $t \times b_{vec}$ *bits y almacénalos en el vector variable* v,
   (d) *calcula* $O_i = \sigma u v^T$.
3. *Calcula* $b_s = Int\,(\log_2 r) + 1$.
4. *Hasta que el cliente pare la transmisión o se llegue al final del archivo,*
   (a) *lee los primeros* b_s *bits y almacénalos en la variable* n,
   (b) *lee los siguientes* b_{val} *bits y almacénalos en la variable* σ,
   (c) *lee los siguientes* $s \times b_{vec}$ *bits y almacénalos en el vector variable* u,
   (d) *lee los siguientes* $t \times b_{vec}$ *bits y almacénalos en el vector variable* v,
   (e) *calcula* $O_n = O_n + \sigma u v^T$, *(mejora del corte* n*).*

Debido a los pocos datos que contiene cada región de forma individual, sería recomendable mejorar o actualizar la imagen visualizada, no en el momento en que llega cada región, sino por ejemplo cada vez que se reciben al menos un 1% ó un 2% más de datos adicionales.

Los datos hasta el paso 2 incluido, del algoritmo 6.3, se pueden enviar totalmente en un "primer paquete", ya que las primeras regiones de cada corte no constituyen una gran cantidad de datos y son un magnífico punto de partida

para realizar una primera reconstrucción y visualización. Además, su transmisión completa evita la interpolación o técnicas similares para aproximar cortes sin ninguna región transmitida.

Por ejemplo, en el ejemplo del TAC de 93 cortes dado en la figura 6.3, el tamaño de los datos transmitidos hasta el paso 2, incluido éste, son:

$$74 + 93\,(256 \times 16 + 32 + 256 \times 16) = 764906 \text{ bits} = 93.372 \text{ Kb},$$

que corresponde solamente al 1.21% del tamaño total del archivo codificado.

6.5.2 Transmisión y reconstrucción de ROIs. Preliminares

Supongamos que durante el proceso o ejecución del algoritmo 6.3 y más exáctamente en el paso 4, el cliente observa o visualiza alguna o algunas características de cierto interés que aparecen en la reconstrucción de la imagen. En tal situación, el cliente puede parar el proceso y seleccionar dichas regiones de interés.

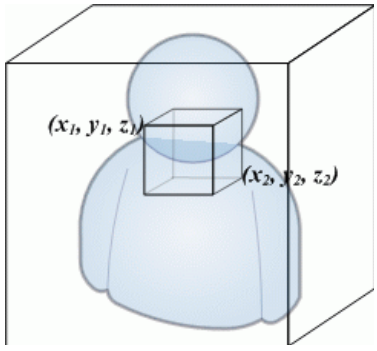


Figura 6.5: Coordenadas superior izquierda e inferior derecha de una ROI en una imagen 3-D.

Luego vamos a suponer que el cliente selecciona p ROIs cuyas coordenadas de la esquina superior izquierda y de la esquina inferior derecha son respectivamente, $\left(x_1^i, y_1^i, z_1^i\right)$ y $\left(x_2^i, y_2^i, z_2^i\right)$, $i = 1, 2, ..., p$ y donde $x_1^i, y_1^i, z_1^i, x_2^i, y_2^i, z_2^i$ son números enteros cumpliendo $x_1^i \leq x_2^i$, $y_1^i \leq y_2^i$, $z_1^i \leq z_2^i$ (véase la figura 6.5). Se definen entonces los conjuntos:

$$X = [x_1^1, x_2^1] \cup \cdots \cup [x_1^p, x_2^p], \tag{6.5}$$

$$Y = [y_1^1, y_2^1] \cup \cdots \cup [y_1^p, y_2^p], \tag{6.6}$$

$$Z = [z_1^1, z_2^1] \cup \cdots \cup [z_1^p, z_2^p]. \tag{6.7}$$

Estas coordenadas se transmiten al servidor y, éste y el cliente que seleccionó las diferentes ROIs continúan con la transmisión-recepción de datos.

Tras el envío de la situación de cada ROI, en vez de regiones completas de la imagen se van a transmitir partes de ellas, realizándose las reconstrucciones correspondientes según la idea vista en la figura 6.2. Por lo tanto, la secuencia de la transmisión-recepción de datos tiene que cambiar, tienen que ser desarrolladas ciertas estrategias para evitar la repetición de los datos y debe ser incluido en la transmisión un cierto número de datos adicionales para la identificación de las regiones que estamos transmitiendo (aunque sólo sea una parte de ellas).

Para evitar redundancia de datos, deben ser definidos en ambos lados del servidor y del cliente dos matrices y un vector de la siguiente forma: supongamos que hasta ahora se han transmitido-recibido q regiones completas. Entonces en el lado del servidor se definen:

- la matriz U_s de tamaño $(q\,(O,\epsilon) - q) \times s$,
- la matriz V_s de tamaño $(q\,(O,\epsilon) - q) \times t$,
- y el vector S_s de tamaño $q\,(O,\epsilon) - q$.

Además debe cumplirse que la entrada (i,j) de U_s valdrá 1 si la componente j del vector u de la región $q+i$ se ha transmitido, y 0 en cualquier otro caso. Sucede igualmente para $V_s\,(i,j)$ con la componente j del vector v de la región $q+i$ y para $S_s\,(i)$ con el valor singular de la región $q+i$.

En el lado del cliente, se definen también:

- la matriz U_c de tamaño $(q\,(O,\epsilon) - q) \times s$,
- la matriz V_c de tamaño $(q\,(O,\epsilon) - q) \times t$,
- y el vector S_c de tamaño $q\,(O,\epsilon) - q$.

Las entradas $U_c\,(i,j)$ y $V_c\,(i,j)$ son respectivamente la componente j recibida de los vectores u y v de la región $q+i$, y 0 en cualquier otro caso. La entrada $S_c\,(i)$ almacena el valor singular de la región $q+i$ si éste se ha recibido y si no se ha recibido, dicha entrada valdrá 0.

Llegados hasta aquí debemos observar tres hechos importantes. El **primero** es que el control de las regiones o de cualquier parte de éstas, está basado en el orden de las mismas, por lo tanto, el número de la región es uno de los datos adicionales que el servidor tiene que transmitir al cliente. En el algoritmo 6.3 esto no era necesario porque se transmitían regiones completas.

El **segundo** hecho es que las matrices y los vectores auxiliares descritos anteriormente aparecen sólo cuando se hace necesaria la transmisión de una ROI y serán suprimidos de la memoria del servidor y del cliente cuando terminen la transmisión-recepción progresivas.

El **tercero** ocurre en el caso en que la ROI, llamémosla A, haya sido transmitida y, posteriormente, el cliente selecciona una nueva ROI, llamada B, con datos (regiones) en común con A. En este caso sólo tienen que ser transmitidas

las componentes de los vectores u y v de las regiones comunes no transmitidas, porque las restantes componentes fueron ya almacenadas en U_c y V_c y los valores singulares en S_c durante la transmisión de la anterior ROI A. Por otra parte, hay que observar que en la reconstrucción puede producirse un solapamiento, una coincidencia parcial entre las ROIs.

Por ejemplo, podemos suponer que durante la reconstrucción de la ROI A cierta parte de un corte se reconstruye como sigue

$$O_n = \sigma \begin{bmatrix} 0 \\ u_2 \\ u_3 \end{bmatrix} [0, v_2, v_3] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \sigma u_2 v_2 & \sigma u_2 v_3 \\ 0 & \sigma u_3 v_2 & \sigma u_3 v_3 \end{bmatrix},$$

y, además, durante la reconstrucción de la ROI B, el mismo corte se mejora también como

$$O_n = O_n + \sigma \begin{bmatrix} u_1 \\ u_2 \\ 0 \end{bmatrix} [v_1, v_2, 0] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \sigma u_2 v_2 & \sigma u_2 v_3 \\ 0 & \sigma u_3 v_2 & \sigma u_3 v_3 \end{bmatrix} + \begin{bmatrix} \sigma u_1 v_1 & \sigma u_1 v_2 & 0 \\ \sigma u_2 v_1 & \sigma u_2 v_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

y aparece entonces un solapamiento en la componente común $(2, 2)$. Para evitar que este posible solapamiento sobreescriba el valor correcto en la reconstrucción, vamos a definir una suma de matrices distinta de la usual.

