Universidad Nacional de General Sarmiento

DOCTORADO EN CIENCIA Y TECNOLOGIA

Supercompresión de video y aplicaciones

Trabajo de tesis para optar por el título de Doctor en Ciencia y Tecnología
de la Universidad Nacional de General Sarmiento

Autor: **Mario Mastriani**

Director: Dra. Juliana Gambini

Fecha: 21 de Octubre de 2011

Universidad Nacional de General Sarmiento

DOCTORADO EN CIENCIA Y TECNOLOGIA

FORMULARIO "E"
TESIS DE POSGRADO

Niveles de acceso al documento autorizados por el autor

El autor de la tesis puede elegir entre las siguientes posibilidades para autorizar a la UNGS a difundir el contenido de la tesis: a)

a) Liberar el contenido de la tesis para acceso público.
b) Liberar el contenido de la tesis solamente a la comunidad universitaria de la UNGS:
c) Retener el contenido de la tesis por motivos de patentes, publicación y/o derechos de autor por un lapso de cinco años.

a. Título completo del trabajo de Tesis: Supercompresión de video y aplicaciones

b. Presentado por: Mastriani Mario

c. E-mail del autor: mmastriani@untref.edu.ar

d. Estudiante del Posgrado: Doctorado en Ciencia y Tecnología

e. Institución o Instituciones que dictaron el Posgrado: Instituto de Ciencias, Universidad Nacional de General Sarmiento

f. Para recibir el título de (consignar completo):
   a) Grado académico que se obtiene: Doctorado
   b) Nombre del grado académico: Doctorado en Ciencia y Tecnología

g. Fecha de la defensa: 21 / 10 / 2011
   día mes año

h. Director de la Tesis: Gambini, Juliana

i. Tutor de la Tesis: Acosta Rodríguez, Gabriel

j. Colaboradores con el trabajo de Tesis: -

k. Descripción física del trabajo de Tesis: 358 páginas, 36 tablas, 210 figuras y 10 algoritmos.

l. Alcance geográfico y/o temporal de la Tesis: El alcance geográfico es internacional, mientras se encuentre en vigencia la necesidad de comprimir imágenes.

m. Temas tratados en la Tesis: Blur, compresión de imágenes y video, interpolación bidimensional, JPEG/JPEG2000, nitidez, realce, super-resolución, y TV Digital.

Universidad Nacional de General Sarmiento

DOCTORADO EN CIENCIA Y TECNOLOGIA

n. Resumen:

Este trabajo de tesis propone el uso de la técnica conocida como super-resolución en compresión de imágenes, sean estas imágenes fijas, cuadros de un video, TV Digital de alta definición, Cine Digital, y cualquier otro tipo de imagen a comprimir para su posterior almacenamiento y/o transmisión.

Mediante la técnica de super-resolución se disminuye precisamente la resolución de la imagen en el codificador con lo cual se baja notablemente su peso en bytes, para luego comprimirlo mediante cualquier técnica conocida y almacenarlo y/o transmitirlo. Posteriormente, y ya en el decodificador, se descomprime la imagen y se restaura su resolución y nitidez mediante técnicas de super-resolución.

Originalmente, las técnicas de super-resolución son empleadas en aquellos casos donde se desea obtener una imagen de mayor nitidez y resolución a partir de un conjunto de imágenes de baja resolución y poco nítidas, las cuales son el resultado de una notable diferencia de velocidad entre el objeto a capturar en pantalla vs la velocidad del obturador de la cámara, por lo cual las imágenes salen movidas, con un desplazamiento uniforme (es decir, en un solo sentido) o no y que en inglés se lo conoce como blur, mientras que en español es llamado emborronamiento. Cabe destacarse que todo el proceso puede verse también afectado por ruido, por lo cual deberá recurrirse en dicho caso a una técnica robusta de super-resolución, las cuales exigen para su tratamiento una elevada complejidad computacional.

En este trabajo se lidia con imágenes sin ruido, el cual ha sido suprimido con otra técnica en un lugar diferente de aquellos donde actúa la super-resolución.

Por otra parte, mientras la compresión consiste en bajar en promedio el número de bit-por-pixel de la imagen, la super-compresión se define como una combinación de la disminución de la resolución de una imagen seguida de la compresión de esta. En otras palabras, la super-compresión es una sobre-compresión o compresión adicional de la imagen.

Consecuentemente, mientras la disminución y aumento de la resolución se llevan a cabo mediante un sub y sobre-muestreos, respectivamente, la restauración de la nitidez se realiza mediante una pequeña máscara convolutiva bidimensional, la cual realiza un barrido sobre la imagen. Estos procesos son de una elevada complejidad computacional, por ende, tanto en el codificador, como en el decodificador son implementados sobre placas de cálculo de Propósitos Generales en Unidades de Procesamiento Gráfico (en inglés, GPGPU). De hecho, la máscara mencionada es codificada en la memoria de texturas (la cual es la más pequeña de los cuatro tipos de memoria de estas placas, no obstante, la más rápida de todas) de una GPGPU.

Específicamente, los aportes del presente trabajo son:

1. Empleo de la técnica conocida como super-resolución de una imagen o video para ser utilizado en un proceso que será llamado super-compresión, sobre-compresión o compresión adicional, el cual consiste en los siguientes pasos:
   1.1. codificador:
      1.1.1. Bajar la resolución de una imagen o cada cuadro de un video (submuestreo)

1.1.2. Comprimir con algún compresor (con o sin pérdidas)
1.2. transmitir y/o almacenar
1.3. decodificador:
1.3.1. Descomprimir con el descompresor (con o sin pérdidas) respectivo como proceso inverso del item 1.1.2.
1.3.2. Restaurar la resolución original de la imagen o cuadro (sobre-muestreo)
1.3.3. Restaurar la nitidez original de la imagen o cuadro (realce)

2. Establecer las diferencias fundamentales entre compresión y super-compresión
3. Desarrollar un esquema de super-resolución para compresión
4. Establecer las diferencias fundamentales entre super-resolución y realce
5. Deducción de las herramientas más conspícuas a los efectos de los items anteriores
6. Aplicaciones adicionales: Imágenes médicas, satelitales, biométricas y documentales, incluyendo cine digital

Para terminar, es importante mencionar que los módulos algorítmicos del decodificador, es decir, sobre-muestreo y restauración de nitidez mediante máscara convolutiva bidimensional son fácilmente implementables también en un chip CMOS en inglés: Complementary Metal–Oxide–Semiconductor) y tecnología ARM (en inglés: Advanced RISC Machine, donde RISC en inglés: Reduced Instruction Set Computer), lo cual permite un reemplazo económicamente accesible de la GPGPU, además de bajo consumo de energía y una notable reducción del tamaño del dispositivo de decodificación, el cual, por ejemplo en el caso de la TV Digital se lo conoce como set-top-box.

Universidad Nacional de General Sarmiento

DOCTORADO EN CIENCIA Y TECNOLOGIA

o. Sumário:

Esta tese propõe o uso da técnica conhecida como super-resolução, compressão de imagem, são essas imagens estáticas, caixas de vídeo, digital de alta definição, cinema digital, e qualquer outro tipo de imagem comprimido para armazenamento adicional e / ou transmissão.

Pela técnica de super-resolução é diminuída exatamente a resolução da imagem no codificador, assim, o peso significativamente menor em bytes, e depois comprimilo por qualquer técnica conhecida e armazena / ou transmissão. Mais tarde, como o decodificador descompacta e restaura a resolução de imagem e nitidez de super-resolução técnicas.

Originalmente, as técnicas de super-resolução são usados em casos em que você deseja obter maior nitidez de imagem e resolução de um conjunto de imagens de baixa resolução e um pouco afiada, que são o resultado de uma diferença notável de velocidade entre a fim de capturar na tela versus a velocidade do obturador da câmera, para que as imagens são borradas, com um deslocamento uniforme (ie, one-way) ou não e em Inglês é conhecido como blur, enquanto em espanhol é chamado de manchas. Note-se que o processo também pode ser afetada pelo ruído, que deve ser utilizada neste caso uma técnica robusta super-resolução, os quatroles para o tratamento requer uma alta complexidade computacional.

Neste trabalho trata imagens sem ruído, o que foi reprimida com uma outra técnica em um lugar diferente de onde as obras super-resolução.

Além disso, enquanto a compressão é menor em número médio de bits por pixel da imagem, o super-compressão é definida como uma combinação de compressão mais baixa resolução de imagem seguiu este. Em outras palavras, a compressão super-compressão é um super-compressão ou adicionais da imagem.

Conseqüentemente, enquanto a diminuição eo aumento da resolução é realizada por uma subamostra e, respectivamente, restaurando a nitidez é através de uma máscara de convolução-dimensional pequeno, que varre a imagem. Estes processos são de alta complexidade computacional, portanto, tanto o codificador eo decodificador são implementadas na placa de ligação de cálculo Gráficos Uso Geral Processing Units (em Inglês, GPGPU). Na verdade, a máscara em questão é codificado na memória de textura (que é o menor dos quatro tipos de memória dessas placas, no entanto, mais rápido de todos) a GPGPU.

Especificamente, as contribuições deste trabalho são:

1. Usando uma técnica conhecida como super-resolução de imagem de vídeo ou para ser utilizado em um processo chamado de super-compressão, ao longo de compressão-Comissão ou compressão adicional, que consiste das seguintes etapas:
   1.1. encoder:
      1.1.1. Diminuir a resolução de uma imagem ou vídeo cada quadro de um (subamostragem)
      1.1.2. Comprimido com um compressor (com ou sem perda)
   1.2. transmitir e / ou loja
   1.3. decodificador:

1.3.1. Descompacte o descompactador (com ou sem perda) os respectivos como processo inverso do item 1.1.2.
1.3.2. Restaurar a resolução original da imagem ou foto (acima de amostratreo)
1.3.3. Restaurar a nitidez da imagem original ou a imagem (avançado)
2. Estabelecer as diferenças fundamentais entre compressão e super-compressão
3. Desenvolver um sistema de super-resolução para a compressão
4. Estabelecer as diferenças fundamentais entre super-resolução de melhoria
5. Dedução dos efeitos mais visíveis dos itens pré-anterior
6. Aplicações adicionais: Medical Imaging, satélite, biométricos e documentário tais, incluindo cinema digital

Finalmente, é importante notar que os módulos algorítmicos do decodificador dor, ou seja, mais de amostragem e restauração de nitidez por convolução máscarativa bidi-mensionais são facilmente implementáveis também em um chip também em Inglês CMOS: Complementary Metal-Oxide Semiconductor) e tecnologia ARM (em Inglês: Máquina Advanced RISC, onde o Inglês RISC: instruções reduzido Set Computer), permitindo uma alternativa acessível de GPGPU, além de baixo consumo de energia e uma diminuição significativa do dispositivo de decodificação que, por exemplo no caso da TV Digital é conhecido como set-top box.

DOCTORADO EN CIENCIA Y TECNOLOGIA

p. Abstract:

This thesis proposes the use of the technique known as super-resolution image compression, are these still images, video boxes, Digital TV HD, Digital Cinema, and any other type of image compressed to subsequent storage and / or transmission.

By the technique of super-resolution is diminished precisely the image resolution in the encoder thus significantly lower weight in bytes, and then compress it by any known technique and store and / or transmit. Later, as the decoder decompresses and restores the image resolution and sharpness by super-resolution techniques.

Originally, the super-resolution techniques are used in cases where you want to get higher picture sharpness and resolution from a set of low-resolution images and a little sharp, which are the result of a noticeable difference in speed between in order to capture on screen vs. the shutter speed the camera, so images are blurred, with a uniform displacement (i.e., one-way) or not and in English is known as blur, while in Spanish is called emborronamiento. It should be noted that the process can also be affected by noise, which should be used in this case a technique robust super-resolution, which require treatment for high computational complexity.

This work deals with images without noise, which has been suppressed with another technique in a different place than where the super-resolution works.

Moreover, while compression is lower on average the number of bit-by-pixel image, the super-compression is defined as a combination of lower resolution image com-pression followed this. In other words, the super-compression is an over-compression or additional compression of the image.

Consequently, while the decrease and increase of the resolution is carried out by a sub-sample and, respectively, restoring sharpness is through a small bidimensional convolutive mask, which sweeps over the image. These processes are of high com-putational complexity seasonally, therefore, both the encoder and the decoder are implemented calculation results on plates of general purpose processing units growth chart (in English, GPGPU). In fact, the mask in question is encoded in the texture memory (which is the smallest of the four types of memory of these plates, however, the fastest of all) a GPGPU.

Specifically, the contributions of this work are:

1. Using a technique known as super-resolution image or video to be used in a process called super-compression, over-compression or additional compression on further, which consists of the following:
   1.1. encoder:
      1.1.1. Lowering the resolution of an image or each frame of a video (sub-sampling)
      1.1.2. Compressed with a compressor (with or without loss)
   1.2. transmit and/or store
   1.3. decoder:
      1.3.1. Unzip the decompressor (with or without loss) corresponding as a reverse pro-

cess of item 1.1.2.

1.3.2. Restore the original resolution of the image or frame (oversampling)

1.3.3. Restore the original sharpness of the image or frame (enhanced)

2. Establish the fundamental differences between compression and super-compression
3. Develop a system of super-resolution for compression
4. Establish the fundamental differences between super-resolution enhancement
5. Deduction of the most conspicuous effects of the above items
6. Additional applications: medical, satellite, biometric and document imaging, including digital cinema

Finally, it is important to note that the algorithmic modules of the decoder, i.e. over-sampling and restoration of sharpness by a bidimensional convolutive mask are easily implementable on a chip also in English CMOS: Complementary Metal-Oxide-Semiconductor) and ARM technology (in English: Advanced RISC Machine, where English RISC: reduced Instruction Set Computer), allowing an affordable replace-ment of GPGPU, plus low power consumption and a significant downsizing of the decoding device which, for example in the case of Digital TV is known as set-top box.

DOCTORADO EN CIENCIA Y TECNOLOGIA

q. Aprobado por:
Dra. Ballarín, Virginia Laura
Dra. Del Fresno, Mirta Mariana
Dr. Julián, Pedro

Firma y aclaración de la firma del Presidente del Jurado:

Firma del autor de la tesis:

Universidad Nacional de General Sarmiento

DOCTORADO EN CIENCIA Y TECNOLOGIA

Supercompresion de video y aplicaciones

Publicaciones:

M. Mastriani and J. Gambini, "*Uso de la Transformada de Haar para el incremento de la eficiencia y rapidez de la Transformada Discreta de Karhunen-Loève en le compresión de imágenes con pérdidas*," 11º Argentine Symposium on Tecnology (AST), 39 Jornadas Argentinas de Informática (39 JAIIO), Volume 1, pp.1563-1574, 30 de Agosto al 3 de Septiembre de 2010, Buenos Aires, Argentina.

M. Mastriani and J. Gambini, "*Fast Cosine Transform to increase speed-up and efficiency of Karhunen-Loève Transform for lossy image compression*," International Journal of Engineering and Mathematical Sciences, Volume 6, Number 2, pp.82-92, 2010.

M. Mastriani, "*Single frame supercompression of still images, video, HDTV and Digital Cinema*," International Journal of Information and Mathematical Sciences, Volume 6, Number 3, pp.143-159, 2010.

M. Mastriani, "*Supercompression for Full-HD and 4k-3D (8k) Digital TV Systems*," International Journal of Information and Mathematical Sciences, Volume 6, Number 3, pp.186-199, 2010.

Aportes Originales:

Página

Agradecimientos

Desearía expresar mi agradecimiento y gratitud a la Profesora Dra. Juliana Gambini por su constante guía, consejo y estímulo desde que nos conocimos.

Quiero agradecer también al Dr. Gabriel Acosta Rodriguez, por su predisposición y apoyo.

Un particular agradecimiento a los miembros del jurado, Profesora Dra. Ing. Virginia Ballarín, Profesora Dra. Ing. Mariana Del Fresno y Profesor Dr. Ing. Pedro Julián cuyos aportes, comentarios y observaciones enriquecieron este trabajo.

Un agradecimiento a ANSES, UNTreF y CEMA sin cuyo soporte no podría haber llevado adelante este trabajo a esta altura de mi vida.

Mi agradecimiento y disculpas para Paola y Agustín por sacar de nuestro tiempo en común para hacer este trabajo.

A todos los que de una u otra forma, directa o indirectamente me ayudaron, les doy las gracias.

A Paola.

Indice General

Indice de Tablas

Indice de Figuras

Indice de Algoritmos

Página

Capítulo 1

Estado del arte

En este capítulo se aborda el estado del arte de la compresión de imágenes, sus ventajas y desventajas, pero enfocado fundamentalmente al área médica, donde las pérdidas por compresión son menos admisibles, no obstante, sin pérdida de generalidad en relación a otras áreas de aplicación. Por otra parte, se analizan las condiciones bajo las cuales se puede alcanzar un mejor rendimiento de compresión (con o sin pérdidas) ya sea mediante métodos conexionistas y no-conexionistas.

1.1 Introducción

Una imagen no variable con el tiempo (ya sea una fotografía, un fotograma, …) se puede definir como una función $\Re^2$ en $\Re^3$, si la imagen es en color o de $\Re^2$ en $\Re$ si es con niveles de gris. En ambos casos las dos variables independientes representan el espacio bidimensional. Nos centraremos en las imágenes con niveles de gris, ya que de este tipo son las imágenes médicas que analizaremos, pero la ampliación a tres variables para la representación del color, como en los modelos RGB, YIQ o HSI [1-9], es directa, por lo tanto, no perdemos generalidad y a la vez simplificamos notablemente los conceptos.

Como ocurre para cualquier tipo de función, una digitalización supone una discretización de los valores tanto de las variables independientes como de las dependientes. De esta forma podríamos decir que una imagen digital es una función discreta de dos variables discretas de la forma:

$$y[n_1, n_2] \quad n_1 = 1, ..., M; \quad n_2 = 1, ..., N \tag{1.1}$$

donde n_1 y n_2 son las variables dimensionales discretas, $M \times N$ es el tamaño de la imagen e y es el valor de la intensidad en cada punto. A cada punto de la imagen dado por las coordenadas (n_1, n_2) se le denomina píxel.

La intensidad, por el carácter digital de la imagen, no puede tomar cualquier valor sino que está limitado a un rango discreto. Esto podría alejar una imagen digitalizada de su original continua, pero hay que tener en cuenta el funcionamiento del ojo humano. La visión humana no puede percibir cualquier variación en la intensidad ΔI sino que es necesario que dicha variación supere un mínimo para que pueda ser distinguida la diferencia entre la intensidad I e $I + \Delta I$. De hecho, la variación mínima perceptible es proporcionalmente constante ($\Delta I / I = Cte$), es decir, que el ojo tiene una respuesta logarítmica. A esto hay que añadir que existe una pérdida de sensibilidad a la variación en los extremos del intervalo de intensidades, con lo cual dicho intervalo a percibir no es infinito. El efecto conjunto de ambas características conlleva que el ser humano sólo puede percibir un número limitado de niveles de intensidad o niveles de gris. En diversos estudios realizados con múltiples observadores tipo se concluyó que el ser humano puede distinguir una media de 100 niveles logarítmicos de intensidad. De esta forma, nos bastaría con que el conjunto de valores posibles de la intensidad discreta tuviera 100 componentes para que el ser humano no percibiera diferencia entre la imagen original y la digitalizada. Dado que se trabaja con bits, serían suficientes 7 bits (2^7 = 128 niveles) por píxel para evitar pérdidas perceptibles en el proceso de digitalización. Sin embargo, normalmente se utilizan 8 bits (2^8 = 256 niveles) por

píxel, dado que es el número de bits con el que se trabaja normalmente (8 bits = 1 byte); además, se ofrece un margen para que los posibles procesos a los que se sometan las imágenes no repercutan en una pérdida de fidelidad con respecto al original. Debemos tener en cuenta que todo lo expuesto es válido si la imagen tiene como destinataria la visión humana. Para el análisis computarizado sería necesario un número considerablemente mayor de bits por píxel. Sin embargo, en nuestro caso es bastante asumible que las imágenes médicas serán analizadas por el ojo humano, de hecho sus destinatarios serán los especialistas en la materia. En las imágenes médicas en general los distintos niveles de gris que estudiará el especialista representan alguna propiedad química o física de la estructura objeto de análisis. Por ejemplo en una radiografía de rayos-X digitalizada el valor del nivel de gris de cada punto expresa la densidad óptica del área correspondiente; en una tomografía el valor del nivel de gris está relacionado con el coeficiente de atenuación lineal de tejido, etc.

Hasta ahora hemos hablado del carácter discreto de la percepción humana en lo que se refiere a niveles de intensidad. Sin embargo, el ojo humano también realiza un procesado en frecuencia, de tal forma que lo que se percibe en un punto depende de lo percibido en puntos vecinos. Mediante diversos experimentos se ha llegado a la conclusión de que el ojo realiza un filtrado paso bajo, manteniendo la componente continua. De este hecho se deducen dos conclusiones:

- Existe una frecuencia espacial máxima por encima de la cual no se perciben las variaciones, sino que se uniformiza lo observado. Será esta característica la que nos permita discretizar el espacio en píxels sin que el ojo humano detecte el salto entre puntos consecutivos, siempre que los pixels sean lo suficientemente pequeños y estén lo suficientemente juntos.

- Dentro del margen de frecuencias espaciales perceptibles no podemos despreciar las altas frecuencias, pues el ojo humano es muy sensible a los cambios bruscos de intensidad, dado que los potencia.

Por último, debemos reseñar que una imagen digital con niveles de gris - dado su carácter discreto - puede definirse también como una matriz de la forma:

$$A_{MN} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \dots & \dots & \dots & \dots \\ a_{M1} & a_{M2} & \dots & a_{MN} \end{pmatrix} \tag{1.2}$$

Las dimensiones de la matriz ($M \times N$) es el tamaño de la imagen en pixels y el valor en cada punto ($a_{ij}\ i = 1,2,...,M;\ j = 1,2,...,N$) es la intensidad del píxel especificado dentro de un rango discreto, tal como explicamos. Por tanto, una imagen digital tiene un tamaño en bits igual a $M \times N \times p$ bits donde 2^p es el número de niveles de gris que tiene la imagen (p es el número de bits por pixel).

A continuación vamos a ver unos ejemplos de imágenes digitales y podremos constatar algunos de los puntos expuestos anteriormente. Una imagen digital del tipo RX de parte del torso, hombro/clavícula y brazo se muestra en la Fig.1.1. En esta imagen, para especificar el valor de la intensidad en cada punto, se utilizan 8 bits. Comprobemos lo que ocurre si lo reducimos a 3 bits, de tal forma que el número de niveles de cuantización es menor que los escalones de intensidad perceptibles. El resultado se observa en la Fig.1.2. Vemos que se hace patente la

transición entre los distintos niveles de gris (de entre los 8 posibles), mientras que en la anterior dicha transición era, al menos para nuestros ojos, continua. La Fig.1.3 es una pequeña parte de la Fig.1.1 aumentada en tamaño. Ahora volvemos a tener 8 bits por píxel pero el tamaño de los pixels es tal que la visión humana percibe las transiciones espaciales, la frecuencia de muestreo espacial es inferior a la necesaria. Podemos observar perfectamente los bordes de cada pequeño cuadrado de intensidad distinta, dado que dichos cuadrados son lo suficientemente grandes para ser percibidos.

Figura 1.1: Ejemplo de imagen digitalizada sin pérdidas visuales.

Figura 1.2: Ejemplo de imagen digitalizada con 8 niveles de cuantización.

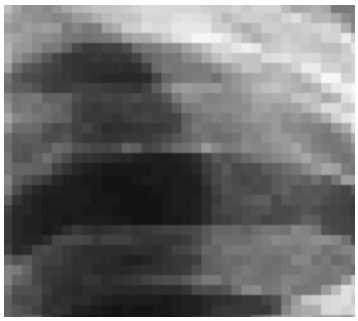


Figura 1.3: Ejemplo de imagen digitalizada con frecuencia espacial baja.

1.2 Definiciones y Clasificación

El objetivo primordial de la compresión de imágenes digitales es la reducción del número de bits que requieren dichas imágenes con la menor pérdida de calidad posible. Aunque la capacidad de los soportes para almacenamiento en el mercado aumenta considerablemente día a día, hay que tener en cuenta las enormes necesidades que implica un número elevado de imágenes de gran tamaño, como es el caso de los historiales radiológicos que maneja cualquier hospital. Por otro lado, la compresión cobra considerable importancia en aquellas aplicaciones que requieren la transmisión de imágenes por un medio que siempre tendrá un ancho de banda limitado, ya sea por naturaleza o por precio.

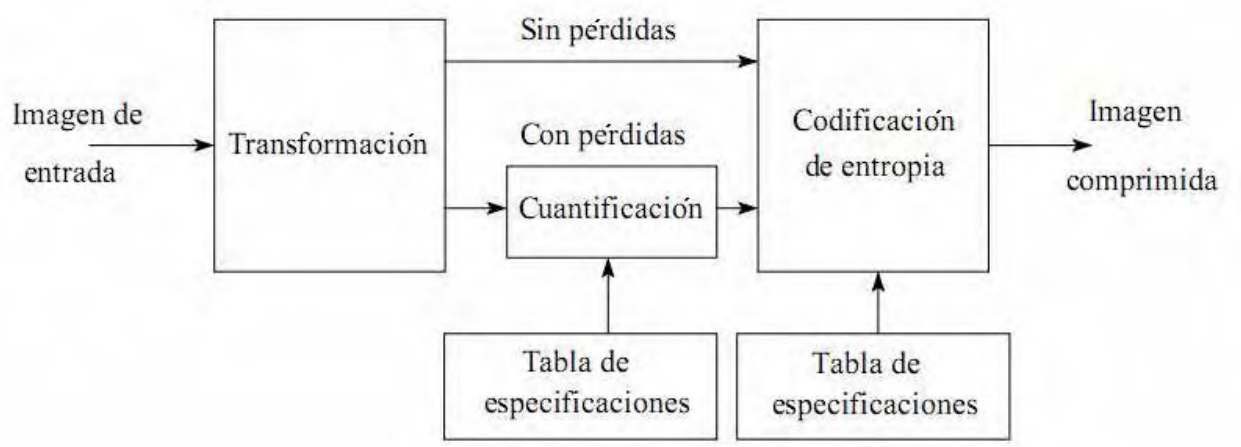


Figura 1.4: Diagrama general de un compresor de imágenes.

El diagrama en bloques del compresor tipo se muestra en la Fig1.4. Las técnicas de compresión de imágenes, como muchas técnicas de compresión de señales digitales, constan de tres etapas bien definidas:

- Transformación de la imagen. Supone una recorrelación de la señal, una reducción de su rango dinámico que elimine información redundante. Los coeficientes transformados deben ser estadísticamente independientes y la energía de la imagen transformada debe compactarse en un número mínimo de ellos.

- Cuantificación. Consiste en reducir la precisión de los coeficientes transformados sin sobre-

pasar los límites de calidad marcados. Esta fase implica siempre algún tipo de pérdida, con lo que las técnicas sin pérdidas carecerán de ella.

- Codificación de entropía. Aumenta la compresión sin aumentar las pérdidas codificando los coeficientes transformados de forma más compacta. Tras esta codificación tendremos una ristra de bits sin separación distinguible en vez de una serie de coeficientes o símbolos.

Existen diversas formas de clasificar las técnicas de compresión de imágenes digitales, según la característica específica que se quiera diferenciar. Podemos clasificar los métodos de compresión en función de la naturaleza de la transformación que se lleva a cabo. En este caso tenemos tres grandes grupos:

- Codificadores de forma de onda. La codificación se realiza directamente sobre la intensidad de la imagen como función espacial de n_1 y n_2, número de columna y número de fila. Normalmente se lleva a cabo algún tipo de procesado antes de la fase de cuantificación, de tal forma que se elimine parte de la información redundante. Aunque no consiste en una codificación por transformada, genera un aumento de la complejidad computacional tanto en el codificador como en el decodificador, razón por la cual a veces se elimina este procesamiento (como por ejemplo en el caso de la codificación PCM o de la cuantificación vectorial, que veremos más adelante).

- Codificadores de transformada. Sobre la imagen se realiza algún tipo de transformación para tratar de eliminar la dependencia estadística entre los pixels consecutivos antes de llevar a cabo la cuantificación, normalmente escalar. Por tanto, lo que se codifica no es la imagen en sí sino sus coeficientes en el espacio transformado. La transformación es equivalente a un cambio de la base en la que se expresa la imagen (o uno de los bloques de tamaño $P \times Q$ en los que se divide). Una imagen puede definirse como una matriz $M \times N$, pero también puede entenderse como un vector de dimensión $M\ N$. Si modificamos la base del espacio de forma conveniente obtendremos que la mayoría de la energía se concentra en un número pequeño de componentes, que deberemos cuantificar con mayor resolución. Por ejemplo, si la transformación es a un espacio de frecuencias espaciales, observaremos que la mayor parte de la energía de la señal transformada se concentra en las bajas frecuencias [10-12].

- Codificadores basados en modelos. Estas técnicas tratan de modelar la generación de la imagen de tal forma que lo que se codifica son los parámetros del modelo. El proceso comienza con una segmentación de la imagen en distintas zonas según el color, la textura, entre otras; aplicándosele un modelo matemático distinto según el caso. El siguiente paso será la extracción de las características de cada una de las zonas para obtener el valor específico para la imagen de entrada de los parámetros variables de cada modelo. Son estos parámetros los que se codificarán, de tal forma que el decodificador a partir de ellos debe ser capaz de sintetizar cada región. También es necesario almacenar la segmentación que se realizó sobre la imagen para determinar las áreas en las que el decodificador aplicará cada modelo. Estas técnicas logran unas tasas de compresión muy altas, aunque la calidad se resiente considerablemente.

Otra clasificación más general divide las técnicas de compresión en dos tipos:

- Compresión sin pérdidas. Son aquellas técnicas en las que la imagen original puede recuperarse de su versión codificada sin ningún tipo de alteración. Se trata de compresiones rever-

sibles y en consecuencia no incluyen una fase de cuantificación, pues este proceso es intrínsicamente con pérdidas. Sin embargo, una codificación directa de las imágenes con algún tipo de codificador de entropía no supone una tasa de compresión relevante.

- Compresión con pérdidas. En el proceso de compresión se realizan modificaciones irreversibles de tal forma que existen diferencias entre la imagen original y la imagen que se obtiene tras la decodificación. Esta modificación irreversible es básicamente una cuantificación de algún tipo. Dentro de este grupo es importante diferenciar las técnicas denominadas visualmente sin pérdidas. En este tipo de compresión, la imagen decodificada es distinta a la original pero las pérdidas sufridas no son detectadas por el ojo humano; tienen por tanto en consideración las características de la visión humana a la hora de decidir que alteraciones puede sufrir la imagen durante el proceso de compresión.

1.3 Cuantificación

En este apartado trataremos de los métodos de compresión basados en la cuantificación. Como dijimos al principio de esta sección, la cuantificación es una operación que suele formar parte del proceso de compresión. Sin embargo, existen técnicas que reducen e incluso eliminan el procesado anterior a la cuantificación y basan la compresión en esta fase. La recorrelación de los pixels, si se lleva a cabo, se realizará por tanto como consecuencia de las características de la cuantificación, ver Apéndice A.

Existen principalmente tres tipos de cuantificación, la cuantificación escalar, la cuantificación multiescalar y la cuantificación vectorial. Las dos primeras se suelen utilizar como paso posterior a la recorrelación mientras que la cuantificación vectorial suele ser por sí sola un método de compresión.

A continuación veremos cada tipo con más detalle.

1.3.1 *Cuantificación Escalar*

La cuantificación consiste en la transformación de un conjunto de valores de entrada, que puede ser finito o infinito y discreto o continuo, en otro conjunto menor, siempre discreto y finito. En el caso que nos ocupa el conjunto de valores de entrada serán directamente los niveles de gris de cada píxel o los coeficientes obtenidos por cualquier transformación. Por tanto, los valores de entrada se moverán dentro de un rango. Es ese rango el que dividimos en intervalos, a cada uno de los cuales asignamos un valor representativo, de tal forma que a cualquier valor de entrada perteneciente a un cierto intervalo $\boldsymbol{i}$ se le aproximará por el valor representativo de dicho intervalo. Cada intervalo estará limitado por unos umbrales de cuantificación. El número de intervalos vendrá condicionado por las necesidades de compresión y por el nivel de pérdida fijado aunque siempre tomará la forma $L = 2^n$, donde n es el número de bits asignado a cada valor de salida del cuantificador.

El valor representativo o nivel de cuantificación se escoge de manera que minimice la distorsión para cada intervalo por lo que dependerá de la medida de la distorsión utilizada. La forma de medición más común es la distorsión cuadrática, es decir:

$$d(y,\hat{y}) = (y-\hat{y})^2 \tag{1.3}$$

donde y es el valor original e $\hat{y}$ es el valor cuantificado, el valor representativo del intervalo al que pertenece y. Utilizando la distorsión cuadrática, el valor representativo óptimo es el punto medio del intervalo.

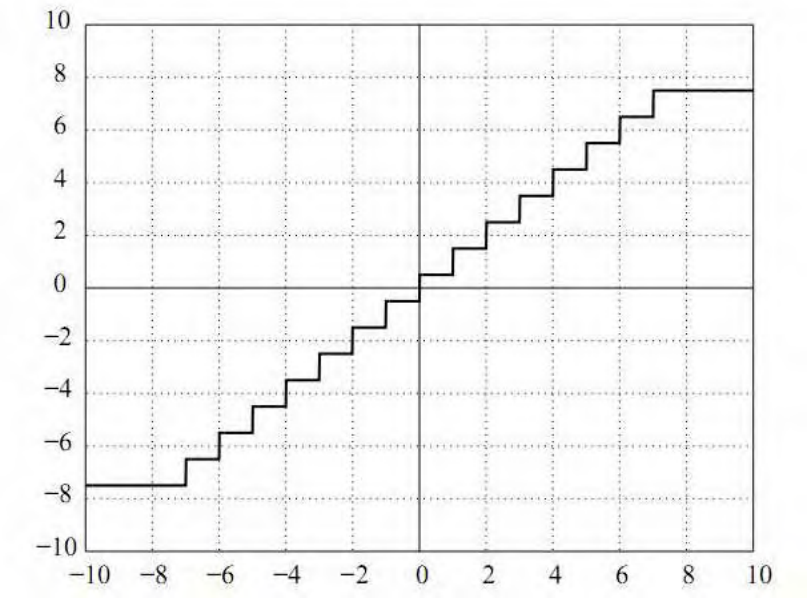


Figura 1.5: Cuantificador uniforme con 16 niveles.

Existe una clasificación de los cuantificadores escalares en función de la naturaleza de los escalones de cuantificación. En este sentido, el más sencillo es el cuantificador uniforme, que es aquel en el que el rango de variación es dividido en intervalos iguales y el nivel de cuantificación toma el valor del punto medio de cada uno de ellos. En la Figura 1.5 vemos la característica entrada/salida de un cuantificador uniforme para $L = 16$ niveles. La relación señal a ruido (SNR) de un cuantificador uniforme viene dado por:

$$\text{SNR} = 6n + K \ \text{[dB]} \tag{1.4}$$

siendo $2^n = L$ el número de niveles de cuantificación y K una constante que depende de las características de la señal de entrada.

El cuantificador uniforme será óptimo si la función de densidad de probabilidades del conjunto de posibles valores de entrada es uniforme. En caso contrario, la optimización se consigue con un cuantificador no uniforme, que consta de una transformación (tras la cual los valores de entrada adquirirán una función de densidad de probabilidades uniforme), una cuantificación uniforme y la necesaria transformación inversa. Este proceso se denomina compactación. El efecto conjunto es equivalente a asignar más niveles de cuantificación en la zona del rango de variación en la que la función de entrada toma valores con más frecuencia. Centrándonos en el caso que nos ocupa, el ojo humano tiene una respuesta logarítmica, es sensible a los saltos del logaritmo de la intensidad lumínica, con lo que será menos dañina perceptualmente una cuantificación logarítmica que una cuantificación uniforme.

1.3.2 *Cuantificación Multiescalar*

Este tipo de cuantificación consiste básicamente en la agrupación de múltiples cuantificadores

escalares de características distintas dentro de un mismo esquema de codificación. Por tanto, está concebido para ser utilizado tras una fase de transformación. Los coeficientes calculados pueden tener rangos y características probabilísticas muy distintas en función de su frecuencia o pseudocaracterísticas probabilísticas muy distintas en función de su frecuencia o pseudofrecuencia, por lo que es conveniente aplicar una cuantificación distinta a cada conjunto de coeficientes con características similares para una optimización del proceso.

1.3.3 *Cuantificación Vectorial*

La cuantificación vectorial consiste básicamente en agrupar k muestras de la señal de entrada (para ello habrá que dividir o segmentar la imagen original en sub-bloques) [2-9], formando vectores k-dimensionales y representar cada uno de éstos por un vector representativo denominado centroide.

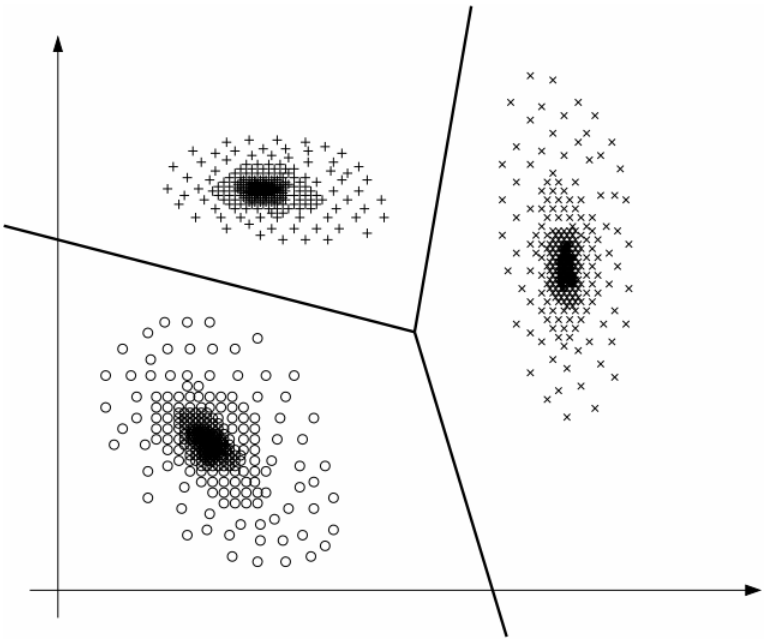


Figura 1.6: Ejemplo de cuantificación vectorial.

En la Fig.1.6 podemos ver un ejemplo para $k = 2$ dimensiones y tres centroides. Es decir, en vez de dividir el rango de la recta real en el que se mueven los valores de entrada en intervalos (cuantificación escalar) dividimos el espacio k-dimensional en celdas o subespacios. De esta forma se aprovechará la correlación entre pixels consecutivos que hará que los vectores se concentren en una determinada zona del espacio. La principal complicación de este tipo de métodos es la división en celdas y el consecuente cálculo de los centroides.

La cuantificación vectorial es la versión multidimensional de la cuantificación escalar. En la cuantificación escalar el análisis de la entrada se realiza elemento a elemento, es decir, el valor de la entrada en cada momento y_n se compara con un conjunto de niveles de decisión ordenados y_i hasta encontrar el par entre los que se encuentra: $y_i < y_n < y_{i+1}$. En consecuencia, el valor de salida será un nivel fijo de reconstrucción para el intervalo al que pertenece la entrada: $\hat{y}_i$.

La determinación del valor de los niveles de decisión y de los niveles de reconstrucción depende del número total de niveles disponibles, el criterio de error y las posibles restricciones que

impongan factores como la entropía o la tasa de bits de salida mínima.

La cuantificación vectorial es una extensión de este planteamiento a varias dimensiones. La entrada en cada momento no es un elemento aislado, sino un bloque de elementos, o un vector de dimensión *k*. De esta forma todas las componentes de dicho vector $Y = [y_1, y_2, \ldots, y_k]$ se cuantificarán unidas, llevándose a cabo la división no en la recta real sino en el espacio *k*-dimensional. Los niveles de decisión serán sustituidos por los bordes de una determinada región C_i de dicho espacio y el nivel de reconstrucción es sustituido por un punto determinado en dicha región $\hat{Y}_i = [\hat{y}_{i1}, \hat{y}_{i2}, ..., \hat{y}_{ik}]$. A estos vectores de reconstrucción se los denomina centroides o vectores código. De esta forma la cuantificación vectorial es equivalente a un mapeo del espacio euclídeo *k*- dimensional $\Re^k$ en un subconjunto finito formado por N_c centroides según la función dada por:

$$Y \rightarrow \hat{Y}_i \quad si \quad Y = [y_1, y_2, ..., y_k] \in C_i \tag{1.5}$$

Al subconjunto finito de $\Re^k$ mencionado se le denomina librería de centroides. Será en esta librería en la que se lleve a cabo la búsqueda del centroide más cercano al vector de entrada. Lo que se transmite para cada vector de entrada es por tanto el índice ***i*** del centroide que lo representa, que es reconstruido en el decodificador. El tamaño de la librería es típicamente una potencia de 2, es decir, $N_c = 2^b$, de tal forma que el índice pueda ser expresado como un número de *b* bits.

La complejidad de la búsqueda aumenta linealmente con el número de centroides y exponencialmente con la dimensión. Esta dimensión está determinada por el tamaño de los subbloques en los que se divida la imagen para su codificación. El método de compresión por cuantificación vectorial es por tanto un método por subbloques. Debido al aumento de complejidad con la dimensión, la división típica que se suele utilizar es en subbloques cuadrados de 4 x 4 pixels. Al no ser subbloques muy grandes se evita también que aparezcan efectos graves por el procesado por subbloques.

Dentro de la cuantificación, la distorsión cobra importancia en dos operaciones. Por un lado, en la fase de entrenamiento, los centroides $\hat{Y}_i$ deben ser elegidos de tal forma que minimicen la distorsión media dentro de la región C_i. Por otro lado, en la fase de codificación, para cada vector de entrada debe elegirse el centroide con el que se logre una distorsión mínima. De todo esto se desprende la importancia que tiene la elección de una correcta medida de la distorsión. La medida más utilizada sigue siendo el error cuadrático medio, dado que es una medida de la energía, no de la amplitud. En el caso de la cuantificación vectorial, siguiendo la nomenclatura, el error cuadrático medio entre el vector original y el de reconstrucción es:

$$d_2(Y, \hat{Y}_i) = \frac{1}{k} \sum_{n=1}^{k} (y_n - \hat{y}_{in})^2 \tag{1.6}$$

Dado que la operación expresada en la ecuación (1.6) será reiteradamente realizada durante la fase de búsqueda de cualquier algoritmo de cuantificación vectorial, una manera de ahorro del coste computacional sería reescribir cada sumando de la forma:

$$(y_n - \hat{y}_{in})^2 = y_n^2 + \hat{y}_{in}^2 - 2\, y_n\, \hat{y}_{in} \tag{1.7}$$

De esta manera, el error cuadrático medio resulta:

$$d_2(Y,\hat{Y}_i)=\frac{1}{k}\left(\sum_{n=1}^{k}y_n^2+\sum_{n=1}^{k}\hat{y}_{in}^2-2\sum_{n=1}^{k}y_n\hat{y}_{in}\right) \tag{1.8}$$

Dado que la primera sumatoria es fija para cada vector de entrada (es su módulo al cuadrado), la determinación del centroide más cercano (de distorsión mínima) para cada entrada se basa únicamente en las dos últimas sumatorias. Por lo tanto, la búsqueda de dicho centroide es equivalente a la maximización del valor:

$$M(Y,\hat{Y}_i)=\sum_{n=1}^{k}y_n\,\hat{y}_{in}-\frac{1}{2}\sum_{n=1}^{k}\hat{y}_{in}^2 \qquad i=1,...,N_c \tag{1.9}$$

La diferencia principal dentro del conjunto de algoritmos basados en la cuantificación vectorial es el método empleado en el diseño de la librería de centroides. A continuación realizaremos un repaso de algunos de ellos:

- Cuantificación vectorial clasificada [1]. Surge en respuesta al problema de los bordes que apare-cen en la imagen. Como bordes entendemos las zonas de la imagen en las que se produce un cambio brusco en el nivel de luminancia, marcando el contorno de algún objeto. Dichos bor-des, aunque ocupan espacios reducidos dentro de la imagen necesitan una reproducción exac-ta para satisfacer el criterio perceptual del ojo humano. En el algoritmo básico de la cuanti-ficación vectorial el número de centroides adecuados a los bordes es muy pequeño, debido a su escasa frecuencia de aparición, con lo que la búsqueda para los vectores de borde puede dar como resultado un centroide que no se ajusta bien a él. Para solucionar este problema sur-ge la clasificación. Los vectores de entrenamiento se dividen en borde y sombra y se generan centroides distintos para cada tipo. Los vectores de entrada se clasifican también para codifi-carlos con el tipo de centroide adecuado. Este concepto se puede ampliar aumentando las posibilidades de la clasificación a más de dos tipos. Para la clasificación se utiliza algún algo-ritmo de detección de bordes.

- Cuantificación vectorial de media-y-forma [1]. Se basa en la cuantificación por separado de las propiedades de cada vector. El procedimiento consiste en sustraer a cada elemento del vector el valor medio de este, de modo que el resultado de todas las sustracciones, la forma, se cuantifica vectorialmente, mientras que la media única extraída de dicho vector se cuantifica escalarmente por separado. Este método deriva de la afirmación de que muchas entradas tienen formas parecidas pero difieren en su luminancia media.

- Cuantificación vectorial interpolativa [1]. En este caso el proceso comienza con la transmi-sión mediante alguna codificación sencilla de un elemento representativo de cada bloque. A partir de él, tanto en el codificador como en el decodificador se obtiene dicho bloque representativo mediante interpolación, la diferencia entre éste y la distribución de luminancia del bloque real en la parte codificadora es lo que se cuantificará vectorialmente y se enviará.

- Cuantificación vectorial multietapa [1]. Este método consiste en realizar una primera cuantifi-cación gruesa a la imagen y a continuación someter a la diferencia entre la salida de esta cuantificación y la imagen original a uno o más procesos de cuantificación posteriores. Los resultados son buenos en imágenes que difieren sustancialmente de las imágenes de entrena-miento.

- Cuantificación vectorial de estado finito [1]. Se basa en la transición entre estados que sufren el codificador y el decodificador. Para la codificación de los vectores de entrada se utilizan una colección de pequeñas librerías, de entre las que se elige una en función del estado del codificador y del decodificador, los cuales son básicamente iguales. El estado en cada momento depende del estado en el instante anterior y de la transición asociada al centroide que se usó en dicho instante.

- Cuantificación vectorial predictiva [1]. Aprovecha las ventajas de la cuantificación vectorial pero no para la codificación de la imagen sino para la codificación del residuo de una predicción, sustituyendo a la cuantificación escalar.

- Cuantificación vectorial adaptativa [1]. Se basa en el principio fundamental de la compresión de datos que implica la obtención de mejores resultados cuando se utilizan algoritmos de codificación adaptados a las propiedades estadísticas de los datos de entrada, siempre que el diseño del sistema conlleve que la información requerida para el establecimiento de los parámetros variable sea mínima. Así, estos tipos de algoritmos en general fijan una serie de variables en el codificador según la distribución estadística de la imagen.

- Transformada/Cuantificación vectorial [1]. Resulta de la combinación de una transformación con la cuantificación vectorial posterior de los coeficientes transformados. De esta forma se superan dos problemas de la cuantificación vectorial. Por un lado, la necesidad de utilizar sub-bloques pequeños para disminuir la complejidad computacional, siendo menos complicado aumentar el tamaño de estos subbloques en el dominio transformado. Por otro lado, la distribución estadística de los valores de luminancia en una imagen natural está erráticamente definida, de tal forma que son necesarios complejos procesos de diseño de librería, complejidad que se reduce en el dominio transformado.

1.4 Algoritmos de Entrenamiento para Cuantificación Vectorial (métodos conexionistas)

El principal elemento diferencial entre los distintos métodos de cuantificación vectorial es el algoritmo utilizado para la generación de la librería de centroides a utilizar. En el desarrollo de un proceso de cuantificación vectorial es necesaria la existencia de una librería de centroides, que será el conjunto de vectores k-dimensionales que se utilizarán como representativos del espacio $\Re^k$. En este punto, la cuantificación vectorial difiere de su versión unidimensional. Debido a la difícil definición de la distribución estadística de los niveles de luminancia en una imagen no se suelen utilizar las distribuciones analíticas multidimensionales como base para establecer los intervalos de decisión del cuantificador, sino que su diseño suele basarse en la adaptación de métodos de mínima distorsión aplicados a una secuencia de datos de entrenamiento [13-16]. Dado que el valor de cualquier medida de la distorsión incrementa con la distancia entre los vectores de entrada y los vectores código correspondientes, estos métodos establecerán la correspondencia óptima según la regla del vecino más próximo:

$$Y \to \hat{Y}_i \quad si \quad d\left(Y,\hat{Y}_i\right) < d\left(Y,\hat{Y}_j\right) \quad para\ todo\ j \neq i \tag{1.10}$$

Para el caso $j = i$, nos encontraríamos en las fronteras de lo que se conoce como regiones de Voronoi [13-16] y las cuales no se pueden asociar estrictamente a ningún centroide en particular, en otras palabras, dichas fronteras no aportan nada al proceso de compresión/descompresión. Esta regla también definirá las regiones de cuantificación C_i.

1.4.1 *Algoritmo Linde-Buzo-Gray (LBG)*

En esta línea, un algoritmo clásico es el debido a Y. Linde, A. Buzo y R. M. Gray [13], conocido por las iniciales de los autores como LBG. Cada paso de este algoritmo se compone de dos partes. En la primera, partiendo de una librería de vectores código obtenida en el paso anterior, se buscará la partición óptima del espacio, en la que cada vector pertenecerá a la región asociada al vector de reproducción más cercano. En la segunda, basándose en dicha partición, se hallarán los nuevos centroides de cada región, que serán los vectores de reproducción óptimos para este paso. El proceso comienza con un conjunto de valores iniciales que se describen más adelante. Cada iteración del algoritmo de entrenamiento LBG, basado en un conjunto de vectores de entrenamiento, tiene dos partes:

1. Partición del conjunto de vectores de entrenamiento en un número de categorías o regiones igual al número requerido de vectores código N_c. En esta primera parte de la iteración se utiliza el conjunto de vectores código $\hat{\mathbf{Y}}(n-1) = \left\{\hat{Y}_i(n-1),\ i = 1,...,N_c\right\}$ que se calculó en la segunda parte de la iteración anterior. En este caso la selección para cada vector de entrenamiento $\boldsymbol{Y}$ del miembro más cercano del conjunto $\hat{\mathbf{Y}}(n-1)$ tendrá como resultado una partición $\mathbf{P}\left(\hat{\mathbf{Y}}(n-1)\right)$ que será la óptima.