Definición 6.4 *Si $A = (a_{ij})$ y $B = (b_{ij})$ son matrices del mismo tamaño, definimos la adición sin solapamiento de matrices como*

$$A \oplus B = (a_{ij} \oplus b_{ij}),$$

donde

$$a_{ij} \oplus b_{ij} = \begin{cases} a_{ij}, & a_{ij} \neq 0, \\ b_{ij}, & a_{ij} = 0. \end{cases}$$

6.5.3 Algoritmos para la transmisión y reconstrucción de las ROIs

Estos tres hechos ya comentados nos conducen a considerar sobre ambos lados de la conexión, servidor y cliente, la presentación por separado de los siguientes algoritmos para la transmisión progresiva de la ROI. Las condiciones iniciales con las que comienzan los algoritmos 6.5 y 6.6 que vamos a presentar y que serán utilizados para la transmisión de la ROI por el servidor y la reconstrucción de ésta por el cliente son:

- el cliente ha parado la transmisión en cierta región q,
- el cliente ha seleccionado una o varias ROIs, cuyas coordenadas de las esquinas opuestas se transmiten al servidor, el cual obtiene los conjuntos X, Y y Z tal como están definidos en (6.5), (6.6) y (6.7). Estos conjuntos quedan entonces definidos en ambas partes: en la del servidor y en la parte del cliente.

Por otro lado, en la parte del servidor, se requiere solamente una vez y para la primera transmisión de la ROI:

- definir la matriz **nula** U_s del tamaño $(q\,(O,\epsilon)-q)\times s$ (ceros en todas sus entradas),
- definir la matriz **nula** V_s del tamaño $(q\,(O,\epsilon)-q)\times t$ (ceros en todas sus entradas),
- definir el vector **nulo** S_s del tamaño $(q\,(O,\epsilon)-q)$ (ceros en todas sus entradas),

Por lo tanto, asumimos que estas matrices, U_s y V_s, así como el vector S_s están ya definidos y quizá, algunas de sus entradas ya no son cero.

Algoritmo 6.5 (Transmisión por el servidor de las ROIs seleccionadas) *El servidor comienza la transmisión de la región* $q+1$.

DESDE $i=1$ *HASTA* $(q\,(O,\epsilon)-q)$ *o hasta que el cliente pare la transmisión,*

1. *Lee los primeros* b_s *bits y almacénalos en la variable* n (n^o *de corte*),
2. *SI* $n\in X$, (*la región tiene datos de la ROI actual*),
   (a) *codifica* i *a* 16 *bits y transmítelo,* (*codificación y transmisión del número de la región*),
   (b) *transmite* n, (*número del corte correspondiente a la transmisión de la región actual*),
   (c) *SI* $S_s(i)=0$, *transmite los siguientes* b_{val} *bits,* (*transmisión del valor singular*) *y asigna* $S_s(i)=1$,
   (d) *DESDE* $j=1$ *HASTA* s, (*transmisión del vector singular* u),
      i. *SI* $j\in Y$ *y* $U_s(i,j)=0$, *transmite los* b_{vec} *bits de la componente* j *del vector* u *de la región actual y asigna* $U_s(i,j)=1$,
   (e) *DESDE* $j=1$ *HASTA* t, (*transmisión del vector singular* v),
      i. *SI* $j\in Z$ *y* $V_s(i,j)=0$, *transmite los* b_{vec} *bits de la componente* j *del vector* v *de la región actual y asigna* $V_s(i,j)=1$.

Ahora, en la parte del cliente se requiere, solamente una vez y para la primera transmisión de la ROI:

- definir la matriz **nula** U_c del tamaño $(q\,(O,\epsilon)-q)\times s$ (ceros en todas sus entradas),
- definir la matriz **nula** V_c del tamaño $(q\,(O,\epsilon)-q)\times t$ (ceros en todas sus entradas),

- definir el vector **nulo** S_c del tamaño $(q\,(O,\epsilon)-q)$ (ceros en todas sus entradas).

Por lo tanto, asumimos que estas matrices, U_c y V_c, así como el vector S_c están ya definidos y, quizá, algunas de sus entradas ya no son cero.

Algoritmo 6.6 (Reconstrucción de las ROIs) *El servidor comienza con la transmisión de la región* $q+1$.

Asigna la variable solapamiento = *Falso.*

Hasta que el cliente pare la transmisión o no se reciban más datos:

1. *Lee los siguientes* 16 *bits y almacénalos en la variable* i, (*número de la región*).

2. *Lee los siguientes* b_s *bits y almacénalos en la variable* n, (*número del corte*).

3. ***Construcción del valor singular.***

   - *SI* $S_c(i)=0$, *lee los siguientes* b_{val} *bits y almacénalos en la variable* σ *y asigna a* $S_c(i)$ *el valor* σ, $(S_c(i)=\sigma)$.
   - *SI NO el valor singular ya fue transmitido antes y entonces* $\sigma = S_c(i)$, (*en este caso el solapamiento puede ser posible*) *y asignamos* solapamiento = *Verdadero.*

4. ***Construcción del vector u.***

   - *Las componentes* j *tales que* $j \notin Y$ *toman el valor* 0.
   - *Lee los* b_{vec} *bits siguientes y almacénalos en la variable* aux.
   - *PARA TODO* $j \in Y$.
     - *SI* $U_c(i,j)=0$, *asigna a la componente* j *de* u *el valor* aux $(u(j)=\text{aux})$, *asigna* $U_c(i,j)=\text{aux}$, *lee los* b_{vec} *bits siguientes y almacénalos en la variable* aux.
     - *SI NO asigna a la componente* j *de* u *el valor* $U_c(i,j)$.

5. ***Construcción del vector v.***

   - *Las componentes* j *tales que* $j \notin Z$ *toman el valor* 0.
   - *Lee los* b_{vec} *bits siguientes y almacénalos en la variable* aux.
   - *PARA TODO* $j \in Z$.
     - *SI* $V_c(i,j)=0$, *asigna a la componente* j *de* v *el valor* aux $(v(j)=\text{aux})$, *asigna* $V_c(i,j)=\text{aux}$, *lee los* b_{vec} *bits siguientes y almacénalos en la variable* aux.
     - *SI NO asigna a la componente* j *de* v *el valor* $V_c(i,j)$.

6. ***Mejora de la reconstrucción de la ROI en el corte n.***

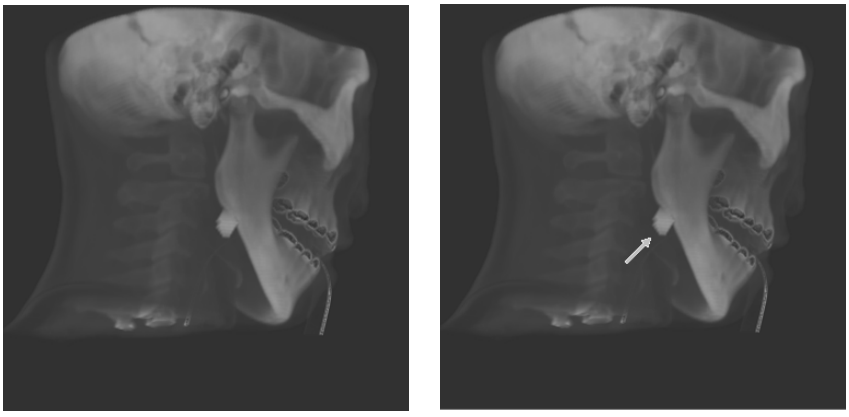
- *SI* solapamiento = *Verdadero*, *mejora* $O_n = O_n \oplus \sigma uv^T$.
- *SI NO, mejora* $O_n = O_n + \sigma uv^T$.

Hay que observar que en ambos algoritmos, cada vez que se requiere una nueva transmisión de la ROI, el servidor comienza a transmitir datos de la región $q + 1$, es decir, de la primera región no transmitida totalmente.

Ambos algoritmos están preparados para transmitir y recibir secuencialmente las ROIs seleccionadas, y cuando no se requieran más ROIs, permiten la transmisión del resto de datos no transmitidos y su uso para mejorar la reconstrucción, todo ello hasta que se hayan transmitido completamente los datos codificados de la imagen tridimensional.

6.6 Ejemplo

La figura 6.6 es una reconstrucción lateral, utilizando el método *Ray Casting*[1], de la cabeza humana de la figura 6.3 a la que se le ha "implantado un tumor". Todo esto nos proporciona un pequeño detalle interno para que podamos reconocerlo durante la exploración visual. Este tumor ficticio está localizado en el lado derecho de la mandíbula inferior (figura 6.6).



(a) (b)

Figura 6.6: (a) Reconstrucción lateral mediante el método Ray Casting de los 93 cortes del TAC de una cabeza humana con un tumor implantado en la mandíbula. (b) El tumor se indica claramente con una flecha.

Esta imagen ha sido codificada por el algoritmo 6.2 tomando $\epsilon = 0.01$, $b_{val} = 32$ bits, $b_{vec} = 16$ bits y $b_s = 7$ (véase la sección 6.4.1). El servidor comienza con la transmisión progresiva a un cierto cliente y después de la transmisión de 476

[1] Algoritmo de determinación de superficies visibles de Arthur Appel (1968). Método muy usado (entre otros por el Raytracing, Turner Whitted 1980) para obtener imágenes de alta calidad de objetos sólidos.

regiones de la imagen codificada, es decir, al 6.20% de los datos transmitidos o después de transmitir 478.196 Kb. Podemos ver la reconstrucción obtenida en la figura 6.7.

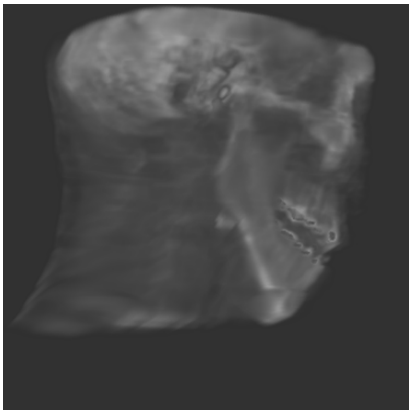


Figura 6.7: Al 6.20% de transmisión de la imagen 3D codificada, es decir, tras 476 regiones transmitidas, ya se aprecia algo extraño debajo de la mandíbula.

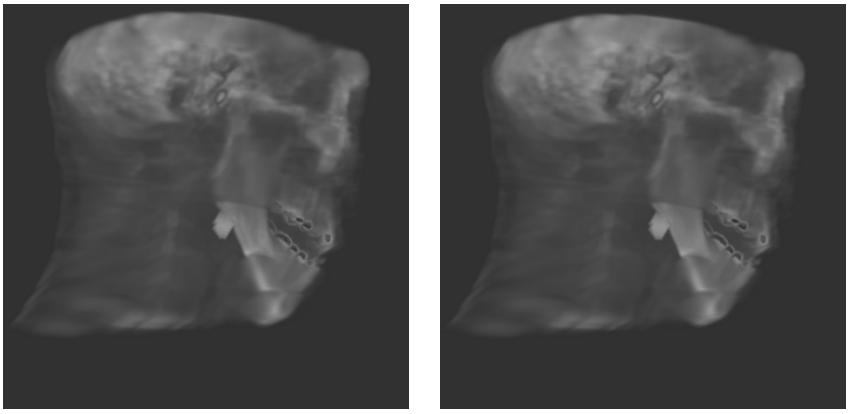
Se puede observar cómo el tumor aparece en la parte inferior del lado derecho de la mandíbula representada. Es entonces cuando el cliente detiene la transmisión y selecciona la ROI determinándola según las coordenadas espaciales de sus esquinas opuestas y que en este caso corresponden a las coordenadas $(x_1, y_1, z_1) = (58, 75, 130)$ y $(x_2, y_2, z_2) = (72, 170, 200)$. Estas coordenadas se transmiten al servidor que simplifica la transmisión de la región 477, transmitiendo solamente los datos correspondientes de la ROI seleccionada.

Las figuras 6.8(a) y 6.8(b) son reconstrucciones utilizando un 6.20% de regiones transmitidas completamente, y el 25% y el 100% respectivamente, de los datos de la ROI. La figura 6.8(b) se reconstruye con 476 regiones completas y 1837^2 regiones que corresponden a la ROI, es decir:

$$\begin{gathered} 478.196 \text{ Kb } + 1837 \times (16 + b_s) + \\ 1837 \times 32 + 1837 \times 16 \times ((y_2 - y_1 + 1) + (z_2 - z_1 + 1)) \text{ bits} = \\ 8926878 \text{ bits} = 1.06417 \text{ Mb}, \end{gathered}$$

(donde $b_s = 7$, $y_1 = 75$, $z_1 = 130$, $y_2 = 170$, $z_2 = 200$) y que corresponden al 14.12% de la imagen tridimensional codificada. Así pues, se alcanza el objetivo de visualizar la ROI de la imagen tridimensional y se hace ahorrando en la transmisión el 85.88% de los datos codificados.

[2] Descontando las regiones enviadas de los cortes $X = [58, 72]$ entre las 476 regiones iniciales, 1837 es el número de regiones no rechadas (por el umbral ϵ) que pertenecen al "corte" X y que quedan por transmitir.



(a) (b)

Figura 6.8: Reconstrucción de la ROI, (a) al 25% de los datos transmitidos de la ROI, (b) Al 100% de datos transmitidos de la ROI.

Capítulo 7

SVD y transformada Wavelet para una codificación eficiente de vídeo digital

7.1 Introducción

A medida que la tecnología ha ido avanzando, las imágenes digitales se han convertido en una pieza muy importante de nuestro día a día. Actualmente surgen cada vez más entornos gráficos orientados a múltiples y variadas aplicaciones en los que las imágenes juegan un papel prioritario. En los comienzos de la informática sólo había texto en los ordenadores, el código ASCII, con lo que la compresión de éste era sencilla y con buenos resultados. Por ejemplo, hace unos 30 años la compresión de una secuencia de imágenes se limitaba a comprimir el ancho de banda en las transmisiones de vídeo mediante métodos analógicos. Con la llegada de las computadoras digitales, los métodos de compresión analógicos fueron dejando paso a la compresión digital.

Hoy en día la compresión digital de imágenes es crucial para el desarrollo de la información multimedia (las computadoras se utilizan para la videoproducción, difusión, etc.). También es muy relevante el papel que desempeña en temas como videoconferencias, transmisión y custodia de imágenes médicas, o el control remoto en aplicaciones militares, etc.

La realidad es que el campo de la compresión digital de vídeo ha avanzado de manera considerable en estos últimos años. Esto en parte ha sido debido a la puesta en práctica de los trabajos teóricos que iniciara C.E. Shannon hacia 1940 [59] y que en la actualidad han derivado en la creación de diversos y muy eficaces estándares internacionales para la compresión digital de vídeo, como por ejemplo MPEG [47] y [53].

Considerando las secuencias de imágenes de vídeo como una imagen 3D, es posible aplicar los estudios ya realizados en este trabajo sobre este tipo de imágenes. La realización de diferentes experimentos con estas técnicas nos ha llevado a la conclusión de que no es viable el uso de este tipo de metodología cuando lleva asociada la interpolación. Todo esto nos llevó a realizar otro tipo de planteamientos a la hora de abordar el problema de la compresión de vídeo digital basados en la descomposición en valores singulares y en la aplicación de la transformada wavelet.

7.2 Códecs

En los códecs[1] de vídeo que utilizan la DCT (sección 3.2.1) cada imagen de la secuencia de entrada es dividida en bloques de $h \times h$ píxeles. El tamaño del bloque se escoge considerando los requisitos de compresión y calidad de la imagen. En general, a medida que el tamaño del bloque es mayor, la relación de compresión también resulta mayor, y esto es debido a que se utilizan más píxeles para eliminar las redundancias. Pero al aumentar demasiado el tamaño del bloque, la suposición de que las características de la imagen se conservan constantes no se cumple y ocurren algunas degradaciones en ella, como por ejemplo: bordes sin definir. Los resultados en la experimentación demostraron que el tamaño del bloque más conveniente es el de 8×8 píxeles. A continuación, los coeficientes de la transformada son cuantificados según un umbral para obtener el mayor número de ceros posibles. Para la cuantificación se utiliza una matriz de normalización estándar y se redondean los resultados a números enteros; este es el proceso donde se produce la pérdida de información. El paso siguiente consiste en reordenar en zigzag la matriz de coeficientes cuantificados para que la información quede ordenada de mayor a menor relevancia. Aparte, se hace una predicción intercuadros y una compensación de movimiento [38] y [47].