2. Determinación del vector código óptimo para cada categoría. En esta segunda fase tenemos la partición $\mathbf{S}(n) = P\left(\hat{\mathbf{Y}}(n-1)\right)$, siendo conocido todo el conjunto de vectores de entrenamiento pertenecientes a cada región $S_i(n)$. De esta forma, estableceremos el vector código $\hat{Y}_i(n)$ de la región $S_i(n)$ como el centro de gravedad generalizado o centroide de los vectores de dicha región de tal forma que:

$$E\left[d\left(Y,\hat{Y}_i(n)\right)/\, Y \in S_i(n)\right] = \min_u E\left[d\left(Y,u\right)/\, Y \in S_i(n)\right] \tag{1.11}$$

Donde *E{.}* es la esperanza matemática de ".". Por otra parte, si asumimos la medida de la distancia euclídea y dado que en la práctica los vectores código óptimo se calculan con un conjunto finito de vectores de entrenamiento $\mathbf{Y} = \{Y_1, Y_2, ..., Y_T\}$ *con* $T >> N_c$, podemos aproximar el centroide de cada región por:

$$\hat{Y}_i(n) \approx \frac{1}{t_i(n)} \sum_{Y \in S(n)} Y_i\,,\ i = 1,...,N_c \tag{1.12}$$

donde $t_i(n)$ es el número de vectores de entrenamiento pertenecientes a $S_i(n)$. Se deberá cumplir que:

$$\sum_{i=1}^{N} t_i(n) = T,\ \ \forall n \tag{1.13}$$

El criterio de finalización es la variación relativa de la distorsión media $D(n)$ en cada paso, dado que a medida que la distorsión converja a su valor asintótico, se irá reduciendo hasta alcanzar el umbral ε, valor que se considera aceptablemente bajo. La distorsión de cada vector de entrenamiento $D_t(n)$ viene dada por:

$$D_t(n) = \min_{i=1,\ldots,N} \left\{ d\left(Y_t, \hat{Y}_i(n)\right) \right\}, \quad t = 1,\ldots,T \tag{1.14}$$

y la distorsión media viene dada por la ecuación:

$$D(n) = \frac{1}{T} \sum_{t=1}^{T} D_t(n) \tag{1.15}$$

El criterio de parada es:

$$\frac{D(n-1) - D(n)}{D(n)} < \varepsilon \tag{1.16}$$

El único punto que nos queda por determinar es el establecimiento de los valores iniciales de los vectores código que conforman el conjunto inicial $\hat{\mathbf{Y}}(0)$. Para este fin desarrollaremos un proceso que comienza con la determinación del centro de gravedad global del conjunto de entrenamiento completo. Cada una de las componentes de este único vector representativo es perturbada con pequeñas cantidades aleatorias $\pm \delta$. Esta perturbación se realiza dos veces dando lugar a dos nuevos centroides. El proceso continúa de la misma manera, dividiéndose estos dos valores iniciales en otros dos, iterándose hasta alcanzar el número de centroides requerido, que será potencia de dos, de cara a su posterior codificación binaria. Al final de este proceso, la librería de centroides inicial será aproximadamente una nube de vectores alrededor del centro de gravedad del conjunto de entrenamiento.

Figura 1.7: Ejemplo de cuantificación vectorial de imagen utilizando algoritmo LBG. Imagen original y reconstruida.

En la Fig.1.7 se ve un caso a modo de ejemplo de una imagen y su equivalente reconstruida utilizando cuantificación vectorial con librerías de centroide generada mediante algoritmo LBG. En la Tabla 1.1 se puede ver la tasa de compresión y la calidad lograda para la imagen de la Fig.1.7.

Al algoritmo LBG se han ido añadiendo una serie de métodos que pretenden generar librerías de forma más eficiente. A continuación veremos algunos de ellos.

Tabla 1.1: Resultados de la codificación con cuantificación LBG.

SNR (dB)	Tasa de compresión
22.6916	22.5987:1

1.4.2 *Librería en Arbol.*

Se trata de un algoritmo que define la librería mediante una estructura en árbol. El algoritmo parte del centroide o centro de gravedad de la totalidad del conjunto de vectores de entrenamiento. Este vector representativo único es perturbado mediante la suma a cada una de sus componentes de una cantidad diferencial aleatoria δ. Mediante dos perturbaciones obtenemos dos nuevos vectores representativos que serán utilizados para establecer una partición del espacio en dos regiones. Siguiendo el algoritmo LBG, a continuación se hallan los centroides de cada una de estas regiones, que a su vez se vuelven a dividir en dos dando lugar a una partición con el doble de regiones. El proceso se repite hasta alcanzar el número N_c de centroides fijado, siendo N_c una potencia de dos.

Para el establecimiento de la estructura en árbol es necesario que no sólo se almacenen los N_c centroides finales, sino también los que aparecen en los pasos intermedios, estableciendo una sucesión de ramas. De cada centroide calculado en un paso intermedio parten dos centroides pertenecientes a su región asociada, que a su vez dividen dicha región en dos subregiones. El nodo inicial del árbol es el centro de gravedad del conjunto de vectores de entrenamiento, cuya región asociada es todo el espacio. De esta forma se simplifica considerablemente la búsqueda durante el proceso de cuantificación, ya que ésta se realiza siguiendo un camino en el árbol, realizando decisiones binarias en cada nivel de la estructura: cada entrada pertenece a una de las dos regiones en las que se divide el espacio en el primer nivel y dentro de su región pertenece a una de las dos subregiones en las que se dividen las regiones primarias y así sucesivamente.

En su forma más sencilla, cada rama más baja se divide en dos más altas. Los algoritmos de división binaria en cada nodo tiene la ventaja de producir una estructura uniforme pero los resultados que se obtienen son peores a los que se conseguirían si relajamos este requerimiento, dividiendo cada nodo en función de la distorsión que aparece en la región que representa. Surge así un algoritmo de estructura en árbol no uniforme, basado en la división sucesiva sólo de los nodos con más distorsión. Los resultados que se obtienen en este caso se acercan más a los de la búsqueda exhaustiva (comparación con todos los centroides). El número de centroides sólo tiene que ser un entero y no una potencia de dos como en el caso anterior.

Otra propiedad de la estructura en árbol añade el concepto de podado. La idea básica consiste en la eliminación de ciertas ramas para compensar la creación de otras nuevas. Como criterio de eliminación en el proceso se analiza la relación λ_p asociada a cada nodo, definida como la relación entre el incremento de la distorsión con respecto al decremento de la tasa de transmisión que se produce como consecuencia de la eliminación (podado) de la rama que cuelga de dicho nodo. En la dirección de crecimiento (en los nodos finales del árbol, susceptibles de generar nuevas ramas), lo que se analiza es λ_g, es decir, a la inversa, la relación entre el decremento de la distorsión con respecto al incremento de la tasa de transmisión. De esta forma, el árbol es generado mediante la división en dos nuevas ramas de los nodos con los valores más altos de λ_g (con los que se obtienen mejores resultados) y el podado de aquellos cuyo valor de

λ_p sea menor (cuya eliminación implica una pérdida menor). Incluso existen medicaciones que permiten la división de los nodos en tres o cuatro ramas, no sólo en dos.

1.4.3 *Algoritmo Pairwise Nearest Neighbour (PNN).*

Con un planteamiento totalmente diferente encontramos el algoritmo vecinos más cercanos por parejas o PNN (Pairwise Nearest Neighbour) [14]. Este algoritmo opera en sentido inverso a la estructura en árbol, ya que agrupa los vecinos más próximos dentro del conjunto de vectores de entrenamiento hasta que se alcanza el tamaño de librería requerido. En un principio se agrupan los vectores de entrenamiento por parejas, siendo la pareja de cada uno el vector de entrenamiento vecino más cercano, y se calculan los centroides de cada región formada por dos vectores de entrenamiento. Tenemos entonces el espacio dividido en un número de regiones igual a la mitad del número de vectores de entrenamiento, estando asociado a cada región un vector representativo. En cada paso se lleva a cabo la unión de dos regiones en una nueva, y la obtención del centroide de esta nueva región. Las dos regiones vecinas C_i y C_j que se unirán en una sola son elegidas de tal forma que se minimice el error introducido por la unión de dichas regiones dado por la ecuación:

$$e_{ij} = \frac{n_i n_j}{n_i + n_j} \left| \hat{Y}_i - \hat{Y}_j \right|^2 \tag{1.17}$$

En esta ecuación, los enteros n_i y n_j son el número de vectores que contiene cada región e $\hat{Y}_i$ e $\hat{Y}_j$ son los centroides de dichas regiones. Podemos observar que el criterio que se aplica es el de aunar una región C_i con la región vecina C_j cuyo centroide $\hat{Y}_j$ sea el más cercano al centroide de la primera región $\hat{Y}_i$. La unión de regiones termina cuando se alcanza un determinado número de centroides o cuando la distorsión alcanza un umbral determinado.

1.4.4 *Relajación Estocástica.*

La relajación estocástica es un método general para la optimización de problemas complejos. En el caso que estudiamos, para una determinada librería se selecciona una temperatura **T** como parámetro de control del algoritmo y la cantidad exp(-ΔE/T) es calculada cuando la librería es sometida a alguna perturbación tal como mover un vector de entrenamiento aleatoriamente de una partición a otra o añadir una variación aleatoria a algún centroide. En estos casos, se calcula la variación de la energía **ΔE** (distorsión) y si es negativa, se acepta la nueva asignación. En caso contrario, la nueva asignación será aceptada con probabilidad exp(-ΔE/T). Por tanto, a altas temperaturas, las variaciones serán probablemente aceptadas incluso aunque supongan un aumento de la distorsión.

Cuando el proceso se considera terminado a una determinada temperatura, bien sea a causa de una limitación del número de iteraciones o porque la distorsión ha disminuido en una cantidad aceptable, la temperatura se reduce y el proceso se repite. A medida que la temperatura disminuye también lo hace la probabilidad de aceptar una nueva asignación, alcanzándose al final del algoritmo cuando se consigue una librería estable durante sucesivas reducciones de temperatura. La principal ventaja de este algoritmo es su capacidad de converger a un mínimo global, no local, de la distorsión, pero su complejidad computacional es muy alta, lo cual hace lenta su implementación en determinadas tecnologías.

1.4.5 *Redes Neuronales.*

Una de las últimas soluciones para el diseño de librerías de centroides consiste en la aplicación de técnicas basadas en redes neuronales, de amplia aplicación en el procesamiento de señales [15,16]. Una red neuronal podría definirse como un procesador masivamente distribuido, compuesto por unidades de procesamiento simples, cuyos parámetros generan una representación interna del conocimiento experimental almacenado y una disponibilidad para su uso. Esta red se asemeja en su funcionamiento al cerebro humano en dos aspectos:

- El conocimiento es adquirido por la red de su entorno a través de un proceso de aprendizaje.

- Las fuerzas de interconexión entre las neuronas, denominadas pesos sinápticos, son usadas para almacenar el conocimiento adquirido.

El procedimiento usado para llevar a cabo el proceso de aprendizaje se denomina algoritmo de aprendizaje. Será un procedimiento de este tipo el que se empleará para el entrenamiento de la librería de vectores código.

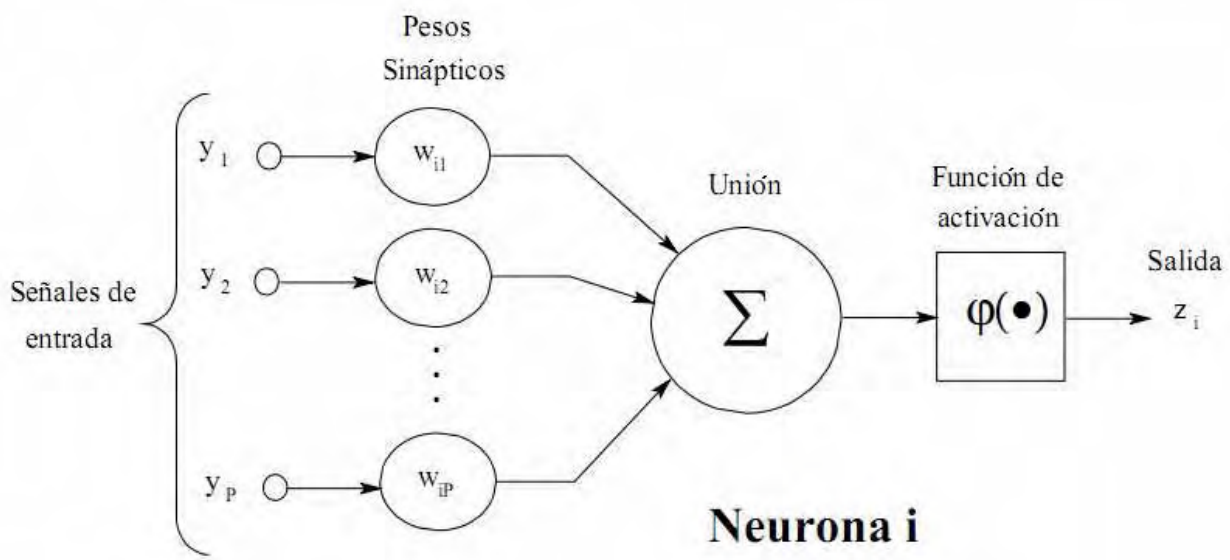


Figura 1.8: Modelo no lineal de una neurona.

La unidad de procesado de información fundamental para la realización de una red neuronal es la neurona. El diagrama de bloques del modelo genérico de una neurona se reproduce en la Fig.1.8. Los tres elementos básicos del modelo neuronal son los siguientes:

- El nivel de conexionismo entre neuronas de diferentes capas recibe el nombre de sinapsis. Cada una de estas sinapsis está caracterizada por un peso propio. Específicamente, una señal y_p a la entrada de la sinapsis p conectada a la neurona i es multiplicada por un peso w_{ip}, el cual las vincula paramétricamente y al mismo tiempo modeliza con un número real regula la mencionada sinapsis.

- Un sumador para sumar las señales de entrada ponderadas por la respectiva sinapsis de la neurona.

• Una función de activación que limita la amplitud de la salida de la neurona a un valor finito.

La teoría expuesta hasta ahora responde al modelo genérico de redes neuronales. Para su adaptación a la cuantificación vectorial de imágenes es necesario redefinir algunos aspectos. De entrada, consideramos la imagen dividida en subbloques como un conjunto de vectores del espacio *k*-dimensional. Son estos vectores Y los que constituyen el conjunto de señales de entrada de la red neuronal, ver Fig.1.8.

Suponemos que el tamaño de la librería de vectores código es N_c, y dichos vectores serán $\hat{Y}_i$; $i = 1, \ldots, N_c$. De esta forma, consideraremos una red neuronal con N_c neuronas o unidades neuronales, cada una de las cuales tiene una sola entrada y por tanto, una sola sinapsis. Haremos que el vector código *i*-ésimo $\hat{Y}_i$ sea el peso asociado a la sinapsis de la neurona i, W_i. Por otro lado, en la definición de neurona vista, la ponderación de la entrada para cada sinapsis se realizaba mediante una simple multiplicación de la entrada por el peso correspondiente. En el caso de la cuantificación vectorial, para esta ponderación es necesario una operación más compleja, el cálculo de la distorsión $d(Y,\hat{Y}_i)$ entre el vector Y y el vector código $\hat{Y}_i$ que lo representa. Será sobre esta distorsión sobre la que se aplicará la función de activación $\varphi(.)$ ver Fig.1.8. Podemos observar que en la estructura definida no aparece el sumador, lo cual es debido a que cada neurona tiene una única sinapsis.

La cuantificación seguirá el siguiente proceso: cada vector Y a cuantificar es alimentado en paralelo a las N_c neuronas de la red. Cada una de ellas proporciona una salida que resulta de ponderar la entrada con el peso asociado a su sinapsis mediante el cálculo de la distorsión entre ambos. Esta salida a su vez será la entrada de la función de activación de cada neurona.

Con respecto a la función de activación, el tipo que utilizaremos es el tipo umbral, es decir, la salida z_i sólo toma el valor 1 ó 0. Existen otros tipos de funciones de activación que, para ciertos valores de entrada, pueden tomar valores intermedios, pero no son adecuadas en este caso. La definición exacta de esta función puede ser distinta en la fase de codificación y en la fase de entrenamiento. En la fase de codificación, el valor 1 significa que la unidad neuronal corresponde al vector código que representará al vector de entrada. Así, para cada entrada, la función de activación sólo puede ser igual a 1 en una neurona. Lógicamente, el valor 1 lo toma la neurona cuya $d(Y,W_i)$ sea menor, es decir, la que corresponde al vector código más cercano al vector de entrada. Matemáticamente, para la neurona i, la salida de la función de activación sería de la forma:

$$z_i = \begin{cases} 1 & d[Y,W_i(n)] \le d[Y,W_j(n)], \qquad j = 1,....,N_c \\ 0 & \textit{en otro caso} \end{cases} \tag{1.18}$$

En lo que se refiere a la fase de entrenamiento, los algoritmos basados en redes neuronales hacen uso de la capacidad de almacenar conocimiento de estas estructuras. Como dijimos, el conocimiento se almacena en los pesos sinápticos, los cuales serán actualizados con cada nueva entrada. La actualización de los pesos de las sinapsis estará en función de la entrada, del peso actual y de la salida de la neurona. Traducido en términos de cuantificación vectorial, tras el paso de un vector de entrada por la red neuronal se actualizan los vectores código (que son los pesos) en función de su diferencia con el vector de entrada si así lo indica la salida de la función de activación correspondiente. Este proceso lo veremos con más detalle en la descripción de los

algoritmos que veremos a continuación.

1.4.5.1 *Red de Aprendizaje Competitivo.*

Para la ejecución de este algoritmo asumimos que la red neuronal es entrenada por un conjunto amplio de vectores de entrenamiento. Partimos de la inicialización de las N_c unidades neuronales con los vectores peso $W_i(0)$, i = 1, …, N_c. Estos pesos pueden ser genera-dos aleatoriamente o ser los primeros N_c vectores del conjunto de entrenamiento. Partiendo de esta base, el algoritmo de entrenamiento realizará diversas pasadas por los vectores de entrena-miento, actualizando los vectores peso de las unidades neuronales después de la entrada de cada vector de entrenamiento. El algoritmo de ajuste está basado en el aprendizaje competitivo. Comienza con la presentación del vector de entrada Y a todas las unidades neuronales cada una de las cuales calcula la distorsión entre su peso y el vector de entrada. La unidad de distorsión menor es designada como ganadora y su peso es ajustado hacia el vector de entrada. Expresado matemáticamente, partiremos del peso $W_i(n)$ de la unidad i en el paso n antes de la entrada del nuevo vector, siendo la salida z_i de la función de activación calculada de la forma:

$$z_i = \begin{cases} 1 & d[Y, W_i(n)] \le d[Y, W_j(n)], \qquad j = 1, ..., N_c \\ 0 & \textit{en otro caso} \end{cases} \tag{1.19}$$

Los nuevos vectores peso $W_i(n+1)$ son calculados entonces como:

$$W_i(n+1) = W_i(n) + \eta(n)[Y - W_i(n)]z_i \tag{1.20}$$

El parámetro $\eta(n)$ es la tasa de aprendizaje y suele decaer monótonamente a cero a medida que avanza el proceso de aprendizaje. El principal problema que surge con este algoritmo es la infrautilización de algunas unidades neuronales que no resultan ganadoras nunca.

1.4.5.2 *Mapas Autoorganizados de Kohonen.*

Los mapas autoorganizados o KSFM (Kohonen Self-organizing Feature Map) conciben una estructura de red parecida al aprendizaje competitivo, pero donde cada unidad neuronal tiene asociada una vecindad topológica con otras unidades neuronales. Durante el proceso de entre-namiento, son actualizadas tanto las unidades neuronales ganadoras como aquellas que se encuentran en su vecindad. El tamaño de esta vecindad decrece a medida que avanza el entrena-miento hasta alcanzar la unidad, es decir, la red KSFM se transforma en una red de aprendizaje competitivo.

Siendo $W_i(n)$ el peso asociado a la i-ésima unidad neuronal e Y el vector de entrada, se calcula la distorsión $d[Y,W_i(n)]$, i = 1, …, N_c estableciéndose que la neurona de índice i* es la de menor distorsión. Por otro lado, estará definido el subespacio N_i*(n) como la vecindad topológica asociada a la unidad i*. La vecindad topológica incluirá a aquellos vectores que se encuentren a una distancia inferior a un umbral (que irá disminuyendo) de la unidad ganadora. Con todos estos elementos la ecuación de actualización del peso, de la que se puede deducir la función de activación utilizada, será:

$$W_i(n+1) = \begin{cases} W_i(n) + \eta(n)[Y - W_i(n)] & i \in N_{i^*}(n) \\ W_i(n) & en\ otro\ caso \end{cases} \quad (1.21)$$

Es lógico deducir que la estructura KSFM implica una aumento considerable de la carga computacional con respecto al aprendizaje competitivo, pues las actualizaciones afectan a toda la vecindad. Sin embargo, es este mismo hecho el que permite solucionar el problema de la infrautilización de unidades neuronales visto. Por otro lado, también supone un notable incremento de la complejidad el cálculo de las vecindades en cada paso del algoritmo.

1.5 Codificación

En la sección anterior hemos discutido el problema de cuantificar escalar o vectorialmente una fuente de información. Como resultado de esta cuantificación, se obtiene un nivel específico de reconstrucción. Para transmitir hasta el receptor esos posibles $L = 2^n$ niveles de cuantificación (o para representar en la imagen comprimida) se necesita asignar una palabra código a cada nivel. Así, el decodificador podrá identificar el nivel de reconstrucción correspondiente mirando a la entrada apropiada de tabla de palabras código. Para que esto sea posible a cada nivel de cuantificación se le debe asignar una palabra código diferente. Además, como se transmitirá (o se tendrá almacenado) muchas palabras código de forma secuencial, se deben diseñar de forma que se puedan identificar cada una de ellas en cualquier secuencia. A los códigos mencionados se los denomina unívocamente decodificables.

1.5.1 *Códigos de Longitud Fija.*

Supongamos que el resultado de la cuantificación escalar o vectorial es un mensaje con L posibles valores a_i, $i = 0, \ldots, L\text{-}1$ cada uno de ellos correspondiendo a un nivel de reconstrucción. El método más sencillo de obtener el conjunto de palabras código es utilizando un código de longitud fija. Usando este método cada elemento del mensaje se codifica mediante una palabra código que tiene la misma longitud que el resto. La forma más sencilla de construir estos códigos es representando cada a_i pasando el valor del índice i a base binaria usando $n = \log_2(L)$ dígitos binarios o bits. En la Tabla 1.2 podemos ver un ejemplo para $L = 8$ niveles.

Tabla 1.2: Ejemplo de un código de longitud fija.

Nivel de reconstrucción	Palabra de código
a_0	000
a_1	001
a_2	010
a_3	011
a_4	100
a_5	101
a_6	110
a_7	111

1.5.2 *Códigos de Longitud Variable y Entropía.*

La codificación Huffman es una codificación de longitud variable que consiste en asignar a cada posible salida de un codificador de datos un símbolo de canal según su probabilidad de aparición. Se trata de un codificador de entropía, pues trata de aproximar la longitud media de

símbolo a la entropía del sistema.

La entropía (h) de un sistema de comunicaciones caracteriza la información que los símbolos o señales de dicho sistema $\boldsymbol{S}$ contiene y viene dada por:

$$h = -\sum_i p(i) \log_2 p(i) \quad i \in S \tag{1.22}$$

La sumatoria se aplica a todo símbolo i posible del sistema, cuya probabilidad de ocurrencia es $p(i)$. La entropía constituye el límite inferior de tamaño medio de código alcanzable.

El algoritmo de codificación Huffman (usada en el estándar JPEG [17-24], ver Apéndice B), diseñado por D. A. Huffman, trata de acercarse al límite impuesto por la entropía del sistema. Para entenderlo es conveniente estudiar un ejemplo sencillo, el que mostramos en la Figura 9. Partimos de un conjunto de 5 símbolos, los cuales ordenamos en orden descendente según su probabilidad de ocurrencia, tal como vemos en la Fig.1.9. A continuación agrupamos los dos símbolos inferiores creando uno nuevo de probabilidad igual a la suma de las probabilidades de dichos símbolos y reordenamos. Esto se repite hasta llegar a la probabilidad unidad. En este punto recorremos la cadena hacia atrás, asignando valores distintos (uno o cero) a cada rama de una unión, concatenándolo con los valores asignados anteriormente, de tal forma que el código para cada símbolo inicial se forma recorriendo su trayectoria de derecha a izquierda.

En este ejemplo, si la codificación hubiera sido de tamaño de código fijo, este tamaño hubiera sido igual 3 bits por símbolo. Con la codificación Huffman, multiplicando el tamaño de cada símbolo por la probabilidad de que aparezca y sumándolo todo, obtenemos un tamaño medio igual a 1.98 bits por símbolo. Este número se acerca considerablemente a la entropía, que en este caso es igual a 1.955, según la ecuación (1.22). Resulta por tanto notable la conveniencia de la utilización de este tipo de codificación.

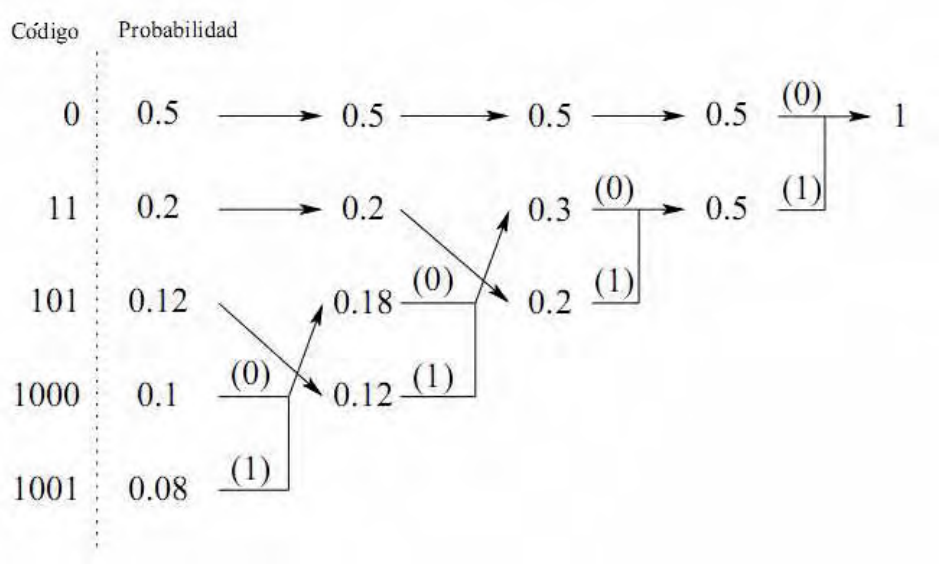


Figura 1.9: Ejemplo de generación de códigos Huffman.

Debemos destacar además que tal como se forman las palabras códigos, es imposible que coincida un segmento inicial de una de ellas con otra completa. Si el cero es utilizado como símbolo aislado, no será utilizado como principio de ningún otro símbolo y así con cualquier

combinación. Esto evita en la decodificación cualquier confusión, aunque no se indique explícitamente el límite entre símbolos, siempre que se conozcan las palabras código fijadas.

1.6 Codificadores de Forma de Onda (método no-conexionista)

En la codificación de forma de onda se codifica directamente los niveles de intensidad de la imagen o alguna variación sencilla de estos niveles de intensidad como la diferencia entre pixels consecutivos. La ventaja más importante de este tipo decodificación es su simplicidad. Los codificadores de forma de onda no pretenden explotar las características específicas de una clase particular de señales y por lo tanto se van a poder utilizar para un rango muy amplio de señales (voz e imagen). En algunos casos la tasa de compresión alcanzada puede ser similar a los codificadores de transformada (ver Apéndice A), pero en otros casos las tasas alcanzadas son significativamente menores.

En principio, se podría utilizar diferentes tipos de cuantificación y codificación en los codificadores de forma de onda, sin embargo, en general, se utiliza cuantificación escalar y codificación de longitud fija.

1.6.1 *Pulse Code Modulation* (*PCM).*

Es la forma más sencilla de codificación de forma de onda. Simplemente, la imagen de intensidad se pasa a través de un cuantificador uniforme. En la Fig.1.10 podemos ver el esquema de este tipo de codificación. El sistema PCM no sólo se puede utilizar para codificar los niveles de intensidad, sino que se puede utilizar para codificar los coeficientes transformados (en el caso de codificación de transformada) y los parámetros (en el caso del codificación basada en modelos).

Una forma sencilla de mejorar el sistema PCM básico es utilizando un cuantificador no uniforme en lugar del uniforme ya que los niveles de intensidad no suelen estar distribuidos de forma uniforme. Para ello se suele utilizar la técnica de compactación ya explicada.

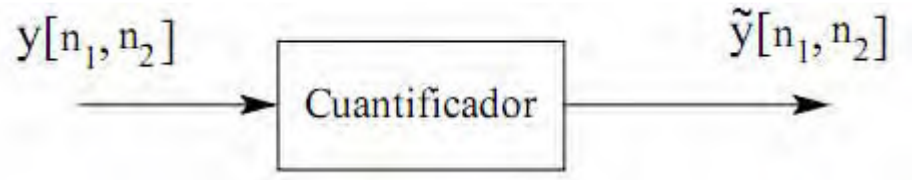


Figura 1.10: Sistema PCM.

1.6.2 *Modulación Delta (DM).*

En un sistema PCM la intensidad de la imagen se cuantifica escalarmente, pero no se explota para nada la correlación existente entre los pixels de la imagen. Una forma de hacer esto manteniendo la cuantificación escalar es la modulación delta (DM). En un sistema DM, se codifica con un bit (2 niveles) la diferencia entre la intensidad de dos pixels consecutivos. Aunque el rango dinámico de la señal diferencia es el doble, la varianza de la señal diferencia es significativamente menor debido a la elevada correlación existente entre los niveles de intensidad de dos pixels cercanos en la imagen. En la Fig.1.11 podemos ver el esquema del codificador y del decodificador DM. La señal error $e[n_1, n_2]$ (diferencia) del píxel actual con respecto al anterior se cuantifica con un bit de forma que $e_q[n_1, n_2]$ toma el valor $\Delta/2$ si $e[n_1, n_2]$ es positivo y $-\Delta/2$ si

es negativo, donde Δ es el tamaño del escalón del cuantificador (por lo general un entero). Esta señal cuantificada se codifica con un bit y se transmite. En el receptor la señal recuperada se obtiene sumando el error cuantificado $e_q[n_1, n_2]$ a la señal reconstruida para el píxel anterior. En la misma Fig.1.11, z^{-1} representa un retardo unitario, es decir, de un ciclo en el lazo cerrado de la figura y proviene de la Transformada Z, con $z = e^{ST}$, $s = \alpha+j\beta$, $j = \sqrt{-1}$ y T es el período, Por otra parte, e_q es la señal codificada, siendo la Fig.7(a) en codificador y la 7(b) 4l decodificador.

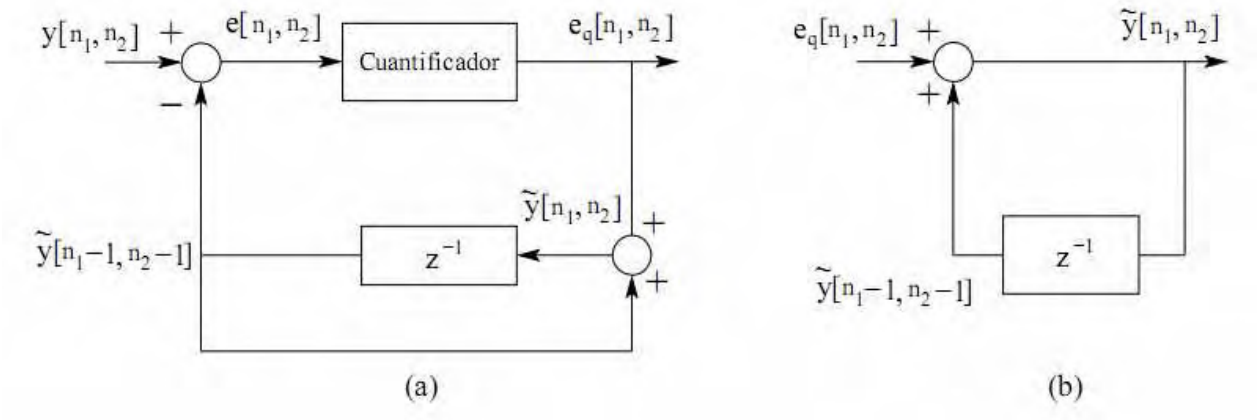


Figura 1.11: Sistema DM. (a) Codificador. (b) Decodificador.

Un parámetro importante de diseño del sistema DM es el tamaño del escalón Δ. Cuando la señal varía lentamente, la señal recuperada varía rápidamente en torno de la señal deseada ($\pm\Delta/2$ en torno a ella). Este tipo de error se conoce como error granular. Un valor grande de Δ dará lugar a un error granular mayor, por lo que se prefieren valores de Δ pequeños. Cuando la señal varía (crece o decrece) rápidamente, si Δ es pequeño la señal reconstruida tardará bastante en alcanzar a la señal deseada. Este error se conoce como error de sobrecarga dependiente. Para este caso valores de Δ grandes son deseables. Es por tanto necesario tomar una solución de compromiso entre el error granular y el de sobrecarga dependiente.

1.6.3 *Differential PCM (DPCM).*

DPCM es un método predictivo que se puede considerar una generalización del método DM: se utilizan más de un píxel para predecir el actual y se utiliza un cuantificador de más de un bit. Los métodos de codificación predictivos pretenden eliminar la correlación existente entre pixels consecutivos antes de la cuantificación. Para ello tanto en el codificador como en el decodificador se realizan predicciones de la muestra actual en función de las muestras anteriores, de tal manera que el valor que se cuantifica es la diferencia entre la predicción y el valor real del pixel. Mediante esta resta suprimimos en cada punto la información redundante dada ya por los puntos anteriores. El diagrama de bloques genérico de este tipo de compresores se muestra en la Fig.1.12, tanto la parte del codificador como la parte del decodificador.

La señal de partida está constituida por una secuencia de valores discretos que forman bidimensionalmente la imagen digitalizada original, $y[n_1, n_2]$. A cada valor se le sustrae la pre-dicción calculada, la señal $\hat{y}[n_1, n_2]$. Esta predicción no se realiza directamente a partir de la señal $y[n_1, n_2]$, sino que el predictor tiene como entrada la señal que se obtendrá en el decodificador, $\tilde{y}[n_1, n_2]$. Esto se hace para evitar la acumulación de errores en el decodificador. El error

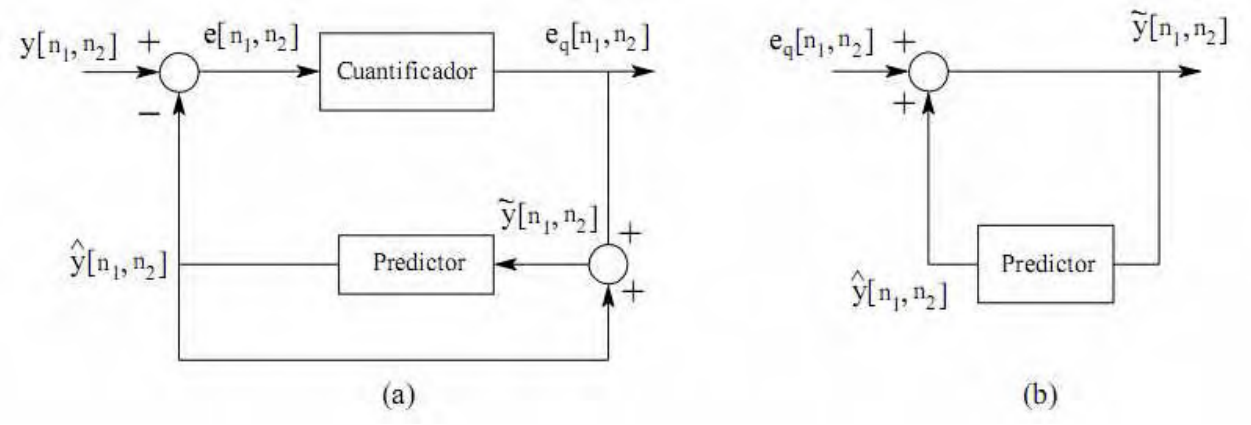


Figura 1.12: Diagrama de un codificador predictivo. (a) Codificador. (b) Decodificador.

De predicción será por lo tanto:

$$e[n_1, n_2] = y[n_1, n_2] - \hat{y}[n_1, n_2] \tag{1.23}$$

Sobre esta secuencia se realiza una cuantificación reduciendo el número de niveles, lo que añadirá cierto ruido de cuantificación $q[n_1, n_2]$. Es este error de predicción cuantificado lo que constituirá la imagen comprimida. La señal que se recibirá en el lado del decodificador será por tanto:

$$e_q[n_1, n_2] = e[n_1, n_2] + q[n_1, n_2] \tag{1.24}$$

En el decodificador vemos que se lleva a cabo exactamente la misma predicción que en el codificador a partir de la misma señal $\tilde{y}[n_1, n_2]$, añadiéndole la secuencia $e_q[n_1,n_2]$ a la señal predicha. La señal reconstruida es por tanto:

$$\begin{aligned} \tilde{y}[n_1, n_2] &= e_q[n_1, n_2] + \hat{y}[n_1, n_2] \\ &= e[n_1, n_2] + q[n_1, n_2] + \hat{y}[n_1, n_2] \\ &= y[n_1, n_2] + q[n_1, n_2] \end{aligned} \tag{1.25}$$

El principal problema de este tipo de compresores es la determinación de los coeficientes del predictor. Sabemos que la señal predicha será una combinación lineal fija de muestras anteriores ponderadas, lo cual se puede expresar matemáticamente de la forma:

$$\hat{y}[n_1, n_2] = \sum_{i=1}^{p1} \sum_{j=1}^{p2} a_{i,j}\, \tilde{y}[n_1 - i, n_2 - j] \tag{1.26}$$

donde $a_{i,j}$ serán los coeficientes del predictor y p_1, p_2 los niveles del predictor en ambas direcciones. Normalmente para el cálculo de coeficientes que optimicen el predictor se suele aproximar $\tilde{y}[n_1, n_2]$ por $y[n_1, n_2]$ para evitar la componente no lineal $q[n_1, n_2]$. Esta aproximación es válida si el número de niveles del cuantificador del error de predicción no es demasiado pequeño. Los pesos del predictor se determinan mediante un proceso de minimización del error cuadrático medio del error de predicción.

1.7 Codificadores de Transformada (método no-conexionista)

El objetivo principal de los codificadores de transformada es alterar la distribución de los valores que representan el nivel de intensidad para que muchos de ellos puedan ser eliminados o por lo menos cuantificados con muy pocos bits. De esta forma conseguimos, al igual que en los métodos que utilizan predicciones, una disminución de la dependencia estadística de los pixels antes de pasar a la fase de cuantificación.

La transformación no se realiza sobre toda la imagen sino que ésta es dividida en bloques de tamaño fijo (normalmente de 8 x 8 o 16 x 16 pixels) y sobre ellos se realiza la transformación. De esta forma reducimos el coste computacional considerablemente.

Una forma de entender el resultado de la transformación es asimilar cada bloque de 8 x 8 puntos de la imagen original como un vector en un espacio de 64 dimensiones, expresado en los ejes canónicos. La transformación realizará un cambio a otra base con el objetivo de que un número pequeño de los nuevos vectores base sigan las direcciones principales de los datos, de tal forma que en las nuevas coordenadas tengamos dos tipos de componentes: unas (en menor número) tomarán valores grandes y otras (en mayor número) se aproximarán a cero. Por lo tanto el número de niveles con los que se cuantificará cada una de las nuevas coordenadas variará de unas a otras.

Matemáticamente, la transformación consiste en expresar la función bidimensional original $y[n_1, n_2]$ como combinación lineal de una familia de funciones $a^*_{k_1,k_2}[n_1,n_2]$ (con el símbolo * nos referimos a la conjugación) ponderada cada una por un coeficiente $TU[k_1, k_2]$. La secuencia bidimensional de estos coeficientes constituye la señal transformada. Hablamos de transformaciones unitarias, de tal forma que las funciones utilizadas en la transformación deben ser ortonormales entre sí. Todo esto podemos expresarlo mediante las siguientes igualdades (suponiendo las señales de tamaño N x N):

$$TU[k_1,k_2]=\sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1} y[n_1,n_2]\, a_{k_1,k_2}[n_1,n_2] \tag{1.27}$$

$$y[n_1,n_2]=\sum_{k_1=0}^{N-1}\sum_{k_2=0}^{N-1} TU[k_1,k_2]\, a^*_{k_1,k_2}[n_1,n_2] \tag{1.28}$$

La ortonormalidad de la familia de funciones implica lo siguiente:

$$\sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1} a_{k_1,k_2}[n_1,n_2]\, a^*_{k_1',k_2'}[n_1,n_2]=\begin{cases}0 & si\ k_1\neq n_1\ o\ k_2\neq n_2\\ 1 & si\ k_1=n_1\ y\ k_2=n_2\end{cases} \tag{1.29}$$

Como dijimos anteriormente, los subbloques de las imágenes pueden entenderse como matrices de tamaño N x N. Por otro lado, en la mayoría de los casos, la familia de funciones ortonormales de la transformación podrán formar una matriz separable. Por tanto, las transformaciones equivalen a la premultiplicación de cada bloque de la imagen original por una matriz de transformación de tamaño N x N y la postmultiplicación por la traspuesta de dicha matriz de transformación. La operación matricial de la transformación será de la forma:

$$[TU] = [A][Y][A]^T \tag{1.30}$$

Siendo A la matriz para las $a_{k_1,k_2}[n_1,n_2]$, A^T para $a^*_{k_1',k_2'}[n_1,n_2]$, e Y para las $y[n_1,n_2]$.

Algunas de las transformaciones que veremos no parten directamente de una familia de funciones sino que están basadas en matrices de transformación cuyas filas son ortonormales entre sí. Sin más, pasamos a ver algunas de las transformaciones más utilizadas.

1.7.1 *Transformada de Fourier Discreta (DFT).*

Es la particularización de la transformada de Fourier para las funciones discretas bidimensionales. La familia de funciones ortonormales es de la forma:

$$a_{k_1,k_2}[n_1,n_2] = \frac{1}{N} e^{-jk_1\frac{2\pi}{N}n_1} e^{-jk_2\frac{2\pi}{N}n_2} \tag{1.31}$$

Una de las características de la imagen es que en el dominio transformado la energía se concentra en los coeficientes con k_1 y k_2 pequeños (bajas frecuencias espaciales), lo cual es muy conveniente para la compresión. Sin embargo no se pueden obviar los coeficientes de las frecuencias más altas pues contienen la información de los bordes a los cuales es muy sensible el ojo humano. Por lo tanto se opta por una cuantificación con un número de niveles variable según el coeficiente, disminuyendo conforme aumenta la frecuencia. Una de las ventajas que presenta esta transformación es la separabilidad de cada función base bidimensional en dos unidimensionales idénticas. Además, existe un algoritmo que permite calcular la transformada de una función con una reducción considerable del coste computacional, el algoritmo FFT [25-32]. Por el contrario presenta dos grandes inconvenientes. Uno de ellos es la posibilidad de que, aunque partamos de una imagen con valores reales, los coeficientes transformados puedan ser números complejos. Además, esta transformación equivale a la construcción de una imagen que es la repetición periódica de la original, por lo que cada punto del marco de la imagen original pasa a colindar con los puntos del borde opuesto, apareciendo variaciones bruscas que se traducen en un aumento de la energía que se encuentra en las altas frecuencias. Debido a este último efecto, la DFT no es la transformada óptima para la compresión de imágenes.

1.7.2 *Transformada del Coseno Discreta (DCT) [2, 33-51].*

La transformada del coseno discreta equivale a una DFT realizada sobre una imagen $y_a[n_1, n_2]$ de tamaño $2N$ x $2N$ que se forma mediante dos reflexiones (una según un eje horizontal y otra según un eje vertical) de la imagen original:

$$\begin{aligned} DCT\{y[n_1,n_2]\} &= DFT\{y_a[n_1,n_2]\} \\ &= \frac{1}{2N}\sum_{n_1=0}^{2N-1}\sum_{n_2=0}^{2N-1} y_a[n_1,n_2]\, e^{-jk_1\frac{2\pi}{2N}n_1} e^{-jk_2\frac{2\pi}{2N}n_2} \end{aligned} \tag{1.32}$$

Si sobre esta fórmula reagrupamos y realizamos un cambio de variable obtenemos:

$$DCT\{y[n_1,n_2]\} = \frac{2}{N}\sum_{n_1=0}^{2N-1}\sum_{n_2=0}^{2N-1} y[n_1,n_2]\cos\left[\frac{k_1\pi}{2N}(2n_1+1)\right]\cos\left[\frac{k_2\pi}{2N}(2n_2+1)\right] e^{j(k_1+k_2)\frac{\pi}{2N}} \tag{1.33}$$

Vemos que la transformada no es real porque aparece el término $e^{j(k_1+k_2)\frac{\pi}{2N}}$. Esta exponencial compleja desaparecerá si consideráramos el origen en el centro de la imagen formada $y_a[n_1, n_2]$, que realmente es el origen de la imagen original. De hecho, dado que es un término de fase que no afecta a la imagen, podemos obviarlo. Así, podemos expresar cada una de las funciones base de la forma:

$$a_{k_1,k_2}[n_1,n_2]=\frac{2}{N}\cos\left[\frac{k_1\pi}{2N}(2n_1+1)\right]\cos\left[\frac{k_2\pi}{2N}(2n_2+1)\right] \tag{1.34}$$

La ventaja principal de la DCT frente a la DFT es que los pixels de los bordes de la imagen original se convierten en adyacentes de sí mismos al realizar las reflexiones, eliminando los cambios abruptos que aparecían en la DFT y por tanto no aparecen nuevas componentes en frecuencias altas. A esto hay que añadir que la DCT tiene todas las ventajas de la DFT y siempre es real, lo cual la convierte en una buena opción como fase de transformación en un algoritmo de compresión. De hecho es la transformación utilizada en el conocido estándar JPEG [17-24].

1.7.3 *Transformada de Walsh-Hadamard [2].*

Esta transformación normalmente es definida de forma matricial. Las matrices de transformación separables de distinto orden parten de la matriz de orden más bajo (2 x 2) que es de la forma:

$$[WHT]_1=\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \tag{1.35}$$

Las matrices de orden superior se generan sustituyendo cada elemento de la matriz $[WHT]_1$ por la matriz misma, obteniéndose por tanto:

$$\begin{aligned}[WHT]_2&=\frac{1}{\sqrt{2}}\begin{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} & \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\\ \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} & -\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\end{pmatrix}\\ &=\frac{1}{2}\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & -1 & 1 & -1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\end{pmatrix}\end{aligned} \tag{1.36}$$

y así sucesivamente, generando matrices mayores aunque siempre de tamaño $2n$ x $2n$. Tal como dijimos, cada fila de las matrices son los vectores base de la transformada. Dado que definimos la transformación mediante matrices $[A]$ los coeficientes transformados se obtienen de la forma descrita en la ecuación (1.30).

La transformada de Walsh-Hadamard al igual que la DCT produce un pseudo-espectro espacial de la imagen. En la DCT cada fila de la matriz de transformación es el producto de dos funciones coseno con una frecuencia ascendente. En el caso de la transformada de Walsh-Hadamard, la frecuencia de cada vector fila se refleja en los cruces por cero. Para que la frecuencia de los

vectores base sea ascendente con el número de fila se suele realizar una reordenación de la matriz de la ecuación (1.36) de la forma:

$$[WHT]_2 = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix} \begin{matrix} \text{cambios de signo} \\ 0 \\ 1 \\ 2 \\ 3 \end{matrix} \tag{1.37}$$

De esta forma, los coeficientes transformados se relacionan con la rapidez de cambio en los vectores de datos, lo cual corresponde a la idea intuitiva de frecuencia.

Es notable la sencillez de cálculo de esta transformación dado que no se producen multiplicaciones para cada pixel, sólo hay sumas (+1/2) y restas (-1/2) entre ellos.

1.7.4 *Transformada de Karhunen-Loève.*

En esta transformación el objetivo es la máxima adaptación a un conjunto de imágenes. La matriz de transformación será la matriz de varianza-covarianza del conjunto de imágenes, específica por tanto para dicho conjunto [10-12, 52-112].

La principal ventaja de esta transformación es su adaptación total a los datos, demostrándose matemáticamente que es la transformación que alcanza la máxima descorrelación posible. Sin embargo es poco viable pues la matriz de transformación cambia con cada conjunto de imágenes, y cuanto más amplio sea este conjunto menos eficaz es la transformación. Además, se trata de una transformada no separable y que no cuenta con un algoritmo para una computación rápida como en el resto de los casos.

1.8 Codificadores de Resolución Variable (método no-conexionista)

Agrupamos bajo este epígrafe a un conjunto de métodos de compresión basados esencialmente en la generación progresiva de la imagen reconstruida a partir de versiones de distinta resolución de la imagen original. Las diferencias entre los métodos surgen de su forma de obtener dichas versiones de distintas resolución y del posterior procesado al que son sometidas.

1.8.1 *Interpolación Jerárquica.*

La interpolación jerárquica (HINT) es un método piramidal basado en el submuestreo. Partiendo de una versión de baja resolución de la imagen original (S_0), obtenida submuestreando dicha imagen, genera versiones sucesivamente de más resolución utilizando interpolación. De esta forma, lo que se transmitirá o almacenará tras ser codificado con algún codicador de entropía será en primer lugar la imagen de menor resolución S_0. De ella se obtiene la versión con el siguiente nivel de resolución situando nuevos pixels entre los anteriores. Para hallar el valor de estos nuevos pixels se utiliza algún método de interpolación aplicada sobre los pixels ya conocidos (interpolación lineal, exacta, etc). En transmisión esta operación también se realiza y el valor calculado es sustraído al valor del píxel correspondiente en la imagen original y es la diferencia obtenida la que se transmitirá para la reconstrucción final de la imagen. Se trata por tanto de un método concebido en principio para una compresión sin pérdidas.