Una alternativa al uso de la DCT la encontramos en [60] donde se sugiere la idea de aplicar SVD en bloques de 8×8 ó 16×16 y reconstruir cada uno de ellos con el 21% de los valores singulares de ese bloque. El resto son truncados según una función de auto-correlación.

SVD tiene excelentes propiedades de compactación de la energía, lo que resulta interesante para un codificador de vídeo digital, pero el principal problema que tiene SVD es que genera más del doble de la información original. Salvo en el caso de tener filas/columnas linealmente dependientes, los valores singulares van a ser todos significativos y serán necesarios para una reconstrucción aceptable. ¿Habrá alguna manera de conseguir que las matrices que forman los canales de color de las imágenes de una secuencia de vídeo tengan filas/columnas linealmente dependientes?

[1] Abreviatura de Compresor-Descompresor. Describe una especificación desarrollada en software, hardware o una combinación de ambos capaz de transformar (en operaciones de transmisión, almacenaje o cifrado) un archivo, un flujo de datos o una señal y recuperarlos o descifrarlos del mismo modo para su reproducción o manipulación en un formato más apropiado para dichas operaciones.

Para contestar a esta pregunta hay que tener en cuenta el tipo de dato con el que se está trabajando. En las imágenes hay amplias zonas con valores de color o luminancia muy parecidos cuya diferencia es inapreciable para el ojo humano, es decir están muy correladas. Al aplicar una transformada wavelet, los coeficientes wavelets serán cercanos a cero en esas regiones y las técnicas de truncamiento nos permitirán eliminar esas pequeñas diferencias entre los píxeles. Obtendremos imágenes que son muy parecidas a la original pero con valores, ahora sí, iguales en esas zonas.

Esto contesta la anterior cuestión. Al aplicar una transformada wavelet y truncar los datos de los coeficientes wavelets, las pequeñas variaciones de una fila/columna de la matriz que conforma cada uno de los canales de color de la imagen con las de alrededor, desaparecen. De esta manera, para el ojo humano la imagen es la misma y al aplicar SVD aparecerán valores singulares con valor cero o muy próximo a él. La eliminación de esos valores dará un error mínimo en la reconstrucción y un ahorro en almacenamiento, pues tampoco tendremos que guardar sus vectores singulares correspondientes. Además, es de esperar que las imágenes no se degraden en exceso puesto que las características diferenciadoras como los bordes, tienen coeficientes wavelets que no se ven descartados en el truncamiento.

Motivados por todas estas ideas vamos a exponer los siguientes algoritmos que pueden ser utilizados como base para la elaboración de un códec para vídeo digital que se servirá tanto de la descomposición SVD como de la transformada wavelet [7].

7.2.1 Codificación

El siguiente algoritmo para la codificación de vídeo se basa en la siguiente idea: una secuencia de vídeo digital A no es más que una sucesión de I imágenes digitales $\{A^1, A^2, \ldots, A^I\}$ de tamaño $m \times n$, llamadas normalmente *frames* o *fotogramas*, y que a su vez están formadas por tres matrices de tamaño $m \times n$, una para cada canal de color

$$\left\{ \left(A_R^1, A_G^1, A_B^1\right), \left(A_R^2, A_G^2, A_B^2\right), \cdots, \left(A_R^I, A_G^I, A_B^I\right) \right\}.$$

Por tanto se pueden aplicar las ideas antes mencionadas a estas matrices para conseguir una reducción de los datos a almacenar.

Algoritmo 7.1 (Codificación) *Sea* $A = \left\{A^i\right\}_{i=1}^{I}$ *una secuencia de vídeo, donde* A^i *representa un fotograma.*

1. *Haz* $i = 1$.
2. *Separa* A^i *en los canales de color (matrices) que la conforman*

$$\begin{array}{ll} A_R^i & \textit{canal rojo (Red)} \\ A_G^i & \textit{canal verde (Green)} \\ A_B^i & \textit{canal azul (Blue)} \end{array}$$

es decir $A^i = \left\{A^i_j, \quad j = \{R, G, B\}\right\}$.

3. *Aplica una transformada wavelet, denotada por* TW, *a cada uno de los canales de color de cada imagen* A^i_j.

$$A^i_j \Longrightarrow TW(A^i_j) = \left\{\{s_1, \ldots, s_c\}^i_j, \{w_1, \ldots, w_d\}^i_j\right\} \quad con\ j = \{R, G, B\},$$

donde $\{s_f\}$ *son los coeficientes scaling con* $f = 1, \ldots, c$ *y* $\{w_g\}$ *son los coeficientes wavelets con* $g = 1, \ldots, d$.

4. *Trunca los coeficientes wavelets según un umbral* ε *dado.*

$$\tilde{w}_g = \begin{cases} w_g, & si\ |w_g| > \varepsilon \\ 0, & si\ |w_g| \leq \varepsilon \end{cases} \quad g = 1, \ldots, d.$$

5. *Reconstruye los canales de color de la imagen con la transformada wavelet inversa,* TW^{-1}, *para la obtención de la aproximación* $\tilde{A}^i_j$.

$$\tilde{A}^i_j = TW^{-1}\left(\left\{\{s_1, \ldots, s_c\}^i_j, \{\tilde{w}_1, \ldots, \tilde{w}_d\}^i_j\right\}\right) \quad con\ j = \{R, G, B\}.$$

6. *Aplica SVD dada por el teorema 2.3 a las aproximaciones* $\tilde{A}^i_j$ *obtenidas en el paso 5.*

$$(U^i_j)^T \tilde{A}^i_j V^i_j = diag\left(\sigma^i_{j_1}, \ldots, \sigma^i_{j_{t^i_j}}\right) = S^i_j, \quad t^i_j \leq p = \min\{m, n\}$$

con $j = \{R, G, B\}$.

7. *Elimina un determinado porcentaje* l *de los valores singulares, los menos significativos de los que no son nulos.*

$$\sigma^i_{j_k} = 0, \quad \forall\ k \geq l \times t^i_j \quad con\ j = \{R, G, B\}.$$

8. *Cuantifica mediante partes enteras los valores de las matrices* U^i_j, V^i_j *y* S^i_j *obtenidas mediante SVD y almacénalas. Aunque* S^i_j *es una matriz diagonal de dimensiones* $m \times n$, *sólo almacenaremos los valores singulares en un vector al que denotaremos, por analogía,* S^i_j.

$$\left.\begin{array}{l} U^i_j \Longrightarrow \tilde{U}^i_j = \left[\tilde{u}^i_{j_1}, \ldots, \tilde{u}^i_{j_{t^i_j}}\right] \\ V^i_j \Longrightarrow \tilde{V}^i_j = \left[\tilde{v}^i_{j_1}, \ldots, \tilde{v}^i_{j_{t^i_j}}\right] \\ S^i_j \Longrightarrow \tilde{S}^i_j = \left[\tilde{\sigma}^i_{j_1}, \ldots, \tilde{\sigma}^i_{j_{t^i_j}}\right] \end{array}\right\} \ con\ j = \{R, G, B\}.$$

9. *Si $i < I$, haz $i = i + 1$ y vuelve al paso 2.*

10. *Fin.*

Como se puede observar, hemos aplicado SVD a toda la imagen y no sólo a bloques de ella, puesto que esta descomposición puede ser tratada como una transformada global, ahorrando así tiempo de cálculo. También se evita de esta forma el efecto píxelado que aparece cuando aplicamos compresiones elevadas a los datos. Este efecto se produce cuando se reconstruye cada bloque en los que se divide la imagen original con pocos coeficientes de la transformada obteniendo bloques con una única intensidad de gris o un único color, el de la media de los datos del bloque.

Por otra parte debemos comentar que como los vectores singulares son ortogonales, toman valores menores, en valor absoluto, que la unidad y entonces la cuantificación de las matrices U y V se realiza multiplicando sus elementos por un factor y luego tomando la parte entera del resultado. Después, al reconstruir las imágenes mediante SVD, dividiremos por dicho factor.