1.8.2 *Pirámide Diferencia.*

Este método de resolución variable se basa en la construcción de una pirámide de medias y el cálculo de una pirámide diferencia que contiene las diferencias entre los niveles de la pirámide de medias, ver Fig.1.13.

El algoritmo parte de la división de la imagen original, que será la base de la pirámide de medias, en grupos de cuatro pixeles sobre los que se halla la media redondeada al entero más cercano, conformando estos valores medios el siguiente nivel de la pirámide, de tamaño cuatro veces menor. Este proceso se repite de nuevo para cada nivel progresivamente de menor resolución. Paralelamente se forma la pirámide diferencia siendo igual en cada punto de cada nivel a la diferencia que se obtiene al sustraer a cada uno de los pixeles de cada grupo del mismo nivel de la pirámide de medias el valor medio correspondiente recogido en el siguiente nivel de las pirámides de medias. De esta forma, la imagen original se reconstruirá a partir del vértice de la pirámide de medias y de la pirámide diferencia. Al vértice de la pirámide de medias, que es un solo valor, se le suma cada uno de los cuatro valores del último nivel de la pirámide diferencia, obteniéndose los cuatro valores del penúltimo nivel de la pirámide de medias. El proceso se repite con cada uno de estos cuatro valores y los 16 valores del siguiente nivel de la pirámide diferencia, obteniéndose los 16 valores del antepenúltimo nivel de la pirámide de medias, y así sucesivamente hasta recuperar la imagen original. De nuevo estamos ante un método sin pérdidas.

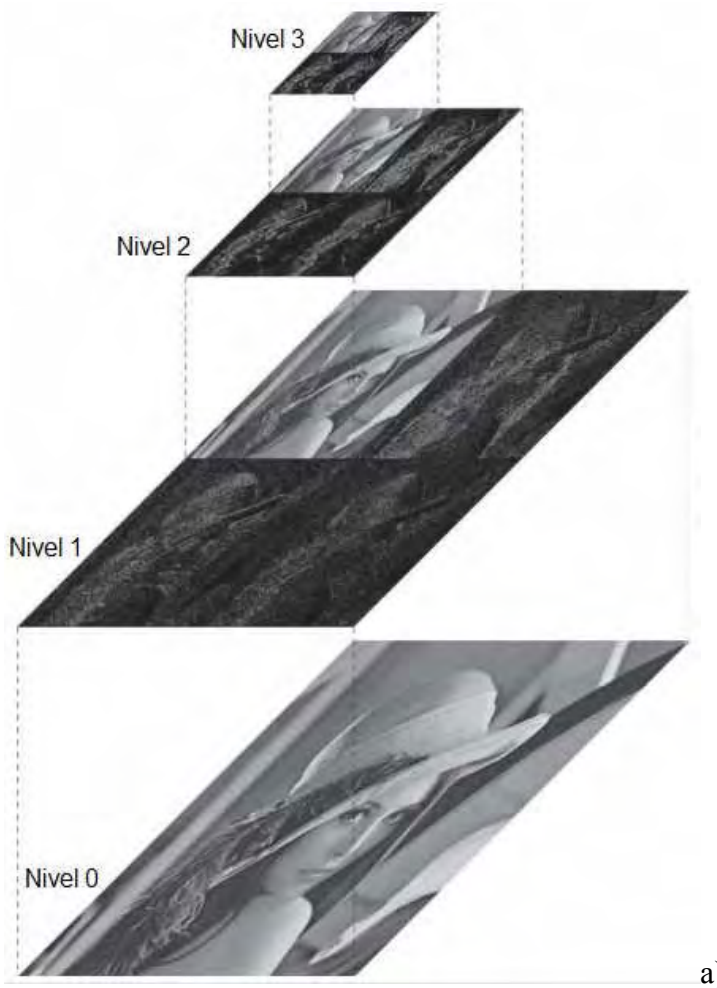
De este método existen variantes como la pirámide diferencia reducida (RDP) o la Transformada-*S* [2-4].

1.8.3 *Codificación por Plano de Bits.*

La codificación por planos de bits se basa en la separación de la imagen original con p bits por píxel en p imágenes de 1 bit por pixel. Partiendo de la representación en binario de los valores de los pixeles, seleccionamos un solo bit de la misma posición para todos los pixeles. De esta manera formaremos una nueva imagen de tamaño $M \times N$ con la particularidad de que es binaria, es decir, todos sus pixeles toman el valor 1 ó 0. Esta imagen se denomina plano de bits. Por ejemplo, el plano de bits más significativo es una imagen $M \times N$ que contiene los bits más significativos de los valores de los pixeles en la imagen original. Repitiendo este proceso para el resto de las posiciones de los bits, se llega a la descomposición de la imagen en los p planos de bits, imágenes binarias de tamaño $M \times N$. En las Fig.1.14 tenemos un ejemplo de esta descomposición. La primera imagen corresponde a la radiografía original y las siguientes son sus 8 planos de bits, del más significativo al menos.

Como podemos observar, los planos de bits más significativos (MSB) contienen la información estructural y a medida que los bits son menos significativos la naturaleza del plano es cada vez más ruidosa, añadiendo menos información para la reconstrucción de la imagen. Sin embargo, estos últimos planos no pueden obviarse porque aparecerían falsos bordes.

La descomposición en planos de bits facilita la transmisión progresiva. Cada plano, de más a menos significativo, se codifica y transmite secuencialmente. En recepción, la imagen reconstruida inicialmente es una imagen binaria, el plano MSB, pero a medida que se reciben más planos se van añadiendo más niveles de grises. Este método de compresión es en un principio sin pérdidas, pero la reconstrucción puede detenerse en cualquier plano, obteniendo tasas de compresión notablemente mayores.



a)

b)

Figura 1.13: Pirámide de medias y diferencias para tres niveles jerárquicos: a) Estructura piramidal donde la porción nítida de Lena se corresponde con las medias, mientras que los mosaicos difusos se corresponden con las diferencias, b) Idem a) con la ubicación de cada nivel en la estructura final.

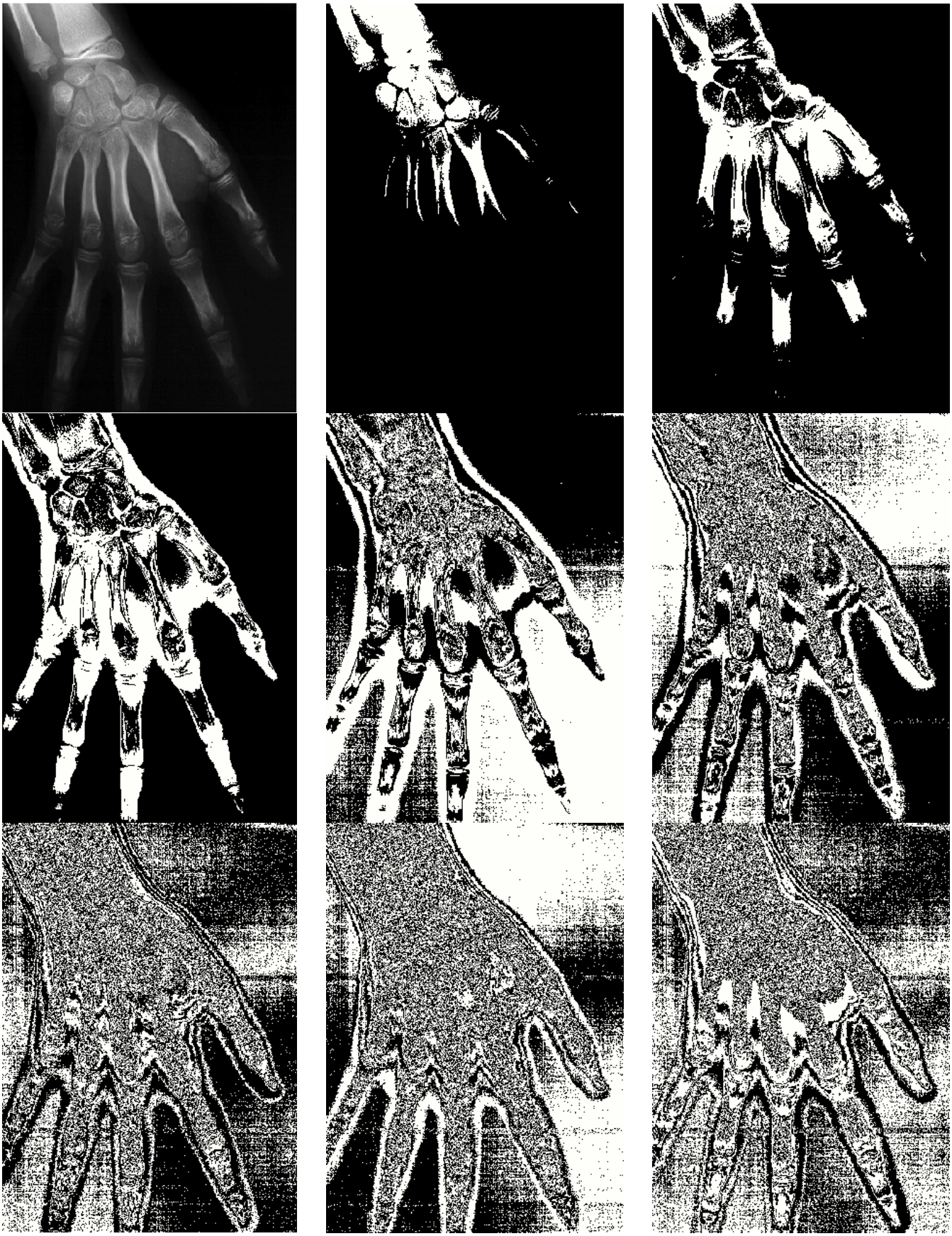


Figura 1.14: Ejemplo de descomposición de una imagen en sus planos de bits. Imagen original [1,1], planos de bits: 1º [1,2], 2º [1,3], 3º [2,1], 4º [2,2], 5º [2,3], 6º [3,1], 7º [3,2] y 8º [3,3].

Para la codificación se aprovecha la existencia de grandes áreas uniformes en los planos de bits más significativos. El método típicamente utilizado para la compresión de datos con grupos amplios de ceros y unos consecutivos es la codificación run length encode o RLE, utilizada ampliamente en la codificación de documentos en el campo de los facsímiles. Básicamente consiste en codificar sólo el primer bit y las longitudes en sistema binario de los grupos de unos y ceros, formando una serie de símbolos RLE. La naturaleza de la ristra de bits que cada símbolo codifica se establece mediante una regla de alternancia, es decir, tras una ristra de unos siempre se codifica una ristra de ceros y viceversa. Sin embargo, la codificación se complica por la limitación del número representable de componentes de cada ristra al establecer un número de bits por símbolo. No podemos codificar con un determinado número de bits una ristra de un número ilimitado de componentes al no ser posible la traducción a binario de su longitud utilizando sólo dichos bits. Lo dicho no necesariamente obliga a romper la alternancia e incluir un símbolo especial de flag. Simplemente, si el tamaño máximo de corrida es ***n***, en el caso de que corresponda a continuación representar una secuencia de ***m*** (***m > n***) ceros, por ejemplo, se debería codificar como: ***n 0*** (***m-n***), o sea una corrida de ***n*** ceros, ***0*** unos y el resto de ceros (en este caso suponiendo que ***m < 2*n***, sino se debería repetir el mismo mecanismo).

La codificación RLE pierde gran parte de su utilidad en los planos menos significativos ya que, tal como se observa en la Fig.1.14 las áreas uniformes van haciéndose cada vez más pequeñas, tendiendo a una estructura ruidosa. Estos planos menos significativos deberán codificarse de forma distinta o, debido a que no aportan información estructural, eliminarse, aunque eso implicaría que el método de codificación dejaría de ser sin pérdidas. Además, a la imagen reconstruida habría que añadirle ruido aleatorio para evitar la aparición de falsos bordes, tal como dijimos. La codificación RLE por sí sola no es por tanto adecuada para estos últimos planos. Sin embargo, gracias a la codificación de longitud variable que se aplicará sobre los símbolos RLE se consigue un tamaño medio de código pequeño, de tal manera que, en conjunto, se logra una cierta compresión, aunque mínima, sobre los planos menos significativos.

Como decíamos, tras la codificación RLE tenemos una serie de símbolos en los que hemos codificado cada uno de los planos de bits de más a menos significativo de tal forma que la reconstruíción será multirresolutiva. A cada uno de estos símbolos RLE es conveniente asignarles un código adecuado para su transmisión. El método más sencillo sería asignar a cada símbolo una palabra de longitud fija, pero este método es muy ineficiente. El conocimiento de las probabilidades de cada símbolo RLE nos permite alcanzar codificaciones de canal más eficientes, las denominadas codificaciones de longitud variable o codificaciones de entropía. La codificación Huffman es la codificación de entropía más utilizada.

Para la aplicación de la codificación de entropía es necesaria una fase de adquisición de experiencia en la que obtengamos las probabilidades de cada símbolo RLE. Estas probabilidades se aproximan realizando un muestreo sobre un universo de imágenes de entrenamiento con las características del tipo de imagen objetivo, aplicándoles RLE y promediando la frecuencia de aparición de cada símbolo.

Un ejemplo de las distintas versiones que se generan durante la reconstrucción progresiva que se lleva a cabo se muestra en la Fig.1.15. En ella vamos observando que la imagen reconstruida tiene cada vez más resolución, pues está formada por un número sucesivamente mayor de planos. En el punto en que se reciben los 5 primeros planos, la imagen reconstruida apenas tiene pérdidas perceptibles. La imagen reconstruida con 7 planos es visualmente sin pérdidas. Esto es debido a la limitación del ojo humano, que sólo percibe un número de niveles de gris aproximadamente igual a 100. Con 7 bits se tienen ya 128 niveles, con lo que la escisión de cada uno de ellos en otros

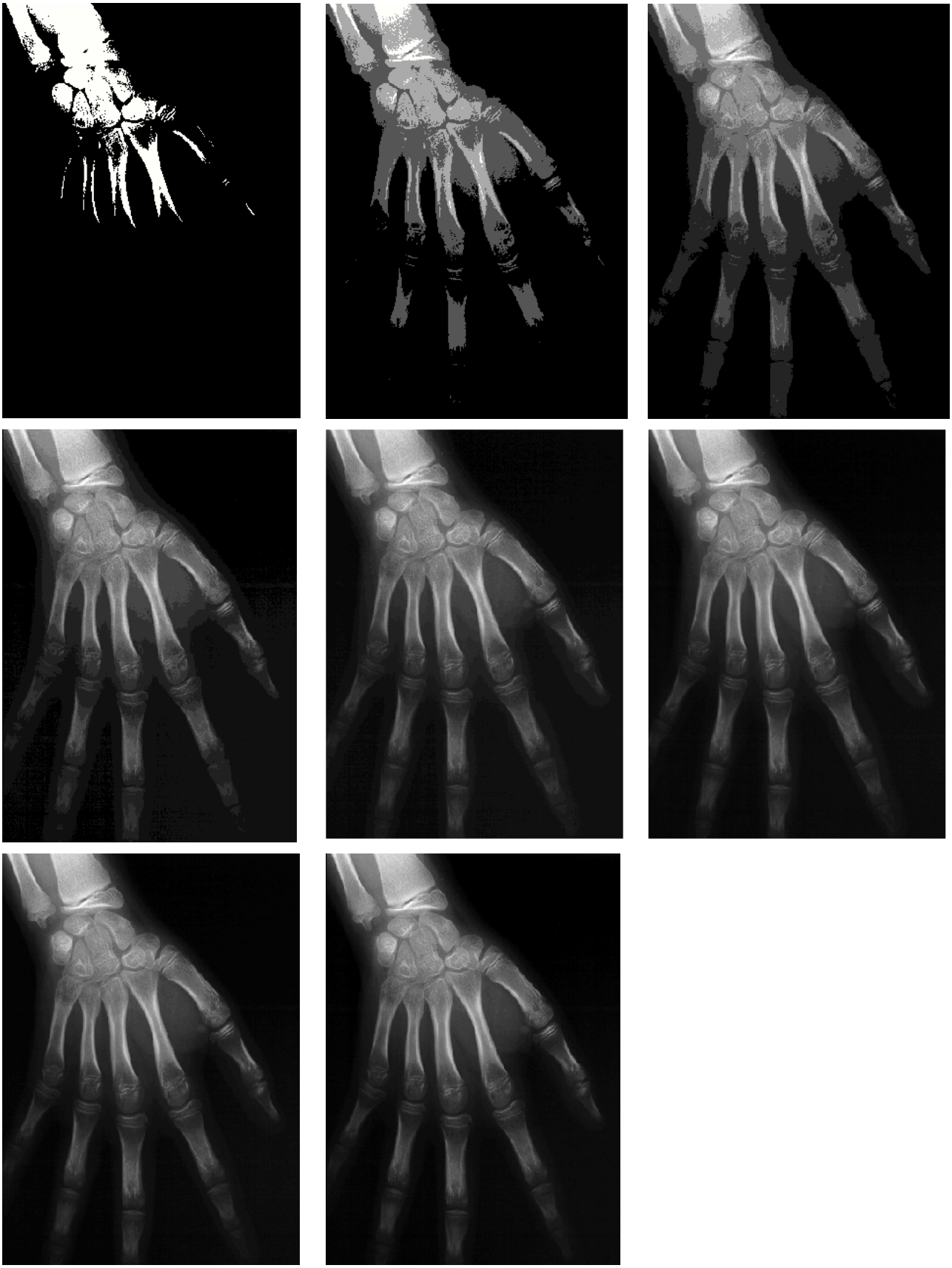


Figura 1.15: Ejemplo de reconstrucción progresiva de una imagen codificada. Cada radiografía tiene un plano más de resolución. La [1,1] tiene el 1º, la [1,2] 1º y 2º, la [1,3] 1º a 3º, la [2,1] 1º a 4º, la [2,2] 1º a 5º, la [2,3] 1º a 6º, la [3,1] 1º a 7º, y la [3,2] todos.

Tabla 1.3: Resultados de la codificación por planos de bits.

	SNR [dB]	Tasa de compresión
1 Plano	3.190	76.211:1
2 Planos	7.097	25.056:1
3 Planos	11.510	11.889:1
4 Planos	17.556	6.432:1
5 Planos	24.317	4.007:1
6 Planos	30.378	2.756:1
7 Planos	39.072	2.125:1
8 Planos	∞	1.681:1

dos, al añadir el último plano, no será percibida por un receptor humano. En la Tabla 1.3 se recogen las tasas de compresión y la calidad lograda para las imágenes de la Fig.1.15.

1.9 Codificación por Sub-bandas y Wavelets (método no-conexionista)

Los codificadores por sub-bandas y los basados en wavelets [10-12, 51, 113-244] aprovechan la existencia de diferentes densidades de energía en cada frecuencia espacial, aplicando el hecho de que para una codificación más adecuada es conveniente procesar cada intervalo de frecuencias de forma distinta.

1.9.1 *Codificación por Sub-bandas.*

En los codificadores por sub-bandas el proceso de codificación comienza con un banco de *M* filtros paso-banda que se aplica a la imagen original (*M* suele ser una potencia de dos). Esto podría parecer un aumento considerable de la información, pues de una señal pasamos a *M* señales. Sin embargo, por los principios del muestreo, cada señal de ancho de banda *M* veces menor extraída de cada filtro puede ser diezmada por un factor igual a dicho *M* sin que se produzca ningún tipo de pérdidas, de tal forma que el conjunto global de información a codificar permanece inalterado, no aumenta. El banco de filtros que descompone la señal debe satisfacer ciertos requerimientos de error de reconstrucción, retraso del procesado e interferencias entre bandas. De la Fig.1.16 podemos observar el proceso completo formado por filtros de aplicación entre pixeles vecinos alineados horizontalmente de baja frecuencia $H_L(z)$ y alta frecuencia $H_H(z)$, siendo z la misma definida anteriormente, mientras que $\downarrow 2$ significa submuestreo en 2, y $\uparrow 2$ sobremuestreo en 2.

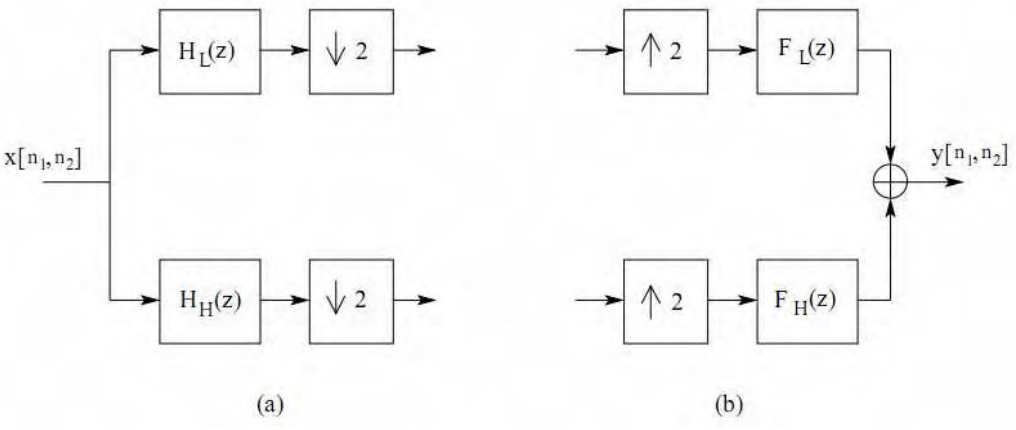


Figura 1.16: Ejemplo sencillo (*M* = 2) de codificación por sub-bandas. (a) Análisis. (b) Síntesis.

La mayor parte de la energía en las imágenes reales se encuentra en las bajas frecuencias, de tal forma que el cuantificador asignado a esta banda contaría con un número mayor de niveles que el asignado a frecuencias altas. Por otro lado no podemos obviar la energía a altas frecuencias pues en ellas se recoge la información de bordes. Esta sería la principal base de la asignación de bits a cada subbanda. Normalmente se lleva a cabo, para el grupo de imágenes a comprimir, un estudio más detallado de su distribución espectral, deduciendo las características más adecuadas para los cuantificadores de cada banda de frecuencia.

Es importante reseñar que en nuestro caso el espectro espacial de una imagen es bidimensional, mientras que la explicación y el diagrama dado es unidimensional. Para dos dimensiones, la descomposición se complica pues no dividimos un rango de frecuencias en intervalos, sino que dividimos un área de frecuencias bidimensionales, normalmente rectangular, en áreas más pequeñas, normalmente cuadrantes, de tal forma que el filtrado se deberá llevar a cabo para las dos componentes de las frecuencias.

1.9.2 *Codificación por Wavelets.*

La utilización de wavelets para la codificación de una imagen no dista demasiado de la división por subbandas, no obstante, con wavelets, dicha división se realiza de forma más conveniente.

La razón de la aparición de las transformadas wavelet surge de las limitaciones en tiempo o frecuencia de las señales reales. Debido a la interdependencia temporal (o espacial) y frecuencial de las señales es imposible tener una señal continua limitada temporalmente, dado que esto implicaría un espectro infinito. Sin embargo, en la digitalización, se registra una función continua a través de una ventana limitada, lo cual se traduce, en el dominio frecuencial, en una convolución con la transformada de dicha ventana y la consecuente expansión del espectro. En general, no es posible lograr de forma independiente exactitudes arbitrarias en el tiempo y en la frecuencia simultáneamente, sino que un aumento de precisión en un dominio disminuirá la precisión en el otro.

Partiendo de este hecho, la transformación mediante wavelets se realiza a partir de una función $\Psi(x)$ apropiada para ser escalada y desplazada de tal forma que, al aplicar la convolución de la función temporal o espacial con ella, podamos centrar nuestro análisis en cierta área rectangular dentro del espacio bidimensional que forman los ejes de la variable independiente x (ya sea espacio o tiempo) y su equivalente frecuencial f. Dicho rectángulo satisfacerá las necesidades específicas del proceso global que utiliza este tipo de transformaciones. Matemáticamente queda reflejado en la siguiente ecuación, aplicando la transformación a la función $f(x)$:

$$W f(s,u) = < f(x), \Psi_s(x-u) > \tag{1.38}$$

donde

$$\Psi_s(x) = \sqrt{s}\, \Psi(s\, x) \tag{1.39}$$

Según modifiquemos los parámetros s y u, varia la forma y la posición de la ventana rectangular en el plano formado por los ejes de tiempo (o espacio) y frecuencia. El producto base (intervalo en x) por altura (intervalo en f) de este rectángulo se mantendrá constante, pues depende de la función wavelet aplicada en la transformación, es decir, una expansión en tiempo supondrá una compresión en frecuencia y viceversa.

En las aplicaciones para la compresión de imágenes materializamos toda esta teoría en una descomposición de la imagen de entrada en versiones de distinta resolución al aplicarle una familia de transformadas wavelets tal como la familia ortonormal y discreta de bases wavelet de Meyer, teniendo en cuenta que los parámetros de escalado (s) y de translación (u) no deben tomar cualquier valor arbitrario. Proyectar la función según el diferente escalado y translación de la wavelet es equivalente a filtrarla con un banco de filtros. De esta forma conseguimos separar la función en subbandas manteniéndose la mejor relación posible entre la precisión espacial y la frecuencial, dadas las características de las transformadas wavelets.

Análogamente al proceso seguido en la codificación por subbandas, se realiza una descomposición en ambas direcciones (una versión separable en dos dimensiones de la transformación unidimensional descrita), separando las bajas frecuencias de las altas. Este filtrado se repetirá el número de veces necesario para alcanzar el número final de subbandas establecido, obteniéndose nuevas imágenes de distintas características, cada una de las cuales será tratada de forma diferente, pero manteniéndose el número total de elementos. De hecho, la codificación mediante wavelets no es sino un caso particular de codificación por subbandas.

1.10 Codificadores Basados en Modelos (método no-conexionista)

En la codificación basada en modelos, la imagen o una parte de ésta se representa según un modelo estadístico y se emplean los parámetros del modelo para sintetizar la imagen posteriormente. En el extremo transmisor se estiman los parámetros del modelo analizando la imagen y en el extremo receptor se sintetiza la imagen a partir de los parámetros del modelo estimados y cuantificados.

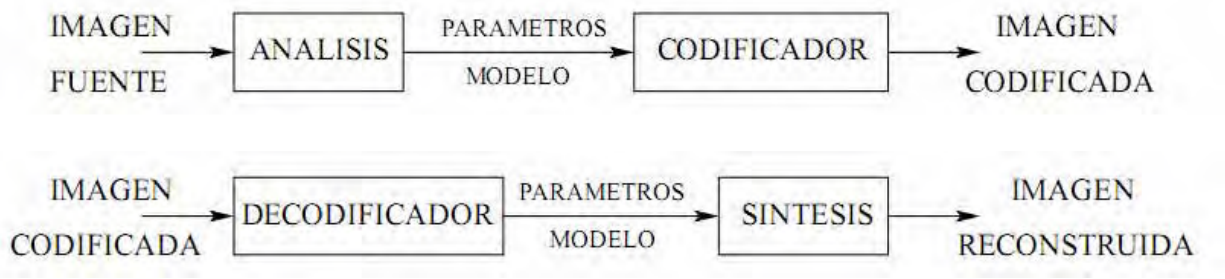


Figura 1.17: Esquema de codificación basada en modelos.

En la Fig.1.17 podemos observar esquemáticamente el transmisor y el receptor de este tipo de codificación como procesos de análisis y síntesis. Los codificadores basados en modelos logran sintetizar imágenes empleando una tasa binaria considerablemente menor que en los codificadores de forma de onda o de transformada. Sin embargo, están todavía en desarrollo y es necesario seguir en esta línea durante algún tiempo para que sean viables en la práctica. Desarrollar un modelo sencillo de imagen a partir del cual se pueda sintetizar una imagen con significado es un problema muy complicado. Además, estimar los parámetros del modelo y sintetizar la imagen a partir de estos parámetros es computacionalmente muy caro.

La codificación basada en modelos tiene su razón de ser en la noción de que para sintetizar imágenes con significado no es necesario reproducir de forma detallada las intensidades de la imagen. Por ejemplo, el fondo de la imagen, como por ejemplo hierba, el cielo, una pared, etc. No

son esenciales en la inteligibilidad de la imagen de forma que se pueden reemplazar por otros fondos con características similares que se puedan sintetizar con un modelo sencillo. En un modelo de imagen, se pretende mantener las características esenciales de la imagen y aproximar de forma no precisa las características no esenciales utilizando para ello modelos sencillos. Este método contrasta enormemente con las codificaciones de forma de onda y de transformada para las que se pretende reconstruir la imagen de intensidad. En estos casos la diferencia entre la imagen original y la reconstruida depende de la cuantificación de los parámetros. Si esta cuantificación no se lleva a cabo la imagen reconstruida es exactamente igual a la original. En la codificación basada en modelos, la diferencia entre la imagen sintetizada y la original es debida a la cuantificación de los parámetros y al error en el modelo. No es posible reconstruir la imagen original incluso aunque los parámetros del modelo no se cuantificaran. El número de parámetros para la codificación basada en modelos es mucho menor que el caso de codificación de forma de onda o codificación de transformada. De esta forma se logra una tasa tan baja en la imagen codificada.

Una imagen típica consiste en varias regiones con diferentes características. Será conveniente modelar cada una de las clases de regiones con modelos diferentes. Regiones tales como hierba, agua, cielo, pared, etc. tienen propiedades de orden o estructuras repetitivas denominadas textura. Hay dos técnicas básicas para modelar texturas:

- Utilizar un patrón elemental básico y repetirle de acuerdo a una regla determinística o probabalística.

- Modelar la textura como un campo aleatorio con ciertas propiedades estadísticas. Dos texturas con estadística de segundo orden similar aparecen como texturas muy similares para el sistema visual humano. Se han desarrollado muchos modelos de campos aleatorios de segundo orden para modelar texturas [2, 5].

Para poder utilizar este método es necesario emplear una técnica que permita segmentar la imagen en regiones con textura similar para poder modelar cada una de ellas por separado con un modelo diferente. Sin embargo, no es posible modelar adecuadamente regiones de la imagen que contienen o forman parte de objetos como texturas puras. Para modelar estas regiones podemos utilizar la idea de que un conjunto reducido de contornos situados adecuadamente pueden representar la mayor parte de la inteligibilidad presente en estos objetos. Así se puede representar una región de la imagen con un conjunto de contornos y, si se desea, rellenar las regiones entre estos contornos empleando las texturas adecuadas.

1.11 Compresión de Imágenes Médicas

Hoy en día el desarrollo de nuevas tecnologías para el tratamiento, almacenamiento y transmisión de imágenes médicas en formato digital está siendo muy estudiado ya que han surgido problemas legales con respecto a los codificadores que actualmente están en el mercado. Es evidente que el codificador ideal para este tipo de imágenes ha de ser sin pérdidas, debido a la importancia de la calidad de la imagen decodificada a la hora de hacer un diagnóstico. Sin embargo, existen codificadores con pérdidas que logran tasas de compresión bastante elevadas sin sacrificar mucho la calidad de la imagen. A estos últimos se les denomina visualmente sin pérdidas (o de pérdidas admisibles), ya que no presentan pérdidas visuales bajo condiciones normales de observación.

El estudio sobre este tipo de técnicas está chocando actualmente con todo tipo de problemas legales que se plantean para su implementación. Evidentemente, los compresores sin pérdidas no tie-

nen problemas legales, pero ofrecen una modesta tasa de compresión de 2:1 aproximadamente. Por esto, se está llevando a cabo un gran esfuerzo de investigación centrado en técnicas de compresión con pérdidas de baja tasa binaria (compresiones de 10:1), que descarten la información de poca relevancia para el diagnóstico. El estándar DICOM está trabajando en esta línea. La política reguladora en esta materia se ocupa menos de las patentes y de los estándares que de los dispositivos médicos que ofrecen software de compresión.

Actualmente, los radiólogos son muy escépticos respecto al uso de compresión con pérdidas para diagnóstico primario debido a la posible pérdida de detalles importantes. Sin embargo, estos algoritmos sí se están empleando para almacenamiento de imágenes ya diagnosticadas. Una línea de estudio interesante es catalogar el grado de compresión necesario para los diferentes tipos de aplicaciones radiológicas.

Actualmente, no existe ningún estándar legal de compresión de imágenes médicas. Esto significa que no hay ninguna referencia clínica para un tribunal a la hora de juzgar un caso en el que intervenga un dispositivo que use un método de compresión con pérdidas. Se traslada la responsabilidad a los usuarios de estos dispositivos y no a los fabricantes. Los usuarios deben conocer las especificaciones de los aparatos y escoger aquellos que les interesa.

Un estándar sería muy importante sobre todo en la transmisión de imágenes a distancia debido al desconocimiento de los dispositivos médicos del otro extremo de la línea de transmisión. En cualquier caso no es lógico que este estándar dependa de la decisión de un tribunal que cree jurisprudencia en todo este asunto, sino que esta tarea se debería llevar a cabo por un profesional de la medicina y del tratamiento de imagen. Esto puede requerir un gran esfuerzo pero evitará problemas en el futuro.

1.12 Implementación de un Sistema de Codificación Multirresolución

La fusión de un algoritmo sin pérdidas (la codificación por planos de bits) y un algoritmo con pérdidas (la cuantificación vectorial) se basa en la idea de la estratificación de la codificación de la imagen. En una primera fase se realiza la codificación con pérdidas, obteniéndose una imagen reconstruida con unas determinadas características de distorsión. Para las siguientes fases es necesaria la extracción en el lado del codificador de la imagen diferencia o imagen error que aparece entre la versión original y la reconstruida de la imagen de entrada. Es esta imagen diferencia la que se somete a las sucesivas etapas de la codificación sin pérdidas, durante la que se generan una serie de símbolos que codificados son añadidos a la trama de bits que constituirá la imagen codificada. En dicha trama se unifican por tanto las distintas fases de la codificación.

En la Fig.1.18 puede observarse el diagrama de bloques para el codificador y en la Fig.1.19 para el decodificador.

En la Fig.1.20 podemos ver un ejemplo de la reconstrucción de la imagen a partir de la codificación con cuantificación vectorial y los planos de bits de la imagen diferencia. Podemos observar un aumento progresivo de la calidad de la imagen final desde el margen superior izquierdo (con la mayor tasa de compresión) al margen inferior derecho (con la menor tasa de compresión), ver Tabla 1.4. En la Fig1.21 se puede ver la imagen diferencia entre la imagen original y la imagen cuantificada y en la Fig.1.22 podemos ver la imagen diferencia representada como el plano de signo y sus 8 planos de bits. En la Tabla 1.4 se recogen los resultados de las tasas de compresión y la calidad lograda para cada imagen de la Fig.1.20.

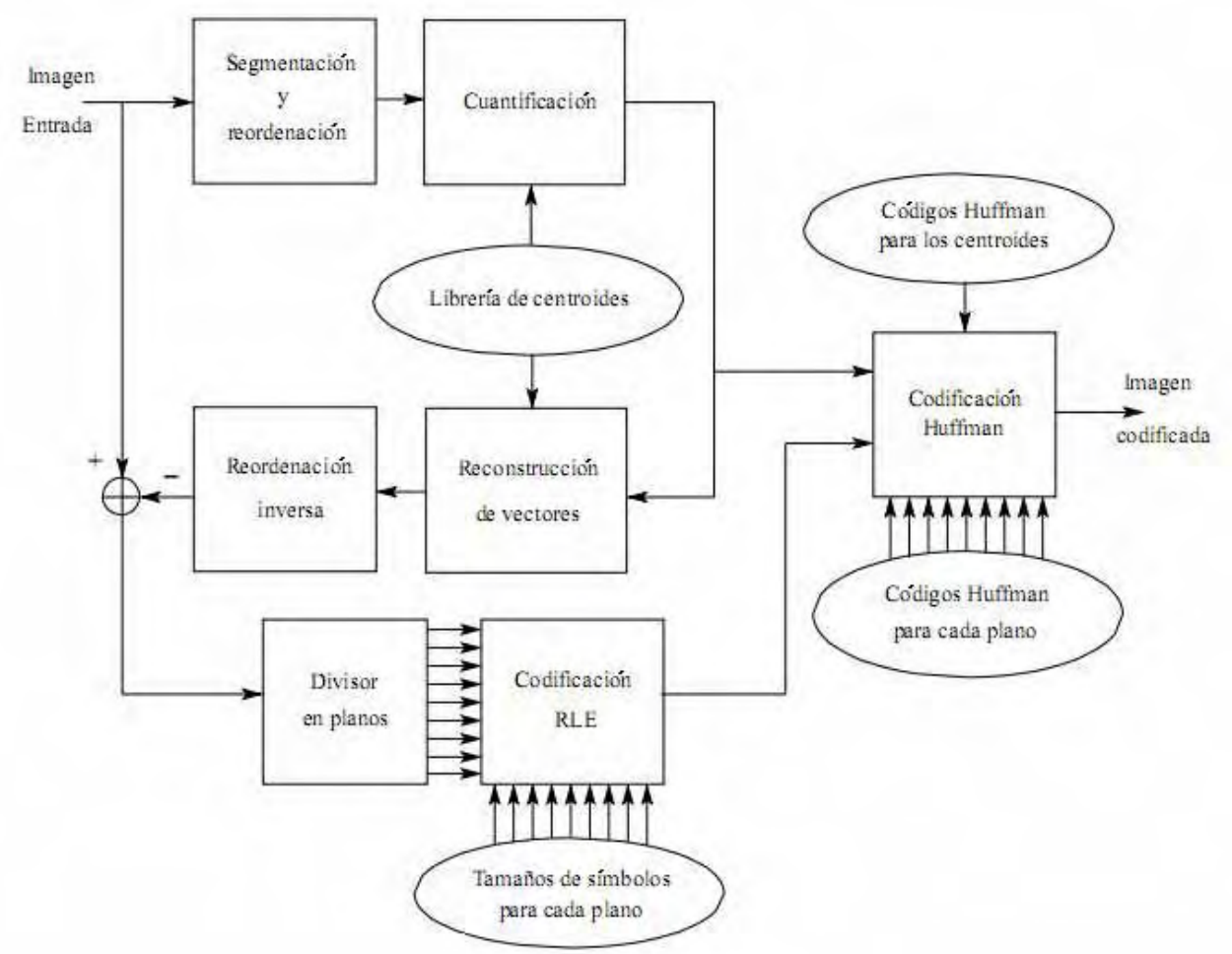


Figura 1.18: Diagrama de bloques del codificador del sistema multirresolutivo.

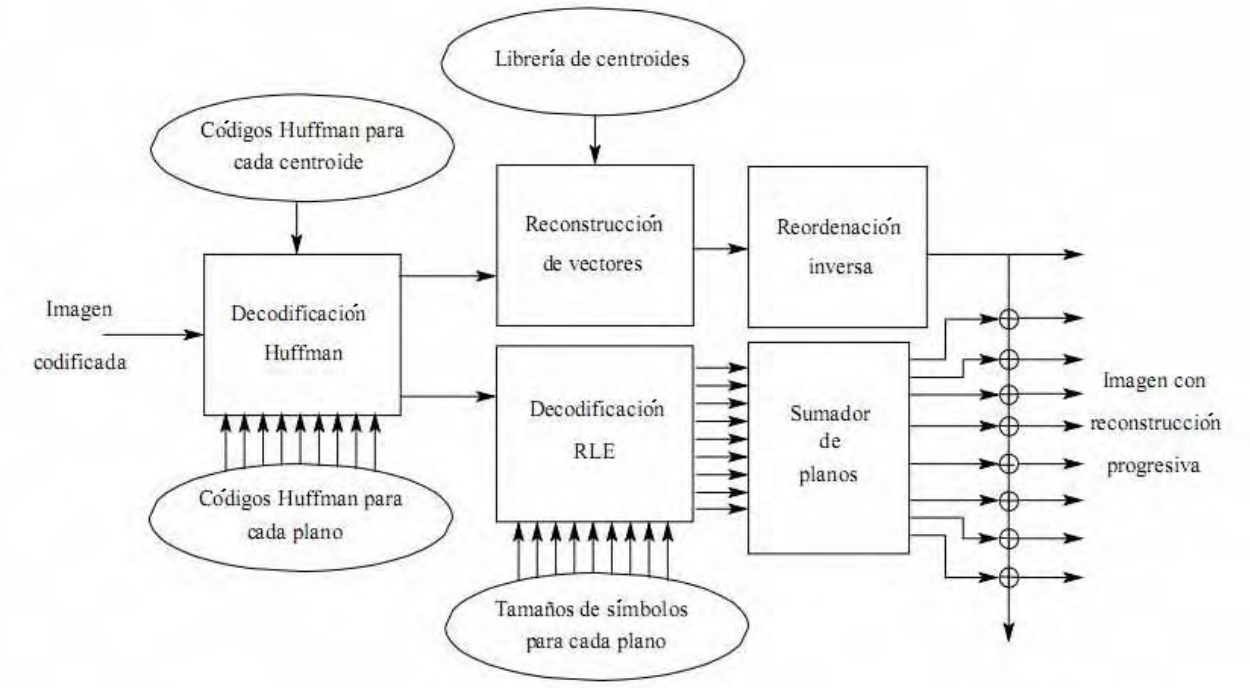


Figura 1.19: Diagrama de bloques del decodificador del sistema multirresolutivo.

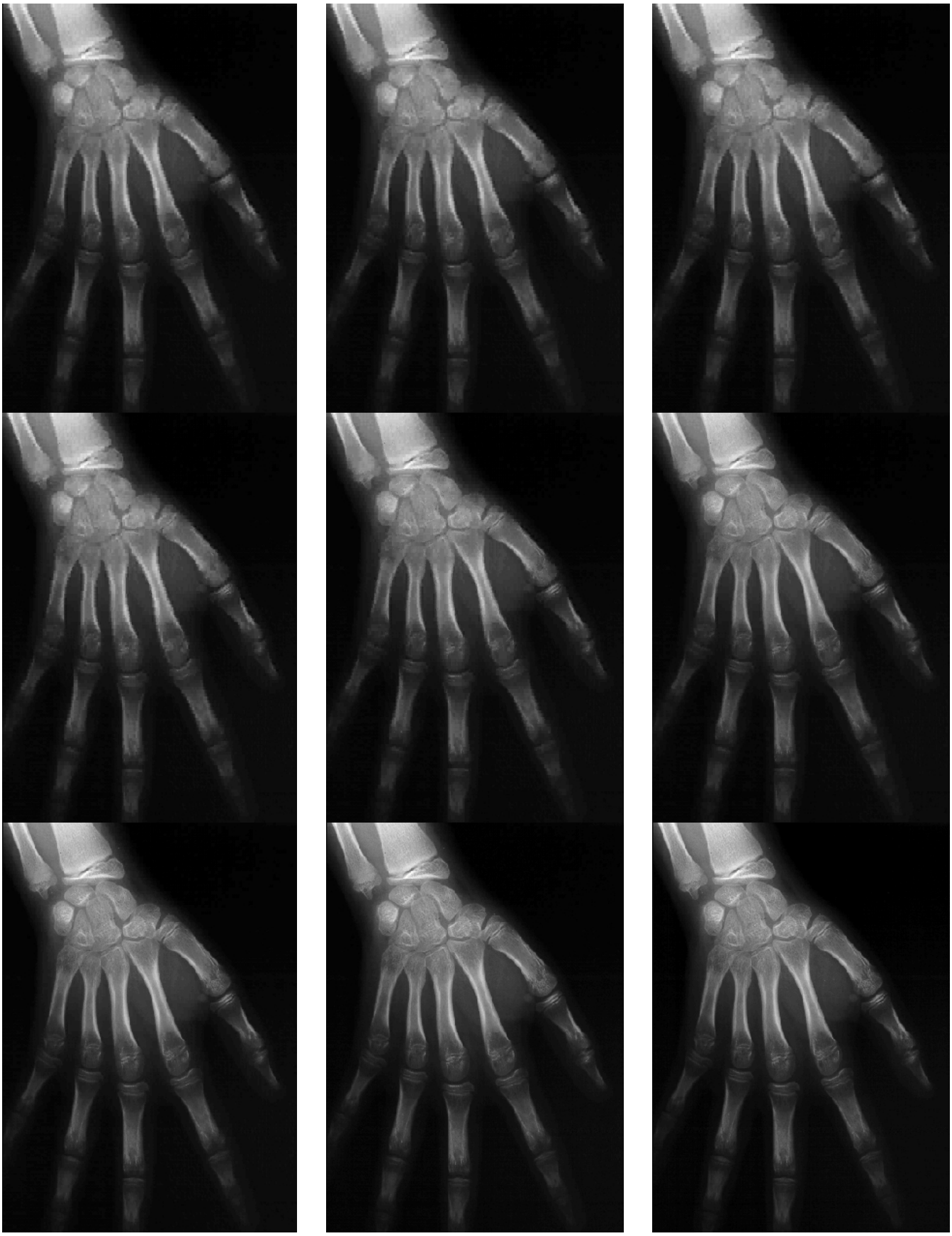


Figura 1.20: Ejemplo de reconstrucción de una imagen codificada. Peor calidad y mayor tasa de compresión (margen superior izquierdo) y mejor calidad y menor tasa de compresión (margen inferior derecho), ver Tabla 1.4.

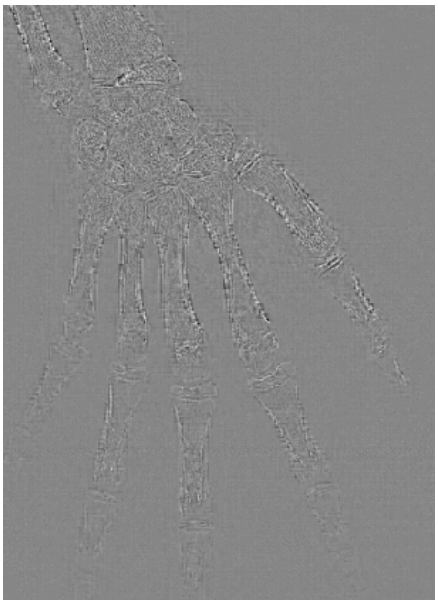


Figura 1.21: Imagen diferencia entre la imagen cuantificada y la original.

Tabla 1.4: Resultados de la codificación.

	SNR [dB]	Tasa de compresión
Imagen cuantificada	22.857	17.433:1
Imagen cuantificada + 1 Plano	22.857	5.785:1
Imagen cuantificada + 2 Planos	22.863	5.739:1
Imagen cuantificada + 3 Planos	23.248	5.682:1
Imagen cuantificada + 4 Planos	25.069	5.229:1
Imagen cuantificada + 5 Planos	28.472	4.381:1
Imagen cuantificada + 6 Planos	32.477	3.447:1
Imagen cuantificada + 7 Planos	39.650	2.441:1
Imagen cuantificada + 8 Planos	∞	1.878:1

La Tabla 1.4 muestra que a medida que la imagen cuantificada no involucra planos, tenemos la más baja relación señal a ruido (SNR) pero la más alta tasa de compresión, Fig.1.20 imagen [1,1]. Pero a medida que vamos involucrando más planos, se incrementa la SNR y baja notablemente la tasa de compresión, Fig.1.20 imagen [3,3]. Esto forma parte del día a día de los profesionales que utilizan la compresión de imágenes para Medicina, debiendo lidiar con el sutil compromiso entre espacio de los archivos almacenados y calidad de la imagen. Esta tarea es crítica, por cuanto un exceso de compresión genera archivos de menor tamaño, fáciles de almacenar y transmitir en un contexto de Telemedicina, pero por otra parte, la compresión excesiva trae aparejada la posibilidad de enmascarar un diagnóstico, es decir, que no aparezca lo que está, o que aparezca lo que no está. El exceso de compresión lesiona fundamentalmente los bordes y textura de una imagen [2].

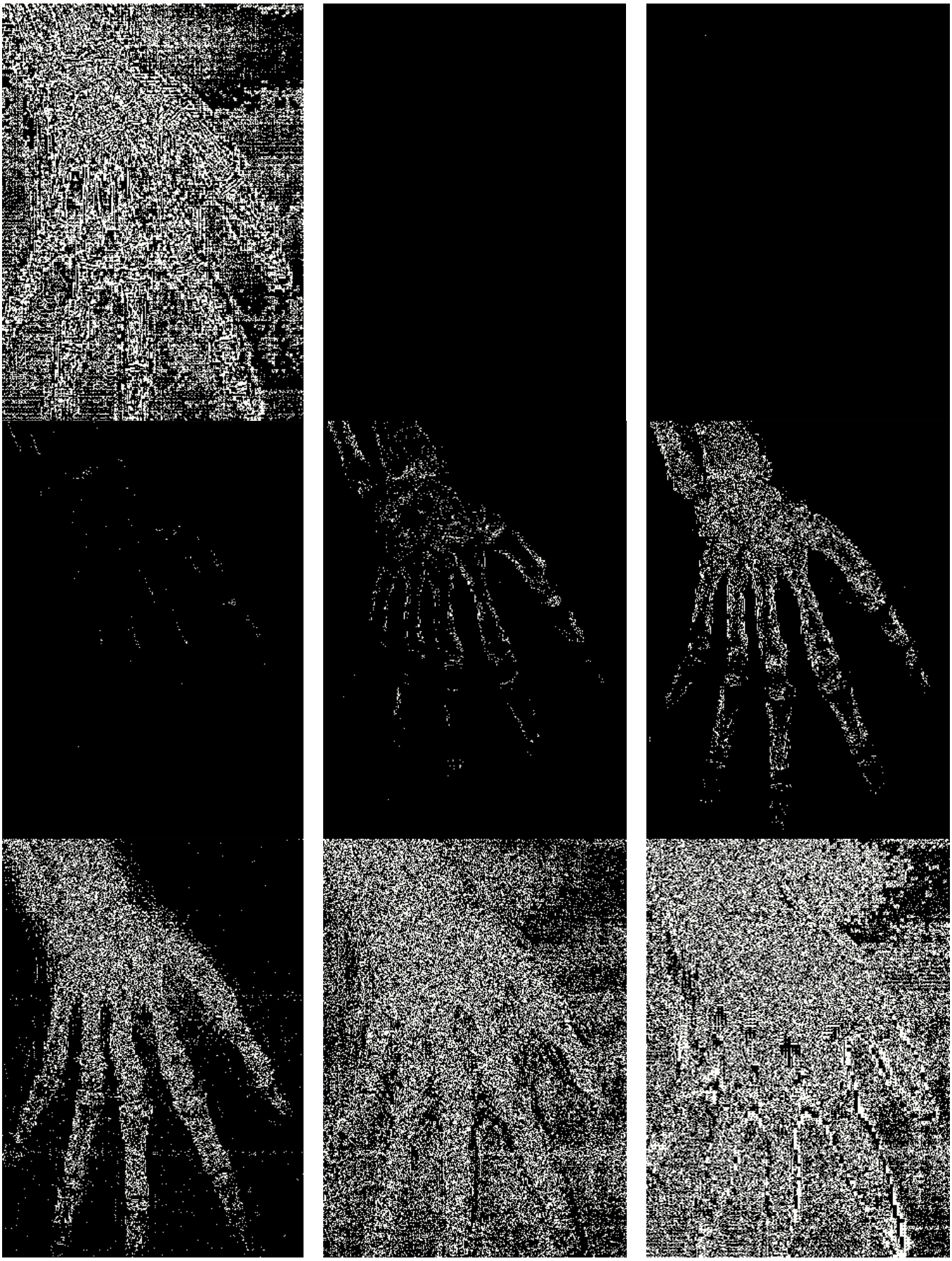


Figura 1.22 Ejemplo de descomposición de una imagen diferencia en sus planos de bits.
(plano de signo y planos del 7 al 0).