7.2.2 Decodificación

El algoritmo de decodificación o reconstrucción de cada uno de los fotogramas que componen la secuencia de vídeo utiliza la definición de SVD y el producto de matrices que aparece en la descomposición SVD, es decir, $A = USV^T$.

Algoritmo 7.2 (Decodificación) *Sea* $\left\{ \left[\tilde{U}_j^i, \tilde{V}_j^i, \tilde{S}_j^i \right] \right\}_{i=1}^{I}$, *con* $j = \{R, G, B\}$, *los datos de la secuencia de vídeo obtenidos y almacenados por el algoritmo 7.1.*

1. *Haz* $i = 1$.
2. *Lee* $\left[\tilde{U}_j^i, \tilde{V}_j^i, \tilde{S}_j^i \right]$, *con* $j = \{R, G, B\}$.
3. *Reconstruye* $\bar{A}_j^i$ *según el producto*

$$\bar{A}_j^i = \tilde{U}_j^i \; \tilde{S}_j^i \; (\tilde{V}_j^i)^T, \;\; con \; j = \{R, G, B\}.$$

4. *Si $i < I$, haz $i = i + 1$ y vuelve al paso 2.*
5. *Fin.*

No obstante, los productos de matrices del paso 3, son simplemente una suma de los productos de los vectores singulares ponderados por los valores singular asociados

$$\bar{A}_j^i = \sum_{k=1}^{t_j^i} \tilde{\sigma}_{j_k}^i \, \tilde{u}_{j_k}^i \left(\tilde{v}_{j_k}^i \right)^T, \;\; \text{con } j = \{R, G, B\},$$

lo que hace que el algoritmo de decodificación sea muy rápido. Además, el corolario 2.6 nos asegura que ésta es la mejor reconstrucción posible con matrices de ese rango, salvo cuantificación.

7.3 Resultados

En el experimento que realizamos se ha utilizado una secuencia de vídeo de 678 fotogramas que representan 27120 milisegundos de vídeo a razón de 25 fotogramas por segundo. Cada fotograma está compuesto por 3 matrices, cada una de ellas para un canal de color (rojo, verde y azul) de dimensiones 420×720. La utilización de SVD para cualquier canal de color genera 420 valores singulares mayores que 0.1, es decir, el 100% de los valores singulares son significativos.

Para el algoritmo 7.1 se realizaron previamente diferentes experimentos con el fin de determinar los parámetros óptimos de cada uno de los pasos que lo componen y concretamente, en los los pasos 3 y 5 de dicho algoritmo se realizaron pruebas con diferentes familias de wavelets (Haar, Daubechies y biortogonales de splines) y número de escalas. Los mejores resultados se obtuvieron aplicando el wavelet biortogonal 5.5. Las pruebas para determinar el número de escalas óptimas reflejaron que la calidad es mejor cuando se realizan menos escalas, por lo que se determinó hacer una única escala [7]. El truncamiento de los datos de las escalas mayores elimina datos que a esos niveles tienen una información muy relevante y su ausencia degrada mucho las imágenes de las secuencias.

Para el paso 4 del algoritmo 7.1 se decidió tomar un umbral $\varepsilon = 50$. Los experimentos realizados nos indicaron que el truncamiento para valores más altos de dicho umbral, nos ofrecía peor calidad de imagen. Al quitar esos detalles, aunque sí se consigue una mayor compresión, obteníamos imágenes de peor calidad visual. Por el contrario, con valores más pequeños no se mejora sustancialmente la calidad de la imagen pero aumentan tanto el tiempo de cálculo como la cantidad de datos a almacenar.

Para el paso 7 del algoritmo 7.1 observamos que si bien no se puede aplicar un truncamiento de los valores singulares al 21% como se propone en [60], debido a una degradación considerable de la imagen, sí es posible descartar una gran parte de ellos manteniendo una calidad aceptable. Las pruebas realizadas indican que el mejor ratio de calidad-coste es cuando l es aproximadamente el 50% de los valores singulares. El utilizar más valores singulares no mejora sustancialmente la calidad de la imagen, pero sí aumenta el coste de almacenamiento de ésta, al tener que guardar sus vectores singulares asociados. Por el contrario, con menos valores singulares la imagen comienza a degradarse teniendo una peor calidad de visión.

Despues de realizar todas las pruebas pertinentes, podemos afirmar que para conseguir la mejor ratio calidad-coste en el almacenamiento de los fotogramas ya codificados, se debe trabajar en la codificación con los siguientes parámetros:

- Wavelet: Biortogonal 5.5 con 1 escala.
- Umbral de truncamiento: $\varepsilon = 50$.
- Porcentaje de eliminación de valores singulares: $l = 50\%$.

La aplicación de estos parámetros nos obliga a tener que almacenar una media de 52 valores singulares, junto con sus respectivos vectores singulares, por

Figura 7.1: Fotograma obtenido de la decodificación de nuestra secuencia de vídeo utilizando los algoritmos 7.1 y 7.2 con 52 valores singulares.

Figura 7.2: Ampliación de la figura 7.1. No hay píxelado.

Figura 7.3: Reconstrucción con el método propuesto en [60] en bloques de 16×16. Se aprecia el efecto de píxelado en el perfil del rostro del personaje.

Figura 7.4: Ampliación de la figura 7.3. Detalle del píxelado.

canal de color de cada fotograma de la secuencia de vídeo. Esto supone trabajar con sólo el 12.38% de los valores singulares originales. Su reconstrucción para un único fotograma de la secuencia de vídeo utilizando el algoritmo 7.2 es la que se puede ver en la figura 7.1. Se puede observar mediante una ampliación (figura 7.2) cómo no se llegan a producir los típicos efectos de píxelado, ya que la transformación utilizada es una transformada global de la imagen.

El método propuesto en [60] con bloques 16×16 nos ofrece la reconstrucción que podemos ver en la figura 7.3, para un único fotograma de la secuencia de vídeo. En ella podemos apreciar los efectos de píxelado. Esto es debido a que la reconstrucción se efectuó con solo 3 valores singulares por bloque. Una ampliación de la imagen (figura 7.4) nos ofrece este efecto con más detalle.

La tabla adjunta nos da una comparación con los datos que tendríamos que almacenar para un único canal de un fotograma de la secuencia de vídeo. Hemos comparado la cantidad de información almacenada en el caso de tener la imagen completa o después de realizar: una descomposición en valores singulares; el método propuesto en [60] considerando primero bloques de 8×8 y después bloques de 16×16; y por último utilizando el algoritmo de codificación 7.1 (TW + SVD) [7].

Método	**N^o de datos almacenamiento/canal**	**%**
Imagen completa	$420 \times 720 = 302400$	100
SVD	$420 \times 420 + 720 \times 720 + 420 = 695220$	229.90
SVD al 21% en 8×8	$9360 \times 8 + 9360 \times 8 + 9360 = 159120$	52.62
SVD al 21% en 16×16	$3510 \times 16 + 3510 \times 16 + 3510 = 115830$	38.30
TW + SVD	$52 \times 420 + 720 \times 52 + 52 = 59332$	19.62

7.4 Conclusiones

En la sección 7.2 de este capítulo se han propuesto dos algoritmos, uno de codificación y otro de decodificación, para la compresión de secuencias de vídeo digital basados en la descomposición en valores singulares y la utilización de wavelets.

Como se puede apreciar, los algoritmos propuestos en este capítulo proporcionan una reconstrucción de calidad utilizando una menor cantidad de información. En una comparación con lo que es la secuencia original y considerando solamente datos a almacenar, el método propuesto tiene una compresión del 19.62%, esto es aproximadamente de $5:1$ frente al método propuesto en [60] que es de $2:1$ ó de $3:1$.

Otra mejora que observamos a lo largo de los experimentos es que al tratarse de una transformada global de la imagen no hay problemas a la hora de tener que almacenar las posibles posiciones de los bloques lo que ahorra un almacenamiento adicional y se evita también el efecto de píxelado. Además, el algoritmo 7.2 de decodificación únicamente consiste en una suma de productos de los vectores singulares ponderados por el valor singular asociado, lo que lo hace muy rápido en su ejecución.

Hay que tener en cuenta que estos algoritmos son de codificación *intra-frame*, es decir, se aplican a cada fotograma de la secuencia independientemente de los demás. Para futuras líneas de investigación queda la aplicación de codificadores de datos como los códigos de longitud variable, los algoritmos de predicción y compesación de movimiento[2] y la introducción de predictores *inter/intra-frames* adaptativos[3] con el fin de aumentar los ratios de compresión y hacer funcionales los algoritmos 7.1 y 7.2.

[2] La compensación de movimiento ocurre cuando un bloque o conjunto de bloques se repite en el siguiente fotograma pero no en la misma posición (movimiento de cámara o de objeto), entonces el sistema almacena un vector que indica donde desplazar el bloque al formar el siguente fotograma.

[3] La predicción inter-frames sirve para codificar fotogramas consecutivos (con el mínimo número de bits posibles) explotando la correlación temporal que pueda existir entre ellos.

Capítulo 8

Conclusiones y reflexiones finales

8.1 Introducción

En este trabajo se ha intentado, entre otras cosas, presentar un novedoso y sobre todo, sencillo y eficaz proceso de codificación, organización, almacenamiento y transmisión de imágenes digitales 3D (incluido vídeo digital) y muy especialmente de subimágenes o regiones de interés de éstas.

Los métodos tradicionales de transmisión de datos son secuenciales, enfoque (éste) con una importante desventaja en cuanto a imágenes se refiere: hasta que no se recibe la imagen al completo no podemos observarla. ¿Y si por un error, la imagen recibida no es la deseada? Éste y otros motivos son los que nos motivaron a la realización de este trabajo desde la perspectiva de la transmisión progresiva. Pero desde el comienzo de este trabajo hemos estado especialmente interesados en una situación muy particular: cuando durante la transmisión progresiva de una imagen el cliente observa uno o varios detalles de interés y entonces, después de seleccionarlos, solamente desea detalle o mejora de dichas regiones (transmisión bajo demanda).

Y bajo esta motivación y con la finalidad primordial de que por un lado (el lado del cliente) la reconstrucción o visualización fuese de la forma más sencilla posible (una multiplicación de vectores) y por otro, no incurrir en el envío de información redundante, de optimizar nuestros recursos en lo que a transmisión de datos se refiere, realizamos este trabajo.

Para todo ello elegimos basicamente la técnica matricial basada en la SVD; a veces con ayuda de la interpolación, otras sin ella y con la ayuda también de la transformada wavelet, pudimos realizar diferentes pruebas sobre codificación, transmisión y posterior reconstrucción de imágenes digitales 2D, 3D y secuencias de vídeo.

8.2 Imágenes 2D

Realmente nuestros comienzos fueron poner en contacto o más bien comparar nuestro método SVD con los potentes métodos de compresión de imágenes 2D como son JPEG y JPEG2000. Tras realizar unos pequeños cambios para adecuar las condiciones de la comparación y que ésta se realizase bajo un objetivo común, transmisión progresiva de ROIs, pudimos observar como tanto JPEG y JPEG2000 son dos métodos más eficaces, sobre todo en la obtención de altas tasas de compresión. Este hecho era de esperar, conocida la desventaja del SVD a la hora transformar la imagen, con el aumento a casi el doble de los datos originales.

Aún así pudimos observar cómo los procesos de reconstrucción de la imagen o de sus ROIs (única en el caso del JPEG2000) son **costosos**, tanto en lo que se refiere al tiempo de cálculo como a la redundancia en la transmisión de datos. Si a esta situación le añadimos una cuestión que se nos planteó durante la utilización del JPEG2000 en los experimentos: ¿Por qué siendo un método tan bueno, si no el mejor a la hora de comprimir imágenes, y teniendo la posibilidad de tratar ROIs, no es tan popular o más utilizado, sobre todo en Internet, que su "pariente" el JPEG? ¿Temas económicos o de patentes son el motivo? Todo ello nos lleva a la posibilidad, en un futuro, de poder seguir experimentando con imágenes 2D utilizando SVD siempre y cuando podamos llegar a resolver el verdadero inconveniente de la transformada SVD: la gran cantidad de datos que genera.

8.3 Imágenes 3D

Nuestro siguiente objetivo fueron las imágenes 3D (paso previo al vídeo digital) y comenzamos utilizando la interpolación matricial lineal por trozos de Newton. Su uso fue debido principalmente a estudios previos a este trabajo realizados con imágenes 3D y que dieron buenos resultados. Enseguida observamos que el uso de la interpolación nos obligaba, por lo menos al principio de las transmisiones, a utilizar la norma Frobenius (la misma que se había utilizado en trabajos anteriores).

Pero con la utilización ahora de la SVD, el cálculo de valores singulares y su relación tan explícita con la norma-2 de una matriz, nuestra nueva elección estaba clara. Cambiar y utilizar la norma-2, así como evitar todo tipo de interpolación nos ahorró mucho sobre todo, el tiempo de reconstrucción en el servidor así como el tiempo en decidir qué región de la imagen debíamos transmitir en todo momento (la ordenación de los valores singulares nos lo daba hecho). Considerando los resultados del experimento efectuado entonces, solamente cabe recordar su notable mejoría respecto a cuando utilizabamos interpolación.

8.3.1 ROIs tridimensionales

Nuestro siguiente paso fue detenernos en el **detalle**. Queríamos tener más y mejor detalle de algunas imágenes, sobre todo imágenes médicas 3D, donde una pequeña parte de la imagen, lo puede ser *todo*. Y también, de paso, comprobar cómo podría afectar una cuantificación adecuada a nuestra gran cantidad de datos SVD generados en todo nuestro proceso.

Propusimos entonces un nuevo algoritmo para la codificación adaptativa con pérdida de imágenes digitales 3D basado, como hasta ahora, en la SVD de matrices y aplicado sobre todo a ciertas regiones de interés seleccionadas de las imágenes 3D. La principal novedad del algoritmo/s (codificación, transmisión y reconstrucción) que presentamos era que la selección de las ROIs, del detalle que deseabamos observar podía conseguirse en cualquier momento, sin ninguna nueva codificación de la imagen y por tanto sin repetición en la transmisión de datos.

Varias ROIs a la vez, sin necesidad de transformar la imagen una y otra vez, sin reincidencia en el envío de datos y una reconstrucción muy sencilla por parte del cliente son en su conjunto razones más que poderosas para darle sentido a este trabajo. Si además le añadimos nuestro interés en cuantificar adecuadamente todos nuestros datos para reducir el tamaño de éstos y en observar la buena calidad de nuestras reconstrucciones, incluso en los primeros pasos de la transmisión, podemos concluir que estamos satisfechos con nuestra aportación.

Para un futuro podríamos insistir más aún en la reducción del tamaño de los vectores singulares. Propiedades como su ortogonalidad o que sus componentes sean todas ellas, en valor absoluto, menores que la unidad, podrían en un futuro ser apropiadas para realizar una óptima compresión de cada vector por si sólo y reducir considerablemente el tamaño en bits de éstos. Y aún así y considerando que al buscar el detalle el tamaño de dichos vectores se reduce considerablemente, encontramos también un campo por delante para investigar cuando podamos trabajar con imágenes muchísimo más grandes y con resoluciones también más amplías y donde el detalle será muy importante. Estamos hablando, por ejemplo, de las imágenes digitales en astronomía, captadas sobre todo por telescopios de altísima resolución situados en algún lugar del espacio y donde el problema de la comunicación-transmisión podría sencillamente dejar de ser eso, un problema.

8.4 Vídeo digital

Y para finalizar quisimos adentrarnos de forma muy modesta en el mundo del séptimo arte. Considerando las secuencias de imágenes de vídeo como una imagen 3D, nos fue posible aplicar los estudios ya realizados y realizar diferentes experimentos con estas técnicas. Ésto nos llevó a otro tipo de planteamiento a la hora de abordar el problema de la compresión de vídeo. Para utilizar SVD necesitabamos previamente de la aplicación de la transformada wavelet.