1.13 Conclusiones del capítulo

En este capítulo se analizaron en forma comparativa las diferentes técnicas de compresión de imágenes, en el ámbito de aplicación más sensible, es decir, el caso de imágenes médicas donde las pérdidas por compresión son menos admisibles, no obstante, sin pérdida de generalidad en relación a otras áreas de aplicación. Por otra parte, se analizaron las condiciones bajo las cuales se puede alcanzar un mejor rendimiento de compresión (con o sin pérdidas) ya sea mediante métodos conexionistas (Sección 1.4) y no-conexionistas (resto del capítulo).

Ninguno de los métodos aludidos alcanza - por sí mismo - el nivel de compresión y simultáneamente la calidad visual de recuperación que requieren los sistemas de TV Digital. La razón descansa en el hecho de que todos estos métodos (ya sean, con o sin pérdidas, conexionistas o no conexionistas, codificadores de onda o por transformada, etc.) involucran en su funcionamiento solo los parámetros tradicionales de una imagen en estos casos, es decir, la redundancia espacial de los pixeles de la misma (o temporal en el caso de video). Por lo tanto, una estrategia razonable tendiente a incrementar la tasa de compresión parece ser aquella que involucre adicionalmente algún parámetro que jamás haya sido considerado en un algoritmo de compresión de imágenes, y este resulta ser la resolución. Por lo tanto, este trabajo involucra con objeto de incrementar la tasa de compresión de los algoritmos de compresión actualmente en uso a la resolución de la imagen, mediante técnicas de reducción de resolución en el codificador y aumento o restauración de la misma en el decodificador. Estas técnicas son agrupadas bajo el nombre de técnicas de super-resolución. En consecuencia, la super-resolución sumada a las técnicas actualmente en uso de compresión constituirán desde este momento una nueva técnica de compresión conocida como supercompresión. La acción combinada de super-resolución más compresión permitirá multiplicar las tasas de compresión individuales de ambas, alcanzando así los estándares requeridos para la posible transmisión de servicios de TV Digital nunca antes posibles hasta la fecha en ninguna norma de TV Digital actualmente en uso.

Capítulo 2

Super-resolución, restauración de nitidez e interpolación bidimensional

En este capítulo se abordan las herramientas necesarias para la supercompresión de imágenes, es decir, la super-resolución como una técnica más completa que la simple restauración de la nitidez, así como la interpolación bidimensional como el mecanismo fundamental que da lugar primero al sub-muestreo en el codificador y luego al sobre-muestreo en el decodificador.

2.1 Introducción

La captura digital de imágenes produce representaciones discretas de escenas continuas. Esta discretización, en ambos, espacio e intensidad, es un proceso de muestreo que crea aliasing (un efecto despreciable en bordes y líneas de falta de continuidad en la forma de sierra o escalón), e información que se pierde en frecuencias por encima de la tasa de Nyquist. Es un hecho común desear obtener una imagen de mayor resolución a partir de una grilla de la imagen o un conjunto de imágenes, pero el aliasing y la pérdida de información de frecuencias convierte esto en un problema mal planteado (problema inverso). La solución típica para este problema (conocida en la literatura como reconstrucción de super-resolución de la imagen, o simplemente super-resolución) es usar un conjunto de imágenes de menor resolución relacionadas entre sí. Como cada una de estas imágenes tiene aliasing, la información de mayor frecuencia es ligeramente diferente, y bajo ciertas condiciones es posible "restaurar" algunas de las altas frecuencias perdidas por aliasing y reconstrucción.

Existen numerosos métodos [245-251] de realizar super-resolución. Muchos de ellos son computacionalmente costosos en su naturaleza, no obstante, permiten una considerable mejora en la resolución para modelos de movimiento complicados con un nivel de ruido y degradación de la imagen importantes, además de otros aspectos que no son considerados en este trabajo. Dadas las suposiciones de movimiento traslacional global, bajo ruido, espacio lineal y blur invariante en el tiempo debido a la PSF (en inglés, *point spread function*, es decir, *función de dispersión del punto*) del sensor de imagen (cámara), la reconstrucción por super-resolución de la imagen puede ser dividida en tres diferentes pasos:

- Registración
- Reconstrucción/Interpolación
- Restauración de Nitidez (Deblurring)

La registración de una imagen es una técnica que puede ser usada para determinar la traslación relativa entre las imágenes de entrada. Generalmente, lo deseable es hacer esto a partir del contenido de una imagen única, sin ningún conocimiento previo. Existen muchos métodos diferentes para llevar a cabo la registración [247]. En el contexto de la super-resolución de una imagen, la registración es requerida para determinar las compensaciones entre las imágenes con una precisión de hasta una pequeña facción de un pixel [247].

Una vez que las imágenes han sido registradas, todos los pixeles del conjunto pueden ser combinados para formar una imagen compuesta. La imagen resultante ya no es muestreada sobre una

grilla rectangular uniforme, sino que debido al movimiento traslacional global, tiene una estructura semi-uniforme, como se puede apreciar en la Figura 1 (donde AR significa alta-resolución y BR significa baja-resolución). Reconstruir los datos de la imagen en todos los puntos sobre la grilla de alta-resolución requiere que los datos de la estructura semi-regular sea interpolada y remuestreada. Es este problema de interpolación el que es tratado en este trabajo.

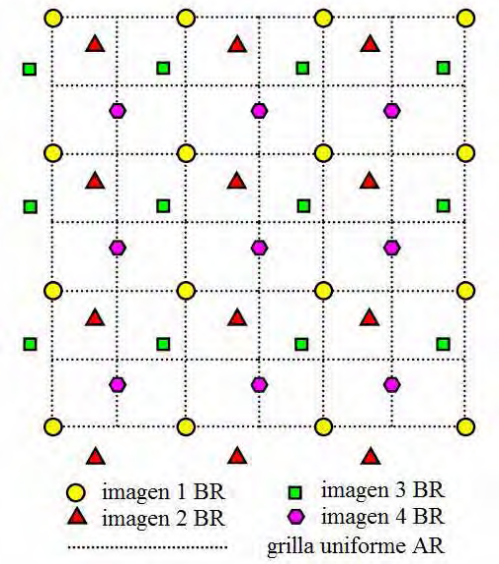


Figura 2.1: Imagen compuesta que muestra una estructura semi-uniforme.

Para la aproximación de super-resolución completa, podemos aplicar en este estadio un procedimiento de deblurring el cual restaure las altas frecuencias que han sido suprimidas por el proceso de bajada de resolución de la imagen. En este trabajo nosotros realizamos este deblurring después del proceso de interpolación via una convolución entre la imagen sobre-muestreada y una máscara convolutiva deducida específicamente en el próximo capítulo.

El objetivo principal de los métodos de Super-Resolution –en la práctica– es recuperar una imagen de alta-resolución de una o más imágenes de entrada de baja-resolución. Los métodos para SR pueden ser ampliamente clasificados en dos familias:

1. la super-resolución clásica multi-imagen (SRCMI), y
2. la super-resolución basada en ejemplo (SRBE).

En la SRCMI (e.g., [248] para referenciar solo una) son tomadas imágenes de baja-resolución de la misma escena en una suerte de desajuste subpixel (lo que significa que elementos de la grilla de una imagen caen en medio de los elementos de la grilla de otra, ver Fig.2.1). Cada imagen de baja-resolución impone restricciones lineales sobre valores de intensidad de alta-resolución desconocidos. Si hay disponibles suficientes imágenes de baja-resolución (en un desplazamiento subpixel), entonces el sistema de ecuaciones se convierte en determinado y puede ser resuelto para recuperar la imagen de alta-resolución. Prácticamente, no obstante, esta aproximación es numéricamente limitada solo a pequeños incrementos en resolución [248] (para factores más pequeños que 2).

Estas limitaciones han conducido al desarrollo de la SRBE también conocida como “alucinación

de imagen" (ver referencias en [248]). En la SRBE, la correspondencia entre una imagen de baja y su contraparte de alta resolución es aprendida como un par de entrenamiento (usualmente con un factor de escala relativo de 2), y luego aplicada a una nueva imagen de baja-resolución para recuperar su versión de alta-resolución más parecida. Las componentes de más altas frecuencias de SR han sido frecuentemente obtenidas por aplicaciones repetidas de este proceso. Se ha demostrado que la SRBE ha excedido los límites de calidad de la SR clásica. No obstante, y a diferencia de la SR clásica, los detalles de alta-resolución reconstruidos ("alucinados") para SRBE no garantizan proveer los detalles reales de alta-resolución (desconocidos).

También han sido propuestos sofisticados métodos para el sobre-escalado de imágenes basados en modelos de aprendizaje de bordes (e.g., [249-251]). El objetivo de estos métodos es magnificar (sobre-escalar) una imagen mientras mantiene la nitidez de los bordes y los detalles de la misma. En contraste con esto, en la SR (tanto en la basada en ejemplos como en la clásica) el objetivo es recuperar *detalles perdidos en la nueva imagen de alta-resolución* que no son explícitos y fácil de encontrar en la correspondiente imagen de baja-resolución (detalles más allá de la frecuencia de Nyquist de la imagen de baja-resolución). En la SR clásica, esta información de alta frecuencia se asume repartida a través de múltiples imágenes de baja-resolución, implícitamente encontrada ahí de una forma suavizada. En la SRBE, esta información perdida de alta-resolución se asume disponible en la base de datos de alta-resolución, y aprendida de los pares de ejemplos de baja-res/alta-res de dicha base de datos. No obstante, más allá de lo dicho anteriormente, en este trabajo, consideraremos la interpolación de a una sola y única imagen.

Por otro lado, el rendering de imágenes de baja resolución sobre displays de más alta resolución se ha convertido en una tarea muy común, en particular por el incremento de la popularidad de las webcams, cámaras en teléfonos celulares, video streaming de poco ancho de banda.

Entonces, existe una fuerte demanda en tiempo real, para la magnificación de imágenes de alta calidad. En este trabajo, sugerimos explotar la alta performance computacional de las *general-purpose computation on programmable graphics processing units* (GPGPUs) para el método de magnificación de la imagen original. A tal fin, proponemos un algoritmo amigable a la GPGPU para el sobre-muestreo de la imagen mediante una interpolación con restauración de los bordes, el cual evite los ringing artifacts (defectos de reconstrucción), excesivo blurring, y efecto serrucho o escalones de escalera en los bordes oblicuos. Al mismo tiempo, y dado que se cuenta con invarianza de escala de grises, se puede aplicar a imágenes color, lo cual permite un procesamiento en tiempo real de las imágenes completas sobre las GPGPUs utilizadas en la actualidad [6-9] con un gran aumento en la velocidad (speed-up).

2.2 Interpolación bidimensional

Una imagen digital no es una instantánea exacta de la realidad, sino solamente una aproximación discreta. Este hecho debe ser evidente para el internauta promedio, el cual emplea comúnmente imágenes convertidas a bloques o fragmentos después de haber sido redimensionadas para ser ajustadas en una página en el navegador. El objetivo de la interpolación de imágenes es producir imágenes aceptables en diferentes resoluciones de una imagen única de baja-resolución. En la actualidad, la resolución de una imagen se define como el número de pixeles, no obstante, la resolución efectiva es una cantidad mucho más difícil de definir, dado que depende del juicio y percepción subjetivos de un ser humano. El objetivo de esta sección es explorar diferentes formulaciones matemáticas de esta magnitud esencialmente estética.

El problema de la interpolación de imágenes recibe muchos nombres, dependiendo de la aplicación: redimensionamiento de imágenes, sobre-muestreo/sub-muestreo de imágenes, zooming digital, magnificación de imágenes, mejoramiento de la resolución, etc. El término super-resolución es empleado algunas veces, aunque en la literatura este se refiere generalmente a la producción de una imagen de alta-resolución a partir de múltiples imágenes tales como una secuencia de video. Si definimos interpolación como "rellenar el espacio inter-pixel," el problema de la interpolación de imágenes puede ser encarado como un subconjunto del problema de inpainting, el cual consiste en un proceso de reconstrucción de las partes perdidas o deterioradas de una imagen o video (ver Fig.2.2).

Figura 2.2: Interpolación de imágenes usando el método de Lagrange, en su forma opción "bilineal". Arriba: imagen original. Medio: aproximación al ojo en la imagen. Abajo: imagen interpolada.

Las aplicaciones de la interpolación de imágenes van desde el lugar común de la visualización de las imágenes en línea hasta la más sofisticada magnificación de las imágenes satelitales. Con el auge de la fotografía digital de alto consume, los usuarios esperan tener un mayor control sobre

sus imágenes digitales. El zooming digital tiene un rol muy importante a la hora de recoger pistas y detalles en imágenes o videos de vigilancia. Con la entrada de la tecnología de televisión de alta-definición (en inglés, HDTV: High-Definition TV) en el mercado, los ingenieros están interesados en algoritmos rápidos de interpolación para disfrutar de programas tradicionales de TV de baja-definición sobre HDTV. Las imágenes astronómicas provenientes de los vehículos espaciales y las sondas son recibidas en una tasa de transmisión extremadamente baja (cerca de 40 bytes por segundo), haciendo la transmisión de datos de alta-resolución inviable [252]. En imágenes médicas, a los neurólogos les encantaría disponer de la habilidad de magnificar una parte específica en las imágenes de tomografía cerebral. Esto constituye solo una pequeña lista de aplicaciones, pero la gran variedad nos advierte que nuestro resultado deseado de interpolación debería variar dependiendo de la aplicación y el usuario.

A. El Problema de la Interpolación de una Imagen

En esta sección, establecemos la notación para interpolación de imágenes usada a lo largo del trabajo. Supongamos que nuestra imagen está definida sobre algún rectángulo $\Omega \subset \Re^2$. Sea la función $f : \Omega \to \Re$ nuestra imagen continua ideal. En un sentido abstracto, podemos pensar en f como nuestra "realidad" y Ω como nuestra "ventana de visión". Nuestra imagen observada u_0 es un muestreo discreto de f en puntos igualmente espaciados en el plano. Si suponemos que la resolución de u_0 es $\delta x \times \delta y$, podemos expresar u_0 como

$$u_0(x,y) = C_{\delta x,\delta y}(x,y) f(x,y), \qquad (x,y) \in \Omega \tag{2.1}$$

donde C denota el peine de Dirac (tren regular de deltas de Dirac):

$$C_{\delta x,\delta y}(x,y) = \sum_{k,l \in Z} \delta(k\delta x, l\delta y), \qquad (x,y) \in \Re^2. \tag{2.2}$$

El objetivo de la interpolación de una imagen es producir una imagen u en una resolución diferente $\delta x' \times \delta y'$. Por simplicidad, asumiremos que las coordenadas Euclideas se encuentran escaladas por el mismo factor arbitrario K:

$$u(x,y) = C_{\frac{\delta x}{K},\frac{\delta y}{K}}(x,y) f(x,y), \qquad (x,y) \in \Omega. \tag{2.3}$$

Dado solo la imagen u_0, tendremos que idear alguna reconstrucción de f en los valores especificados de los pixeles para esta nueva resolución. Nos referiremos a K como nuestro zoom o factor de magnificación. Obviamente, si $K = 1$ recuperaremos trivialmente u_0. La imagen u_0 es sobre-muestreada si $K \uparrow \kappa$ y sub-muestreada si $K \downarrow \kappa$, con $\kappa > 1$. En este trabajo, nos enfocaremos en el caso de sobre-muestreo cuando $K \uparrow \kappa$ es un entero. Por otra parte, $\Omega_K \subset \Omega$ denota el entramado inducido por (2.3) para un zoom fijo K. Notemos que el entramado de la imagen original u_0 en (2.2) es Ω_1. También notemos que para el caso de magnificación infinita obtenemos $\Omega_K \subset \Omega$ para $K \to \infty$. Por propósitos computacionales, podemos desplazar el entramado para los enteros positivos. De esta manera, si la imagen observada u_0 es una imagen de m x n,

$$\Omega_K = [1,2,\ldots,Km] \times [1,2,\ldots,Kn], \qquad K \in Z_+. \tag{2.4}$$

Muchas técnicas de interpolación imponen la condición $\Omega_1 \subseteq \Omega_K$. En este caso, solo un subconjunto de los pixeles en Ω_K necesitan estar determinados y el problema de interpolación se convierte en una versión del problema de inpainting.

Mediante la notación empleada hasta ahora, podemos establecer el problema de interpolación de una imagen sucintamente: Dada una imagen de baja-resolución $u_0 : \Omega_1 \to \Re$ y un zoom $K \uparrow 1$, encontrar una imagen de alta-resolución $u : \Omega_K \to \Re$. Obviamente, este es un problema mal planteado (problema inverso). Necesitamos imponer suposiciones sobre la reconstrucción de *f* en la ecuación (2.3). La elección de la técnica de interpolación depende de la elección de las suposiciones. En otras palabras, necesitamos un entendimiento matemático de lo que constituye nuestra percepción de la "realidad" *f*.

Los métodos de interpolación difieren en su descripción matemática de una "buena" imagen interpolada. Aunque es difícil comparar métodos y juzgar sus salidas, [252] proponemos aquí 9 criterios básicos para un buen método de interpolación. Los primeros 8 son propiedades visuales de la imagen interpolada, el último es una propiedad computacional del método de interpolación.

1) Invarianza Geométrica: Los métodos de interpolación deberían preservar la geometría y las dimensiones relativas de los objetos en una imagen. Es decir, el subconjunto original no debería cambiar bajo interpolación.

2) Invarianza de Contraste: El método debería preservar los valores de luminancia de los objetos en una imagen y el contraste general de la imagen.

3) Ruido: El método no debería adicionar ruido u otros artefactos a la imagen, tales como replicación de los bordes, en las inmediaciones de estos.

4) Preservación de los Bordes: El método debería preservar los bordes y transiciones, conformándolos donde sea posible.

5) Aliasing: El método no debería producir bordes entrecortados o tipo escalones de escalera.

6) Preservación de Texturas: El método no debería generar blur o suavizado de las regiones texturadas.

7) Sobre-suavizado: El método no debería producir regiones constantes indeseables a trozos o a bloques

8) Conciencia de Aplicación: El método debería producir resultados apropiados de acuerdo al tipo de imagen y resolución final. Por ejemplo, los resultados interpolados deberían parecer realistas para imágenes fotográficas, pero para imágenes médicas los resultados deberían tener bordes nítidos y de alto contraste. Si la interpolación es para imágenes en general, el método debería ser independiente del tipo de imagen.

9) Sensibilidad a los Parámetros: El método no debería ser sensible también a parámetros internos los cuales pueden variar de imagen a imagen.

Por supuesto, estos son criterios cualitativos y algo subjetivos. Nosotros en cambio [252], no esperamos desarrollar un modelo matemático de la interpolación de la imagen y el análisis de su error, sino simplemente aplicar el método más eficiente para nuestro desarrollo. En cierto sentido, el método empleado en este trabajo presenta un modelo matemático de estos criterios visuales.

B. Filtros de Interpolación Lineal

La aproximación más simple consiste en asumir que *f* en (2.3) es reconstruída por un kernel de convolución $\varphi : \Re^2 \to \Re$ donde $\int \varphi(x,y)\,dy\,dx = 1$. Por lo tanto, podemos aproximar *f* a

$$f \approx u_0 * \varphi . \tag{2.5}$$

Sustituyendo este en (2.3) da una interpolación lineal general del filtro

$$u(x,y) = C_{\frac{\delta x}{K},\frac{\delta y}{K}}(x,y).(u_0 * \varphi)(x,y), \qquad (x,y) \in \Omega. \tag{2.6}$$

Los filtros lineales más simples lo constituyen la interpolación bilineal y la bicúbica, los cuales asumen que los valores de los pixeles se pueden ajustar localmente a funciones lineales y cúbicas, respectivamente [252]. Junto con la interpolación más simple que es la del vecino más cercano, estos dos filtros son los más comunes esquemas de interpolación en software comercial. Estos métodos son fáciles de codificar como multiplicaciones de matrices de u_0. No obstante, una imagen contiene bordes y texturas, en otras palabras, discontinuidades. De manera tal que las suposiciones acerca de que los valores del pixel se ajustan localmente a una función polinómica producirá resultados indeseables. Los métodos de interpolación bilineal y bicúbica [252] pueden introducen blurring, crear defectos de reconstrucción, además de producir efecto aliasing tipo dientes de serrucho o escalones de escalera en los bordes oblicuos de la imagen (ver Fig.2.3). Los efectos blurring surgen del hecho dichos métodos calculan un promedio ponderado en base a las proximidades de los pixeles, al igual que en el blurring Gaussiano. Los efectos aliasing surgen porque los filtros lineales no toman en consideración la presencia de bordes o como reconstruir a estos.

Otros filtros de interpolación lineal incluyen zoom cuadrático, el método B-spline, y el zero-padding (completar con ceros, por ejemplo reemplazar con ceros las sub-bandas de alta frecuencia o detalles que resultan de la aplicación de la transformada de onditas [113-130]). Pero estos esquemas producen los mismos efectos indeseables de los métodos bilineal y bicúbico, como se menciona en [252]. Los filtros lineales difieren en la elección de φ, los cuales esencialmente muestran que determinan el cálculo en base al promedio ponderado de los pixeles cercanos. Mientras este es un esquema de interpolación natural para conjuntos de datos generales, esto no es necesariamente apropiado para datos *visuales*. Con el fin de mejorar estos filtros lineales, necesitamos considerar métodos de interpolación que de alguna manera cuantifique y preserve la información visual.

A este respecto, disponemos de cinco categorías de métodos, las cuales mencionaremos en la próxima sub-sección, los cuales nos ayudarán a disponer de uno de ellos para nuestro trabajo o eventualmente descartarlos todos y proponer una solución radical que tenga en cuenta los requerímientos de almacenamiento, transmisión y calidad de imagen de los sistemas actuales de computación y comunicaciones.

Figura 2.3: Parte de la imagen de Lena sub-muestreada y luego sobre-muestreada por un factor K = 16.
Arriba: Original, Segunda: Vecino más próximo, Tercera: Bilineal, y Abajo: Bicúbica.

C. Cuáles Métodos Considerar?

En términos generales, las aproximaciones matemáticas al procesamiento de imágenes pueden ser divididas en cinco categorías:

1. Métodos basados en Ecuación Diferencial Parcial (EDP) (e.g. difusión del calor, Perona-Malik, Navier-Stokes, curvatura media).

2. Variaciones de Energía (e.g. Variación Total, Mumford-Shah, contornos activos)

3. Análisis Multiescala (e.g. onditas, análisis de Fourier, análisis de Gabor, pirámides Laplaciana)

4. Máquina que Aprende (e.g. aprendizaje no-supervizado, mineo de datos, redes de Markov)

5. Métodos Estadísticos/Probabilísticos (e.g. inferencia Bayesiana, Estadísticas de Escena Natural, teoría de patrones)

Intentamos describir el campo en términos generales, pero no para clasificar o encasillar el trabajo en visión por computadora. De hecho, muchas técnicas como la de reconstrucción de datos incompletos o faltantes mediante onditas para el caso de imágenes de Variación Total, no se encasillan precisamente en una sola categoría. Además, estos métodos difieren en el nivel matemático, no necesariamente en el nivel conceptual. Por ejemplo, algunas versiones de la energía de Variación Total pueden ser minimizadas al resolver EDP u optimizando una variación de energía.

En nuestro intento anterior por describir las categorías de interpolación de una imagen, así como también mostrar la variedad de las matemáticas usadas, vamos a destacar uno de los métodos de de cada una de las cinco categorías [252], donde cada una será una aproximación:

1. basada en EDP: difusión anisotrópica del calor
2. por variación de la energía: reconstrucción de datos faltantes o incompletos por Mumford-Shah
3. multiescala: interpolación basada en onditas
4. máquina que aprende: LLE-based neighbor embeddings (vecinos basados en un embebido localmente lineal)
5. estadística: interpolación NL-means (medios no locales)

Estos métodos son en algunos casos, representativos de aproximaciones matemáticas al problema de interpolación de la imagen y, en un sentido más amplio, al campo de procesamiento de imágenes. Por ejemplo, la ecuación del calor constituye la EDP más estudiada en procesamiento de imágenes y la energía de Mumford-Shah ha generado docenas, sino cientos, de trabajos científicos. No obstante, estos métodos tienen un número de desventajas para su implementación, razón por la cual, elegimos una configuración basada en interpolación lineal. Las principales desventajas son:

1. Son difíciles de codificar
2. Dependen fuertemente de condiciones iniciales
3. Elevada complejidad computacional
4. La calidad visual no es superior a la interpolación lineal, excepto para altos niveles de sub-muestreo/sobre-muestreo, lo que automáticamente significa una alta tasa de compresión/descompresión.

En el último caso, usamos una máscara convolutiva [119, 124, 127, 253, 254] para mejorar los bordes, como se discutirá en el próximo capítulo.

D. Interpolación Bilineal

La Interpolación Bilineal es por lejos, el más común método de interpolación [247-251, 254]. La idea consiste en interpolar a lo largo de una dimension usando valores que a su vez fueron interpolados a lo largo de otra dimensión, ver Fig.2.4.

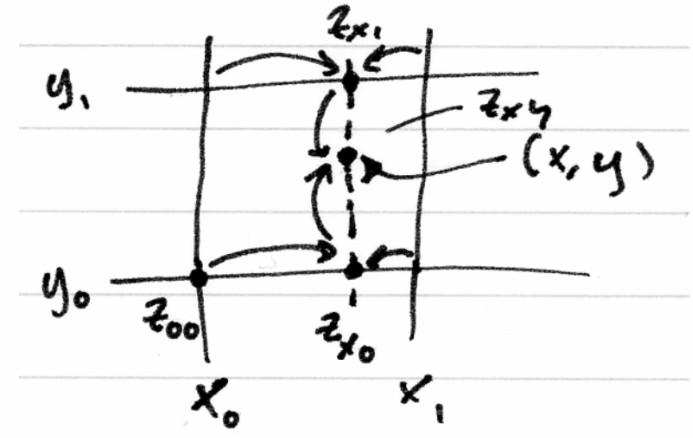


Figura 2.4: Interpolación Bilineal.

En dicha figura vemos que se comienza z_{x1} sobre la línea y_1, y z_{x0} sobre la línea y_0, para luego interpolar z_{xy} a partir de z_{x1} y z_{x0}.

$$z_{x0} = (1 - \alpha)\, z_{00} + \alpha\, z_{10} \qquad \alpha = (x - x_0)/(x_1 - x_0)$$
$$z_{x1} = (1 - \alpha)\, z_{01} + \alpha\, z_{11} \tag{2.7}$$
$$z_{xy} = (1 - \beta)\, z_{x0} + \beta\, z_{x1} \qquad \beta = (y - y_0)/(y_1 - y_0)$$

Tengamos en cuenta que no importa si interpolamos horizontalmente y luego verticalmente, o bien, primero verticalmente y luego horizontalmente (i.e. primero sobre x o sobre y). De todas maneras nos encontramos con

$$z_{xy} = (1 - \alpha)\,(1 - \beta)\, z_{00} + (1 - \alpha)\,\beta\, z_{01} + \alpha\,(1 - \beta)\, z_{10} + \alpha\,\beta\, z_{11} \tag{2.8}$$

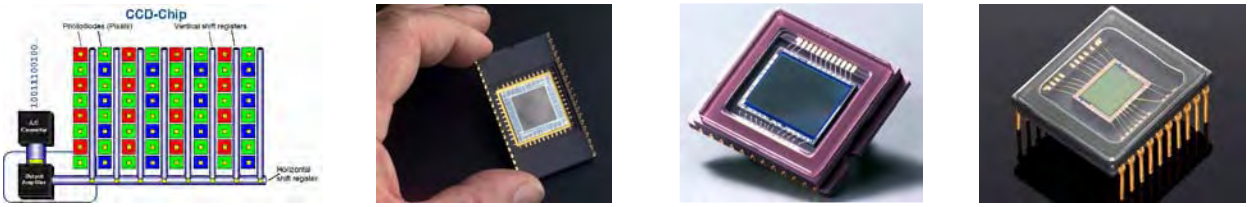
Esto es Interpolación Bilineal. El resultado es una función a trozos que no es lineal a trozos —por supuesto, esto no puede ser, porque coincide con los datos en cuatro puntos diferentes, y solo tres puntos determinan la función lineal. Posee una porción para cada celda en la grilla de puntos datos, pero la interpolación definida sobre ese rectángulo no es lineal. Por otra parte, si observamos la Ecuación (2.8), debemos tener presente que α es una función lineal de x y β es una function lineal de y. La expresión completa para z_{xy} contendrá un término constante, un término x, un término y, y un término xy. Precisamente, por la presencia de este último término, la Ecuación (2.8) no es lineal.

Esta clase de funciones recibe el nombre de bilineal porque es lineal como una función de x cuando y permanece fijo y a su vez lineal como una función de y cuando x permanece invariante. La calidad es muy buena como se observa en la Fig.2.2.

2.3 Super-resolución de imágenes y video

2.3.1 El problema de la superresolución

En numerosos problemas reales son necesarias imágenes de alta calidad espacial (resolución). Por ejemplo: imágenes médicas, imágenes de teledetección, policía forense y en general problemas de visión artificial donde una mayor resolución de las imágenes puede proporcionar un reconocimiento más preciso. No obstante, existen dos limitaciones físicas a este aumento de la resolución. La primera se debe a un límite inferior en el tamaño que un píxel puede tener, y la segunda consiste en que tampoco se puede hacer crecer el tamaño de la CCD (Charge-Coupled Device, dispositivo de carga acoplada, es decir, sensor sensible a la luz que capta la imagen para su posterior almacenamiento) de la cámara (ver Fig.2.5), por un límite superior que haga imposible la lectura de datos en tiempo real, es, por tanto, necesaria una aproximación algorítmica al problema [255].



a b c d

Figura 2.5: Detalle y ejemplos de dispositivos de carga acoplada de una cámara (CCD), a) detalle de acceso a los elementos del chip CCD, b) noción de escala en relación a mano de humano adulto, c) uso espacial, y d) uso multimedial.

2.3.2 Ejemplos más comunes de super-resolución

Los ejemplos más comunes de superresolución son:

a) *Super-resolución a partir de imágenes estáticas:* En este caso disponemos de varias imágenes de baja resolución con pequeños desplazamientos entre ellas, ver Fig.2.6.

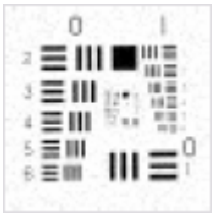


Figura 2.6: Ejemplo de imagen estática de baja resolución.

A los efectos de aumentar la resolución tenemos dos caminos posibles: a) aumentar el tamaño de la imagen, lo cual consiste en repetir cada pixel en un cuadrado (de 2x2, 3x3, etc) pixeles iguales, o b) reconstruirla en una resolución mayor a partir de la combinación de varias imágenes como la de la Fig.2.6. La Fig.2.7 muestra en la imagen de la izquierda que hacer crecer el tamaño de la imagen mediante el procedimiento descripto no es una buena idea en lo visual. Mientras que en la imagen de la derecha, la reconstrucción combinando varias imágenes del tipo de la Fig.2.6, nos proporciona una forma eficaz de alcanzar la resolución deseada, dado que preserva los detalles al mismo tiempo que proporciona una aceptable nitidez de la imagen alcanzada.

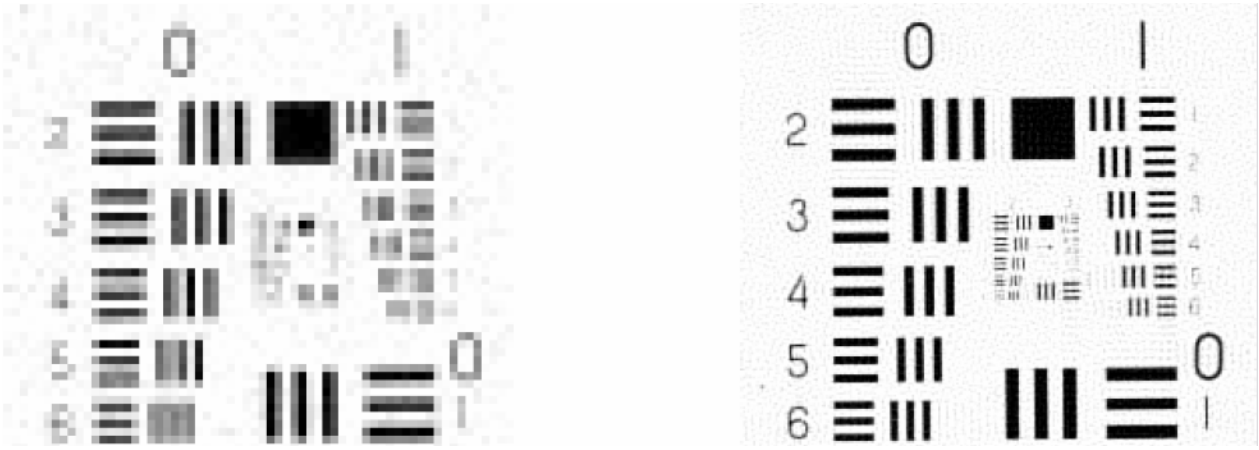


Figura 2.7: Dos formas posibles de intentar un aumento de resolución.

b) *Super-resolución a partir de video (con compresión):* En este caso disponemos de varios cuadros (frames) de baja resolución de un video, el cual ha sido comprimido con algún algoritmo de compresión [256, 257]. En el Apéndice E de desarrollan en detalle los conceptos de resolución baja y alta. Por otra parte, deben emplearse cuadros de una misma escena con pequeñas innovaciones (desplazamientos de los objetos de la escena) de cuadro en cuadro, como el caso de la Fig. 2.8.

Figura 2.8: Dos cuadros (frames) de una misma escena de un video de baja resolución.

El resultado obtenido, es un cuadro solo mucho más nítido, ver Fig.2.9.

Figura 2.9: Cuadro (frame) obtenido a partir del proceso de super-resolución, consistente en generar un cuadro más nítido a partir de la combinación de los cuadros disponibles de la escena

2.3.3 Jerarquía de super-resolución

La Fig.2.10 nos muestra el nivel jerárquico de la super-resolución en forma ascendente, es decir, del caso más sencillo el cual lo constituye la mejora de la resolución espacial de una imagen estática a partir de una imagen de baja resolución (y que es objeto del Capítulo 3), hasta el caso más complejo, el cual esta dado por la mejora espacio-temporal de la resolución de video a partir de cuadros (frames) de un video de baja resolución. En el medio podemos observar algunas interesantes y prácticas posibilidades con notables aplicaciones al caso de imágenes médicas, satelitales, biométricas y documentológicas. Estas técnicas, por lo general, vienen acompañadas de procesos de filtrado, es decir, supresión de ruido [256, 257].

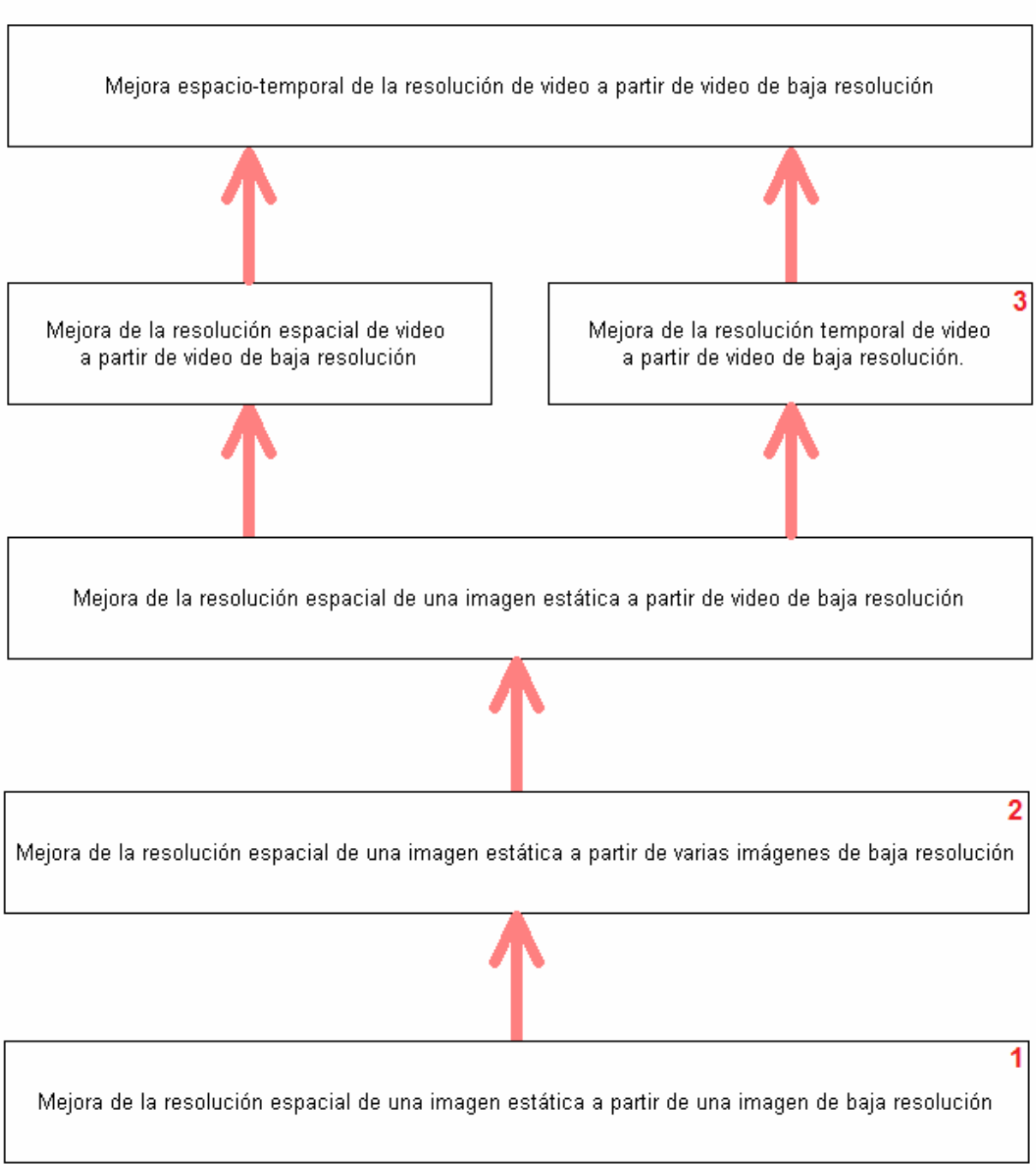


Fig.2.10: Jerarquía de super-resolución.

El módulo rotulado como “1” en la Fig.2.10 hace referencia a la forma más primitiva de super-resolución, consistente en mejorar la resolución de una imagen a partir de una imagen única, mediante alguna técnica de interpolación, o bien, pares de entrenamiento baja-res/alta/res, con estos criterios arribamos a las Fig.2.11 y 2.12 respectivamente.

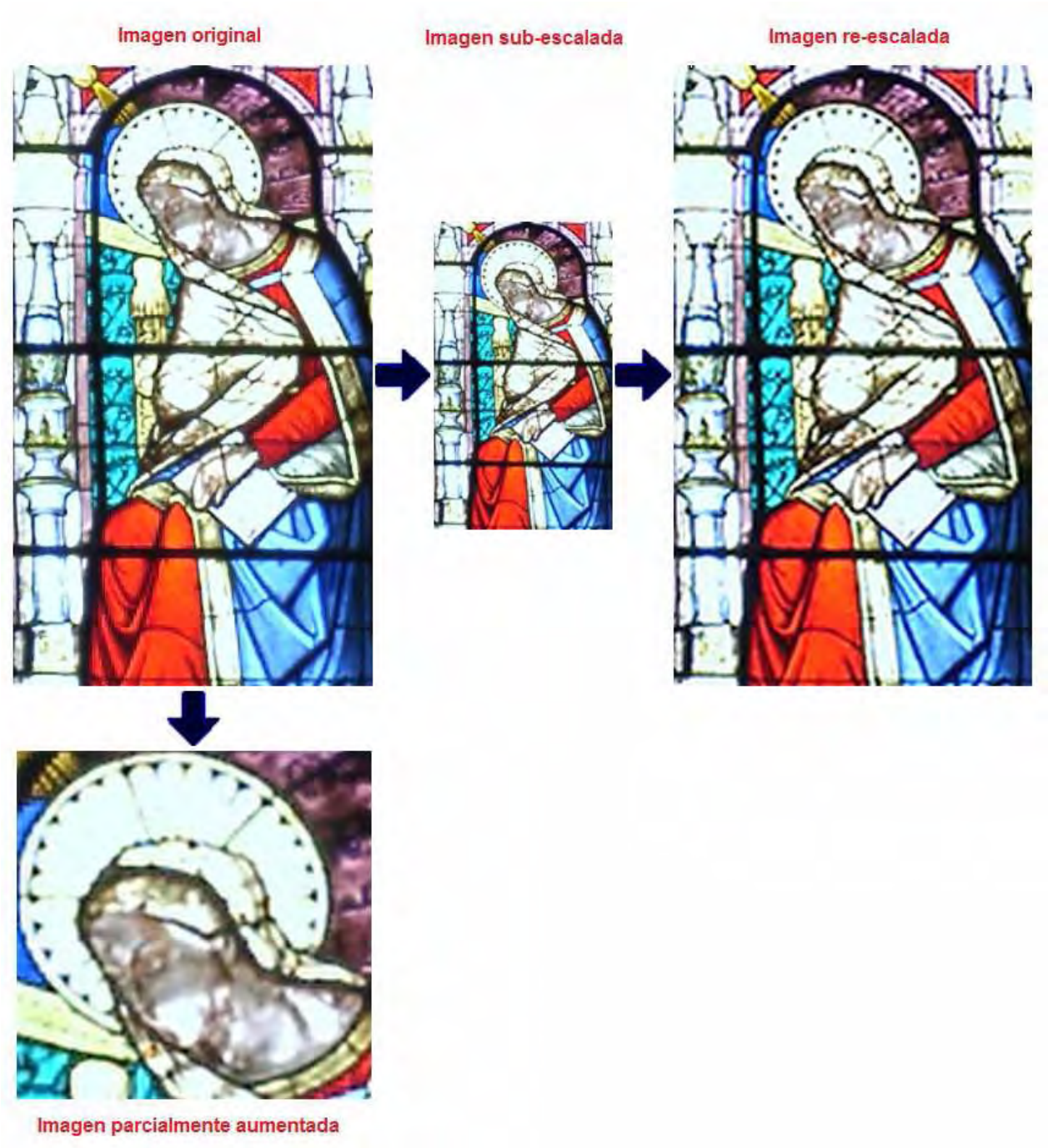


Fig.2.11: Aumento de la resolución por interpolación.

En cambio, el módulo rotulado como “2” en la Fig.2.10 hace referencia a la super-resolución, consistente en mejorar la resolución de una imagen a partir del uso de una secuencia corta (Fig.2.13) o larga (Fig.2.14) de imágenes de baja resolución.

Fig.2.12: Aumento de la resolución por pares baja-res/alta-res.

Fig.2.13: Secuencia corta de imágenes de baja resolución.

Se torna obvio, que la sola inspección visual de ambas imágenes (2.13 y 2.14) nos indica que para casos de identificación o reconocimiento de patrones con fines forenses (por ejemplo, discernir los caracteres alfanuméricos de la matrícula de un vehículo) se hace necesaria una secuencia larga de imágenes de baja resolución. No obstante, para casos de control de acceso, donde la identificación morfológica es uno de los aspectos a tener en cuenta a la hora de franquear e acceso a un recinto de alta seguridad, puede ser de utilidad una secuencia corta, con la diferencia sustancial en el costo del equipamiento relativo que esto conlleva. No obstante, donde sea posible, la secuencia larga es estadística y visualmente ideal.

Fig.2.14: Secuencia larga de imágenes de baja resolución.

En ambos casos, es decir, las secuencias corta y larga, no son artificialmente creadas, sino que provienen de un disparo múltiple de la cámara empleada. Finalmente, el módulo rotulado como "3" en la Fig.2.10 hace referencia a cuadros de la misma imagen generados artificialmente e insertados en la secuencia correspondiente, mitigando así la falta de nitidez que suele darse en estos casos, con la consecuente sensación de movimiento entrecortado, particularmente indeseable en el caso de la TV Digital, ver Apéndice F.

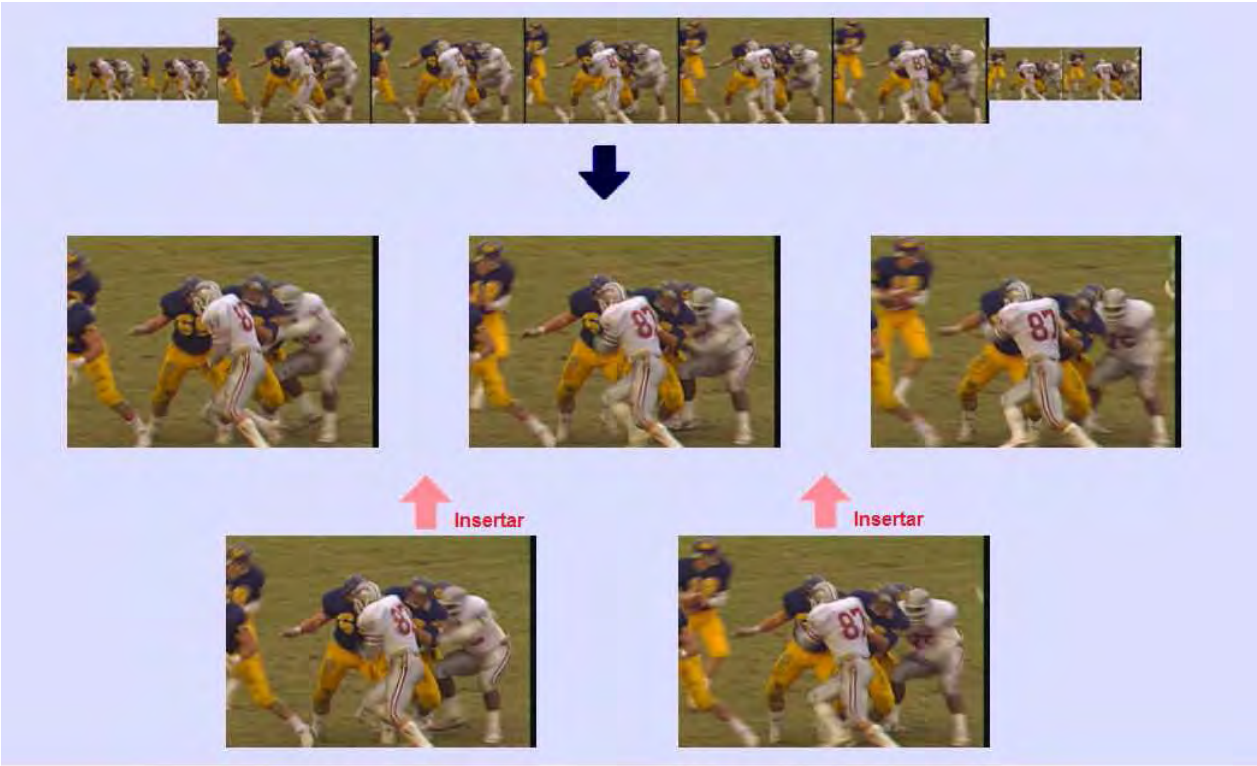


Fig.2.15: Interpolación temporal.

2.3.4 Por qué es posible la super-resolución

La Fig.2.16 nos muestra la hipótesis básica para aumentar la resolución espacial de una imagen, y que consiste en la existencia de varias imágenes de baja resolución de la misma escena con pequeños desplazamientos entre las mismas, las cuales son superpuestas sobre una grilla de manera tal que sus valores se entrelacen, dando lugar a una dramática reducción al tamaño de las celdas de dicha grilla, y por ende del pixel asociado, aumentando automáticamente la resolución. La Fig.2.16 muestra una distribución uniforme de los símbolos, constituyendo este un caso ideal, no siempre presente en la práctica [256, 257], dado que el movimiento puede no ser único y global.