Se propusieron dos algoritmos, uno de codificación y otro de decodificación

para formalizar un sencillo códec. Pero teniendo en cuenta la complejidad que posee intrínsecamente una película de video (25 frames por segundo, decodificación en tiempo real, utilización, compresión y ajuste del sonido, etc.) y la existencia de códecs muy eficientes para vídeo (MPEG-4, Div*X*, WMV, etc.) nos llevó a desistir en realizar un trabajo mucho más amplio sobre este tema. Hay que tener en cuenta que nuestro códec es de codificación *intra-frame*, es decir, se aplica a cada fotograma de la secuencia independientemente de los demás. Así pues, para futuras líneas de investigación quedaría la aplicación de codificadores de datos como los códigos de longitud variable, los algoritmos de predicción de movimiento y la introducción de predictores *inter/intra-frames* adaptativos con el fin de aumentar los ratios de compresión y hacer funcional nuestro códec.

Bibliografía

[1] E. Aboufadel, S. Schlicker, "*Discovering Wavelets*", Wiley-InterScience, 1999.

[2] P. Akhtar, M.I Bhatti, T.J. Ali, M.A. Muqeet, "*Significance of ROI Coding using MAXSHIFT Scaling applied on MRI Images in Teleradiology - Telemedicine*", J. Biomedical Science and Engineering, pp.110-115, 2008.

[3] P. Alonso, E. Defez, A. Hervás, A.G. Law, J. Peinado, R.J. Villanueva, "*Orthogonal matrix polynomials for progressive transmission of 3-D images*", 5th International Symposium on Orthogonal Polynomials (OPSFA), September 1999.

[4] G.K. Anastassopoulos, A.N. Skodras. "*JPEG2000 ROI coding in medical imaging applications*", Visualization Imaging and Image Processing (VIIP 2002, ISBN 0-88986-354-3), Marbella, España, 2002

[5] I. Baeza, A.G. Law, J.A. Verdoy, J. Villanueva-Oller, R.J. Villanueva, "*SVD and matrix polynomial interpolation for lossy progressive transmission of 3D images*", Focus On Robotics And Intelligent Systems Research Nova Publishers (ISBN 1-59454-357-7) 2005.

[6] I. Baeza, A.G. Law, J. Villanueva-Oller, R.J. Villanueva, "*Progressive transmission of images: Adaptive best strategies*", Mathematical and Computer Modelling Vol. 41, pp. 1325-1339, 2005.

[7] I. Baeza, J.A. Verdoy, R.J. Villanueva, "*Descomposición en Valores Singulares y Transformada Wavelet para una compresión y reconstrucción de vídeo eficiente*", Congreso de Métodos Numéricos en Ingeniería 2005 SEMNI (ISBN 84-95999-74-9), Granada, 2005.

[8] I. Baeza, J. Villanueva-Oller, R.J. Villanueva, S. Díez, "*Envío y reconstrucción progresiva de imágenes digitales 3D utilizando interpolación matricial*", III Congreso Internacional sobre Métodos Numéricos en Ingeniería y Ciencias Aplicadas, Monterrey, México, 2004.

[9] I. Baeza, S. Diez, R.J. Villanueva, J. Villanueva-Oller, "*Interpolación Matricial de Newton para la Transmisión Progresiva Adaptiva de Imágenes*

Médicas Tridimensionales", V Jornadas de Investigación y Fomento de la Multidisciplinariedad, Valencia, España, 2003.

[10] J.A. Ball, I. Gohberg, L. Rodman,"*Interpolation of Rational Matrix Functions*", Birkhäuser, 1990.

[11] M.R. Banham, A.K. Katsaggelos, "*Digital image restoration*", IEEE Signal Processing Magazine, Vol. 14, pp. 24-41, 1997.

[12] R.A. Brown, H. Zhu, J.R. Mitchell, "*Distributed vector processing of the S-transform for medical applications*", Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering, Vol. 2, pp. 1129-1133, 2002.

[13] R.C. Calderbank, I. Daubechies, W. Sweldens, B.L. Yeo, "*Wavelet Transforms that Map Integers to Integers*", Applied and Computational Harmonic Analysis, Vol. 5, no. 3, pp. 332-369,1998.

[14] C. Christopoulos, A. Skodras, A. Ebrahimi, "*The JPEG 2000 still image coding system: An overview*", IEEE Transactions on Consumer Electronics. Vol. 46 pp. 1103-1127, 2000.

[15] C.K. Chui, "*An Introduction to Wavelets*", Academic Press, New York, 1992.

[16] C.K. Chui, "*Wavelets: A Mathematical Tool for Signal Analysis*", SIAM, 1997.

[17] C.K. Chui, "*Wavelets: A Tutorial in Theory and Applications*", Academic Press, New York, 1992.

[18] A. Cohen, "*Ondelettes, analyses multiresolutions et filtres miroir en quadrature*", Ann. Inst. H. Poincaré, Anal. non linéaire, Vol. 7, pp. 439-459, 1990.

[19] A. Cohen, "*Ondelettes, analyses multiresolutions et traitement numerique du signal*", Ph. D. Thesis, Université Paris, Dauphine, 1990.

[20] A. Cohen, I. Daubechies, J.C. Feauveau, "*Biorthogonal Bases of Compactly Supported Wavelets*", Communications on Pure and Applied Mathematics, vol XLV, pp. 485-560, 1992.

[21] A. Cohen, I. Daubechies, P. Vial, "*Wavelets and fast wavelets transform on the interval*", AT & T Bell Laboratories, preprint, 1992.

[22] A. Cohen, R.D. Ryan, "*Wavelets and Multiscale Signal Processing*", Chapman and Hall, 1995.

[23] R.R. Coifman, M.V. Wickerhauser, "*Entropy Based Algorithms for Best Basis Selection*", IEEE Transactions of Information Theory, Vol. 38, no. 2, pp. 713-718, 1992.

[24] I. Daubechies, "*Orthonormal Bases of Compactly Supported Wavelets*", Communications. on Pure and Applied Mathematics, Vol 41, pp. 909-996, 1988.

[25] I. Daubechies, "*Ten lectures on wavelets*", CBMS-NSF Lecture Notes no. 61, SIAM, 1992.

[26] I. Daubechies, T. Paul, "*Wavelets-some applications*", Proceedings of the International Conference on Mathematical Physics, eds. M. Mebkout and R. Sénéor, World Scientific, pp. 675-686, Marseille, France, 1987.

[27] E. Defez, A. Hervás, A.G. Law, J. Villanueva-Oller, R.J. Villanueva, "*Progressive transmission of images: PC-based computations using orthogonal matrix polynomials*", Mathematical and Computer Modelling, Vol. 32, pp. 1125-1140, 2000.

[28] E. Defez, A.G. Law, J. Villanueva-Oller, R.J. Villanueva, "*Matrix cubic splines for progressive transmission of images*", Journal of Mathematical Imaging and Vision, Vol. 17, pp. 41-53, 2002.

[29] E. Defez, A.G. Law, J. Villanueva-Oller, R.J. Villanueva, "*Matrix Newton interpolation and progressive 3D imaging: PC-based computation*", Mathematical and Computer Modelling, Vol. 35, no. 3-4, pp. 303-322, 2002.

[30] P. Dev, "*Imaging and visualization in medical education*", IEEE Computer Graphics and Applications, pp. 22-31, 1999.

[31] R.S. Dilmaghani, A. Ahmadian, M. Ghavami, A.H. Aghvami, "*Progressive medical image transmission and compression*", IEEE Signal Processing Letters, 11-10 pp. 806-809, 2004.

[32] J. Froment, S. Mallat, "*Wavelets: a tutorial in theory and applications*", Vol. 2, Academic Press Professional, Inc., New York, 1993.

[33] G.H. Golub, C.F. Van Loan, "*Matrix Computations*", Johns Hopkins Univ. Press, Baltimore, MA., 1989.

[34] R.C. González, R.E. Woods, "*Tratamiento digital de imágenes*", Addison-Wesley Iberoamericana, S.A., Massachusetts, 1996.

[35] P.C. Hansen, M. Jacobsen, J.M. Rasmussen, H. Sørensen, "*The PP-TSVD algorithm for image reconstruction problems*", eds. P.C. Hansen, B.H. Jacobsen, K. Mosegaard, "*Methods and applications of inversion*", Lecture Notes in Earth Science, Vol. 92, pp. 171-186, Springer, Berlin, 2000.

[36] P.C. Hansen, S.H. Jensen, "*FIR filter representation of reduced-rank noise reduction*", IEEE Transactions on Signal Processing, Vol. 46, pp. 1737-1741, 1998.