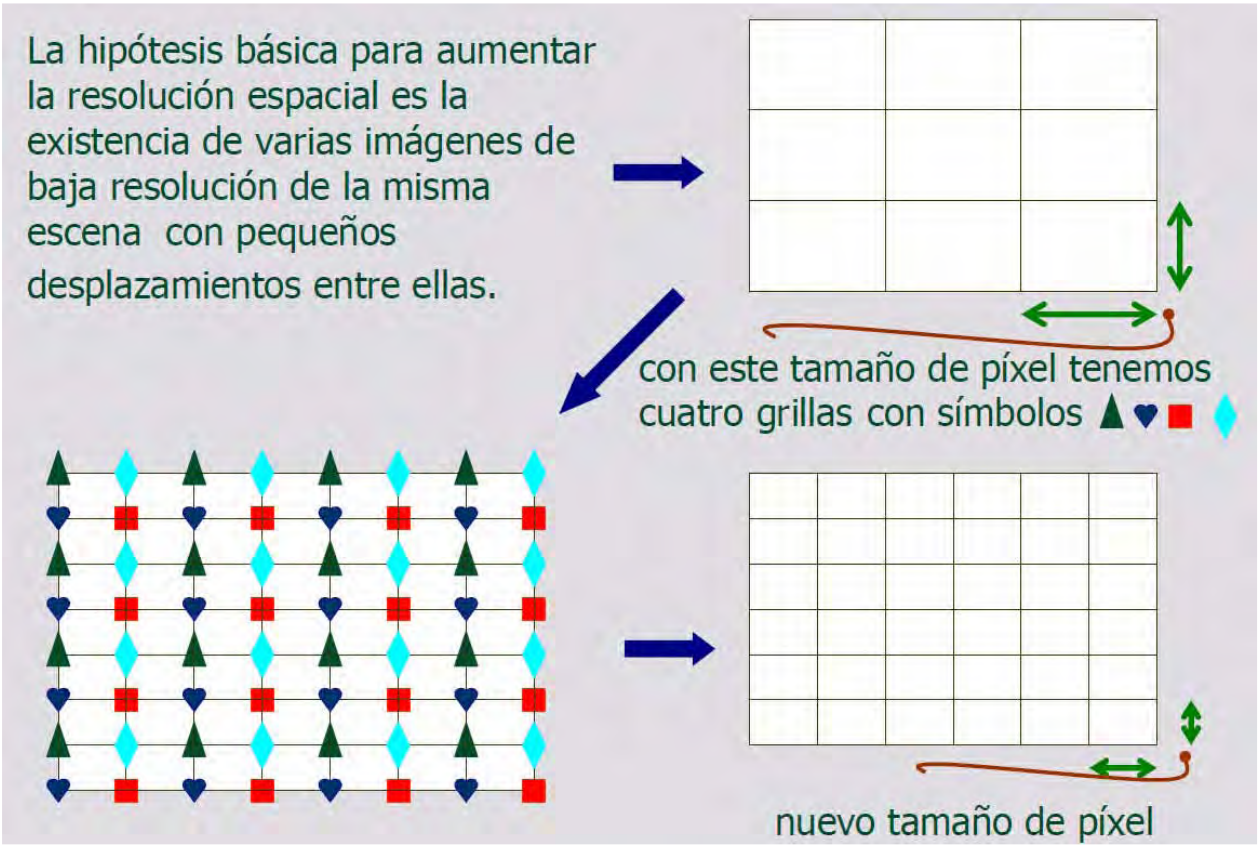


Figura 2.16: Hipótesis básica de sustento a la super-resolución.

La Fig.2.17 nos proporciona una noción de un caso real, es decir, altamente probable en la práctica y mucho más complejo, por cuanto se complica dada la existencia de varios objetos moviéndose simultáneamente.

A continuación, pasaremos a describir el proceso de obtención de las imágenes de baja resolución a partir de las imágenes de alta.

2.3.5. Modelización del proceso de obtención de imágenes de baja resolución a partir de imágenes de alta resolución

a) *Notación:* Consideremos la imagen de alta resolución que buscamos. Dicha imagen tiene un tamaño L_1N_1 x L_2N_2 y la vamos a notar como un vector columna de la forma $\mathbf{x} = [x_1, \ldots, x_N]^T$ siendo $N = L_1N_1$ x L_2N_2 .

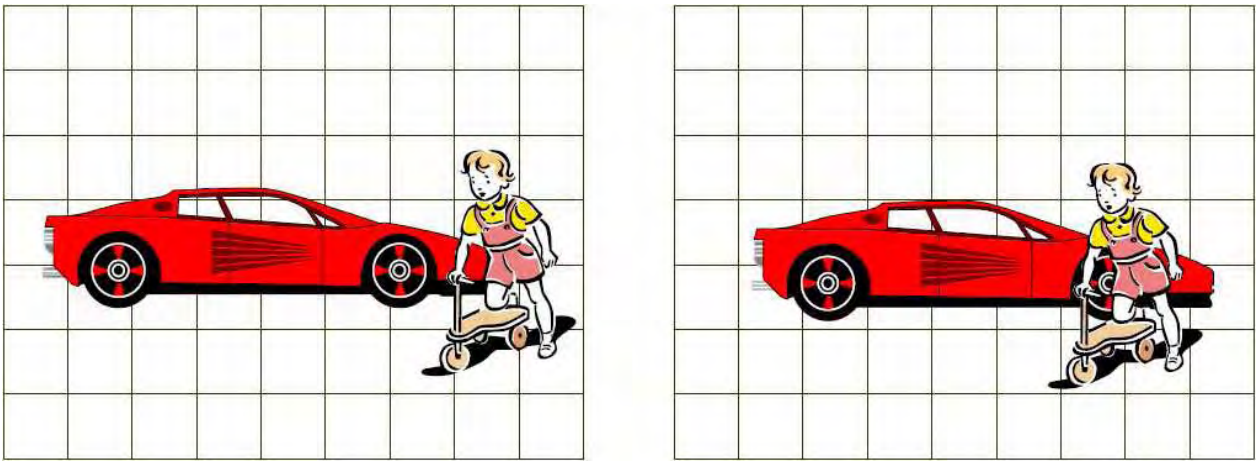


Figura 2.17: Detalles de varios objetos en movimiento entre ambos cuadros.

L_1 y L_2 representan el factor de submuestreo en el modelo de observación en las direcciones vertical y horizontal, respectivamente. Cada imagen de baja resolución tiene, por tanto, tamaño $N_1 \times N_2$.

El objetivo consiste en estimar la imagen $\boldsymbol{x}$ que tendría que haber sido observada en la CCD de alta resolución, a partir de K imágenes $\mathbf{y_k}$, $k = 1,...,K$ observadas en CCDs de baja resolución, donde $\mathbf{y_k} = [\, y_{k,1} ,..., y_{k,M} \,]$ con $M = N_1 \times N_2$, ver Fig.13.

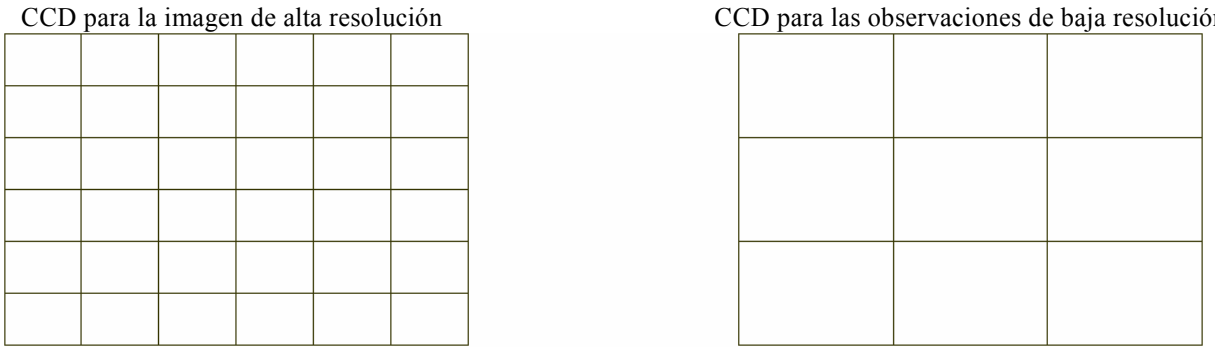


Fig.2.18: $N_1 = 3$, $L_1 = 2$, $N_2 = 3$, $L_2 = 2$.

b) *Modelo de observación:* En este punto cabe una pregunta: Cómo obtenemos las imágenes de baja resolución a partir de las de alta? Para contestar esta pregunta debemos remitirnos al modelo de Elad y Feuer [258]. Este modelo puede ser representado por la siguiente ecuación:

$$y_k = \mathbf{D}\, \mathbf{B}_k\, \mathbf{M}_k\, x_k + n_k \qquad (2.9)$$

donde:
y_k : Imagen de baja resolución
x_k : Imagen de alta resolución
$\mathbf{M}_k$: Matriz de warping (deformación) de orden ($L_1 N_1 L_2 N_2$) x ($L_1 N_1 L_2 N_2$)
$\mathbf{B}_k$: Matriz de emborronamiento de orden ($L_1 N_1 L_2 N_2$) x ($L_1 N_1 L_2 N_2$)
$\mathbf{D}$: Matriz de submuestreo de orden ($N_1 N_2$) x ($L_1 N_1 L_2 N_2$)
n_k : Ruido aleatorio (normal)

b.1) *Descripción del warping,* $\mathbf{M}_k$ *:*

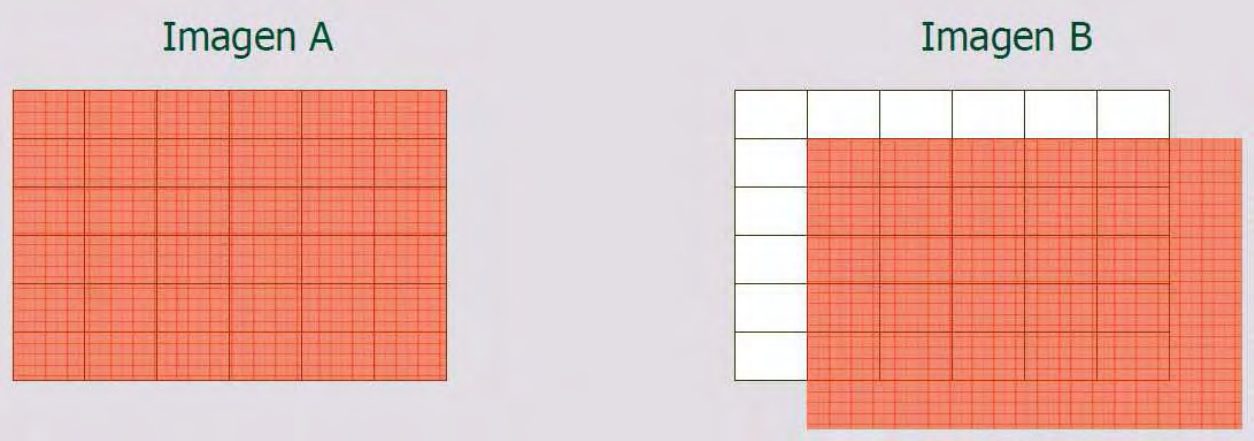


Figura 2.19: Descripción del warping, $\mathbf{M}_k$.

De la Fig.2.19, observemos que en relación a un punto (pixel) genérico de coordenadas (x, y) la relación entre la imagen **A** y **B** es la siguiente:

$$\mathbf{A}(x, y) = \mathbf{B}(x+1, y+1) \tag{2.10}$$

Ahora bien, si definimos

$$\mathbf{W}(\mathbf{A}, (1,1))(x, y) = \mathbf{A}(x-1, y-1) \tag{2.11}$$

y teniendo en cuenta (2.10), surge

$$\mathbf{W}(\mathbf{A}, (1,1))(x, y) = \mathbf{B}(x, y) \tag{2.12}$$

Esto daría la falsa idea que el warping es un proceso simple, no obstante, y como podemos apreciar en la Fig.2.20, desgraciadamente el problema puede ser muy complejo. Cuando hablamos de complejidad, hacemos referencia al empleo de estas imágenes en el contexto de un compresor de video, es decir, dentro de una misma escena, cuanto más se pronuncie el efecto de warping deformando simultáneamente a las múltiples imágenes que forman parte de los cuadros, más se complica el trabajo del módulo conocido como compensador de movimiento, el cual debe seguir vectorialmente a cada objeto solapado visualmente, ver Apéndice F.

No obstante, son numerosas las técnicas de estimación de movimiento, o desplazamientos entre imágenes. Las mencionadas técnicas son fundamentales en los compresores de video [256, 257]. En general, la aproximación es la siguiente:

Dentro de la superresolución, se interpolan las imágenes de baja resolución (se hacen del tamaño de las de alta resolución) y se estiman los desplazamientos entre ellas. Obviamente, el método no es completamente correcto.

Una aproximación alternativa consiste en reestimar el movimiento a partir de las imágenes de alta resolución que se van obteniendo por un proceso iterativo. El mencionado procedimiento, puede tener problemas de convergencia.

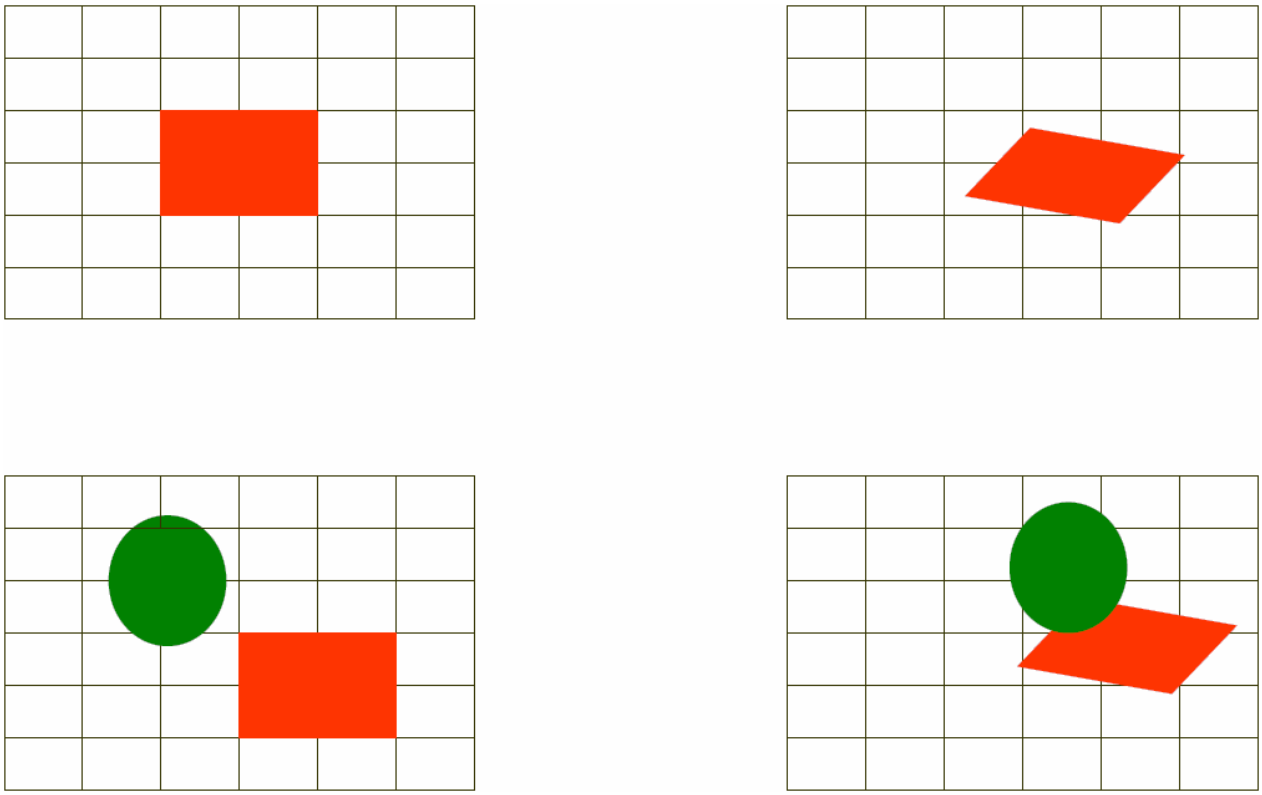


Fig.2.20: Ejemplo complejo de warping.

El trabajo de S. C. Park y colaboradores [255] contiene referencias interesantes sobre aproximaciones clásicas. El trabajo de Segall y colaboradores [259] contiene también una descripción de métodos de estimación de movimiento en super-resolución para imágenes comprimidas. Ver también el tutorial de S. Borman y R. Stevenson [260], y el libro editado por Chaudhuri [261]. El trabajo de Irani y Peleg [262] presenta una nueva metodología para la estimación del movimiento. Sin embargo, son necesarias nuevas aproximaciones entre las que se destacan [263-265].

b.2) *Descripción del emborramiento, decimación y ruido.*

En la Fig.2.21, podemos observar a partir de la imagen original del cameraman los tres efectos perturbadores, en orden y siguiendo la secuencia establecida por las flechas azules, la degradación de la nitidez (blurring), la decimación o sub-muestreo y el ruido, en principio, aditivo [256, 257]. Es decir, en la Fig.2.21 podemos observar 3 de los 4 efectos perturbadores comprendidos en la ecuación (2.9).

Como es comprensible, el intentar solucionar uno de estos efectos afecta la superación y tratamiento de todos los demás. Por ejemplo, es muy común que todo proceso de filtrado tendiente a suprimir o mitigar el ruido, tanto con filtros convolutivos de máscara como con wavelets o alguna otra técnica que lesione los bordes de la imagen haciendo que la misma pierda nitidez [256, 257]. Efecto que debe ser corregido con un filtro tambien de máscara convolutiva llamada sharper [256, 257] o conformador, que restaure los bordes y por ende la pérdida nitidez. No obstante, si no son correcta y exactamente empleados, estos filtros suelen aumentar el brillo de la imagen en demasía, generando un intolerable ruido de alta frecuencia. Lo propio, el proceso de restauración de la decimación empleado para que la imagen alcance su tamaño original suele también lesionar los bordes, generando una desagradable pérdida de nitidez (blurring), espe-

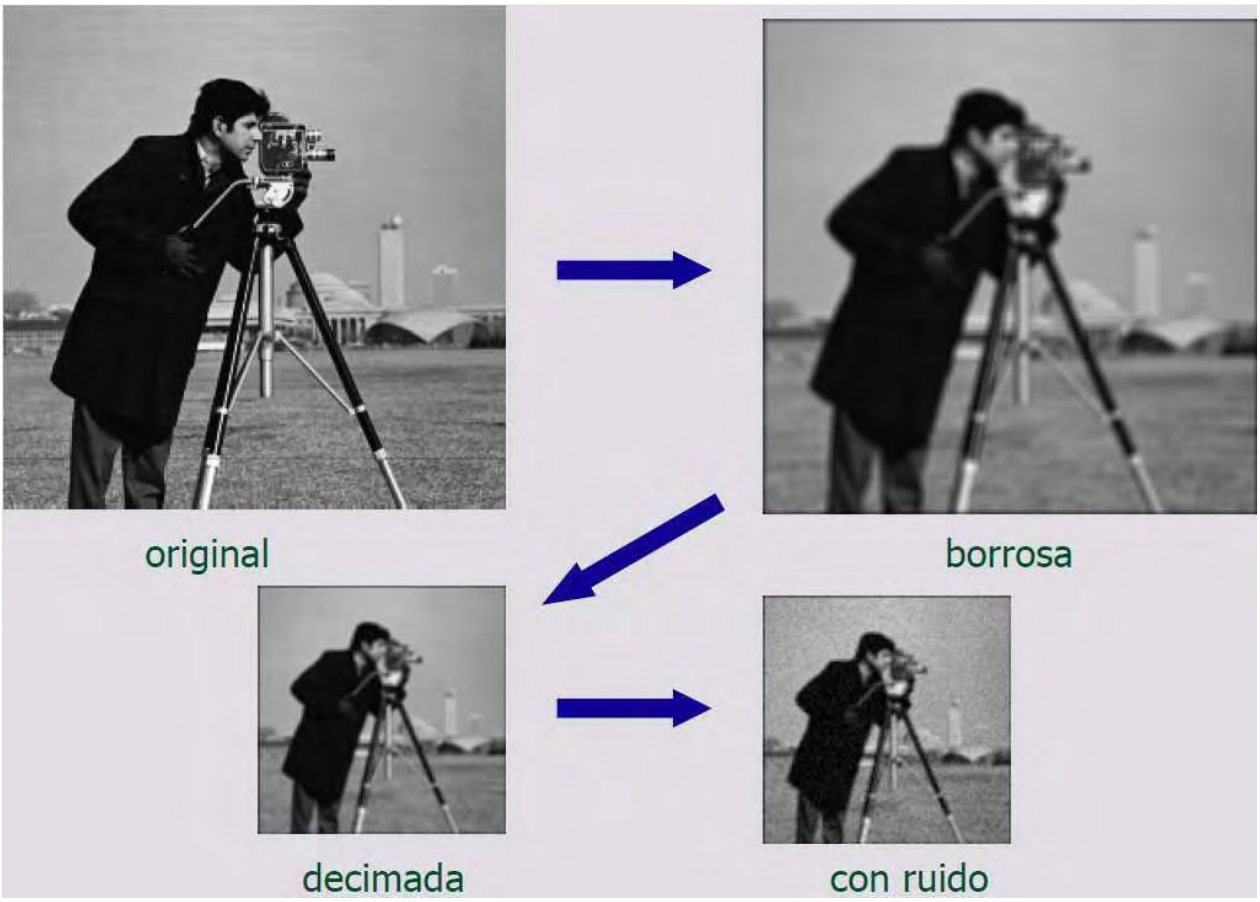


Figura 2.21: Emborramiento, decimación y ruido.

cialmente en aquellos casos donde la diferencia entre x_k e y_k de la ecuación (2.9) es relativamente notorio [245, 246].

2.3.6. Métodos en el dominio de las frecuencias

a) *Introducción:* Vamos a analizar ahora los métodos de procesamiento basados en el dominio frecuencial. La base de esta aproximación es el trabajo [266] de Tsai y Huang de 1984. El análisis de esta aproximación descanza completamente en la transformada discreta de Fourier y sus derivaciones, es decir, la transformada coseno discreta y por ende la versión rápida de ambas. Ver el Capítulo III del libro de Telkap [267].

b) *Métodos basados en la supresión del aliasing:* $\boldsymbol{x}(x_1, x_2)$ es la imagen original. Traslaciones globales de esta función producen **R** imágenes desplazadas

$$\boldsymbol{x}_r(x_1, x_2) = \boldsymbol{x}\,(x_1 + \Delta x_{1r}\,,\, x_2 + \Delta x_{2r}) \quad r = 1, \ldots, \mathbf{R} \tag{2.13}$$

Las imágenes desplazadas son muestreadas produciendo

$$\begin{aligned} \boldsymbol{y}_r(m_1, m_2) &= \boldsymbol{x}\,(m_1\,T_{x_1} + \Delta x_{1r}\,,\, m_2\,T_{x_2} + \Delta x_{2r}) \\ &= \boldsymbol{x}_r(m_1\,T_{x_1}\,,\, m_2\,T_{x_2}) \qquad 0 \le m_1 < M_1\,,\, 0 \le m_2 < M_2 \end{aligned} \tag{2.14}$$

Si la proporción de resolución entre las imágenes de alta y baja resolución es de apenas una octava, podemos relacionar fácilmente la transformada de Fourier de ambas imágenes y considerando la imagen de baja resolución a super-resolver estimamos la imagen de alta resolución a la que queremos llegar.

La pérdida de nitidez en en imágenes debido a múltiples factores es analizada en Tekalp, Ozkany Sezan [268] y el capítulo 17 del libro [267] de Tekalp (que contiene una exposición muy clara).

¿Cómo estimamos los desplazamientos entre imágenes?
En el trabajo de Tsaiy Huang [266] se describe un método que utiliza todas las imágenes simultáneamente, una alternativa simple pero interesante es la propuesta por Kaltenbacher y Hardieen [269] que utiliza una imagen llave (key) como referencia para las interpolaciones multi-grilla.

Kim, Bosey Valenzuela analizan en [270] bajo qué condiciones podemos, a partir de las observaciones (imagen de baja resolución disponible), recuperar la imagen de alta resolución. Proponen también un método recursivo que va incorporando al proceso de estimación las imágenes observadas secuencialmente. El método no tiene en cuenta el posible emborronamiento.

Kim y Su [271] extienden el trabajo de [270] a la presencia de pérdida de nitidez e introducen regularización en la obtención de la solución (un importante aporte). El método vuelve a ser recursivo.

Siguiendo la teoría desarrollada por Davila en [272, 273], Bose, Kim y Valenzuela [274] adaptan su método recursivo descrito anteriormente al problema en el que suponemos que los desplazamientos son estimados y tienen ruido.

Desde el punto de vista de Katssagelos et al [256, 257], la evolución de los métodos frecuenciales es un ejemplo de evolución de la investigación en este tema.

El uso de la transformada discreta coseno en lugar de la transformada rápida de Fourier se describe en Rhee y Kang [275].

Por último, dentro del campo del procesamiento en el dominio frecuencias, Ur y Gross [276] proponen un método para la super-resolución basado en el teorema de muestreo de Papoulis [277] y su versión multicanal descrita por Brown [278].

Es importante notar que el método de Ur y Gross es espacial pero está basado en el análisis de las frecuencias de la imagen.

2.3.7. Interpolación a partir de muestras no uniformes

A continuación se utilizan algoritmos clásicos de supresión de restauración de la nitidez basados en la obtención de imágenes con pequeños desplazamientos de los demás de la grilla, como muestra la Fig.2.22. La idea es simple: Las imágenes de baja resolución son registradas, el resultado es una imagen compuesta de muestras no uniformemente distribuidas sobre el retículo de la imagen de alta resolución, estos puntos muestreados son interpolados y vueltos a muestrear en el retículo de alta resolución. A continuación se aplican técnicas de restauración de imágenes. Es importante tener en cuenta en todos los procesos de interpolación el siguiente resultado de Clark, Palmer y Laurence [279] sobre reconstrucción de funciones a partir de muestras no uniformemente espaciadas.

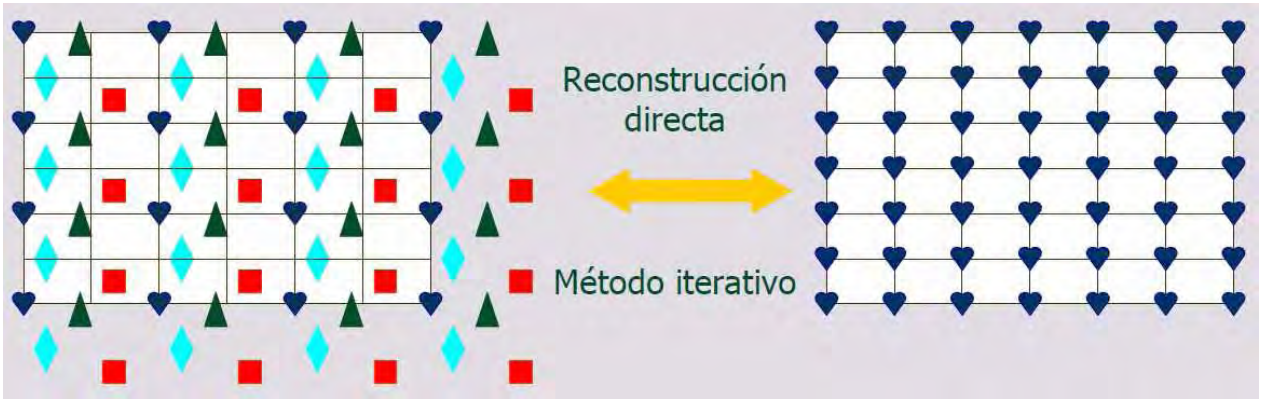


Figura 2.22: Interpolación a partir de muestras no uniformes.

Sea $f(x)$ una función muestreada en los instantes $\{x_m\}_{m=1,2,...}$ que no están uniformemente espaciados. Si existe una función $\gamma(x)$, tal que $mT = \gamma(x_m)$ y si $f(\gamma^{-1}(x))$ tiene una banda limitada a $\omega_0 = \pi/T$, siendo T el período, entonces $f(x)$ puede reconstruirse mediante

$$f(x) = \sum_{m=-\infty}^{m=\infty} f(x_m) \frac{sen(\omega_0(\gamma(x) - mT))}{\omega_0(\gamma(x) - mT)} \tag{2.15}$$

Aizawa, Komatsu, Saito e Igarashi extienden sus resultados sobre visión estéreo en [280] a problemas de super-resolución en [281] y [282].

Todos estos métodos, una vez que **R** imágenes de baja resolución son registradas utilizando compensación de movimiento (ver Apéndice F), se forma una imagen de alta resolución que no está muestreada uniformemente. Se utiliza entonces el algoritmo de Landweber [283] para estimar la imagen de alta resolución muestreada uniformemente. Veamos su descripción:

Si $\boldsymbol{x}$ es la imagen de alta resolución muestreada uniformemente y $\boldsymbol{z}$ es la imagen de alta resolución muestreada no uniformemente resultante del agrupamiento y registrado de las imágenes de baja resolución entonces:

$$z = A\,x \tag{2.16}$$

donde A representa el proceso de muestreo no uniforme. Como esta ecuación no se puede invertir se propone un método iterativo de la forma

$$x_{t+1} = x_t + \alpha\, A^*(z - A\,x) \tag{2.17}$$

donde A^* representa el operador adjunto $[(A^tA)^{-1}A^t]$ y α es un parámetro de control, en todas las ecuaciones hemos de incluir además el proceso de obtención de las imágenes de baja resolución a partir de las de alta.

Alam y colaboradores [284] proponen para imágenes infrarrojas una técnica de super-resolución basada en interpolación con el vecino más cercano, ver Fig.2.23.

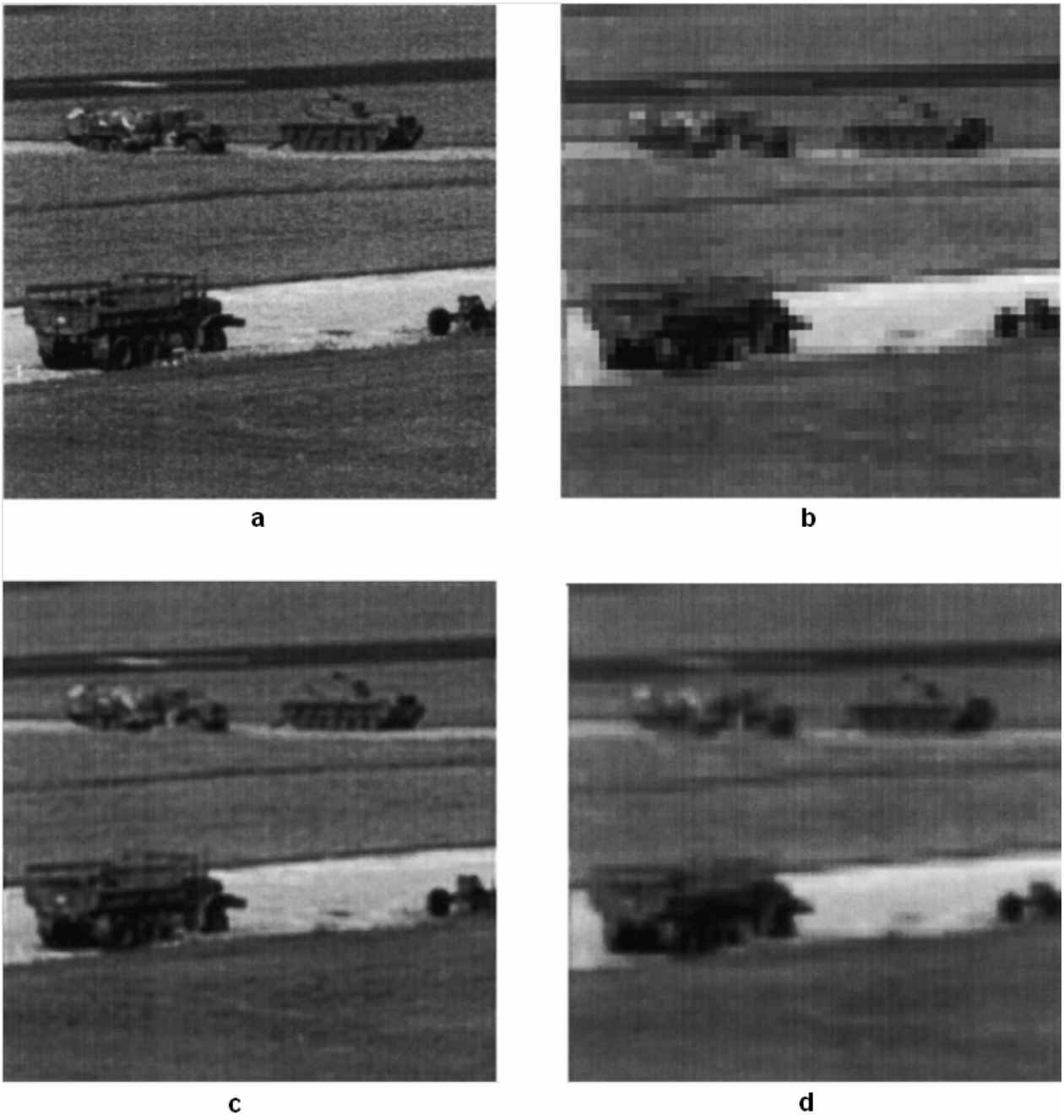


Figura 2.23: (a) Original, (b) Baja resolución, (c) Método de Alam y colaboradores, y (d) Interpolación bilineal.

La utilización de algoritmos de tipo Landweber es también propuesta en los trabajos de super resolución para imágenes en color de Shah y Zahor [285].

Este trabajo está también relacionado con el método de simulación y mejora que veremos con posterioridad.

2.3.8. Métodos en el dominio espacial sin regularización

a) Proyección algebraica hacia atrás: Frieden y Auman [286] consideran el problema de super-resolución 1-D a partir de varias observaciones de una escena estacionaria. El proceso de obser-

vación se define como

$$Y = H\,x \tag{2.18}$$

donde Y denota todos los datos observados, H es la matriz del sistema y x es la señal de alta resolución a ser estimada.

Observemos que esta forma de escribir el problema de super-resolución es equivalente a reescribir la fórmula

$$y_t = D\,B_t\,M_t\,x + n_t \tag{2.19}$$

Con todos los vectores y_t ordenados y llamar a este vector Y y a la correspondiente composición de los $D\,B_t\,M_t$ redefinirla como H.

Frieden y Auman proponen como estimador la proyección hacia atrás de las observaciones

$$\hat{x} = H^T\,Y \tag{2.20}$$

Observemos de las ecuaciones anteriores

$$\hat{x} = H^T H\,x \tag{2.21}$$

por lo tanto, la estimación y la imagen original coinciden *si y sólo si* $H^T H = I$.

Recordemos que el filtrado inverso produciría como estimador de x

$$\tilde{x} = \left(H^T H\right)^{-1} H^T Y \tag{2.22}$$

Donde a $\left(H^T H\right)^{-1} H^T$ es la pseudo-inversa de H.

b) Métodos de simulación y corrección: Los métodos que vamos a considerar ahora tienen en común la aproximación que podríamos llamar de *simulación y corrección* y la no introducción de información sobre la imagen de alta resolución.

Dado un estimador de la imagen de alta resolución y un modelo de formación de las imágenes de baja resolución, estos métodos simulan el proceso de formación de las imágenes de baja, estas imágenes generadas se comparan con las imágenes de baja resolución observadas y el error se usa para corregir la imagen de alta resolución.

El proceso simulación/corrección se itera hasta que se cumple una condición de parada.

Keren, Peleg y Brada [287] describen el siguiente proceso: Registrar las imágenes continuas $f_a(x_1,x_2)$ y $f_b(x_1,x_2)$ utilizando un modelo de rotación y traslación globales,

$$f_b\left(x_1, x_2\right) = f_a\left(x_1\cos\theta - x_2\sin\theta + \Delta x_1, x_2\cos\theta + x_1\sin\theta + \Delta x_2\right) \tag{2.23}$$

Desarrollando seno y coseno hasta el orden 2, tenemos:

$$f_b(x_1,x_2) \approx f_a\left(x_1 + \Delta x_1 - x_2\,\theta + x_1\frac{\theta^2}{2}, x_2 + \Delta x_2 + x_1\,\theta + x_2\frac{\theta^2}{2} \right) \tag{2.24}$$

Desarrollando f_a hasta orden 1, tendremos:

$$f_b(x_1,x_2) \approx f_a(x_1,x_2) + \left(\Delta x_1 - x_2\,\theta - x_1\frac{\theta^2}{2} \right)\frac{\partial f_a}{\partial x_1} + \left(\Delta x_2 + x_1\,\theta - x_2\frac{\theta^2}{2} \right)\frac{\partial f_a}{\partial x_2} \tag{2.25}$$

Se procede ahora a encontrar la solución de

$$\min_{\Delta x_1,\Delta x_2,\theta} E[\Delta x_1,\Delta x_2,\theta] := \sum \left(f_b(x_1,x_2) - f_a(x_1,x_2) - \left(\Delta x_1 - x_2\theta - x_1\frac{\theta^2}{2} \right)\frac{\partial f_a}{\partial x_1} - \left(\Delta x_2 + x_1\theta - x_2\frac{\theta^2}{2} \right)\frac{\partial f_a}{\partial x_2} \right)^2 \tag{2.26}$$

Para encontrar dicha solución se utiliza una pirámide gaussiana (ver Fig.1.13 del Capítulo 1) de menor a mayor resolución. Una vez encontrados los parámetros:

- Sobreponer todas las imágenes registradas en un retículo de alta resolución, usualmente del doble de filas y columnas y con la mitad del tamaño del píxel.

- Realizar en cada píxel de alta resolución un promedio robusto de todos los píxeles de baja cuyos centros se encuentren dentro de dicho píxel de alta resolución.

- Esta imagen de alta resolución es la imagen de partida del proceso de simulación y corrige que discutiremos a continuación.

Cómo se utilizan los errores para corregir la imagen de alta resolución tiene muchas variantes, en cualquier caso todas ellas tienen una forma similar, veámosla.

Si representamos con Y a las R imágenes de baja resolución ordenadas lexicográficamente en la que en cada imagen los píxeles han sido también ordenados lexicográficamente, la imagen de alta resolución es representada por la letra x y el operador que realiza la compensación de movimiento, emborronamiento y submuestreo por H, tendremos la Ecuación (2.18).

Notemos que un problema muy importante es la correcta modelización de H.

Dado un estimador $\hat{x}$ de la imagen de alta resolución, podemos aplicar la fórmula anterior para obtener un estimador de Y, mediante

$$\hat{Y} = H\,\hat{x} \tag{2.27}$$

Podemos entonces diseñar un procedimiento iterativo de propagación hacia atrás (corrección) de los errores

$$x^{j+1} = x^j + H^{BP}\left(Y - Y^j\right) = x^j + H^{BP}\left(Y - Hx^j\right) \tag{2.28}$$

donde el superíndice indica proceso iterativo y BP indica un proceso de distribución de los errores en la imagen de alta resolución. En muchos casos H^{BP} es una aproximación del inverso

del operador H.

Peleg, Keren y Schweitzer [288], Keren, Peleg y Brada [287] e Irani y Peleg [289] proponen dife-rentes formas simples de proyección hacia atrás.

Sin lugar a dudas los trabajos más relevantes dentro del modelo simular y corregir son los trabajos de Irani, Rousso y Peleg [290] e Irani y Peleg [262] que analizaremos a continuación.

El trabajo de Irani, Rousso y Peleg [290] es de hecho una descripción de un proceso de estimación de movimiento y seguimiento de objetos que pueden estar parcialmente ocultos o ser transparentes. Los movimientos que consideran son traslaciones, movimiento afín y de superficies planas y se realiza su estimación en alta resolución.

Por otra parte, Irani y Peleg discuten en [262] la aplicación de los resultados de [290] a la mejora de resolución espacial, a la reconstrucción de oclusiones y a la reconstrucción de objetos semitransparentes. Comenzamos aumentando la resolución de nuestras imágenes observadas, a estas imágenes le aplicamos la metodología de [290] y obtenemos una estimación del movimiento de los objetos de alta resolución. Tenemos, por tanto, una estimación de las matrices M_k. Ver Fig.2.24. Es decir, al ir aumentando la resolución, aumenta nuestra capacidad de estimar el movimiento de los objetos de la escena y por ende compensarlos, ver Apéndice F.

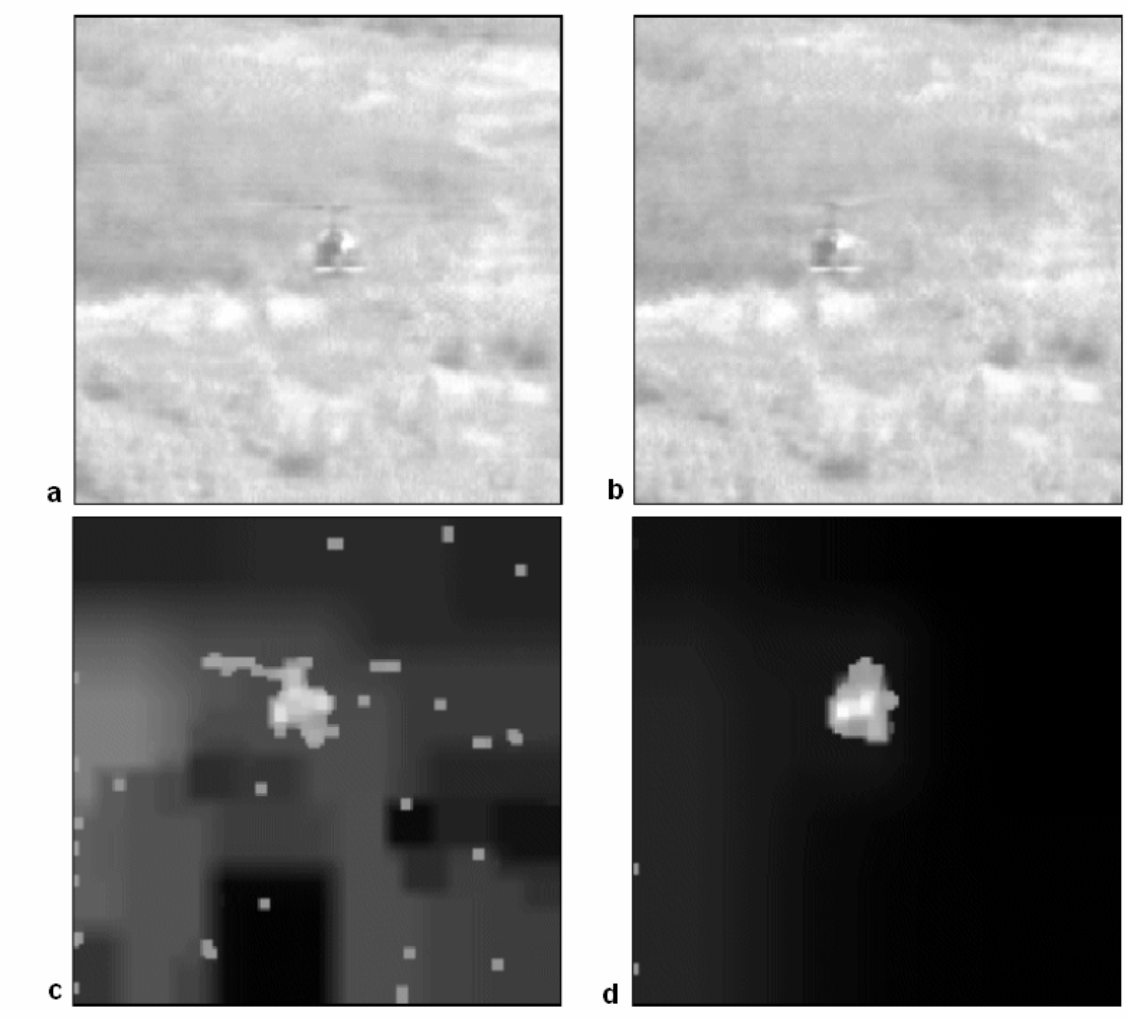


Figura 2.24: Las imágenes son segmentadas teniendo en cuenta el movimiento.

Como puede observarse en la Fig.2.25, vamos obteniendo representaciones de los objetos en las imágenes que no son más que compensaciones del movimiento de los objetos de las imágenes de alta resolución.

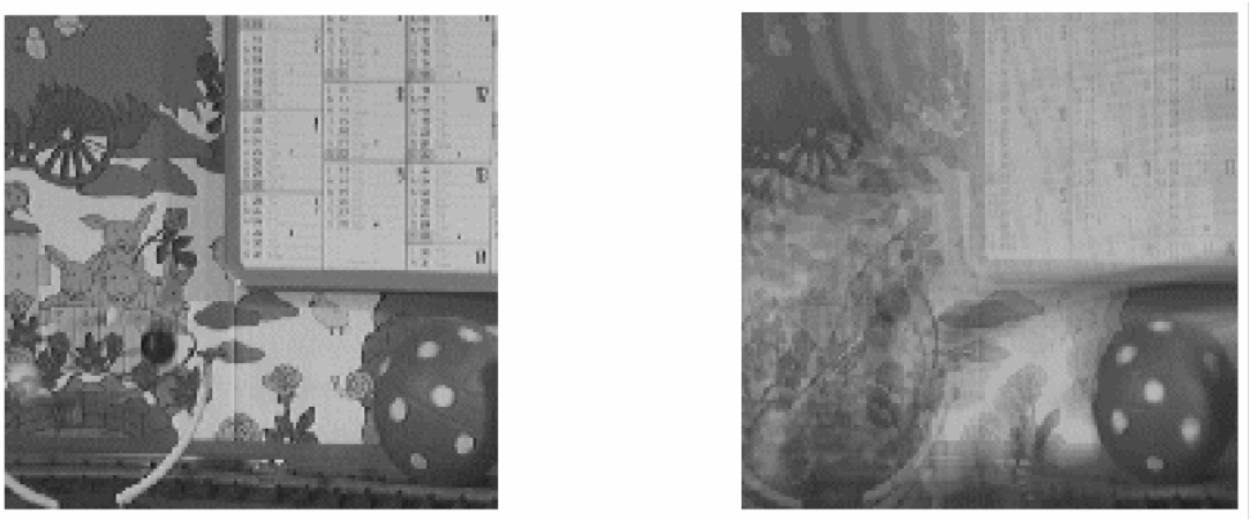


Figura 2.25: Compensación de movimiento de objetos en alta resolución.

Vamos a describir en detalle la mejora de la resolución espacial siguiendo la notación de [262].

Es importante notar que el método propuesto en [262] se basa en la propagación hacia atrás de los errores y que se realiza independientemente para cada objeto que ha sido encontrado en la imagen, ver Apéndice F.

La propagación hacia atrás se hará mediante la utilización de un núcleo de convolución y es interesante destacar que en su forma no permite la incorporación de información a priori.

La secuencia de imágenes de baja resolución $\{y_k\}$ se modeliza mediante

$$y_k = \sigma_k\left(h\left(T_k\left(x\right)\right) + n_k\right) \tag{2.29}$$

donde:

- y_k es la observación de baja resolución del objeto que queremos aumentar su resolución en la imagen k-ésima,
- x es la imagen de alta resolución del objeto que estamos analizando,
- T_k es la transformación geométrica de x a y_k, que se supone invertible y que se realiza en alta resolución para cada objeto,
- h es un operador de convolución que relaciona las imágenes de baja y alta resolución,
- n_k es ruido aditivo es la imagen de alta resolución del objeto que estamos analizando,
- σ_k es el proceso de submuestreo.

Si $x^{(n)}$ es una estimación de la imagen de alta resolución que buscamos en un proceso iterativo, entonces utilizando la fórmula anterior $y_k^{(n)}$ puede escribirse como

$$y_k^{(n)} = \left(T_k\left(x^{(n)}\right) * h\right) \downarrow s \tag{2.30}$$

donde $\downarrow s$ denota submuestreo por un factor s y * convolución.

Podríamos pensar en construir la sucesión $\{x^{(n)}\}$ de forma que se fuese minimizando el error, donde K denota el número de imágenes de baja resolución,

$$e^{(n)} = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\left\|y_k - y_k^{(n)}\right\|^2} \tag{2.31}$$

Para ello el proceso de actualización de la imagen de alta resolución que proponen los autores se escribe como

$$x^{(n+1)} = x^{(n)} + \frac{1}{K}\sum_{k=1}^{K} T_k^{-1}\left(\left(\left(y_k - y_k^{(n)}\right) \uparrow s\right) * p\right) \tag{2.32}$$

donde p es un operador de proyección hacia atrás. Los autores estudian qué tipos de operadores de proyección hacia atrás pueden utilizar y obtienen, en el caso de transformación afín sin disminución de muestreo, una condición suficiente para la convergencia.

2.4 Super-resolución: Regularización e Iteraciones Inversas

Como se mencionó varias veces a lo largo de este capítulo, el método de Super-resolución es usado típicamente para restaurar una imagen de alta-resolución a partir de varias observaciones ruidosas de baja-resolución [249]. En este trabajo, consideramos exclusivamente la interpolación de imágenes únicas. Por lo tanto, vamos a formular el problema como

$$A\,x = y \tag{2.33}$$

donde x es la imagen de alta-resolución desconocida (representada como un vector de los valores del pixel), y es la imagen de baja-resolución conocida, y A es el operador de reducción de escala consistente típicamente de una decimación D seguida de un filtrado pasa-bajo H:

$$A = D\,H \tag{2.34}$$

La elección del operador de filtrado pasa-bajos depende de la PSF del sistema de generación de la imagen digital, el cual produce la imagen de baja-resolución. Si el sistema de generación de la imagen es desconocido, entonces asumiremos que el operador H es una simple caja de filtro.

A. Regularización

La Ec.33 generalmente está mal planteada y un pequeño cambio del vector de entrada y puede causar un gran cambio del vector resultante x. Para la Ecuación (2.33), la solución regularizada se encuentra como:

$$x = \arg\min \|Ax - y\|_n^p + \alpha F(x) \tag{2.35}$$

donde el primer término recibe el nombre de *"discrepancia"*, $F(x)$ es el estabilizador y $\alpha > 0$ es el *coeficiente de regularización* [249].

El estabilizador más popular y universal es el funcional de Tikhonov. Se calcula como una grilla de aproximación del funcional:

$$F(x) = \|\Delta x\|_2^2 \tag{2.36}$$

y $n = 2$, $p = 2$. Para cada $\alpha > 0$ la solución x es correcta: es única, definida para cada y continuamente dependiente de y. Entonces, podemos escribir la ecuación de Euler para este caso:

$$(A^T A + \Delta^2)x = A^T y \tag{2.37}$$

Pero en este caso el algoritmo se convierte en lineal porque x es la solución del sistema de ecuaciones lineales. Por lo tanto, este método hereda inconvenientes de los algoritmos de interpolación lineal y necesitamos encontrar más estabilizadores adaptativos para el re-muestreo de la imagen.

Consideremos los estabilizadores de Variación Total (VT) y VT Bilateral (VTB) [249], los cuales trabajan en la norma l_1 ($n = 1$, $p = 1$):

$$TV(x) = \|\nabla x\|_1 \tag{2.38}$$

donde ∇x es el operador gradiente (su módulo),

$$BTV(x) = \sum_{s,t=-p}^{s,t=p} \gamma^{|s|+|t|} \left\| x - S_x^s S_y^t x \right\|_1 \tag{2.39}$$

donde S_x^s y S_y^t son los operadores de desplazamiento a lo largo de los ejes x e y para los pixeles s y t respectivamente, con $\gamma = 0.8$.

B. Iteraciones inversas
Para resolver la Ecuación (2.35) con el estabilizador (2.39) podemos usar el método iterativo de gradiente descendente:

$$x_{n+1} = x_n - \beta\left\{H^T D^T sign(DHx - y) + \alpha \sum_{s,t=-p}^{s,t=p} \gamma^{|s|+|t|} (I - S_x^{-s} S_y^{-t})\, sign(S_x^s S_y^t x)\right\} \tag{2.40}$$

z = signo de x es un vector con la función signo aplicada por-elemento; D^T es una operación de sobre-escalado. Si D en (2.34) es el operador de decimación más simple que toma cada *k-th* pixel, D^T es el operador de sobre-escalado por la inserción de ceros. Si H en (2.34) es un filtrado simétrico, entonces H^T es igual a H. x_0 es la aproximación inicial a la imagen de alta-resolución.

2.5 Conclusiones del capítulo

Este capítulo se analizaron las principales herramientas de super-resolución, pieza fundamental en el proceso de supercompresión del capítulo siguiente. La super-resolución fue abordada desde todos sus puntos de vista, es decir, desde la simple restauración de la nitidez de una imagen, hasta la recuperación de la resolución en un contexto de supercompresión, el que como en el capítulo siguiente involucra interpolación bidimensional como el mecanismo fundamental que da lugar primero al sub-muestreo en el codificador y luego al sobre-muestreo en el decodificador.

Capítulo 3

Compresión vs Supercompresión

Este capítulo atañe a la diferencia fundamental entre la compresión y la supercompresión, su compatibilidad con esquemas de compresión en uso y sus fundamentales aplicaciones. Por otra parte, contribuye al aporte de la presente tesis.

3.1 Esquema de super-resolución para compresión

Esta sección está organizada en cuatro partes, para un mejor desarrollo de los conceptos:

- Super-resolución vs Restauración de la Nitidez
- Compresión vs Super-compresión
- Deducción de la máscara
- Aplicaciones

3.1.1 Super-resolución vs Restauración de la Nitidez:

Vamos a establecer aquí dos definiciones rigurosas con el propósito de eliminar cualquier confusión entre los conceptos de super-resolución y restauración de la nitidez [2-9].

Definición 1:
Un proceso es de super-resolución, si el mismo restaura la nitidez de una imagen involucrando en dicho proceso un incremento en la resolución de la misma [113, 247-251, 291, 292].

Definición 2:
Un proceso es de restauración de la nitidez, si éste restaura la nitidez de una imagen sin involucrar en dicho proceso un incremento en la resolución de la misma. Esto se aplica cuando la nitidez de la imagen sufre una degradación llamada blur [2-9]*, la cual proviene de una alta velocidad relativa entre un elemento que se intenta enfocar y la cámara que lo capta, o bien la poca rapidéz de apertura y cierre del obturador de dicha cámara, entre otras posibilidades.*

Consideramos importante mencionar que ambos procesos pueden involucrarse mutuamente como parte del procedimiento de mejora de la nitidez de la imagen. De hecho, podemos entender a la super-resolución como un proceso de incremento de la resolución seguida de una restauración de los bordes mediante un proceso de restauración de la nitidez.

3.1.2 Compresión vs Super-compresión:

Definimos compresión como el proceso que reduce en promedio el número de bit-por-pixeles (bpp) de una imagen. En la Fig.3.1, representamos el conjunto de planos de bits en el cual se descompone una imagen en grises o colores. Como se puede apreciar en la Fig.3.1, el proceso de compresión no altera la resolución de la imagen [2-9].

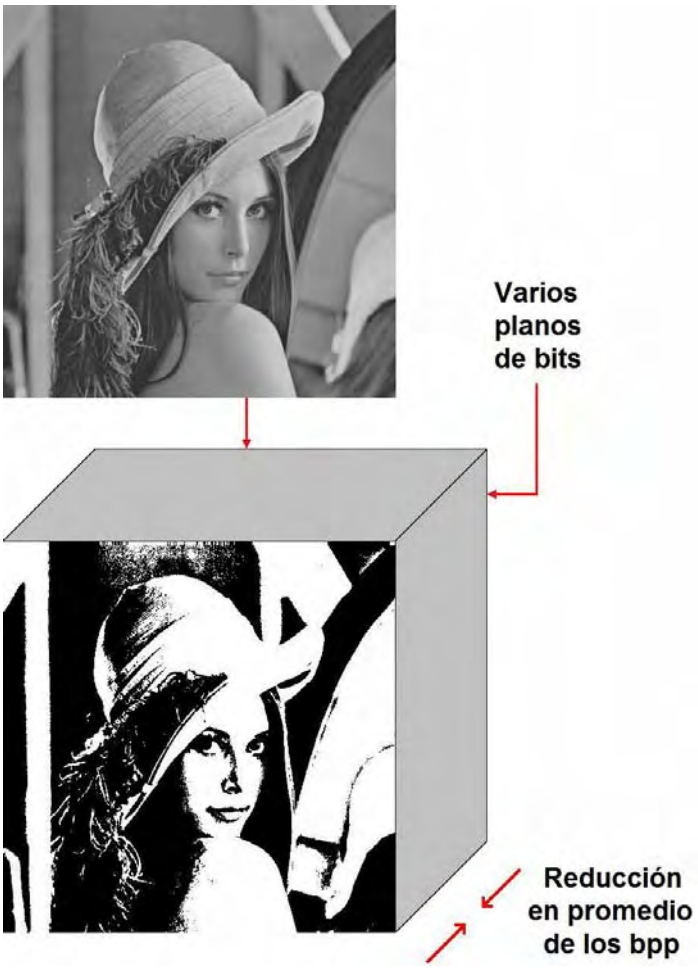


Fig.3.1: Compresión.

En cambio, definimos supercompresión como el proceso que reduce en promedio el número de bit-por-pixeles (bpp) de una imagen luego de una disminución en la resolución de la misma. La Fig.3.2 muestra que este proceso de reducción de la resolución de la imagen se lleva a cabo mediante un sub-muestreo, el cual realiza un encogimiento en filas y columnas simultáneamente, sin obligación de respetar la relación de aspecto, como es el caso de TV Digital, cuya relación de aspecto es 16:9 (para alta resolución, mientras es 4:3 para resolución estándar). De hecho, para la norma ISDB-Tb (Integrated Services Digital Broadcasting–Terrestrial brazilian) del sistema de TV Digital brasilero y argentino, practicaremos aquí una reducción de 5:1 como tasa de compresión por encima de la compresión original del sistema, el cual emplea como estándar de compresión de video [293] al algoritmo Advanced Video Coding (AVC), ver Apéndice F.

En que consiste el AVC?

El AVC, también conocido como H.264 o MPEG-4 parte 10**, e**s una norma que define un códec de vídeo de alta compresión, desarrollada conjuntamente por el ITU-T (Unión Internacional de Telecomunicaciones) Video Coding Experts Group (VCEG) y el ISO/IEC Moving Picture Experts Group (MPEG). La intención del proyecto H.264/AVC fue la de crear un estándar capaz de proporcionar una buena calidad de imagen con tasas binarias notablemente inferiores a los estándares previos (MPEG-2, H.263 o MPEG-4 parte 2), además de no incrementar en forma significativa la complejidad de su diseño.

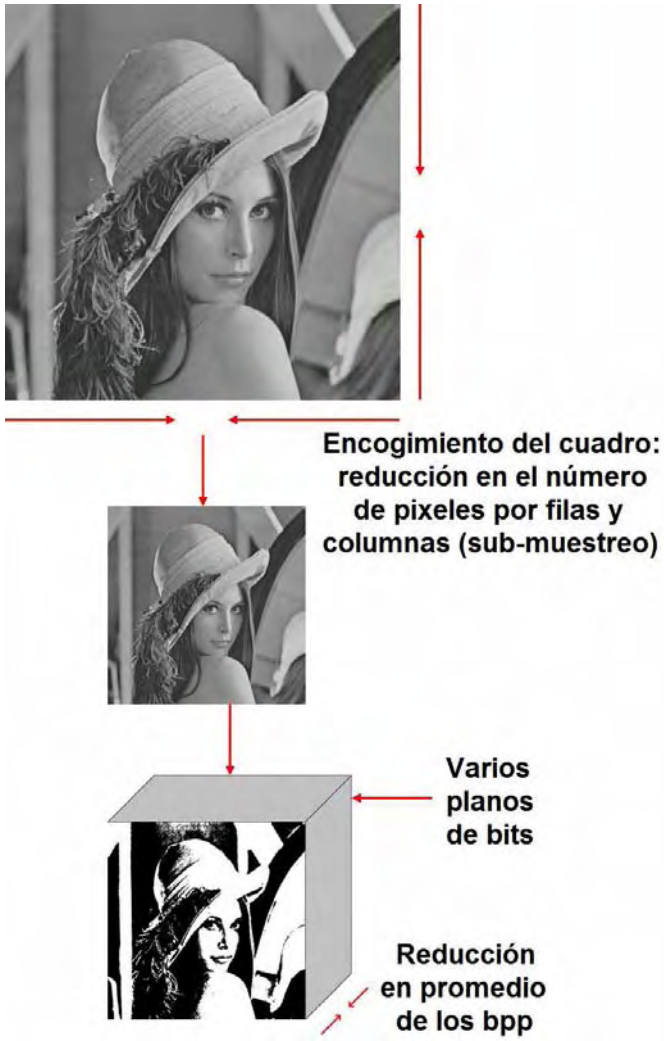


Fig.3.2: Supercompresión.

Para garantizar un ágil desarrollo de dicha norma, la ITU-T y la ISO/IEC acordaron unirse para desarrollar conjuntamente la siguiente generación de compresores de vídeo. El Joint Video Team (JVT) que estaba formado por expertos del VCEG y MPEG y nació en diciembre de 2001 con el objetivo de completar el desarrollo técnico del estándar hacia el 2003. La ITU-T planeó adoptar el estándar bajo el nombre de ITU-T H.264 y ISO/IEC bajo el nombre de MPEG-4 Parte 10 Códec de Vídeo Avanzado (AVC) y de aquí surgió el nombre híbrido de H.264/MPEG-4 AVC. Para empezar a programar el código del nuevo estándar adoptaron las siguientes premisas:

- La estructura DCT (Transformada Coseno Discreta) + Compensación de Movimiento de las versiones anteriores es superior a otros estándares y por esto no hay ninguna necesidad de hacer cambios fundamentales en la estructura.

- Algunas formas de codificación de vídeo que habían sido excluidas en el pasado debido a su complejidad y su alto coste de implementación se volverían a examinar para su inclusión puesto que la tecnología VLSI (Very Large Scale Integration) había sufrido un adelanto considerable y una considerable bajada de costos de implementación.

- Para permitir una libertad máxima en la codificación y evitar restricciones que comprometan la eficiencia, no se contempla mantener la compatibilidad con normas anteriores.

Cuando en los párrafos anteriores decimos que, *incrementamos la compresión del estándar 5 veces*, esto significa que pasamos de una resolución de 1920x1080 (Full-High Definition: Full-HD, i.e., definición alta y completa) a otra 5 veces menor de 720x576 (Standard Definition: SD, definición estándar). El estándar de compresión de video H.264 no se ve afectado por la supercompresión. Como se presentará en la Sub-Sección 3.1.4, la supercompresión requiere mínimo equipamiento en el transmisor, mientras que el procedimiento inverso de la supercompresión, tiene lugar en el receptor, conocido como set-top-box [294]. No obstante, la ausencia de disponibilidad del proceso inverso a la supercompresión en el receptor, obliga a este último a ser compatible con la SD, a efectos de que por lo menos se reproduzcan imágenes en esta resolución con un set-top-box normal, dado que el receptor enviará la señal SD a un televisor LCD (Liquid Display Crystal), el cual realizará automáticamente un sobre-muestreo (a los efectos de completar la pantalla) cambiando la relación de aspecto, como es el caso en el que un TV LCD Full-HD recibe una señal SD. En la Fig.3.2, representamos el conjunto de planos de bits en los cuales se descompone una imagen en grises o colores.

Como también se discutirá en la Sub-Sección 3.1.4, nuestro procedimiento de supercompresión consiste en dos partes bien definidas: el transmisor y el receptor.

En el transmisor tenemos tres pasos:
1. Obtención de los cuadros individuales del video (video slicing): cuadro-por-cuadro
2. Sub-muestreo
3. Reensamblado del video a partir de los cuadros sub-muestreados (video reassembling)

Mientras que en el interior del receptor (set-top-box) tenemos cuatro pasos:
1. Receptor del streaming (flujo de información en el estándar H.264)
2. $H.264^{-1}$ (decodificación del estándar H.264)
3. Sobre-muestreo
4. Restauración de la nitidez

En nuestro caso, tanto el sub-muestreo como el sobre-muestreo se implementan mediante una interpolación bilineal (como se explica en el Capítulo 2), mientras que la restauración de la nitidéz se realiza gracias a una máscara convolutiva bidimensional de NxN pixeles, la cual realiza un barrido fila por fila de izquierda a derecha y de arriba hacia abajo sobre la imagen sobre-muestreada (con blur, del inglés: blurred). Los elementos de esta máscara cuadrada (donde $N = 2 \times M + 1$, con $M \in \aleph$, por lo tanto, $N \in \aleph$ y $N \geq 3$) son críticos, por lo tanto, deben ser calculados y ajustados con exactitud. En la próxima sección, procederemos a deducir la máscara así como el conjunto de relaciones óptimas entre sus parámetros. Luego procederemos a ajustarlos mediante un Algoritmo Genético [295].

3.1.3 Deducción de la máscara:

3.1.3.1. Mediante Filtro de Kalman: En base a la última sección, un único cuadro es recuperado luego de sufrir dos procesos: sub-muestreo y sobre-muestreo, ver el lado izquierdo de la Fig.3.3. En esta figura:

$\mathbf{X_t}$ significa cuadro único y original.
$\mathbf{Y_t}$ significa cuadro único recuperado (blurred).
$\mathbf{M_b}$ significa mascara cuadrada de NxN pixeles ($N \in \aleph$, N es impar y $N \geq 3$).
El sub-índice ***t*** significa *iteración-t*.

↓ significa sub-muestreo.
↑ significa sobre-muestreo.

La máscara M_b produce la pérdida de la nitidez (blur) y es conocida como operador de suavizado o PSF (por sus siglas en inglés, Point Spread Function) [247] y va a sintetizar simultáneamente a los procesos de sub-muestreo y sobre-muestreo (es decir, los modeliza) de manera tal de ser la responsable mediante convolución bilineal de la pérdida de nitidez de la imagen original.

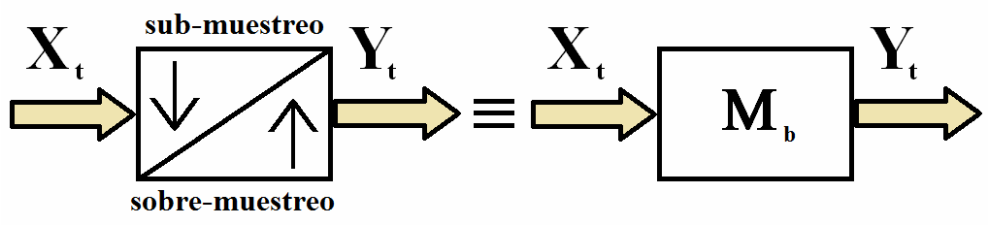


Fig.3.3: Sub-muestreo/sobre-muestreo representados con una máscara convolutiva bidimensional responsable de la pérdida de nitidez de la imagen.

En estos procesos (↓ y ↑), un cuadro de un video (o imagen suelta) es afectado individualmente por blur invariante en el espacio y el tiempo, el cual lo interpretaremos como el resultado de la acción de una convolución bidimensional entre el cuadro (o imagen) original y la máscara, la cual es conocida en Procesamiento Digital de Imágenes como máscara de la media [2-9]. La idea de un filtrado de la media consiste sencillamente en reemplazar el valor del pixel central del sector afectado por la máscara con el valor de la media (`promedio ') entre sus vecinos y el mismo. Esto tiene el efecto de reemplazar el valor de aquellos pixeles que no son representativos de su entorno. El filtrado de la media es usualmente asimilado a un filtro convolutivo [2-9]. Al igual que otras convoluciones basadas en torno a un kernel, este representa la forma y dimensión del vecin-dario o entorno a ser muestreado cuando calculamos la media. Frecuentemente se usa un kernel cuadrado de 3×3, aunque también puede ser usado un kernel más grande (e.g. 5×5) para un suavizado más severo. Notemos que un kernel pequeño puede ser aplicado más de una vez en términos de producir un efecto similar aunque no idéntico a una única pasada con un kernel más grande. En la Fig.3.4, consideramos el caso más general, para un kernel de NxN, siempre con N impar, donde:

$$\varphi = \frac{1}{N \times N} \tag{3.1}$$

Calculando sencillamente la convolución de una imagen con este kernel arribamos a un proceso de filtrado de media.

En base a lo expresado anteriormente, necesitamos un estimador para recuperar el cuadro individual (o imagen) del proceso que lo afecta (blur). Por lo tanto, para una imagen afectada por una convolución bidimensional como la máscara de la Fig.3.4, el mejor estimador es un modelo constante de filtro de Kalman [296].

El conjunto de ecuaciones que refleja el modelo enunciado puede ser dividido en dos etapas: el modelo y el estimador [296].

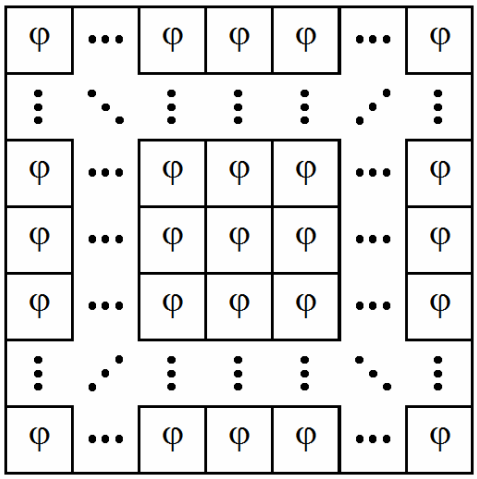


Figura 3.4: Kernel promediador N×N frecuentemente usado en filtrado de media.

Basándonos en la Fig.3.5, donde Δ significa retardo unitario (tiempo discreto entre muestras) para cada elemento del cuadro individual $\boldsymbol{X}_{t+1}$, tenemos:

Modelo de pérdida de nitidez:

$$X_{t+1} = X_t \tag{3.2}$$

$$Y_t = M_b * X_t \tag{3.3}$$

Donde:
* significa convolución bidimensional.
X_t es el cuadro (o imagen) original en el tiempo de iteración *t*
Y_t es el cuadro (o imagen) con blur (medición) en el tiempo de iteración *t*

Estimador del cuadro (o imagen):

$$\widehat{X}_{t+1} = \widehat{X}_t + K \times \varepsilon_t \tag{3.4}$$

$$K = k \times I \tag{3.5}$$

$$\varepsilon_t = Y_t - \widehat{Y}_t \tag{3.6}$$

$$\widehat{Y}_t = M_b * \widehat{X}_t \tag{3.7}$$

$$\begin{aligned} P_t &= (I - k \times I) \times P_t^- \\ &= (1-k) \times P_t^- \end{aligned} \tag{3.8}$$

Donde:
I significa matriz identidad
k es un parámetro constante a ajustar (0 < *k* < *2)*
K es la ganancia de Kalman [296].
$\widehat{X}_t$ es el cuadro (o imagen) estimado en el tiempo de iteración *t*
$\widehat{Y}_t$ es el cuadro (o imagen) estimado con blur (medición) en el tiempo de iteración *t*
ε_t es el error de medición en el tiempo *t*, el cual debe cumplir con $E\left\{\varepsilon_t \times Y_t^T\right\} = 0$

Siendo,

$$P_t = E\left\{\varepsilon_t \times \varepsilon_t^T\right\} \tag{3.9}$$

la matriz de autocorrelación del error de medición del filtro (P_t^- significa muestra de P_t para suavizado [296]). La cual, para un proceso en estado estacionario debe cumplir con

$$\lim_{t\to\infty} \boldsymbol{P_t = 0} \tag{3.10}$$

Siendo:
$\boldsymbol{0}$ la matriz nula (todos sus elementos iguales a cero)
$E\{\bullet\}$ representa la esperanza matemática de "•".

El modelo del proceso de pérdida de nitidez via sub y sobre-muestreo sintetizados en la acción de la máscara convolutiva bidimensional M_b, así como el del Filtro del Kalman invariante que restaura la nitidez al recuperar una versión estimada de X_t, es decir, $\widehat{X}_t$, se los puede observar completos en la Fig.3.5. En la misma figura Δ significa retardo unitario.

Por otra parte, la implementación computacional del conjunto de ecuaciones anterior implica el uso de cuatro *for*'s anidados más un estricto control de la estabilidad de la Ec.3.8 al restringir los posibles valores de *k*, i.e., solo es posible usar $0 < k < 2$.

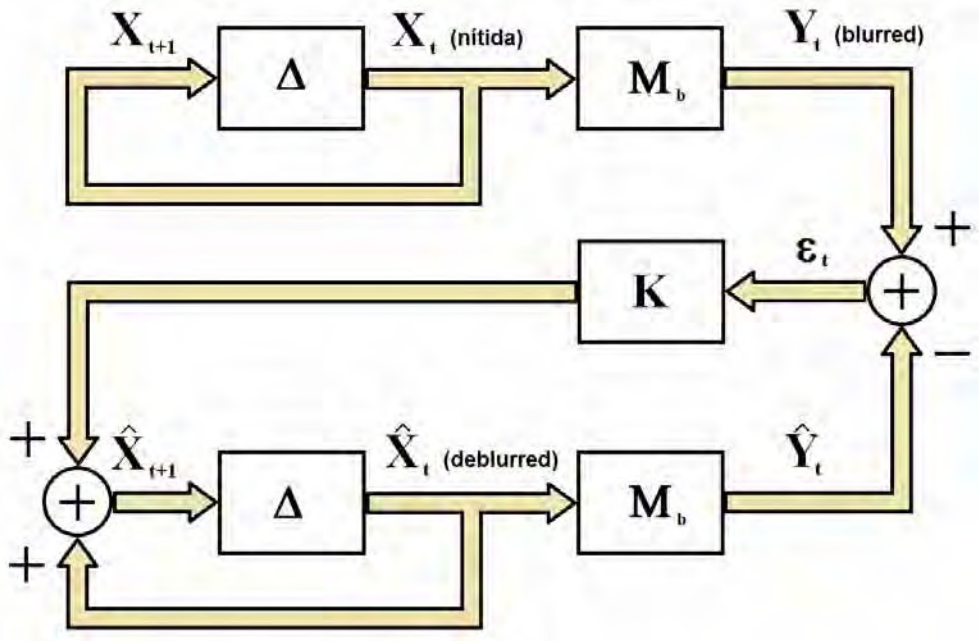


Figura 3.5: Modelo Constante del Filtro de Kalman.

Por lo tanto, es mucho más eficiente implementar tal filtrado a través de una simple convolución con una máscara bidimensional, eliminando el predictor de la Ecuación (3.4), lo cual nos permitirá una implementación más eficiente usando - por ejemplo - una convolución mediante la Transformada Rápida de Fourier (FFT: Fast Fourier Transform) [2-9]. En consecuencia, necesitamos deducir dicha máscara. Si reemplazamos la Ec.3.7 en la Ec.3.6, tenemos,

$$\varepsilon_t = Y_t - M_b * \widehat{X}_t \tag{3.11}$$

Ahora, reemplazamos las Ecuaciones 3.5 y 3.11 en la 3.4, obteniendo,

$$\widehat{X}_{t+1} = \widehat{X}_t + k \times I \times \left(Y_t - M_b * \widehat{X}_t\right) \tag{3.12}$$

Reagrupando términos de la Ecuación 3.12, y teniendo presente un modelo de bajo ruido con un blur invariante en el tiempo y en el espacio lineal, obtendremos,

$$\widehat{X}_{t+1} = M_d * Y_t \tag{3.13}$$

Donde $\boldsymbol{M_d}$ resulta ser una máscara de realce [2-9] con la canonicidad que se muestra en la Fig.3.6.

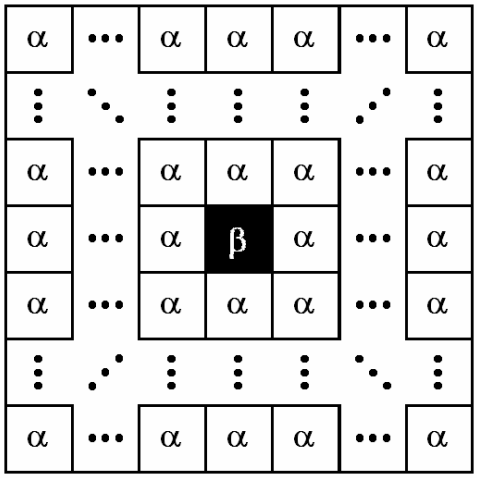


Figura 3.6: Máscara de realce $\mathbf{M_d}$.

Siendo muy importantes las siguientes relaciones a considerar:

$$(N^2 - 1) \times \alpha + \beta = 1, \text{ (para restauración de la nitidez)} \tag{3.14}$$

$$(N^2 - 1) \times \alpha + \beta = 0, \text{ (para detección de bordes)} \tag{3.15}$$

Entonces, aparece en escena un nuevo y simplificado modelo de restauración de la nitidez, ver Fig.3.7, donde $\alpha < 0$ y $\beta > 1$.

Necesitamos establecer con precision ambos parámetros, por lo que, hay dos posibles formas de lograrlo:

1. Elegir N (entero, positivo, impar y pequeño), y $\beta > 1$ (y arbitrariamente menor que 2), entonces α se deriva de la Ecuación (3.14).
2. Comenzar con valores arbitrarios de α y β (sobre ciertas recomendaciones a modo de recetas, e.g., $-0.1 < \alpha < 0$ y $1 < \beta \leq 2$) y generar aleatoriamente una población de pares de $[\alpha, \beta]$, y deducir N de la Ec.3.14.

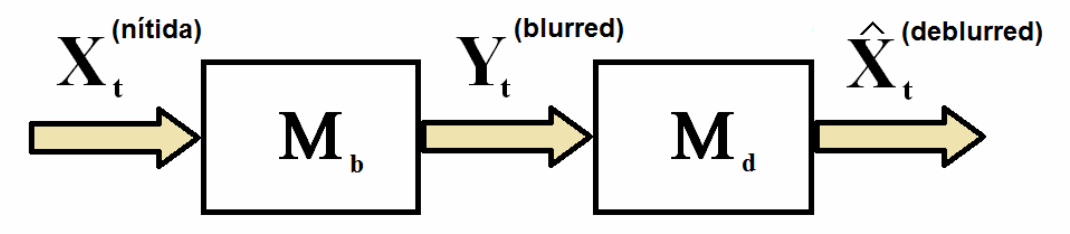


Figura 3.7: Modelo nuevo y simplificado de restauración de la nitidez.

Los mencionados pares sirven de población inicial para el Algoritmo Genético [295] de la Fig.3.8, donde cada par recibe el nombre de cromosoma, y α y β son llamados genes. La métrica para el ajuste es el Error Cuadrático Medio (MSE: Mean Squared Error), el cual se definirá en las Ecuaciones 4.5 y 4.7 de la página 104 del Capítulo 4 [15, 16].

Por otra parte, el Algoritmo Genético utilizado está compuesto de tres grandes módulos:

a) Scoring (puntuación),
b) Operador de Crossover (o apareamiento), y
c) Operador de Mutación

El primero – a su vez – consiste de los siguientes submódulos:

a.1) Set-point (punto de ajuste) donde surge el error pixel-a-pixel
a.2) Cálculo del MSE a partir del error pixel-a-pixel
a.3) Ordenando la camada de cromosomas por MSE, de mínimo (mejor) a máximo (peor)
a.4) Operador de Genocidio, el cual elimina los cromosomas de mayor MSE, i.e., se toma como parámetro de diseño del Algoritmo Genético un número fijo de cromosomas que sobreviven por ciclo, es decir, los que me mejor se adaptan.

El Operador de Crossover (u Operador de Apareamiento) aparea a los cromosomas que harán las veces de padres (los cuales son seleccionados aleatoriamente) generando nuevos cromosomas que serán los hijos de los anteriores hijos, los cuales resultarán mejor y peor que sus padres [295].

El Operador de Mutación debe tener una frecuencia de acción baja a los efectos de no perturbar la naturaleza de la especie (solución fuera del espacio de configuración de trabajo, e.g., solución tipo *síndrome de pegasus*, i.e., caballo con alas, este es un caso muy famoso con la intención de mejorar los purasangre de carreras), o sea, saltar a resolver otro problema [295].

El algoritmo genético arranca su funcionamiento con una imagen tipo nítida y una camada (o población) inicial de cromosomas con genes del tipo (α , β) generados aleatoriamente bajo las condiciones iniciales establecidas en los párrafos anteriores. Se somete a la imagen al proceso de sub y sobre-muestreo, dando lugar a una imagen con blur. Entonces se convoluciona esta última con todas las máscaras surgidas del empleo de los pares (α , β) según el modelo canónico de máscara establecido en la Fig.3.6 y luego se ordenan de mejor a peor las imágenes resultantes, es decir, de menor a mayor Error Cuadrático Medio, quedándonos por ciclo con un número fijo en la población de la camada. Repetimos este proceso siempre que en cada camada surjan imágenes estimadas mejores que las anteriores. Cuando el proceso se estaciona en calidad o incluso empeora (es decir, se alcanza –de un ciclo al siguiente– un MSE igual o mayor), se detiene y así nos quedamos con los mejores (α , β) conocidos hasta ese momento.

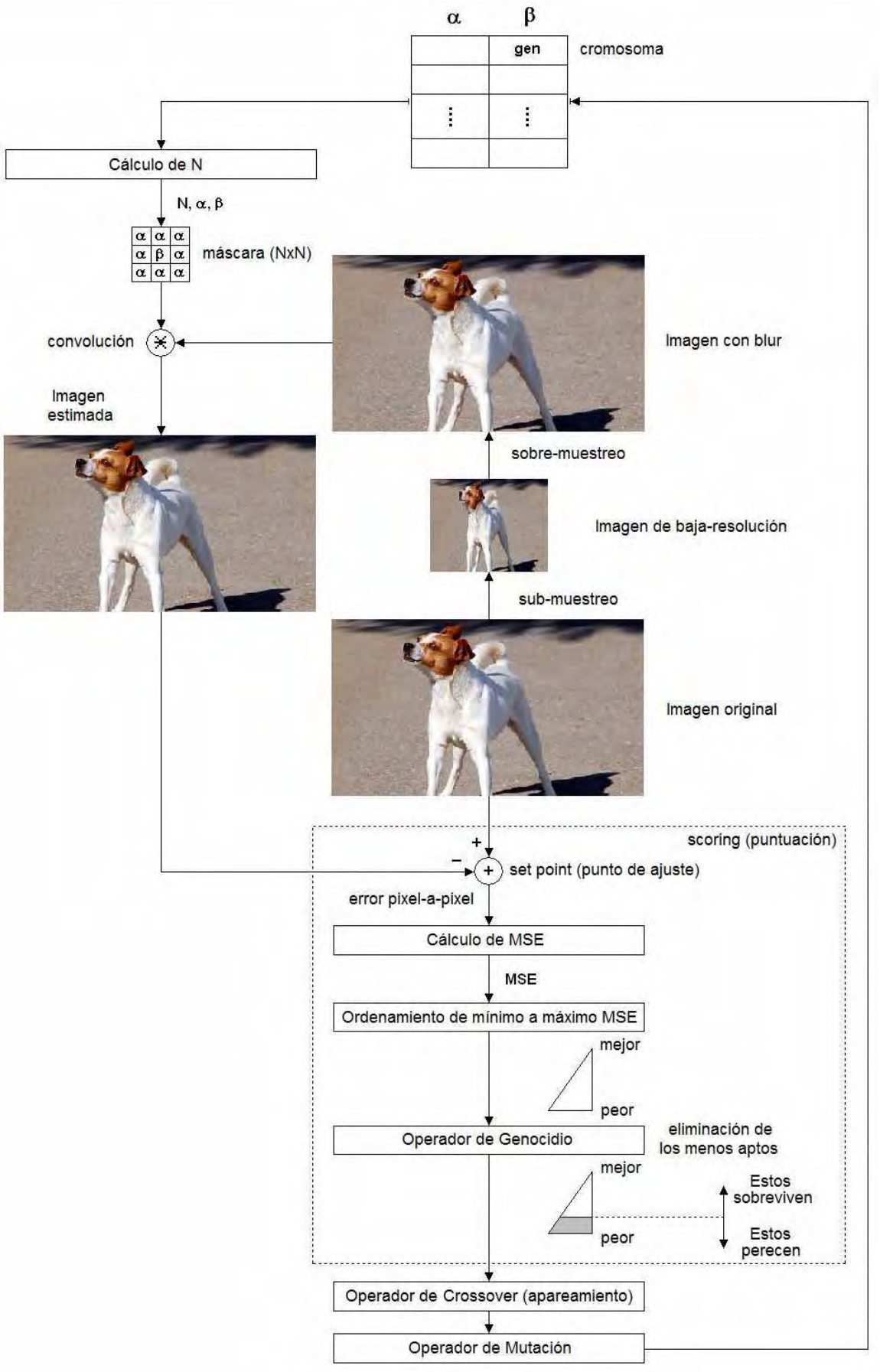


Figura 3.8: Algoritmo Genético para calcular los parámetros de la máscara.

3.1.3.2. Mediante el estimador del algoritmo recursivo de Van Cittert: Sobre la base de lo visto hasta ahora, y teniendo en cuenta que necesitamos un estimador para recuperar el cuadro único del proceso que lo afecta, es decir, un proceso de sub-muestreo seguido de otro de sobre-muestreo, como muestra la Fig.3.3, un candidato para un nuevo estimador es el algoritmo recursivo de Van Cittert [297, 298]. Aquí también la matriz M_b representa una máscara convolutiva y desconocida, la cual sintetiza la acción combinada del sub y sobre-muestreos juntos, siendo el *modelo* el mismo de las Ecuaciones (3.2) y (3.3), nuestro nuevo estimador será:

$$\widehat{X}_{t+1} = \widehat{X}_t + \lambda \times \varepsilon_t \tag{3.16}$$

Donde λ es un parámetro que se selecciona a medida que se va comparando el Error Cuadrático Medio que surge entre la imagen original tipo y aquella resultante del sub + sobre-muestreo y la aplicación del algoritmo recursivo de Van Cittert [297,298]. Por razones relativas a la estabilidad del algoritmo, este parámetro debe encontrarse en el rango $0 < \lambda < 2$. Mientras que la condición inicial del estimador es:

$$\widehat{X}_0 = Y \tag{3.17}$$

Siendo las Ecuaciones (3.6) y (3.7) las mismas para este estimador.

Por otra parte, la implementación computacional del conjunto de ecuaciones de este nuevo estimador implica (al igual que con el primero) el uso de 4 (cuatro) *for*'s anidados más un estricto control (también en este caso) de la estabilidad de la Ec.16 (con forma de predictor) lo que fuerza a una estricta restricción de los posibles valores de λ, i.e., solo es estable su uso para $0<\lambda<2$.

Por lo tanto, es mucho más eficiente implementar tal filtrado a través de una simple máscara de convolución bidimensional, eliminando la forma de predictor de la Ec.3.16, lo cual da lugar – aquí también – a implementaciones mucho más eficientes usando, por ejemplo, una convolución a través de la Transformada Rápida de Fourier (en inglés, Fast Fourier Transform: FFT) [2, 4]. En consecuencia, necesitamos deducir la mencionada máscara. Si reemplazamos la Ec.3.6 en la Ec.3.16, tendremos,

$$\widehat{X}_{t+1} = \widehat{X}_t + \lambda \times \left(Y_t - M_b * \widehat{X}_t\right) \tag{3.18}$$

Reagrupando términos de la Ec.3.18, y teniendo presente un modelo de bajo ruido con un blur invariante en el tiempo y en el espacio lineal, nuevamente arribamos a la Ec.3.13 como en el caso del filtro de Kalman, donde también aquí M_b es la máscara mostrada en la Fig.3.6, a la que obviamente le corresponden las relaciones de las Ecuaciones (3.14) y (3.15), con iguales consideraciones para α y β que para el caso anterior. La acción de ambas mascaras deducidas puede observarse en la Fig.3.9. En primer lugar, llevamos a cabo un sobre-muestreo de la imagen, y en Segundo lugar, aplicamos la máscara de la Fig.3.6 sobre la imagen del medio, entonces, obtenemos una imagen final de muy superior calidad, es decir, menor MSE [245, 246].

En otras palabras, la Fig.3.9 muestra que partiendo de una imagen original y aplicando un simple sobre-muestreo mediante interpolación bidimensional bilineal obtenemos la imagen del medio de dicha figura. En cambio, al aplicar el procedimiento completo propuesto, obtenemos la imagen final del extremo derecho de la Fig.3.9, de mayor calidad visual, o sea, de mayor MSE.

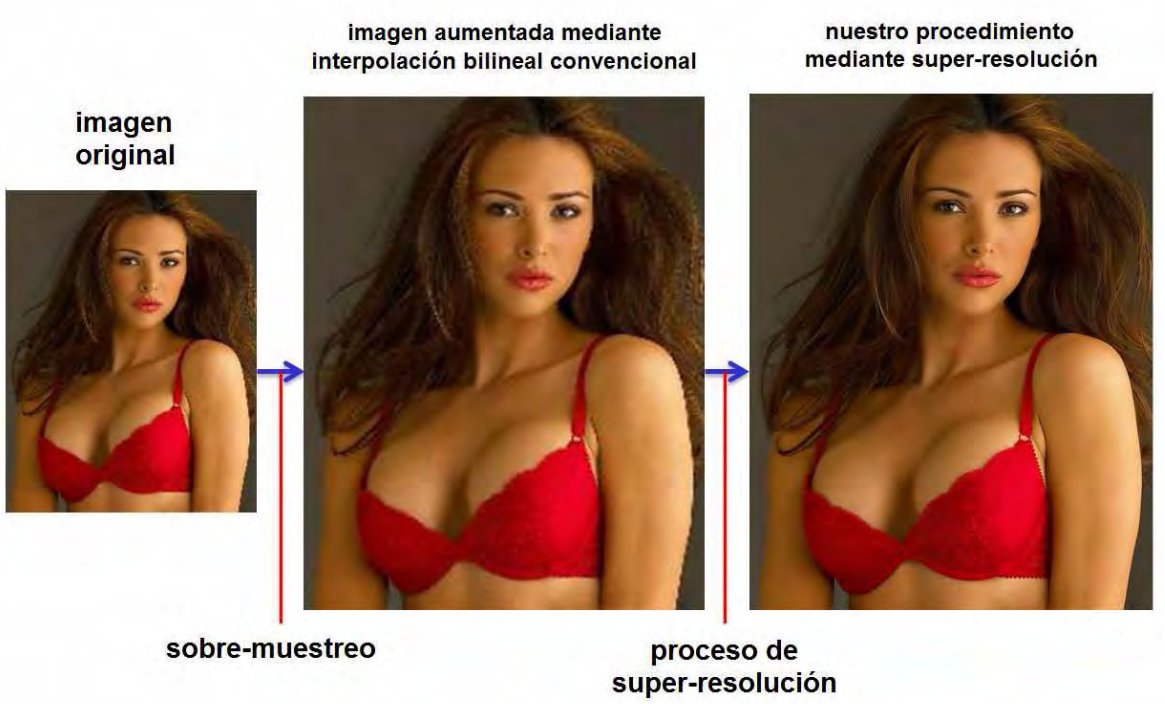


Figura 3.9: Efecto de las máscaras deducidas, siendo aplicadas luego de los procesos de sub y sobre-muestreo.

3.1.4 Ejemplos:

Presentamos aquí varias aplicaciones principales de compresión de video en tiempo real para TV Digital, de acuerdo al estándar ISDB-Tb [299].

En la primera, vamos de una resolución de 1920x1080 Full-HD a otra 5 veces menor de 720x576 SD. Como hemos dicho anteriormente, el estándar de compresión de video H.264 no es afectado por la supercompresión, dado que en lugar de codificar en el estándar a un cuadro de 1920x1080, se codifica en cambio a un cuadro de 720x576. La Fig.3.10 muestra el diagrama del codificador con tres módulos inmersos en placas GPGPUs. La Fig.3.11 en cambio muestra en detalle la tecnología empleada para la implementación real de la Fig.3.10, la cual consiste de 3 GPGPUs Quadro (SDI Capture, 6000 y SDI Output) [300], para la fragmentación cuadro-a-cuadro del video, el sub-muestreo (o decimación espacial [301]) con el filtro anti-aliasing incluido y el reemsamble del mismo.

En la Fig.3.11, TX significa transmisor. Por otra parte, es importante aclarar que el procedimiento de supercompresión de video no afecta al audio, ni el sincronismo con este, pues todo el proceso se lleva a cabo en menos de 40 mseg, es decir, en menos tiempo del que media entre dos cuadros de TV Digital en la norma en uso en la República Argentina. En dicha norma, el códec de audio empleado es el AAC “Advanced Audio Coding” [299], ver Apéndice F.

Es relevante tener en cuenta respecto a las Figuras 3.10 y 3.11 que los tres procesos, es decir, fragmentación cuadro-a-cuadro, el sub-muestreo con filtrado antialiasing y el re-ensamble del video se llevan a cabo íntegramente en la placa GPGPU del medio, es decir la Quadro 6000, mientras que las otras dos hacen las veces de interfaz por un lado y captura por el otro, como es el caso de la Quadro SDI capture. A este respecto, el tipo de entrada de esta última placa condiciona fuertemente el tipo de cámara a ser empleada en el experimento, reduciendo notablemente su

universo de modelos y marcas.

Es importante mencionar en este punto que el filtro anti-aliasing se emplea en el codificador luego del sub-muestreo a los efectos de mitigar el efecto conocido como aliasing en el decodificador, el cual se pondría en evidencia luego del proceso de sobre-muestreo y realce. El aliasing es un desagradable efecto que consiste en un escalonado o serrucho de los bordes y líneas de la imagen recuperada.

Como se describe en detalle en el Apéndice G, existen varias técnicas para suprimir este desagradable efecto, aunque la más recomendada para todo tipo de aliasing en TV Digital es sin duda alguna la basada en máscara convolutiva bidimensional del tipo Gaussiana [2-4], la cual permite una recuperación de alta calidad visual de la imagen procesada mediante una razonable complejidad computacional. Como se verá en el Apéndice F, la máscara del filtro Gaussiano anti-aliasing empleada en el presente trabajo es de 7x7 pixeles, brindando un perfecto equilibrio entre anti-aliasing y razonable complejidad computacional.

Figura 3.10: Codificador.

Figura 3.11: Codificador implementado con GPGPUs.

Por otra parte, la Fig.3.12 muestra un diagrama del decodificador implementado en el interior del set-top-box (STB). De manera tal que se dispone de las siguientes opciones:

1. Si el STB posee la superdescompresión y dependiendo de la resolución del televisor LCD, LED o plasma, obtenemos las resoluciones

   1.1. High Definition (HD) $1280x720_p$ pixeles

   1.2. Full-HD $1920x1080_i$ pixeles

2. No obstante, si el STB no dispone de la superdescompresión, de todos modos el sistema debe ser compatible, por lo tanto obtenemos solo Standard Definition, es decir, $720x576_p$.

Donde el subíndice:

"***p***" significa escaneo progresivo: En cada barrido son exploradas todas las líneas de pixeles del alto de la imagen

"***i***" significa escaneo entrelazado: En cada barrido se exploran primero las líneas impares y en un segundo barrido las pares

Se hace evidente que el escaneo progresivo implica una mejor calidad de imagen a costa de manejar el doble de información visual a transmitir y/o almacenar por el sistema. En el Apéndice F, la Fig.F.12 detalla estos aspectos.

La Fig.3.13 muestra el Módulo de Super-Resolution (MSR) usado en el interior del STB de la Fig.3.12, el cual incluye el sobre-muestreo y el restaurador de la nitidez (realce), que permite restaurar la resolución, es decir, Full-HD.

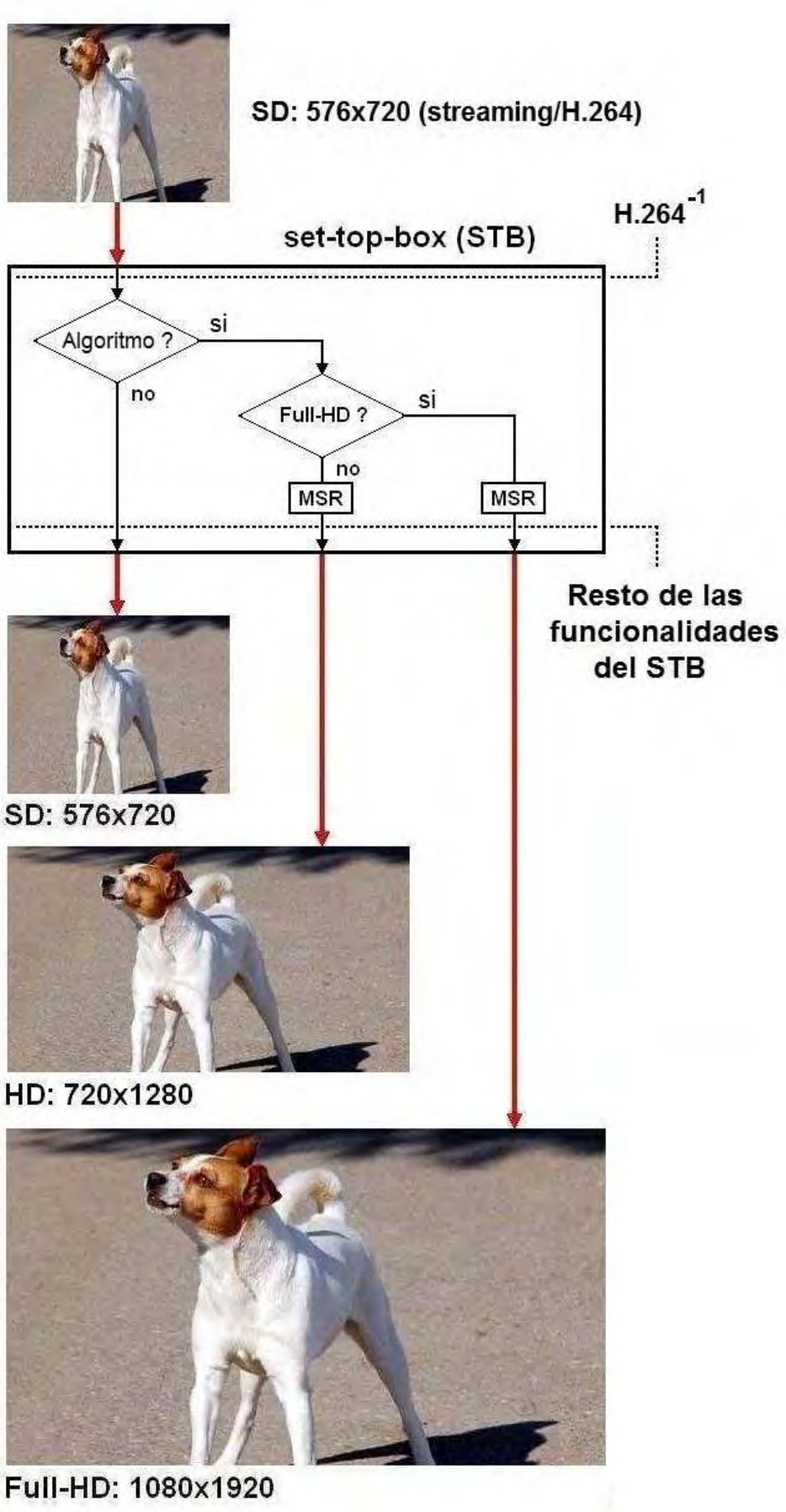


Figura 3.12: Decodificador.

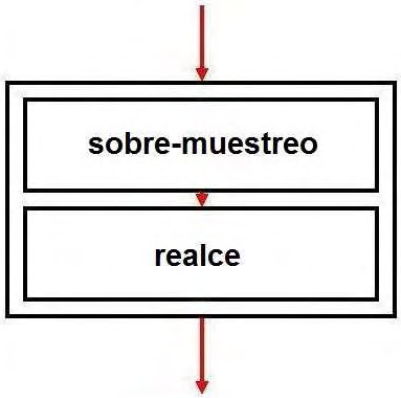


Figura 3.13: Módulo de Super-resolución (MSR).

La Fig.3.14 representa la implementación real de la Fig.3.12, en la cual, podemos ver, el set-top-box experimental usado, simulado en el interior de una CPU. Este STB trabaja igualmente para TV Digital Terrestre, IPTV, WebTV, 3DTV y Cine Digital, y lleva un sintonizador, el cual puede observarse en la imagen del centro de la Fig.3.14 junto con las placas Quadro. Actualmente, estamos trabajando en el diseño de un circuito integrado (chip) [302] para reemplazar las actuales GPGPUs del interior del STB (y por ende a la CPU misma), minimizando el consumo de energía y la geometría del mismo [294].

Figura 3.14: Set-top-box usado.

La segunda aplicación de esta tecnología presentada aquí se muestra en la Fig.3.15, en la cual usamos un teléfono celular con salida de video High-Definition Multimedia Interface (HDMI) como un receptor.

Como se muestra en la mencionada figura, tomamos la salida de video HDMI del teléfono celular y la conectamos al STB. Dependiendo de la resolución de la TV obtenemos resoluciones HD o Full-HD. En este caso, el teléfono celular hace también de sintonizador.

La resolución original de los teléfonos celulares *One-Seg* (uno de los 13 segmentos que forman la norma ISDB-T (de hecho, el 7mo., ver las Figuras 3.16 y 3.17) es Low Definition (LD) $320x240_p$ pixeles. En este caso, la tasa de compresión adicional que aporta esta tecnología sobre H.264 es 27:1 [294]. Podemos ver una referencia completa de ISDB-T y H.264 en el Apéndice F.