[37] N. Johson, S. Jagodia. "*Exploring steganography: seeing the unseen*", Computer, pp. 26-34, 1998.

[38] F. Kelly, A. Kokaram, "*Fast Image Interpolation for Motion Estimation using Graphics Hardware*", IS&T/SPIE Electronic Imaging - Real-Time Imaging VIII, pp. 184-194, San Jose, California, USA, 2004.

[39] Y.S. Kim, W.Y. Kim, "*Reversible decorrelation method for progressive transmission of 3D medical image*", IEEE Transactions on Medical Imaging, Vol. 17, no. 3, pp. 383-394, 1998.

[40] E. Kofidis, N. Kolokotronis, A. Vassilarakou, S. Theodoridis, D. Cavouras, "*Wavelet-based medical image compression*", Future Generation Computer Systems, Vol. 15, pp. 223-243, 1999.

[41] T. Lehmann, C. Gönner, K. Spitzer, "*Interpolation Methods in Medical Image Processing*", IEEE Transactions on Medical Image Processing, Vol. 18(11), pp. 1049-1075, 1999.

[42] T.E. Liptay, J. Barron, I. Gargantini, "*A WWW interactive progressive local image transmission system*" Ph.D., Department of Computer Science, The University of Western Ontario London, Canada, 2000.

[43] S.G. Mallat, "*A theory for multiresolution signal decomposition: The Wavelets representation*", IEEE Transactions on Pattern Recognition and Machine Intelligence, Vol. 11, no. 7, pp. 674-693, 1989.

[44] S.G. Mallat, "*Multiresolution approximation and Wavelets*", Transactions of the American Mathematical Society, Vol. 315, pp. 69-88, 1989.

[45] G. Menegaz, J.P. Thiran, "*Lossy to lossless object-based coding of 3-D MRI data*", IEEE Transactions on Image Processing, Vol. 11-9, pp. 1053-1061, 2002.

[46] Y. Meyer, "*Wavelets and Operators*", Translation of "*Ondelettes et opérateurs*" (Hermann, 1990), Cambridge University Press, 1993.

[47] J. L. Mitchell, W.B. Pennebaker, C. Fogg, D.J. LeGall, "*MPEG video compression standard*", Chapman Hall, 1996.

[48] J. Morlet, "*Sampling theory and wave propagation*", NATO ASI Series, Vol. 1, Issues in Acoustic signal/Image processiong and recognition, ed. C.H. Chen, Springer-Verlag, Berlin, pp. 233-261, 1983.

[49] J. Morlet, G. Arens, I. Fourgeau, D. Giard, "*Waves propagation and sampling theory*", Geophysics, Vol. 47, pp. 203-236, 1982.

[50] G. Nakos, D. Joyner, "*Álgebra Lineal con Aplicaciones*", International Thomson Editores, 1998.

[51] N. Panagiotidis, S. Kollias, "*Region-Of-Interest Based Compression of Magnetic Resonance Imaging Data*", Proceedings IWISP 1996, pp. 31-35.

[52] F. Pedroche, "*On some capabilities of the SVD expansion to handle images*", WSEAS Transactions on Mathematics Vol. 1-2, pp. 67-70, 2002.

[53] W. B. Pennebaker, "*JPEG Still image data compression standard*", Van Nostrand Reinhold, 1993.

[54] L. Prasad, S.S. Iyengar, "*Wavelet Analysis with Applications to Image Processing*", CRC Press. 1997.

[55] R. Pratap, "*Getting started with Matlab:Version 6 A quick introduction for scientists and engineers*", Oxford University Press, 2002.

[56] D.S. Taubman, M.W. Marcellin, "*JPEG2000: Image compression fundamentals, standards and practice*", Kluwer, 2002.

[57] W. Schroeder, K. Martin, "*The VTK user's guide*", Kitware Inc, 1999.

[58] W. Schroeder, K. Martin, B. Lorensen, "*The visualization toolkit. An object-oriented approach to graphics*", Prentice-Hall, 1997.

[59] C.E. Shannon , "*A mathematical theory of communication*", Bell System Technical Journal, Vol. 27, pp. 379-423, 623-656, 1948.

[60] A. Sharifinejad, "*Utilizing Autoregressive Truncated Singular Value Decomposition algorithm for obtaining more efficiently Compressed Images*", Image and Vision Computing New Zealand pp. 199-204, Massey University, Palmerston North, New Zealand, 2003.

[61] T. Sigitani, Y. Iiguni, H. Maeda, "*Progressive cross-section display of 3D medical images*", Physics in Medicine and Biology, Vol. 44, no. 6, pp. 1565-1577, 1999.

[62] E.J. Stollnitz, T.D. DeRose, D.H. Salesin, "*Wavelets for computer graphics: A primer, part 1*", IEEE Computer Graphics and Applications, Vol. 15, no. 3, pp. 76-84, 1995.

[63] G. Strang, T. Nguyen, "*Wavelets and Filter Banks*", Wellesey-Cambridge Press, 1996.

[64] K. Tzou, "*Progressive image transmission: A review and comparison of techniques*", Optical Engineering, Vol. 26, pp. 581-589, 1987.

[65] M. Unser, "*A perfect fit for signal and image processing*", IEEE Signal Processing Magazine, 16(6), pp. 22-38, 1999.

[66] A.Vlaciu, S. Lungu, N. Crisan, S.Persa. "*New compression techniques for storage and transmission of 2-D and 3-D medical images*" In Advanced Image and Video Communications and Storage Technologies, volume 2451, pages 370-377, Amsterdam, 1995.

[67] J.S. Walker, "*A primer on wavelets and their scientific applications*", Chapman & Hall/CRC, 1999.

[68] G. Wallance, "*The JPEG Still Picture Compression Standard*", IEEE Tansactions on Consumer Electronics, Volume 38, Issue 1, pp. 28-34, Feb 1992.

[69] S. Wolfram, "*The Mathematica book*", Wolfram Media Inc., 1996.

[70] W. Wrazidlo, H.J. Brambs, W. Lederer, S. Schneider, B. Geiger, C. Fischer, "*An alternative method of three-dimensional reconstruction from two-dimensional CT and MR data sets*", Medical & Biological Engineering & Computing, Vol. 38, no. 2, pp. 140-149, 2000.

[71] X. Wu, T. Qiu, "*Wavelet coding of volumetric medical images for high throughput and operability*", IEEE Transactions on Medical Image Processing 24-6, pp. 719-727, 2005.

[72] H. Zhu, R.A. Brown, R.J. Villanueva, J. Villanueva-Oller, M.L. Lauzon, J.R. Mitchell, A.G. Law, "*Progressive imaging: S-transform order*", ANZIAM J. 45(E) C1002-C1016, 2004.

[73] *http://www.acm.org/crossroads/xrds6-3/sahaimgcoding.html* Tutorial "on line" sobre wavelets de la Association for Computing Machinery, firmado por Subhasis Saha.

[74] *http://www.cs.wpi.edu/∼matt/courses/cs563/talks/Wavelets_presentation* Explicación y ejemplos de codificación por wavelets de Haar de Benj Lipchak, del Computer Science Department, Universidad Politécnica de Worcester, USA.

[75] *http://www.imaginis.net/ct-scan* Información general sobre sistemas de CT suministrada por la web comercial *Imaginis*, versada sobre diagnósticos de enfermedades de mama.

[76] *http://www.kakadusoftware.com* Software Kakadu. Marco general para el codificador Kakadu basado en el JPEG2000.

[77] *http://www.netlib.org/blas/* Basic Linear Algebra Subprograms (BLAS).

[78] *http://www.scriptics.com* Software Tool Command Language (TCL), lenguaje de *scripting* o programa ayuda para ejecutar archivos que contienen ciertas instrucciones. Utilizado para preparar las representaciones gráficas con VTK.

[79] *http://www.iv.optica.csic.es/* Desarrollo de Algoritmos de Optimizacion Visual en Imágenes Médicas.

[80] *http://www.profundizar.com.ar/compresion* Proyecto final de carrera sobre la compresión de imágenes y de video.

[81] *http://www.ace.ual.es/* Proyecto final de carrera sobre la transmision progresiva de imágenes con JPEG2000.