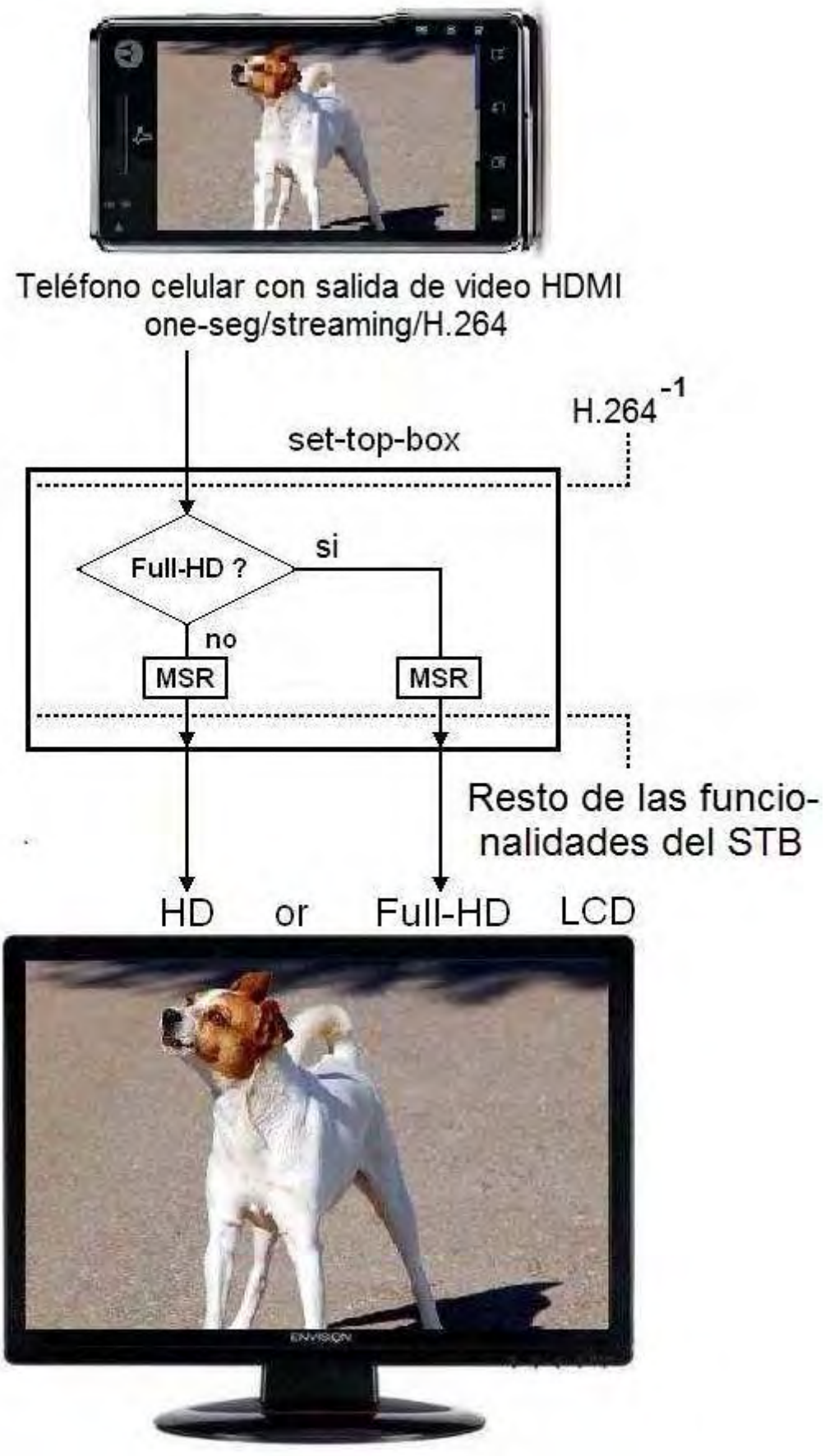


Figura 3.15: Teléfono celular como receptor HD o Full-HD.

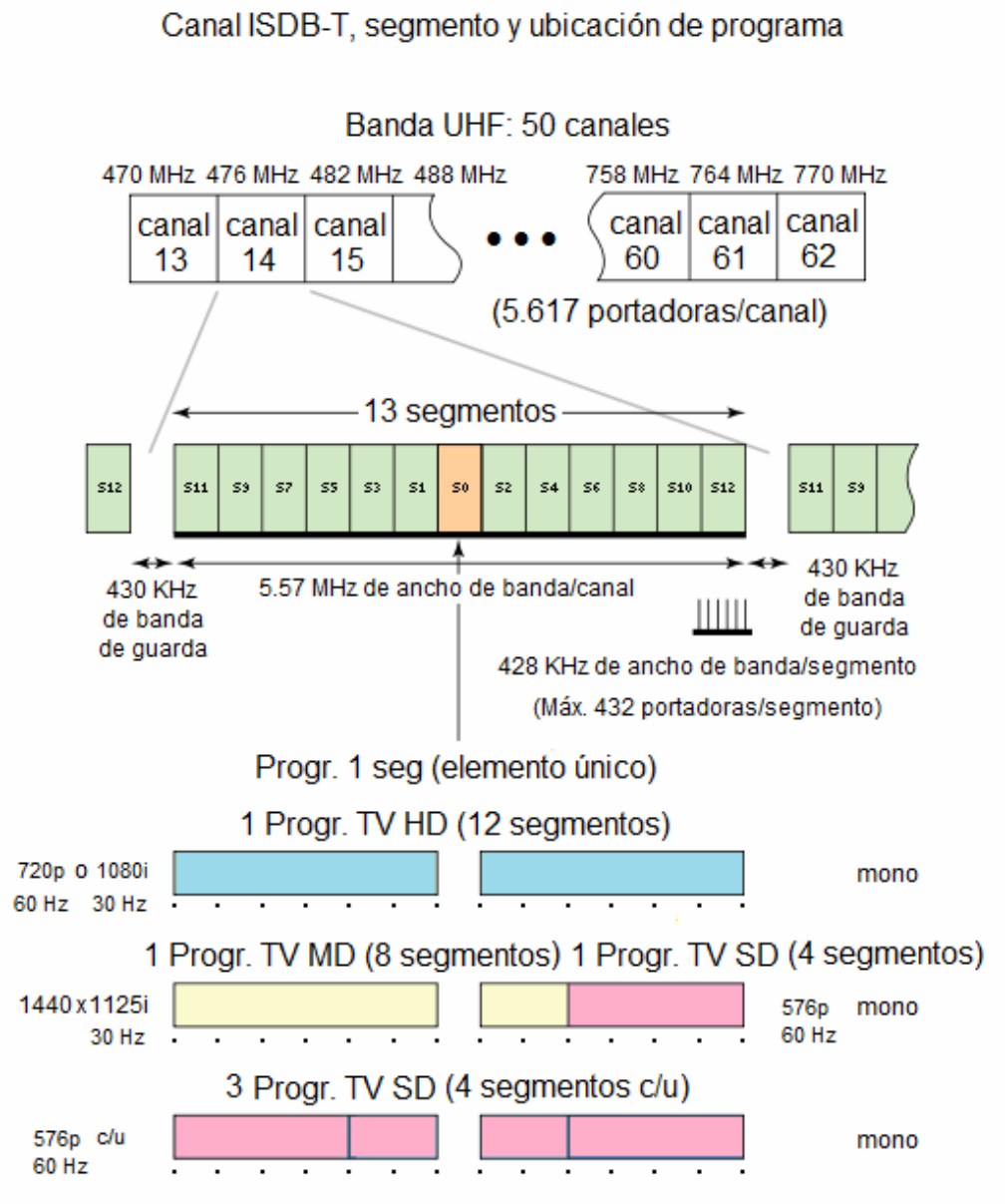


Figura 3.16: Actual organización de servicios en ISDB-T (Japón).

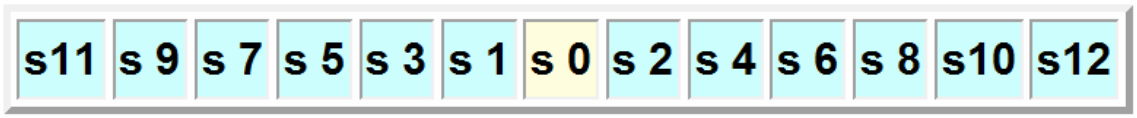


Figura 3.17: Detalle de ordenamiento de los 13 segmentos en ISDB-T.

La Fig.3.16 muestra la organización en 14 segmentos de cada canal de la norma japonesa ISDB-T, la cual consiste de 13 segmentos útiles, dado que desperdicia medio segmento por delante y por detrás de cada canal a los efectos de evitar solapamientos con un canal vecino. Los 13 segmentos útiles se organizan en el orden especificado en la Fig.3.17, donde el segmento central es conocido como *one-seg* y es el empleado en el experimento de la Fig.3.15, Apéndices H, I y J.

La Fig.3.16 muestra en colores la organización de servicios para esta norma:

1. En el nivel más alto y en celeste, un canal HD de 12 segmentos 720_p 0 1080_i y un *one-seg*.
2. En el nivel medio y en amarillo, un canal MD (Middle Definition) de 1440x1125_i más un SD 576_p en rosa.
3. En el nivel más bajo y en rosa, 3 canales SD.

Todos mono, es decir, TV Digital 2D.

Como se desprende de la Fig.3.18, la norma es independiente de la super-compresión. De esta figura tenemos que:
• los módulos en rojo representan SC (supercompresión) y SC^{-1} (superdescompresión)
• *DataStream*: Es el flujo de datos con la información interactiva a ser desplegada sobre la pantalla
• *IP Encap.*: Es el encapsulador de IP (Internet Protocol)
• *MUX*: Multiplexor, el cual mezcla o combina las señales de audio y video codificadas provenientes de la fuente + IP Encap. + DataStream.
• *Middleware*: Es el nombre del estándar abierto para el Sistema Brasilero de TV Digital (SBTVD), el cual recibe el nombre de Ginga. Este está compuesto de dos sub-sistemas: Ginga-NCL y Ginga-J (J de Java). Ginga-NCL es un entorno de presentación multimedia para aplicaciones declarativas escritas en NCL y su lenguaje de scripting LUA [303, 304]. Por otra parte, Ginga-J provee una infraestructura de ejecución de aplicaciones Java y extensiones específicamente adaptadas al ambiente de la TV.
• *IP Desencap.*: Es el desencapsulador de IP (Internet Protocol)
• *DEMUX*: Demultiplexor, el cual separa las señales de audio y video codificadas en la fuente, así como el IP Desencap. y el Middleware.

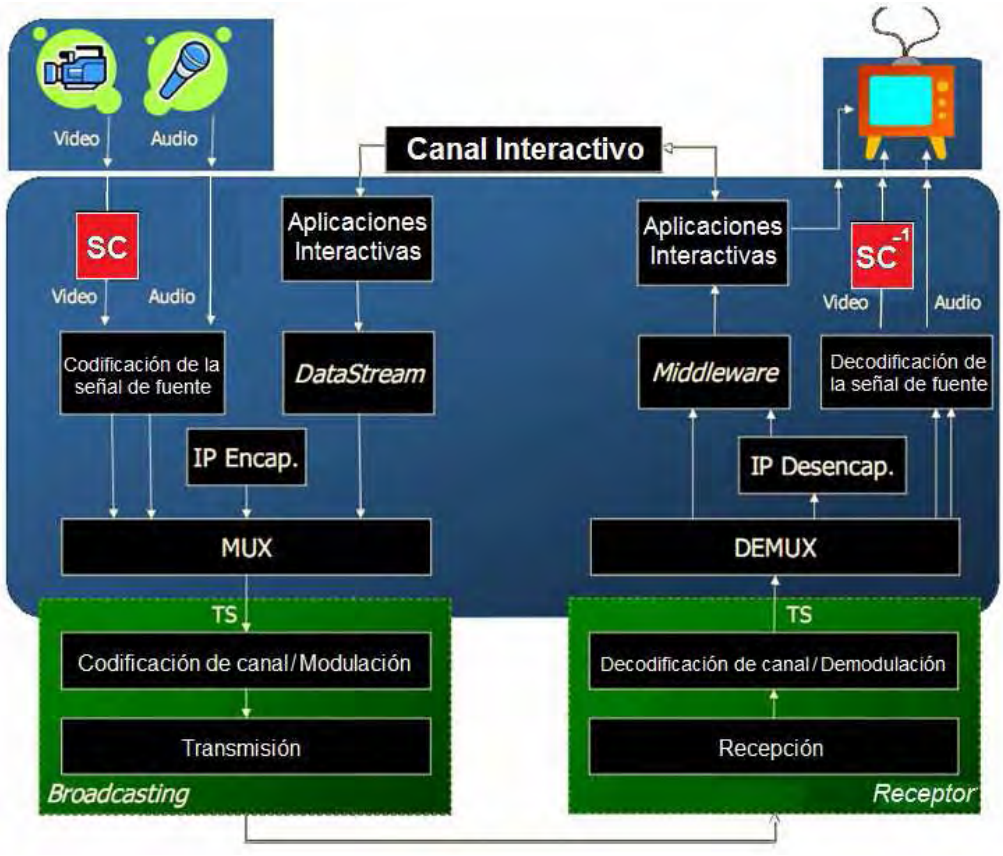


Figura 3.18: Sistema de transmisión/recepción, el cual incluye al canal interactivo, donde los módulos en rojo representan a la SC (supercompresión) y SC^{-1} (superdecompresión), respectivamente.

Como ya hemos mencionado recurrentemente, la supercompresión es transparente a la norma, razón por la cual, puede ser empleada en cualquiera de las normas de TV Digital actualmente en uso en el mundo, así como TV IP, Web TV, TV 3D y Cine Digital. A este respecto y cómo podemos apreciar en las Figuras 3.19 y 3.20, las resoluciones empleadas en Cine Digital son por mucho muy superiores a la Full-HD de TV Digital. A modo de ejemplo, la Norma Red EPIC 645 9K de Cine Digital tiene una resolución de 9334x7000 pixeles, y que en la Fig.3.19 está representada en color violeta, mientras que en la Fig.3.20 la Norma Red EPIC 617 28K de 28000x9334 pixeles está representada en color naranja. En ambas figuras la resolución Full-HD de TV Digital es la tercera empezando del margen superior izquierdo y está representado en celeste.

Como podemos apreciar, la diferencia de resolución es muy evidente, y dado que a diferencia de la TV Digital la cual es en tiempo real, el Cine Digital no es un proceso en tiempo real, no obstante, dados los formidables volúmenes de información involucrados en cada caso, no alcanza con un muy eficiente sistema de compresión para su almacenamiento y eventual transmisión a deshora para su posterior reproducción, sino que es imprescindible una compresión adicional que juegue con la resolución. Esto último es la supercompresión. Es por demás evidente que en la actualidad estas resoluciones solo se manejan mediante la modalidad *storage/no-broadcast* grabando las películas en cintas digitales de 6 cms de ancho, dado que si se deseara transmitirlas en tiempo real, solo se lo podría hacer de a una a la vez (monocanal) y mediante una red fotónica.

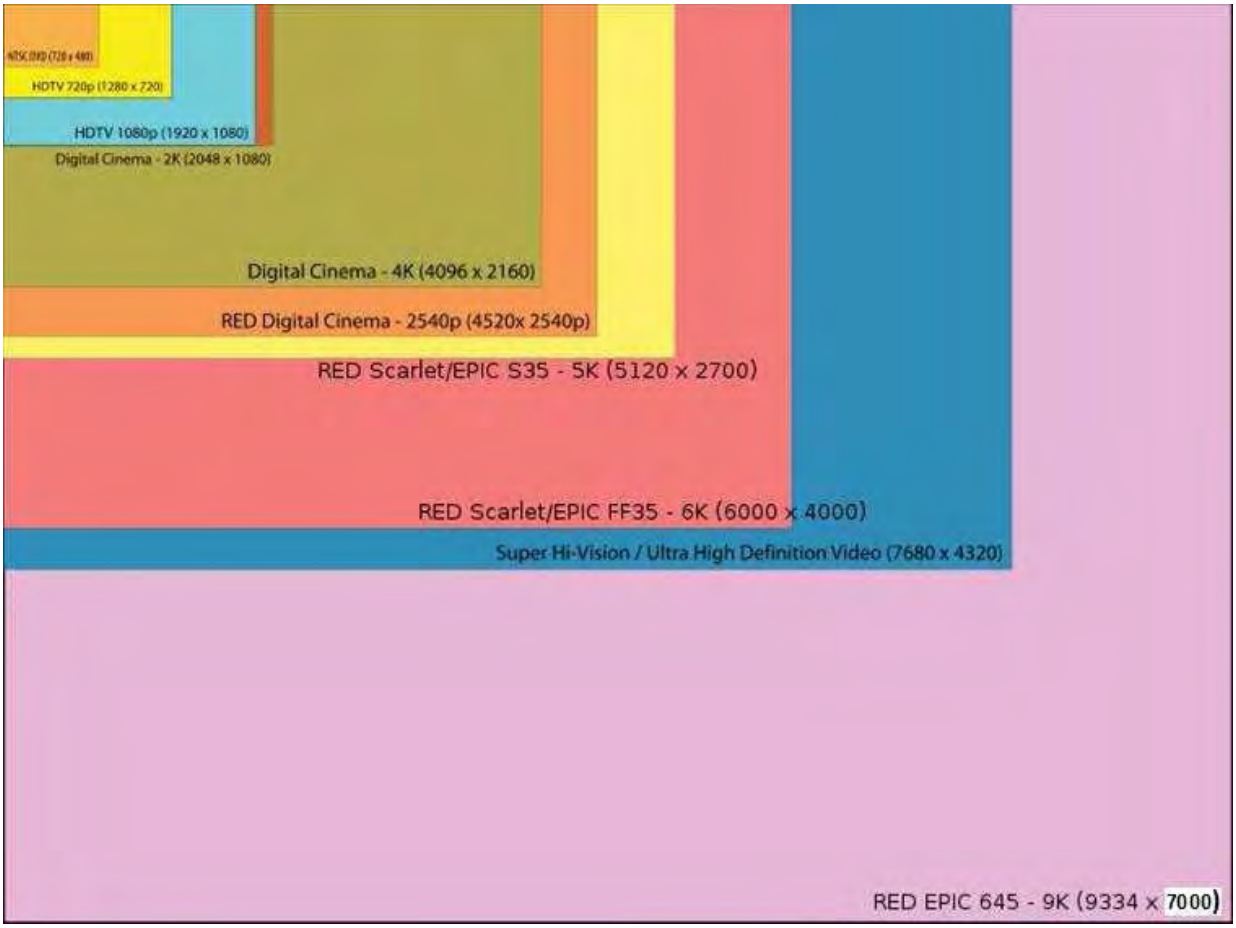


Figura 3.19: Resoluciones comparativas de TV Digital (la 3° empezando en el margen superior izquierdo en celeste) y Cine Digital, incluyendo la norma RED EPIC 645 – 9K (9334 x 7000 pixeles), la más grande en violeta.

Figura 3.20: Resoluciones comparativas de TV Digital (la 3° empezando en el margen superior izquierdo en celeste) y Cine Digital, incluyendo la norma RED EPIC 617 – 28K (28000 x 9334 pixeles), la más grande en naranja.

Para la tercera aplicación, debemos establecer previamente algunos lineamientos concernientes a la Televisión Digital en general y a la norma adoptada por la República Argentina en particular. No obstante, todos los conceptos serán exhaustivamente expuestos en el Apéndice F.

La Fig.3.21 muestra los posibles modelos de programa que permite la norma ISDB-Tb (Brasil y Argentina) teniendo como códec de video a H.264, en comparación con la ISDB-T (Japón) el cual tiene como códec de video a MPEG2, ver Apéndice F.

La Fig.3.21 muestra en colores la organización de servicios para esta norma:

1. En el nivel más alto y en celeste, dos canal HD de 6 segmentos c/u para 720_p 0 1080_i y un *one-seg*.

2. En el nivel más bajo y en amarillo, 4 canales SD.

Todos mono, es decir, TV Digital 2D. Es decir, esta norma tampoco permite TV Digital 3D, como así tampoco 1080_p .

La relación de aspecto para ambas normas vistas hasta el momento son:

a) 16:9 para HD y Full-HD

b) 4:3 para SD

Aunque en la actualidad, tanto en Argentina como en Brasil y al solo efecto de hacer entrar cada canal de HD en solo 6 segmentos de la norma ISDB-Tb, la relación de aspecto se la modifica a 14:9, razón por la cual se ven las imágenes tan deformadas en nuestro medio.

Por otra parte, tanto para la norma ISDB-T (Japón) como para la ISDB-Tb (Brasil y Argentina) cada canal ocupa un ancho de banda de 6 MHz, ver Apéndice F.

Otro aspecto importante a tener en cuenta, es que en la norma ISDB-Tb cada segmento tiene un bit-rate de aproximadamente 1.5 Mbits/seg.

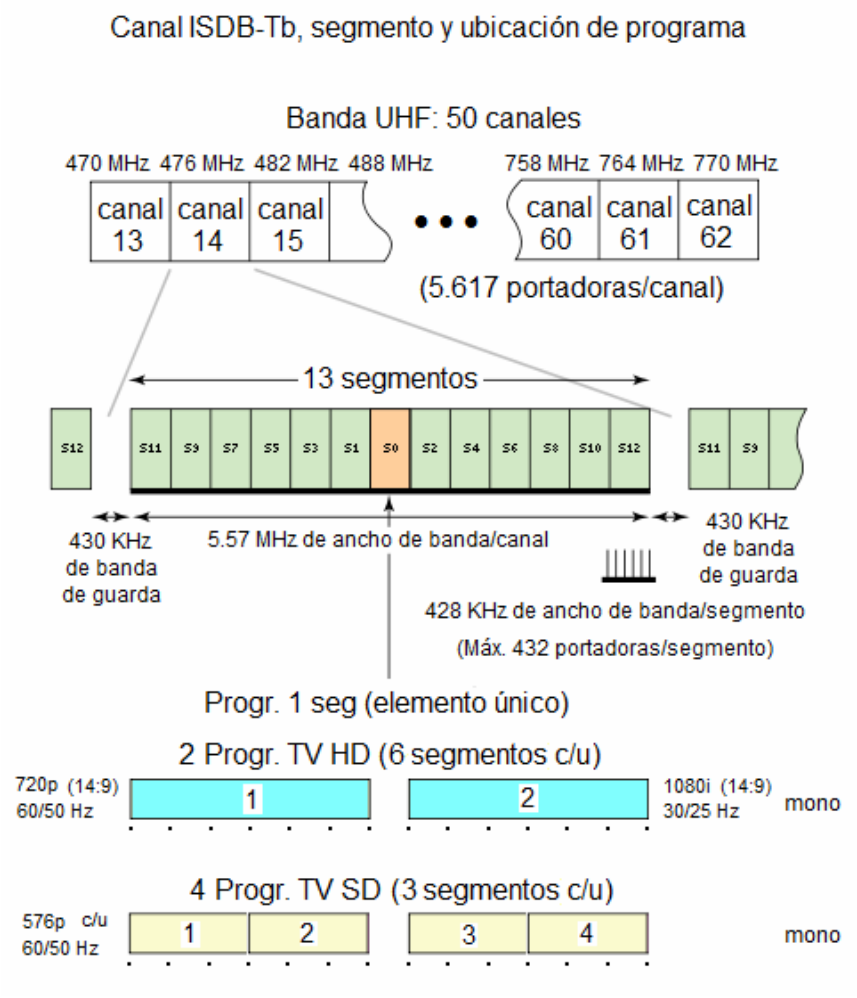


Figura 3.21: Actual organización de servicios en ISDB-Tb (Brasil y Argentina).

Establecidos los puntos anteriores, la tercera aplicación consiste en utilizar la super-compresión para transmitir por primera vez en la historia de la TV Digital en general y la norma ISDB-Tb en particular en 1920 x 1080p stereo, es decir, 1080_p 3D. Para dicho experimento, seguimos los siguientes pasos de la Fig.3.22:

1. *En el codificador*:
   1.1. Obtenemos imágenes de una cámara de resolución 1080p con salida HD-SDI doble, debido a sus dos lentes (izquierdo y derecho),
   1.2. Se practica un sub-muestreo en ambas pasando c/u de una resolución de 1920x1080p a 960x540p
   1.3. Se las une conformando una única imagen de (960+960)x540p
   1.4 Se las codifica en H.264 y se las multiplexa con audio y datos para su posterior transmisión

2. *En el decodificador*:
   2.1. Se las recibe, demultiplexa y decodifica con la inversa de H.264.
   2.2. Se las separa (splitting) en dós imágenes, cada una de 960x540p
   2.3. Se practica un sobre-muestreo en ambas pasando a 1920x1080p
   2.5. Se aplica la máscara de restauración de nitidez a ambas
   2.6. Se exhiben en un TV-3D-2K, es decir, $1920x1080_p$ stereo.

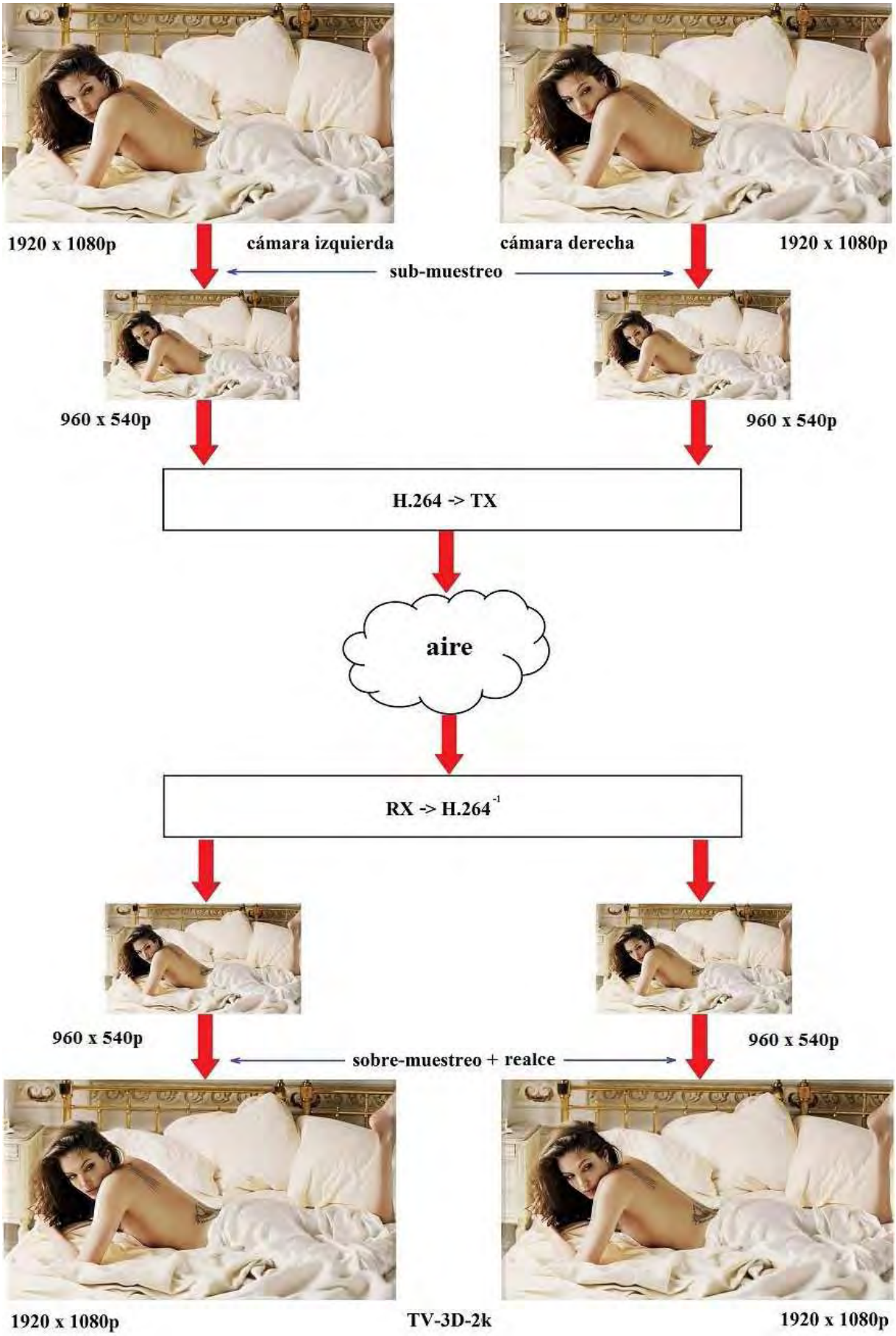


Figura 3.22: Transmisión de 1920x1080p-3D (stereo) mediante H.264.

La Fig.3.23 muestra el diagrama del codificador del experimento real llevado a cabo en las instalaciones de Canal 7 y bajo la supervisión de las autoridades de entidad reguladora de medios, Radio Televisión Argentina (RTA). Desde el MUX de Canal 7 a la antena transmisora se envían las imágenes super-comprimidas por fibra óptica.

Figura 3.23: Captura, super-compresión, codificación, multiplexado y transmisión.

Mientras en la Fig.3.23 observamos en detalles los componentes del codificador, en orden: la cámara 1080p 3D, la combinación CPU+GPGPUs (captura, procesamiento e interfaz), encoder y MUX y finalmente enlace de fibra óptica hasta el lugar de transmisión; la Fig.3.24 muestra el detalle del decodificador, en correcta secuencia: la antena (arreglo planar), el set-top-box, la CPU+GPU (con la super-descompresión) y el TV 3D. Con el desarrollo del chip dedicado de super-descompresión, la CPU y el set-top-box se fusionarán en un único y nuevo set-top-box.

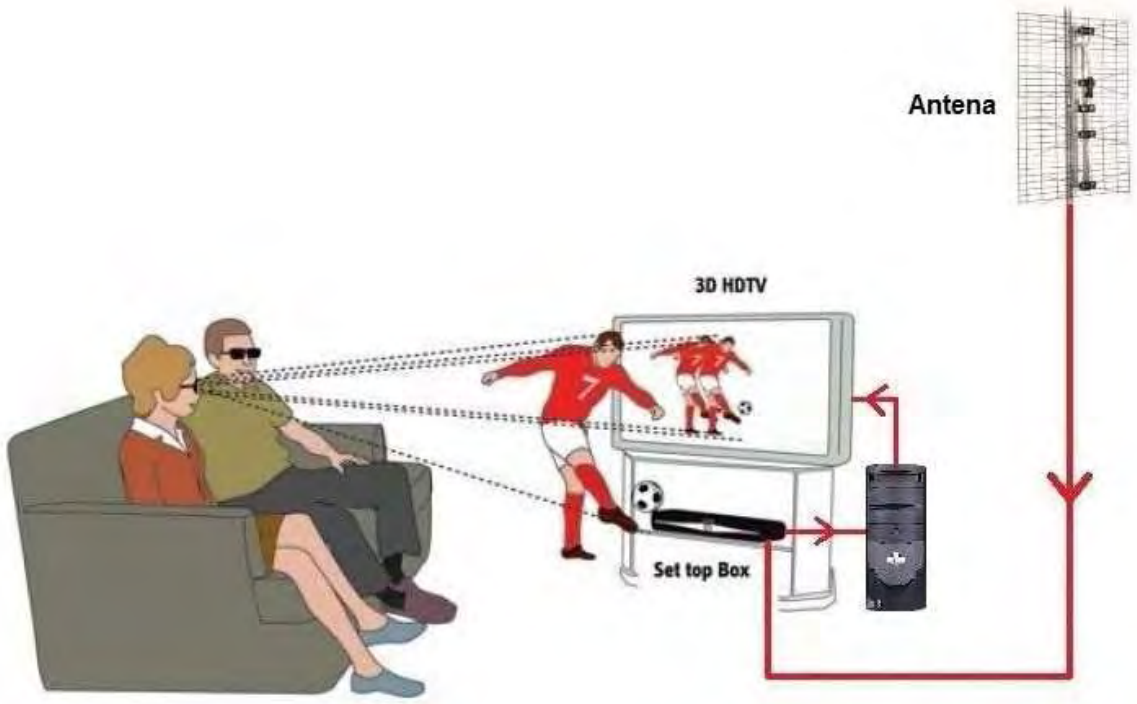


Figura 3.24: Recepción, decodificación, super-descompresión y visualización.

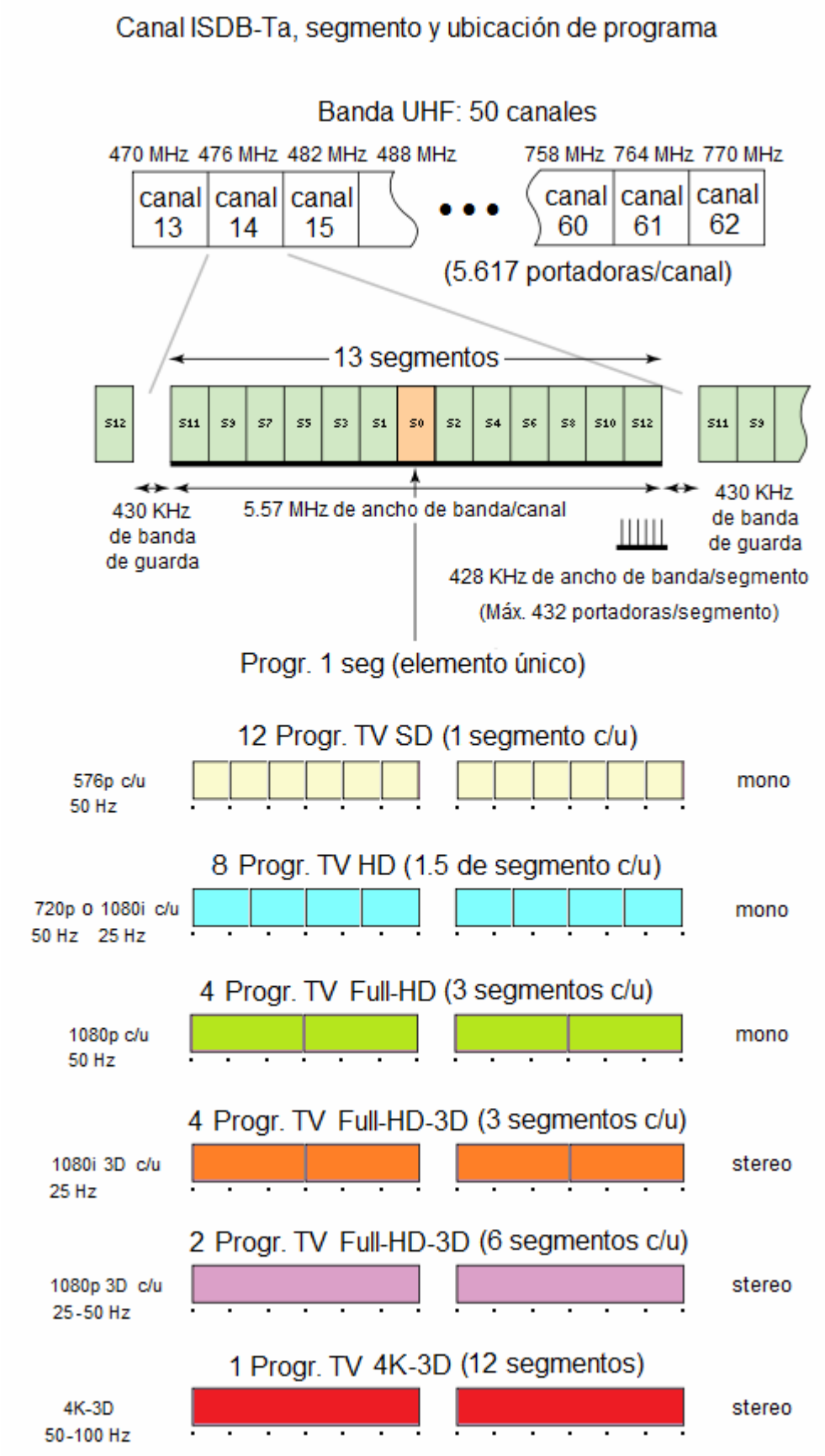


Figura 3.25: Empleando seis segmentos para cada transmisión 1080p stereo y a la vez ahorrando una enorme porción del espectro radioeléctrico (ver parte baja de la figura en violeta), así como pasando de 4 576p a 12 576p y de 2 a 8 1080i/720p.

Este experimento da lugar, primero a una enmienda de la norma ISDB-Tb y posteriormente al nacimiento de una nueva norma conocida como ISDB-Ta. A tal efecto, y al pie de la Fig.3.25 podemos observar la ocupancia de segmentos por parte de las componentes del experimento descripto. Como puede observarse, cada canal 1080p 3D solo ocupa 6 segmentos, pudiendo haber dos, así como 4 de 1080_i 3D, 4 de 1080_p , 8 de 720_p o 1080_i y 12 de 576_p y aun así permitiendo en todos los casos un *one-seg*.

Otro aspecto de importancia no menor a la hora de analizar comparativamente ISDB-Tb e ISDB-Ta es la cantidad de servicios alternativos a la que da lugar la segunda con respecto a la primera con solo observar las Figuras 3.21 y 3.25. Esto es de capital importancia para todos aquellos responsables de generar contenidos para TV Digital, los cuales encontrarán en la tecnología propuesta alternativas que ni siquiera están previstas en la norma ISDB-T y su mejora brasilera, es decir, ISDB-Tb. Como hemos mencionado, la norma ISDB-Tb no solo contempla la posibilidad de 1080p 3D, sino ni siquiera 1080p mono.

La Fig.3.26 hace referencia a un experimento programado conjuntamente con el Ministerio de Planificación Federal, Ingreso Público y Servicios y la UNTreF [294], y mediante el cual las dos señales de salida de una cámara 4k-3D, es decir, 3840x2160p stereo, son sub-muestreadas a 1920x1080p cada una, y posteriormente codificada cuadro a cuadro mediante el algoritmo JPEG2000 (ver Apéndice B). En otras palabras, para este tipo de resoluciones, al igual que 8k-3D (7680x4320p) no existe a la fecha ningún algoritmo de compresión inter-cuadro, solo se practica la compresión intra-cuadro, y solo se transmite por redes fotónicas, es decir, fibras ópticas de 100 MBPS de ancho de banda troncales [305].

Para este experimento, se generan expectativas lógicas respecto de la utilización de la super-compresión, en virtud de la necesidad de Brasil de transmitir su mundial de futbol 2014 y sus olimpíadas 2016 sobre una red fotónica idéntica a "Argentina Conectada" y conocida como "Brasil Conectado". Ver Apéndices H, I y J.

Ambas super-redes, es decir, Argentina Conectada" y "Brasil Conectado" se tocarán en el mes de Septiembre de 2011 a la altura de la Provincia de Misiones en Argentina, a efectos de que en un futuro se vehiculicen libremente servicios del tipo *video-on-demand*, cine digital, TV Digital de alta resolución y 4K-3D con motivo de los eventos anteriormente mencionados. El problema reside en que si Brasil intenta transmitir dichos eventos por estas redes, solo lo podrá lograr a través de muy pocos canales alternativos, incluso de programación alternativa a las imágenes del mundial mismo, dado el desmedido volumen de información que implican las ultra altas definiciones del tipo 4K-3D. Por esta razón, la super-compresión se presenta como una alternativa factible a los efectos de un apropiado y práctico empleo de dichas super-redes.

Otro aspecto importante a tener en cuenta y que distingue a la super-compresión como una solución viable, consiste en que si deseáramos transmitir 4K-3D via la modalidad TV Digital Terrestre (TDT), se lo podría hacer empleando el mismo esquema de la Fig.3.22, dado que la super-compresión resultó ser – a partir de todos los experimentos realizados – absolutamente lineal respecto de la resolución y la cantidad de cuadros por segundo (en inglés, *frames-per-second*: FPS). En otras palabras, si en lugar de usar el esquema de la Fig.3.22 para transmitir 1080_p 3D a 25/50 FPS, se lo usa para transmitir 3840x2160p 3D a 50/100 FPS (ocupando en este caso 12 segmentos de la norma) solo hay que multiplicar la capacidad de cómputo de las placas GPGPUs empleadas, es decir, usar 4 veces más buffering y procesamiento. No obstante, y por una cuestión

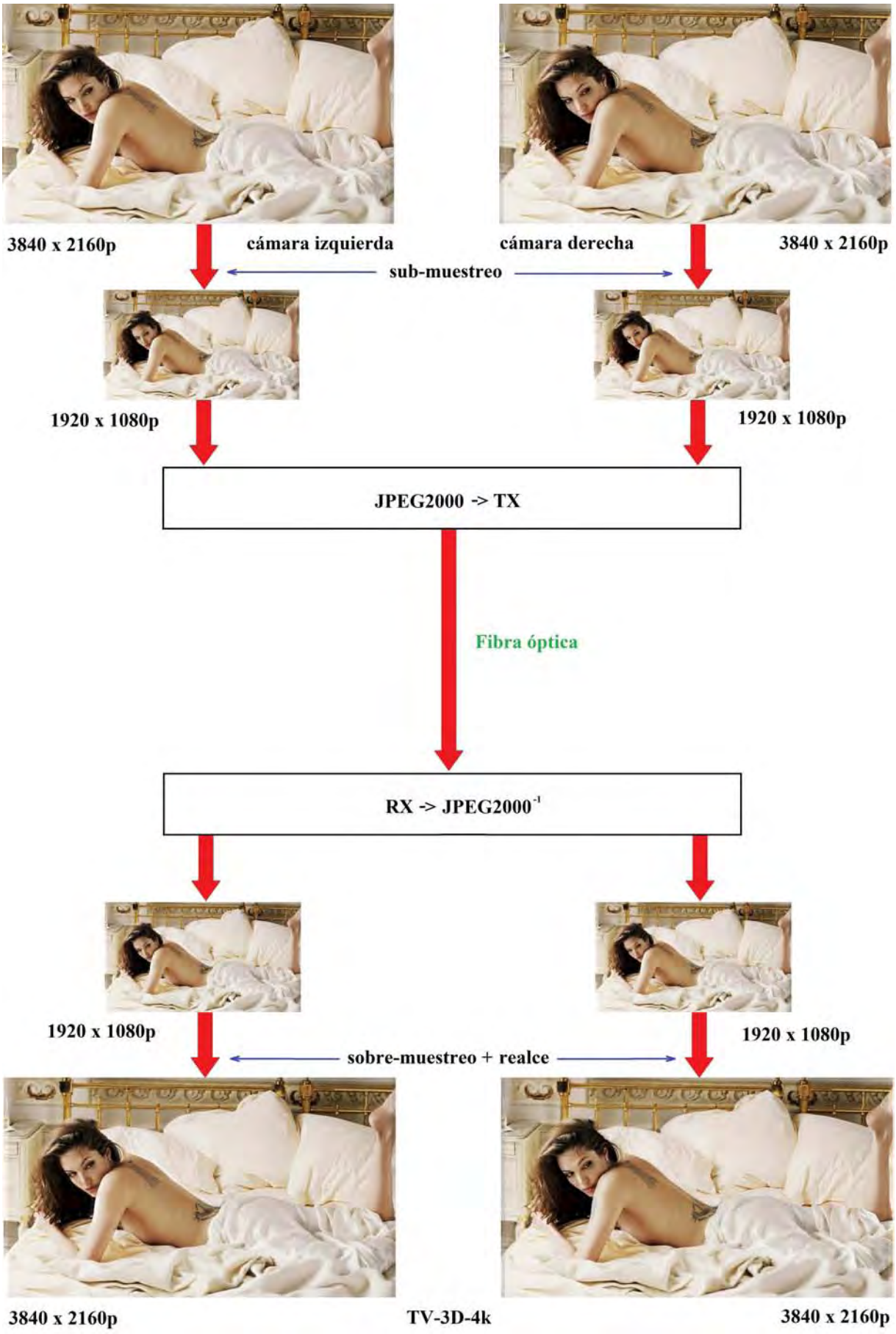


Figura 3.26: Transmisión de 4k-3D mediante JPEG2000 sobre fibra óptica.

de espacio y cantidad de slots PCI Express de las mothers en las CPUs anfitrionas para las GPGPUs Quadro, el esquema de la Fig.3.14 no es una alternativa válida. Esto sucede pues en el mejor de los casos dichos slots PCI Express llegan a 7, no obstante por su ancho estas placas invaden espacio más allá del slot que ocupan. Por lo tanto, deberemos pasar a una configuración rackeable. De esta manera, con placas del tipo Quadro rackeable podremos apilar casi sin límites las mismas para atender a las resoluciones y FPS involucrados. Un aspecto importante a tener en cuenta es que para este tipo de configuración requerimos conexiones del tipo infiniband.

3.2 Conclusiones del capítulo

Este capítulo se puso de relieve la diferencia fundamental entre compresión y supercompresión, su compatibilidad con esquemas de compresión en uso y sus fundamentales aplicaciones. Por otra parte, este capítulo es el anfitrión del principal aporte de la presente tesis.

Capítulo 4

Métricas y evaluación de performance de reconstrucción

Este capítulo atañe a las simulaciones comparativas entre compresión y supercompresión, realizadas primero en MATLAB® y Jacket® en una máquina de 4 procesadores y luego en CUDA y OpenCL sobre placas GPGPUs. Al final de este capítulo se detallarán tales implementaciones.

4.1 Métricas

4.1.1 Tasa de Compresión de Datos (en inglés: Data Compression Ratio, o CR)

La tasa de compresión de datos, también conocida como potencia de compresión, es un término propio de las Ciencias de la Computación usado para cuantificar la reducción en la cantidad de representación de los datos producida por un algoritmo de compresión de datos. La tasa de compresión de datos es análoga a la tasa de compresión física usada para medir la compresión física de sustancias, y se define de la misma manera, como la tasa entre la cantidad sin comprimir y la cantidad comprimida [2-9]:

$$CR = \frac{Cantidad\ sin\ comprimir}{Cantidad\ comprimida} \tag{4.1}$$

Entonces, un algoritmo que comprime un archivo de 10MB a 2MB tiene una tasa de compresión de 10/2 = 5, frecuentemente citado como una tasa explícita, 5:1 (léase "cinco a uno"), o como una tasa implícita, 5X. Observemos que esta formulación se aplica igualmente para compresión, donde la cantidad sin comprimir se corresponde con la imagen original; y para la descompresión, donde la cantidad sin comprimir se corresponde con la imagen recuperada.

4.1.2 Bit-por-pixel (en inglés, bit-per-pixel: bpp)

El término "bits por pixel" se refiere a la suma de los bits en los tres canales de color y representa la suma de colores disponibles en cada pixel antes de la compresión (bpp_{ac}). No obstante, como una métrica para compresión, los bits-por-pixel se refieren al promedio de los bits en los tres canales de color, luego del proceso de compresión (bpp_{dc}).

$$bpp_{dc} = \frac{Cantidad\ comprimida}{Cantidad\ descomprimida} \times bpp_{ac} = \frac{bpp_{ac}}{CR} \tag{4.2}$$

Ademas, bpp se define también como

$$bpp_{dc} = \frac{Número\ de\ bits\ codificados}{Número\ total\ de\ pixeles} \tag{4.3}$$

4.1.3 Error Medio Absoluto (en inglés, Mean Absolute Error: MAE)

El error medio absoluto es una cantidad usada para medir que tan próximos se encuentran los pronósticos o predicciones de las salidas eventuales. El error medio absoluto (MAE) está dado por:

$$MAE = \frac{1}{NRxNC} \sum_{nr=0}^{NR-1} \sum_{nc=0}^{NC-1} \left| \boldsymbol{X}(nr,nc) - \hat{\boldsymbol{X}}(nr,nc) \right| \tag{4.4}$$

La cual para dos imágenes grises $\boldsymbol{X}$ y $\hat{\boldsymbol{X}}$ de dimensiones $NR \times NC$, donde la segunda de las imágenes es considerada una aproximación descomprimida de la primera.

4.1.4 Error Cuadrático Medio (en inglés, Mean Squared Error: MSE)

El error cuadrático medio o MSE es una de las muchas formas de cuantificar la diferencia entre una imagen original y el valor real de la cantidad de la imagen descomprimida, la cual para dos imágenes monocromáticas de $NR \times NC$ (filas-por-columnas) $\boldsymbol{X}$ y $\hat{X}$, donde la segunda de las imágenes es considerada la aproximación descomprimida de la primera se define como:

$$MSE = \frac{1}{NRxNC} \sum_{nr=0}^{NR-1} \sum_{nc=0}^{NC-1} \left(\boldsymbol{X}(nr,nc) - \hat{\boldsymbol{X}}(nr,nc) \right)^2 \tag{4.5}$$

4.1.5 Tasa Señal-a-Ruido Pico (en inglés, Peak Signal-To-Noise Ratio: PSNR)

La frase tasa señal-a-ruido pico, frecuentemente abreviada PSNR, es un término de la Ingeniería para la tasa entre la potencia máxima posible de una señal y la potencia del ruido que afecta la fidelidad de la representación de la misma. Dado que muchas señales tienen un muy amplio rango dinámico, PSNR es usualmente expresado en función de la escala logarítmica en decibeles. La PSNR es más empleada como una medida de la calidad de la reconstrucción en compresión de imágenes, ver [2]. Se define más fácilmente via el error cuadrático medio (MSE), de forma tal que, PSNR se define como [3-9]:

$$PSNR = 10 \log_{10}\left(\frac{MAX_X^2}{MSE}\right) = 20 \log_{10}\left(\frac{MAX_X}{\sqrt{MSE}}\right) \tag{4.6}$$

Aquí, MAX_X es el valor máximo de la imagen. Donde los pixeles están representados usando 8 bits por muestra, esto es 256. Más generalmente, cuando las muestras están representadas usando modulación por codificación de pulso lineal (en inglés, linear pulse code modulation: PCM) con B bits por muestra (bps), con una valor máximo posible de $MAX_X = 2^B - 1$.

Para imágenes color con tres valores por pixel rojo-verde-azul (en inglés, red-green-blue: RGB), la definición de PSNR es la misma excepto que el MSE es la suma sobre todas las diferencias cuadráticas divididas por la dimensión de la imagen y por 3 [2-9].

$$MSE = \frac{1}{NRxNCx3} \sum_{nr=0}^{NR-1} \sum_{nc=0}^{NC-1} \sum_{col=0}^{2} \left(\boldsymbol{X}(nr,nc,col) - \hat{\boldsymbol{X}}(nr,nc,col) \right)^2 \tag{4.7}$$

Donde *col* representa a la componente de color *rojo* para 0, a la *verde* para 1 y a la *azul* para 2. Los 3 de 8 bpp c/u, aunque en TV Digital pueden ser también de 10 bpp c/u. Ver Apéndice F.

Los valores típicos para el PSNR en compresión de imágenes y video con pérdidas están entre 30 y 50 dB, donde, cuanto más alto, mejor.

4.2 Evaluaciones de Calidad y Comparaciones

Las simulaciones son organizadas en once experimentos, sobre imágenes fijas (por razones de claridad en la presentación, no obstante, idénticos resultados fueron obtenidos en video, HDTV y Cine Digital) separadas en dos grupos: siete en color y cuatro en grises. Todos los experimentos incluyen el cálculo de MAE, MSE, PSNR, bpp y CR, e involucran la comparación entre el uso de JPEG vs SC (JPEG+SR), y JPEG2000 vs SC (JPEG2000+SR) para imágenes fijas a color y grises, en ambos casos sobre un archivo de formato gráfico BMP (el cual no tiene compresión), y donde los acrónimos usados significan:

BMP: BitMap file format [23]
JPEG: Joint Picture Group [17-24]
JPEG2000: JPEG with wavelets [306-318]
SC: Super-compression
SR: Super-resolution

4.2.1 Grupo color

Todos estos archivos son de 24 bpp y sin compresión alguna, siendo sus características identificatorias más sobresalientes:

Archivo = angelina.bmp
Tamaño = 1920-por-1080 pixeles

Archivo = flor.bmp
Tamaño = 502-por-502 pixeles

Archivo = texto.bmp
Tamaño = 256-por-256 pixeles

Archivo = tomografía.bmp (un cuadro de una tomografía seriada en corte oxiaxial)
Tamaño = 512-por-512 pixeles

Archivo = quickbird-1.bmp (satélite estadounidense de muy alta resolución)
Tamaño = 4096-por-4096 pixeles

Archivo = eros-b.bmp (satélite israelí de muy alta resolución)
Tamaño = 600-por-600 pixeles

Archivo = quickbird-2.bmp
Tamaño = 1400-por-1442 pixeles

4.2.2 Grupo gris

Todos estos archivos son de 8 bpp y sin compresión alguna, siendo sus características identificatorias más sobresalientes:

Archivo = lena.bmp
Tamaño = 512-por-512 pixeles

Archivo = fingerprint.bmp
Tamaño = 542-por-542 pixeles

Archivo = rodilla.bmp
Tamaño = 512-por-512 pixeles

Archivo = sar.bmp (imagen de Radar de Apertura Sintética: SAR por sus siglas en inglés)
Tamaño = 540-por-552 pixeles

4.3 Simulaciones

En todos los experimentos se comparan primero las compresiones del archivo en formato BMP mediante el algoritmo JPEG, el cual implica en orden:

a) compresión mediante el algoritmo JPEG
b) almacenamiento o transmisión por el canal
c) descompresión mediante el algoritmo inverso al JPEG

vs su compresión mediante Supercompresión (SC = JPEG+SR, donde SR significa Super-Resolución), la cual implica en orden:

a) el sub-muestreo
b) compresión mediante el algoritmo JPEG
c) almacenamiento o transmisión por el canal
d) descompresión mediante el algoritmo inverso al JPEG
e) el sobre-muestreo
f) restauración de la nitidez

Y luego las compresiones del archivo en formato BMP mediante el algoritmo JPEG2000, el cual implica en orden:

d) compresión mediante el algoritmo JPEG2000
e) almacenamiento o transmisión por el canal
f) descompresión mediante el algoritmo inverso al JPEG2000

vs su compresión mediante Supercompresión (SC = JPEG2000+SR, donde SR significa Super-Resolución), la cual implica en orden:

g) el sub-muestreo
h) compresión mediante el algoritmo JPEG2000
i) almacenamiento o transmisión por el canal
j) descompresión mediante el algoritmo inverso al JPEG2000
k) el sobre-muestreo
l) restauración de la nitidez

4.3.1 Grupo color

4.3.1.1 Archivo angelina.bmp

4.3.1.1.1 JPEG vs SC (JPEG+SR)

JPEG: Ver Tabla 4.1, columna JPEG, y Fig.4.1 (2da desde arriba).

Codificador:
1. De BMP (24 bpp, 1920x1080)
2. A JPEG (0.6853 bpp, 1920x1080)

Canal/almacenamiento

Decodificador:
1. De JPEG (0.6853 bpp, 1920x1080)
2. A BMP (24 bpp, 1920x1080)

SC (JPEG+SR): Ver Tabla 4.1, columna SC (JPEG+SR), y Fig.4.1 (3ra desde arriba).

Codificador:
1. BMP (24 bpp, 1920x1080)
2. Sub-muestreo (24 bpp, 720x576)
3. JPEG (0.1445 bpp, 720x576)

Canal/almacenamiento

Decodificador:
1. JPEG (0.1445 bpp, 720x576)
2. Sobre-muestreo (0.4323 bpp, 1920x1080)
3. Deblurring (0.5004 bpp, 1920x1080)
4. BMP (24 bpp, 1920x1080)

4.3.1.1.2 JPEG2000 vs SC (JPEG2000+SR)

JPEG2000: Ver Tabla 4.2, columna JPEG2000, y Fig.4.1 (4ta desde arriba).

Codificador:
1. De BMP (24 bpp, 1920x1080)
2. A JPEG2000 (2.6285 bpp, 1920x1080)

Canal/almacenamiento

Decodificador:
1. De JPEG2000 (2.6285 bpp, 1920x1080)
2. A BMP (24 bpp, 1920x1080)

SC (JPEG2000+SR): Ver Tabla 4.2, columna SC (JPEG2000+SR), y Fig.4.1 (abajo).

Codificador:
1. BMP (24 bpp, 1920x1080)
2. Sub-muestreo (24 bpp, 720x576)
3. JPEG2000 (0.8148 bpp, 720x576)

Canal/almacenamiento

Decodificador:
1. JPEG2000 (0.8148 bpp, 720x576)
2. Sobre-muestreo (1.3903 bpp, 1920x1080)
3. Deblurring (2.2397 bpp, 1920x1080)
4. BMP (24 bpp, 1920x1080)

Las siguientes tablas muestran las métricas vs los Algoritmos para ambos casos, i.e., JPEG y JPEG2000 vs Supercompresión.

TABLA 4.1
ANGELINA (COLOR, 24 BPP, 1920X1080): JPEG VS SC (JPEG+SR)

Métricas	JPEG	SC (JPEG+SR)
MAE	0.5333	1.0009
MSE	2.3137	7.6264
PSNR	43.6693	38.2393
bpp	0.6853	0.1445
CR	35.0210	166.1154

TABLA 4.2
ANGELINA (COLOR, 24 BPP, 1920X1080): JPEG2000 VS SC (JPEG2000+SR)

Métricas	JPEG2000	SC (JPEG2000+SR)
MAE	0.0446	0.2961
MSE	0.0472	1.1385
PSNR	61.3884	47.5673
bpp	2.6285	0.8148
CR	9.1307	29.4538

4.3.1.1.3 Interpretación de los resultados

La Fig.4.1 (arriba) nos muestra la imagen original de **Angelina**, la segunda muestra la imagen recuperada del proceso de codificado y decodificado con JPEG, mientras que la tercera hace lo propio con JPEG+SC. Como puede observarse, no existen diferencias visuales entre las tres primeras imágenes considerando – no obstante – una diferencia importante en las métricas cuantitativas de la Tabla 4.1. El segundo paquete de simulaciones con respecto a **Angelina** lo representan la cuarta y quinta imagen de la Fig.4.1, las cuales representan a la codificación y decodificación con JPEG2000, y lo mismo pero para la codificación y decodificación con JPEG2000+SC, respectivamente. La Tabla 4.2 muestra cambios notables en las métricas comparativas de calidad de ambos ejemplos. Más allá de las mencionadas diferencias en las tablas mencionadas, un análisis del tipo *doble-ciego* con 44 participantes arrojó que los respectivos aciertos acerca de la selección entre la imagen original y la tercera y la original y la quinta, dieron los mismos porcentajes de acierto que entre la original y la segunda, y la original y la cuarta, es decir, aproximadamente el 50 %, o sea, el mismo porcentaje que estar adivinando, ergo, no hay diferencia visual.

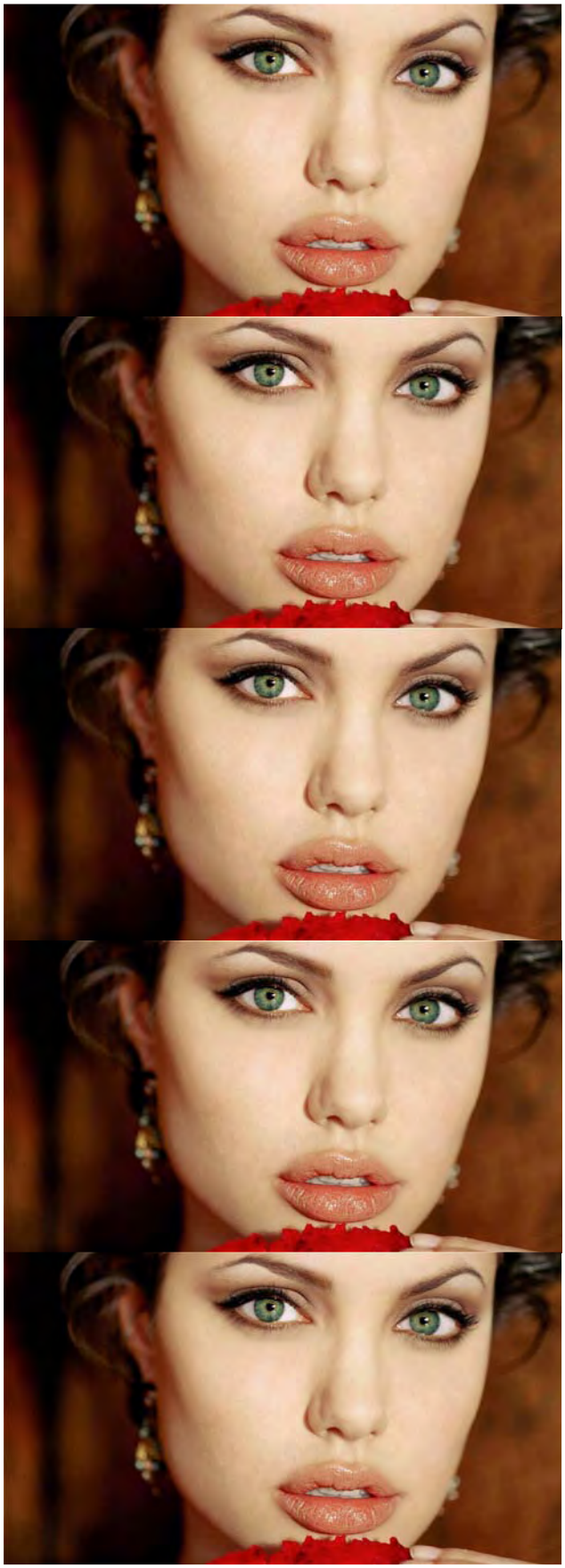


Figura 4.1: Primero (arriba) imagen original, segundo (codificado y decodificado con JPEG), tercero (codificado y decodificado con JPEG+Supercompresión), cuarto (codificado y decodificado con JPEG2000), quinto (abajo, codificado y decodificado con JPEG2000+Supercompresión).

4.3.1.2 Archivo flor.bmp

4.3.1.2.1 JPEG vs SC (JPEG+SR)

JPEG: Ver Tabla 4.3, columna JPEG, y Fig.4.2 (2da desde arriba).

Codificador:
1. De BMP (24 bpp, 502x502)
2. A JPEG (0.6777 bpp, 502x502)

Canal/almacenamiento

Decodificador:
1. De JPEG (0.6777 bpp, 502x502)
2. A BMP (24 bpp, 502x502)

SC (JPEG+SR): Ver Tabla 4.3, columna SC (JPEG+SR), y Fig.4.2 (3ra desde arriba).

Codificador:
1. BMP (24 bpp, 502x502)
2. Sub-muestreo (24 bpp, 251x251)
3. JPEG (0.1398 bpp, 251x251)

Canal/almacenamiento

Decodificador:
1. JPEG (0.1398 bpp, 251x251)
2. Sobre-muestreo (0.2793 bpp, 502x502)
3. Deblurring (0.4504 bpp, 502x502)
4. BMP (24 bpp, 502x502)

4.3.1.2.2 JPEG2000 vs SC (JPEG2000+SR)

JPEG2000: Ver Tabla 4.4, columna JPEG2000, y Fig.4.2 (4ta desde arriba).

Codificador:
1. De BMP (24 bpp, 502x502)
2. A JPEG2000 (2.6471 bpp, 502x502)

Canal/almacenamiento

Decodificador:
1. De JPEG2000 (2.6471 bpp, 502x502)
2. A BMP (24 bpp, 502x502)

SC (JPEG2000+SR): Ver Tabla 4.4, columna SC (JPEG2000+SR), y Fig.4.2 (abajo).

Codificador:

4. BMP (24 bpp, 502x502)
5. Sub-muestreo (24 bpp, 251x251)
6. JPEG2000 (0.9214 bpp, 251x251)

Canal/almacenamiento

Decodificador:

4. JPEG2000 (0.9214 bpp, 251x251)
5. Sobre-muestreo (1.8416 bpp, 502x502)
6. Deblurring (2.1954 bpp, 502x502)
4. BMP (24 bpp, 502x502)

Las siguientes tablas muestran las métricas vs los Algoritmos para ambos casos, i.e., JPEG y JPEG2000 vs Supercompresión.

TABLA 4.3
FLOR (COLOR, 24 BPP, 502X502): JPEG VS SC (JPEG+SR)

Métricas	JPEG	SC (JPEG+SR)
MAE	0.49123	0.7564
MSE	2.2895	7.5144
PSNR	43.2134	38.2765
bpp	0.6777	0.1398
CR	34.9800	164.4399

TABLA 4.4
FLOR (COLOR, 24 BPP, 502X502): JPEG2000 VS SC (JPEG2000+SR)

Métricas	JPEG2000	SC (JPEG2000+SR)
MAE	0.0432	0.2886
MSE	0.0500	1.1781
PSNR	63.4445	48.2714
bpp	2.6471	0.9214
CR	9.2117	30.0007

4.3.1.2.3 Interpretación de los resultados

La Fig.4.2 (arriba) nos muestra la imagen original de **Flor**, la segunda muestra la imagen recuperada del proceso de codificado y decodificado con JPEG, mientras que la tercera hace lo propio con JPEG+SC. Como puede observarse, no existen diferencias visuales entre las tres primeras imágenes considerando – no obstante – una diferencia importante en las métricas cuantitativas de la Tabla 4.3. El segundo paquete de simulaciones con respecto a **Flor** lo representan la cuarta y quinta imagen de la Fig.4.2, las cuales representan a la codificación y decodificación con JPEG2000, y lo mismo pero para la codificación y decodificación con JPEG2000+SC, respectivamente. La Tabla 4.4 muestra cambios notables en las métricas comparativas de calidad de ambos ejemplos. Más allá de las mencionadas diferencias en las tablas mencionadas, un análisis del tipo *doble-ciego* con 44 participantes arrojó que los respectivos aciertos acerca de la selección entre la imagen original y la tercera y la original y la quinta, dieron los mismos porcentajes de acierto que entre la original y la segunda, y la original y la cuarta, es decir, aproximadamente el 50 %, o sea, el mismo porcentaje que estar adivinando, ergo, no hay diferencia visual.

Figura 4.2: Primero (arriba) imagen original, segundo (codificado y decodificado con JPEG), tercero (codificado y decodificado con JPEG+Supercompresión), cuarto (codificado y decodificado con JPEG2000), quinto (abajo, codificado y decodificado con JPEG2000+Supercompresión).

4.3.1.3 Archivo texto.bmp

4.3.1.3.1 JPEG vs SC (JPEG+SR)

JPEG: Ver Tabla 4.5, columna JPEG, y Fig.4.3 (2da desde arriba).

Codificador:
1. De BMP (24 bpp, 502x502)
2. A JPEG (0.6554 bpp, 502x502)

Canal/almacenamiento

Decodificador:
1. De JPEG (0.6554 bpp, 502x502)
2. A BMP (24 bpp, 502x502)

SC (JPEG+SR): Ver Tabla 4.5, columna SC (JPEG+SR), y Fig.4.3 (3ra desde arriba).

Codificador:
1. BMP (24 bpp, 502x502)
2. Sub-muestreo (24 bpp, 251x251)
3. JPEG (0.1346 bpp, 251x251)

Canal/almacenamiento

Decodificador:
1. JPEG (0.1346 bpp, 251x251)
2. Sobre-muestreo (0.2712 bpp, 502x502)
3. Deblurring (0.4982 bpp, 502x502)
4. BMP (24 bpp, 502x502)

4.3.1.3.2 JPEG2000 vs SC (JPEG2000+SR)

JPEG2000: Ver Tabla 4.6, columna JPEG2000, y Fig.4.3 (4ta desde arriba).

Codificador:
1. De BMP (24 bpp, 502x502)
2. A JPEG2000 (2.6421 bpp, 502x502)

Canal/almacenamiento

Decodificador:
1. De JPEG2000 (2.6421 bpp, 502x502)
2. A BMP (24 bpp, 502x502)

SC (JPEG2000+SR): Ver Tabla 4.6, columna SC (JPEG2000+SR), y Fig.4.3 (abajo).

Codificador:
1. BMP (24 bpp, 502x502)
2. Sub-muestreo (24 bpp, 251x251)
3. JPEG2000 (0.8200 bpp, 251x251)

Canal/almacenamiento

Decodificador:
1. JPEG2000 (0.8200 bpp, 251x251)
2. Sobre-muestreo (1.63126 bpp, 502x502)
3. Deblurring (2.9541 bpp, 502x502)
4. BMP (24 bpp, 502x502)

Las siguientes tablas muestran las métricas vs los Algoritmos para ambos casos, i.e., JPEG y JPEG2000 vs Supercompresión.

TABLA 4.5
TEXTO (COLOR, 24 BPP, 256X256): JPEG VS SC (JPEG+SR)

Métricas	JPEG	SC (JPEG+SR)
MAE	0.4876	0.9005
MSE	2.3214	6.8766
PSNR	44.1515	39.9900
bpp	0.6554	0.1346
CR	35.0099	167.2222

TABLA 4.6
TEXTO (COLOR, 24 BPP, 256X256): JPEG2000 VS SC (JPEG2000+SR)

Métricas	JPEG2000	SC (JPEG2000+SR)
MAE	0.05100	0.2883
MSE	0.0561	1.1144
PSNR	60.3111	48.5541
bpp	2.6421	0.8200
CR	9.1110	30.0002

4.3.1.3.3 Interpretación de los resultados

La Fig.4.3 (arriba) nos muestra la imagen original de **Texto**, la segunda muestra la imagen recuperada del proceso de codificado y decodificado con JPEG, mientras que la tercera hace lo propio con JPEG+SC. Como puede observarse, no existen diferencias visuales entre las tres primeras imágenes considerando – no obstante – una diferencia importante en las métricas cuantitativas de la Tabla 4.5. El segundo paquete de simulaciones con respecto a **Texto** lo representan la cuarta y quinta imagen de la Fig.4.3, las cuales representan a la codificación y decodificación con JPEG2000, y lo mismo pero para la codificación y decodificación con JPEG2000+SC, respectivamente. La Tabla 4.6 muestra cambios notables en las métricas comparativas de calidad de ambos ejemplos. Más allá de las mencionadas diferencias en las tablas mencionadas, un análisis del tipo *doble-ciego* con 44 participantes arrojó que los respectivos aciertos acerca de la selección entre la imagen original y la tercera y la original y la quinta, dieron los mismos porcentajes de acierto que entre la original y la segunda, y la original y la cuarta, es decir, aproximadamente el 50 %, o sea, el mismo porcentaje que estar adivinando, ergo, no hay diferencia visual.

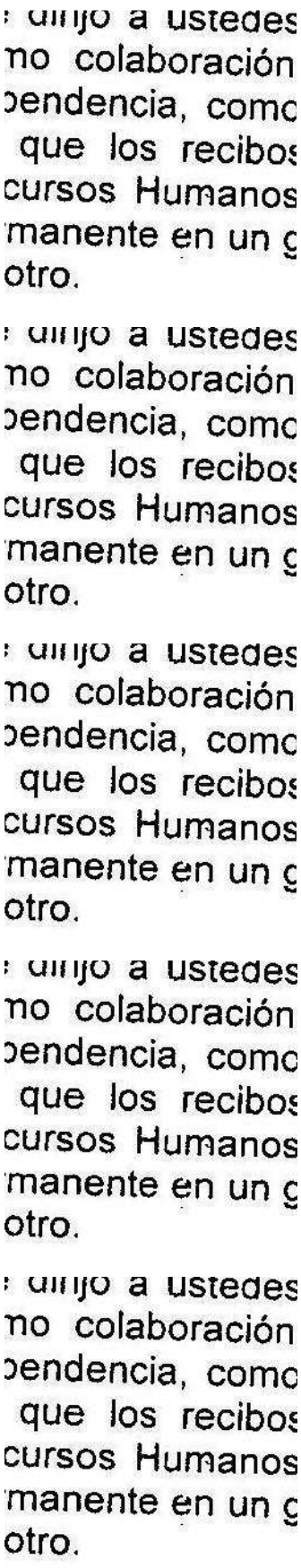


Figura 4.3: Primero (arriba) imagen original, segundo (codificado y decodificado con JPEG), tercero (codificado y decodificado con JPEG+Supercompresión), cuarto (codificado y decodificado con JPEG2000), quinto (abajo, codificado y decodificado con JPEG2000+Supercompresión).

4.3.1.4 Archivo tomografía.bmp

4.3.1.4.1 JPEG vs SC (JPEG+SR)

JPEG: Ver Tabla 4.7, columna JPEG, y Fig.4.4 (2da desde arriba).

Codificador:
1. De BMP (24 bpp, 502x502)
2. A JPEG (0.6771 bpp, 502x502)

Canal/almacenamiento

Decodificador:
1. De JPEG (0.6771 bpp, 502x502)
2. A BMP (24 bpp, 502x502)

SC (JPEG+SR): Ver Tabla 4.7, columna SC (JPEG+SR), y Fig.4.4 (3ra desde arriba).

Codificador:
1. BMP (24 bpp, 502x502)
2. Sub-muestreo (24 bpp, 251x251)
3. JPEG (0.1311 bpp, 251x251)

Canal/almacenamiento

Decodificador:
1. JPEG (0.1311 bpp, 251x251)
2. Sobre-muestreo (0.2745 bpp, 502x502)
3. Deblurring (0.5133 bpp, 502x502)
4. BMP (24 bpp, 502x502)

3.1.4.2 JPEG2000 vs SC (JPEG2000+SR)

JPEG2000: Ver Tabla 4.8, columna JPEG2000, y Fig.4.4 (4ta desde arriba).

Codificador:
1. De BMP (24 bpp, 502x502)
2. A JPEG2000 (2.7171 bpp, 502x502)

Canal/almacenamiento

Decodificador:
1. De JPEG2000 (2.7171 bpp, 502x502)
2. A BMP (24 bpp, 502x502)

SC (JPEG2000+SR): Ver Tabla 4.8, columna SC (JPEG2000+SR), y Fig.4.4 (abajo).

Codificador:

1. BMP (24 bpp, 502x502)
2. Sub-muestreo (24 bpp, 251x251)
3. JPEG2000 (0.8022 bpp, 251x251)

Canal/almacenamiento

Decodificador:

1. JPEG2000 (0.8022 bpp, 251x251)
2. Sobre-muestreo (1.1769 bpp, 502x502)
3. Deblurring (2.3156 bpp, 502x502)
4. BMP (24 bpp, 502x502)

Las siguientes tablas muestran las métricas vs los Algoritmos para ambos casos, i.e., JPEG y JPEG2000 vs Supercompresión.

TABLA 4.7
TOMOGRAFIA (COLOR, 24 BPP, 512X512): JPEG VS SC (JPEG+SR)

Métricas	JPEG	SC (JPEG+SR)
MAE	0.5419	0.9666
MSE	2.4512	7.1912
PSNR	42.7543	39.8746
bpp	0.6771	0.1311
CR	36.0008	166.4234

TABLA 4.8
TOMOGRAFIA (COLOR, 24 BPP, 512X512): JPEG2000 VS SC (JPEG2000+SR)

Métricas	JPEG2000	SC (JPEG2000+SR)
MAE	0.0552	0.3100
MSE	0.0712	1.1177
PSNR	60.6539	48.0000
bpp	2.7171	0.8022
CR	10.0088	30.9121

4.3.1.4.3 Interpretación de los resultados

La Fig.4.4 (arriba) nos muestra la imagen original de **Tomografía**, la segunda muestra la imagen recuperada del proceso de codificado y decodificado con JPEG, mientras que la tercera hace lo propio con JPEG+SC. Como puede observarse, no existen diferencias visuales entre las tres primeras imágenes considerando – no obstante – una diferencia importante en las métricas cuantitativas de la Tabla 4.7. El segundo paquete de simulaciones con respecto a **Tomografía** lo representan la cuarta y quinta imagen de la Fig.4.4, las cuales representan a la codificación y decodificación con JPEG2000, y lo mismo pero para la codificación y decodificación con JPEG2000+SC, respectivamente. La Tabla 4.8 muestra cambios notables en las métricas comparativas de calidad de ambos ejemplos. Más allá de las mencionadas diferencias en las tablas mencionadas, un análisis del tipo *doble-ciego* con 44 participantes arrojó que los respectivos aciertos acerca de la selección entre la imagen original y la tercera y la original y la quinta, dieron los mismos porcentajes de acierto que entre la original y la segunda, y la original y la cuarta, es decir, aproximadamente el 50 %, o sea, el mismo porcentaje que estar adivinando, ergo, no hay diferencia visual.

Figura 4.4: Primero (arriba) imagen original, segundo (codificado y decodificado con JPEG), tercero (codificado y decodificado con JPEG+Supercompresión), cuarto (codificado y decodificado con JPEG2000), quinto (abajo, codificado y decodificado con JPEG2000+Supercompresión).

4.3.1.5 Archivo quickbird-1.bmp

4.3.1.5.1 JPEG vs SC (JPEG+SR)

JPEG: Ver Tabla 4.9, columna JPEG, y Fig.4.5 (2da desde arriba).

Codificador:
1. De BMP (24 bpp, 502x502)
2. A JPEG (0.7111 bpp, 502x502)

Canal/almacenamiento

Decodificador:
1. De JPEG (0.7111 bpp, 502x502)
2. A BMP (24 bpp, 502x502)

SC (JPEG+SR): Ver Tabla 4.9, columna SC (JPEG+SR), y Fig.4.5 (3ra desde arriba).

Codificador:
1. BMP (24 bpp, 502x502)
2. Sub-muestreo (24 bpp, 251x251)
3. JPEG (0.1321 bpp, 251x251)

Canal/almacenamiento

Decodificador:
1. JPEG (0.1321 bpp, 251x251)
2. Sobre-muestreo (0.2618 bpp, 502x502)
3. Deblurring (0.4914 bpp, 502x502)
4. BMP (24 bpp, 502x502)

4.3.1.5.2 JPEG2000 vs SC (JPEG2000+SR)

JPEG2000: Ver Tabla 4.10, columna JPEG2000, y Fig.4.5 (4ta desde arriba).

Codificador:
1. De BMP (24 bpp, 502x502)
2. A JPEG2000 (2.5122 bpp, 502x502)

Canal/almacenamiento

Decodificador:
1. De JPEG2000 (2.5111 bpp, 502x502)
2. A BMP (24 bpp, 502x502)

SC (JPEG2000+SR): Ver Tabla 4.10, columna SC (JPEG2000+SR), y Fig.4.5 (abajo).

Codificador:

1. BMP (24 bpp, 502x502)
2. Sub-muestreo (24 bpp, 251x251)
3. JPEG2000 (0.6900 bpp, 251x251)

Canal/almacenamiento

Decodificador:

1. JPEG2000 (0.6900 bpp, 251x251)
2. Sobre-muestreo (1.7200 bpp, 502x502)
3. Deblurring (3.2876 bpp, 502x502)
4. BMP (24 bpp, 502x502)

Las siguientes tablas muestran las métricas vs los Algoritmos para ambos casos, i.e., JPEG y JPEG2000 vs Supercompresión.

TABLA 4.9
QUICKBIRD-1 (COLOR, 24 BPP, 4096X4096): JPEG VS SC (JPEG+SR)

Métricas	JPEG	SC (JPEG+SR)
MAE	0.6111	1.0018
MSE	2.4622	7.7711
PSNR	42.9678	37.8734
bpp	0.7111	0.1321
CR	34.5353	167.8763

TABLA 4.10
QUICKBIRD-1 (COLOR, 24 BPP, 4096X4096): JPEG2000 VS SC (JPEG2000+SR)

Métricas	JPEG2000	SC (JPEG2000+SR)
MAE	0.0398	0.2701
MSE	0.0399	1.0912
PSNR	63.9834	49.9834
bpp	2.5122	0.6900
CR	9.9976	31.9787

4.3.1.5.3 Interpretación de los resultados

La Fig.4.5 (arriba) nos muestra la imagen original de **Quickbird-1**, la segunda muestra la imagen recuperada del proceso de codificado y decodificado con JPEG, mientras que la tercera hace lo propio con JPEG+SC. Como puede observarse, no existen diferencias visuales entre las tres primeras imágenes considerando – no obstante – una diferencia importante en las métricas cuantitativas de la Tabla 4.9. El segundo paquete de simulaciones con respecto a **Quickbird-1** lo representan la cuarta y quinta imagen de la Fig.4.5, las cuales representan a la codificación y decodificación con JPEG2000, y lo mismo pero para la codificación y decodificación con JPEG2000+SC, respectivamente. La Tabla 4.10 muestra cambios notables en las métricas comparativas de calidad de ambos ejemplos. Más allá de las mencionadas diferencias en las tablas mencionadas, un análisis del tipo *doble-ciego* con 44 participantes arrojó que los respectivos aciertos acerca de la selección entre la imagen original y la tercera y la original y la quinta, dieron los mismos porcentajes de acierto que entre la original y la segunda, y la original y la cuarta, es decir, aproximadamente el 50 %, o sea, el mismo porcentaje que estar adivinando, ergo, no hay diferencia visual.

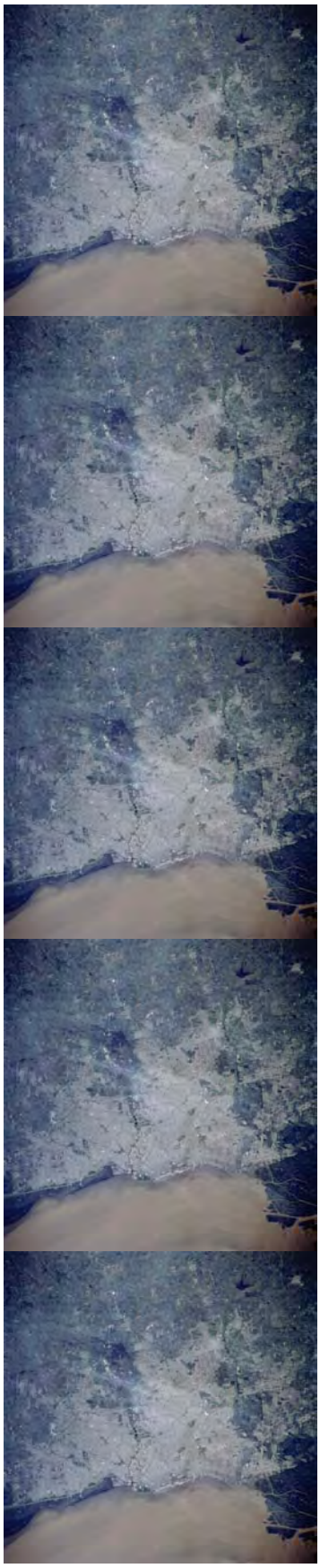


Figura 4.5: Primero (arriba) imagen original, segundo (codificado y decodificado con JPEG), tercero (codificado y decodificado con JPEG+Supercompresión), cuarto (codificado y decodificado con JPEG2000), quinto (abajo, codificado y decodificado con JPEG2000+Supercompresión).

4.3.1.6 Archivo eros-b.bmp

4.3.1.6.1 JPEG vs SC (JPEG+SR)

JPEG: Ver Tabla 4.11, columna JPEG, y Fig.4.6 (2da desde arriba).

Codificador:
1. De BMP (24 bpp, 502x502)
2. A JPEG (0.5524 bpp, 502x502)

Canal/almacenamiento

Decodificador:
1. De JPEG (0.5524 bpp, 502x502)
2. A BMP (24 bpp, 502x502)

SC (JPEG+SR): Ver Tabla 4.11, columna SC (JPEG+SR), y Fig.4.6 (3ra desde arriba).

Codificador:
1. BMP (24 bpp, 502x502)
2. Sub-muestreo (24 bpp, 251x251)
3. JPEG (0.0981 bpp, 251x251)

Canal/almacenamiento

Decodificador:
1. JPEG (0.0981 bpp, 251x251)
2. Sobre-muestreo (0.2113 bpp, 502x502)
3. Deblurring (0.4040 bpp, 502x502)
4. BMP (24 bpp, 502x502)

4.3.1.6.2 JPEG2000 vs SC (JPEG2000+SR)

JPEG2000: Ver Tabla 4.12, columna JPEG2000, y Fig.4.6 (4ta desde arriba).

Codificador:
1. De BMP (24 bpp, 502x502)
2. A JPEG2000 (2.7200 bpp, 502x502)

Canal/almacenamiento

Decodificador:
1. De JPEG2000 (2.7200 bpp, 502x502)
2. A BMP (24 bpp, 502x502)

SC (JPEG2000+SR): Ver Tabla 4.12, columna SC (JPEG2000+SR), y Fig.4.6 (abajo).

Codificador:

1. BMP (24 bpp, 502x502)
2. Sub-muestreo (24 bpp, 251x251)
3. JPEG2000 (0.8298 bpp, 251x251)

Canal/almacenamiento

Decodificador:

1. JPEG2000 (0.8298 bpp, 251x251)
2. Sobre-muestreo (1.7116 bpp, 502x502)
3. Deblurring (2.9981 bpp, 502x502)
4. BMP (24 bpp, 502x502)

Las siguientes tablas muestran las métricas vs los Algoritmos para ambos casos, i.e., JPEG y JPEG2000 vs Supercompresión.

TABLA 4.11
EROS B (COLOR, 24 BPP, 600X600): JPEG VS SC (JPEG+SR)

Métricas	JPEG	SC (JPEG+SR)
MAE	0.6100	1.1114
MSE	2.4660	7.9908
PSNR	41.9873	37.9393
bpp	0.5524	0.0981
CR	36.9893	169.9845

TABLA 4.12
EROS B (COLOR, 24 BPP, 600X600): JPEG2000 VS SC (JPEG2000+SR)

Métricas	JPEG2000	SC (JPEG2000+SR)
MAE	0.05423	0.2998
MSE	0.0674	1.1432
PSNR	60.9834	47.0003
bpp	2.7200	0.8298
CR	9.0008	29.4123

4.3.1.6.3 Interpretación de los resultados

La Fig.4.6 (arriba) nos muestra la imagen original de **Eros B**, la segunda muestra la imagen recuperada del proceso de codificado y decodificado con JPEG, mientras que la tercera hace lo propio con JPEG+SC. Como puede observarse, no existen diferencias visuales entre las tres primeras imágenes considerando – no obstante – una diferencia importante en las métricas cuantitativas de la Tabla 4.11. El segundo paquete de simulaciones con respecto a **Eros B** lo representan la cuarta y quinta imagen de la Fig.4.6, las cuales representan a la codificación y decodificación con JPEG2000, y lo mismo pero para la codificación y decodificación con JPEG2000+SC, respectivamente. La Tabla 4.12 muestra cambios notables en las métricas comparativas de calidad de ambos ejemplos. Más allá de las mencionadas diferencias en las tablas mencionadas, un análisis del tipo *doble-ciego* con 44 participantes arrojó que los respectivos aciertos acerca de la selección entre la imagen original y la tercera y la original y la quinta, dieron los mismos porcentajes de acierto que entre la original y la segunda, y la original y la cuarta, es decir, aproximadamente el 50 %, o sea, el mismo porcentaje que estar adivinando, ergo, no hay diferencia visual.

Figura 4.6: Primero (arriba) imagen original, segundo (codificado y decodificado con JPEG), tercero (codificado y decodificado con JPEG+Supercompresión), cuarto (codificado y decodificado con JPEG2000), quinto (abajo, codificado y decodificado con JPEG2000+Supercompresión).

4.3.1.7 Archivo quickbird-2.bmp

4.3.1.7.1 JPEG vs SC (JPEG+SR)

JPEG: Ver Tabla 4.13, columna JPEG, y Fig.4.7 (2da desde arriba).

Codificador:
1. De BMP (24 bpp, 502x502)
2. A JPEG (0.6969 bpp, 502x502)

Canal/almacenamiento

Decodificador:
1. De JPEG (0.6969 bpp, 502x502)
2. A BMP (24 bpp, 502x502)

SC (JPEG+SR): Ver Tabla 4.13, columna SC (JPEG+SR), y Fig.4.7 (3ra desde arriba).

Codificador:
1. BMP (24 bpp, 502x502)
2. Sub-muestreo (24 bpp, 251x251)
3. JPEG (0.1213 bpp, 251x251)

Canal/almacenamiento

Decodificador:
1. JPEG (0.1213 bpp, 251x251)
2. Sobre-muestreo (0.2379 bpp, 502x502)
3. Deblurring (0.4591 bpp, 502x502)
4. BMP (24 bpp, 502x502)

4.3.1.7.2 JPEG2000 vs SC (JPEG2000+SR)

JPEG2000: Ver Tabla 4.14, columna JPEG2000, y Fig.4.7 (4ta desde arriba).

Codificador:
1. De BMP (24 bpp, 502x502)
2. A JPEG2000 (2.9845 bpp, 502x502)

Canal/almacenamiento

Decodificador:
1. De JPEG2000 (2.9845 bpp, 502x502)
2. A BMP (24 bpp, 502x502)

SC (JPEG2000+SR): Ver Tabla 4.14, columna SC (JPEG2000+SR), y Fig.4.7 (abajo).

Codificador:

1. BMP (24 bpp, 502x502)
2. Sub-muestreo (24 bpp, 251x251)
3. JPEG2000 (0.6649 bpp, 251x251)

Canal/almacenamiento

Decodificador:

1. JPEG2000 (0.6649 bpp, 251x251)
2. Sobre-muestreo (1.8664 bpp, 502x502)
3. Deblurring (2.9595 bpp, 502x502)
4. BMP (24 bpp, 502x502)

Las siguientes tablas muestran las métricas vs los Algoritmos para ambos casos, i.e., JPEG y JPEG2000 vs Supercompresión.

TABLA 4.13
QUICKBIRD-2 (COLOR, 24 BPP, 1400X1442): JPEG VS SC (JPEG+SR)

Métricas	JPEG	SC (JPEG+SR)
MAE	0.4112	0.8756
MSE	2.0012	6.9749
PSNR	45.0454	40.2343
bpp	0.6969	0.1213
CR	36.8976	171.2988

TABLA 4.14
QUICKBIRD-2 (COLOR, 24 BPP, 1400X1442): JPEG2000 VS SC (JPEG2000+SR)

Métricas	JPEG2000	SC (JPEG2000+SR)
MAE	0.0398	0.2612
MSE	0.0311	1.0001
PSNR	64.8433	50.2340
bpp	2.9845	0.6649
CR	8.8734	32.7432

4.3.1.7.3 Interpretación de los resultados

La Fig.4.7 (arriba) nos muestra la imagen original de **Quickbird-2**, la segunda muestra la imagen recuperada del proceso de codificado y decodificado con JPEG, mientras que la tercera hace lo propio con JPEG+SC. Como puede observarse, no existen diferencias visuales entre las tres primeras imágenes considerando – no obstante – una diferencia importante en las métricas cuantitativas de la Tabla 4.13. El segundo paquete de simulaciones con respecto a **Quickbird-2** lo representan la cuarta y quinta imagen de la Fig.4.7, las cuales representan a la codificación y decodificación con JPEG2000, y lo mismo pero para la codificación y decodificación con JPEG2000+SC, respectivamente. La Tabla 4.14 muestra cambios notables en las métricas comparativas de calidad de ambos ejemplos. Más allá de las mencionadas diferencias en las tablas mencionadas, un análisis del tipo *doble-ciego* con 44 participantes arrojó que los respectivos aciertos acerca de la selección entre la imagen original y la tercera y la original y la quinta, dieron los mismos porcentajes de acierto que entre la original y la segunda, y la original y la cuarta, es decir, aproximadamente el 50 %, o sea, el mismo porcentaje que estar adivinando, ergo, no hay diferencia visual.

Figura 4.7: Primero (arriba) imagen original, segundo (codificado y decodificado con JPEG), tercero (codificado y decodificado con JPEG+Supercompresión), cuarto (codificado y decodificado con JPEG2000), quinto (abajo, codificado y decodificado con JPEG2000+Supercompresión).

4.3.2 Grupo gris

4.3.2.1 Archivo lena.bmp

4.3.2.1.1 JPEG vs SC (JPEG+SR)

JPEG: Ver Tabla 4.15, columna JPEG, y Fig.4.8 (2da desde arriba).

Codificador:
1. De BMP (8 bpp, 512x512)
2. A JPEG (0.8953 bpp, 512x512)

Canal/almacenamiento

Decodificador:
1. De JPEG (0.8953 bpp, 512x512)
2. A BMP (24 bpp, 512x512)

SC (JPEG+SR): Ver Tabla 4.15, columna SC (JPEG+SR), y Fig.4.8 (3ra desde arriba).

Codificador:
1. BMP (8 bpp, 512x512)
2. Sub-muestreo (8 bpp, 256x256)
3. JPEG (0.2957 bpp, 256x256)

Canal/almacenamiento

Decodificador:
1. JPEG (0.2957 bpp, 256x256)
2. Sobre-muestreo (0.6502 bpp, 512x512)
3. Deblurring (0.7727 bpp, 512x512)
4. BMP (8 bpp, 512x512)

4.3.2.1.2 JPEG2000 vs SC (JPEG2000+SR)

JPEG2000: Ver Tabla 4.16, columna JPEG2000, y Fig.4.8 (4to desde arriba).

Codificador:
1. De BMP (8 bpp, 512x512)
2. A JPEG2000 (3.7242 bpp, 512x512)

Canal/almacenamiento

Decodificador:
1. De JPEG2000 (3.7242 bpp, 512x512)
2. A BMP (8 bpp, 512x512)

SC (JPEG2000+SR): Ver Tabla 4.16, columna SC (JPEG2000+SR), y Fig.4.8 (abajo).

Codificador:

1. BMP (8 bpp, 512x512)
2. Sub-muestreo (8 bpp, 256x256)
3. JPEG2000 (1.0066 bpp, 256x256)

Canal/almacenamiento

Decodificador:

1. JPEG2000 (1.0066 bpp, 256x256)
2. Sobre-muestreo (1.6421 bpp, 512x512)
3. Deblurring (2.4230 bpp, 512x512)
4. BMP (8 bpp, 512x512)

Las siguientes tablas muestran las métricas vs los Algoritmos para ambos casos, i.e., JPEG y JPEG2000 vs Supercompresión.

TABLA 4.15
LENA (GRIS, 8 BPP, 512X512): JPEG VS SC (JPEG+SR)

Métricas	JPEG	SC (JPEG+SR)
MAE	1.0785	2.0243
MSE	4.4363	14.6230
PSNR	41.6606	36.4804
bpp	0.8953	0.2957
CR	8.9358	27.0526

TABLA 4.16
LENA (GRIS, 8 BPP, 512X512): JPEG2000 VS SC (JPEG2000+SR)

Métricas	JPEG2000	SC (JPEG2000+SR)
MAE	0.0902	1.5312
MSE	0.0905	9.2596
PSNR	58.5647	38.4649
bpp	3.7242	1.0066
CR	2.1481	7.9475

4.3.2.1.3 Interpretación de los resultados

La Fig.4.8 (arriba) nos muestra la imagen original de **Lena**, la segunda muestra la imagen recuperada del proceso de codificado y decodificado con JPEG, mientras que la tercera hace lo propio con JPEG+SC. Como puede observarse, no existen diferencias visuales entre las tres primeras imágenes considerando – no obstante – una diferencia importante en las métricas cuantitativas de la Tabla 4.15. El segundo paquete de simulaciones con respecto a **Lena** lo representan la cuarta y quinta imagen de la Fig.4.8, las cuales representan a la codificación y decodificación con JPEG2000, y lo mismo pero para la codificación y decodificación con JPEG2000+SC, respectivamente. La Tabla 4.16 muestra cambios notables en las métricas comparativas de calidad de ambos ejemplos. Más allá de las mencionadas diferencias en las tablas mencionadas, un análisis del tipo *doble-ciego* con 44 participantes arrojó que los respectivos aciertos acerca de la selección entre la imagen original y la tercera y la original y la quinta, dieron los mismos porcentajes de acierto que entre la original y la segunda, y la original y la cuarta, es decir, aproximadamente el 50 %, o sea, el mismo porcentaje que estar adivinando, ergo, no hay diferencia visual.

Figura 4.8: Primero (arriba) imagen original, segunda (codificado y decodificado con JPEG), tercera (codificada y decodificada con JPEG+Supercompresión), cuarta (codificada y decodificada con JPEG2000), quinta (abajo, codificada y decodificada con JPEG2000+Supercompresión).

4.3.2.2 Archivo fingerprint.bmp

4.3.2.2.1 JPEG vs SC (JPEG+SR)

JPEG: Ver Tabla 4.17, columna JPEG, y Fig.4.9 (2da desde arriba).

Codificador:
1. De BMP (8 bpp, 512x512)
2. A JPEG (0.8556 bpp, 512x512)

Canal/almacenamiento

Decodificador:
1. De JPEG (0.8556 bpp, 512x512)
2. A BMP (24 bpp, 512x512)

SC (JPEG+SR): Ver Tabla 4.17, columna SC (JPEG+SR), y Fig.4.9 (3ra desde arriba).

Codificador:
4. BMP (8 bpp, 512x512)
5. Sub-muestreo (8 bpp, 256x256)
6. JPEG (0.2677 bpp, 256x256)

Canal/almacenamiento

Decodificador:
4. JPEG (0.2677 bpp, 256x256)
5. Sobre-muestreo (0.5365 bpp, 512x512)
6. Deblurring (0.8993 bpp, 512x512)
4. BMP (8 bpp, 512x512)

4.3.2.2.2 JPEG2000 vs SC (JPEG2000+SR)

JPEG2000: Ver Tabla 4.18, columna JPEG2000, y Fig.4.9 (4to desde arriba).

Codificador:
1. De BMP (8 bpp, 512x512)
2. A JPEG2000 (2.8744 bpp, 512x512)

Canal/almacenamiento

Decodificador:
1. De JPEG2000 (2.8744 bpp, 512x512)
2. A BMP (8 bpp, 512x512)

SC (JPEG2000+SR): Ver Tabla 4.18, columna SC (JPEG2000+SR), y Fig.4.9 (abajo).

Codificador:

4. BMP (8 bpp, 512x512)
5. Sub-muestreo (8 bpp, 256x256)
6. JPEG2000 (0.9600 bpp, 256x256)

Canal/almacenamiento

Decodificador:

4. JPEG2000 (0.9600 bpp, 256x256)
5. Sobre-muestreo (1.9211 bpp, 512x512)
6. Deblurring (2.724 bpp, 512x512)
4. BMP (8 bpp, 512x512)

Las siguientes tablas muestran las métricas vs los Algoritmos para ambos casos, i.e., JPEG y JPEG2000 vs Supercompresión.

TABLA 4.17
FINGERPRINT (GRIS, 8 BPP, 542X542): JPEG VS SC (JPEG+SR)

Métricas	JPEG	SC (JPEG+SR)
MAE	1.0003	1.7809
MSE	3.7433	12.7433
PSNR	44.7433	39.9843
bpp	0.8556	0.2677
CR	10.8764	29.8743

TABLA 4.18
FINGERPRINT (GRIS, 8 BPP, 542X542): JPEG2000 VS SC (JPEG2000+SR)

Métricas	JPEG2000	SC (JPEG2000+SR)
MAE	0.07992	1.4310
MSE	0.0788	7.8585
PSNR	60.7473	40.7434
bpp	2.8744	0.9600
CR	3.0403	6.9843

4.3.2.2.3 Interpretación de los resultados

La Fig.4.9 (arriba) nos muestra la imagen original de **Fingerprint**, la segunda muestra la imagen recuperada del proceso de codificado y decodificado con JPEG, mientras que la tercera hace lo propio con JPEG+SC. Como puede observarse, no existen diferencias visuales entre las tres primeras imágenes considerando – no obstante – una diferencia importante en las métricas cuantitativas de la Tabla 4.18. El segundo paquete de simulaciones con respecto a **Fingerprint** lo representan la cuarta y quinta imagen de la Fig.4.9, las cuales representan a la codificación y decodificación con JPEG2000, y lo mismo pero para la codificación y decodificación con JPEG2000+SC, respectivamente. La Tabla 4.19 muestra cambios notables en las métricas comparativas de calidad de ambos ejemplos. Más allá de las mencionadas diferencias en las tablas mencionadas, un análisis del tipo *doble-ciego* con 44 participantes arrojó que los respectivos aciertos acerca de la selección entre la imagen original y la tercera y la original y la quinta, dieron los mismos porcentajes de acierto que entre la original y la segunda, y la original y la cuarta, es decir, aproximadamente el 50 %, o sea, el mismo porcentaje que estar adivinando, ergo, no hay diferencia visual.

Figura 4.9: Primero (arriba) imagen original, segunda (codificado y decodificado con JPEG), tercera (codificada y decodificada con JPEG+Supercompresión), cuarta (codificada y decodificada con JPEG2000), quinta (abajo, codificada y decodificada con JPEG2000+Supercompresión).

4.3.2.3 Archivo rodilla.bmp

4.3.2.3.1 JPEG vs SC (JPEG+SR)

JPEG: Ver Tabla 4.19, columna JPEG, y Fig.4.10 (2da desde arriba).

Codificador:
1. De BMP (8 bpp, 512x512)
2. A JPEG (0.6654 bpp, 512x512)

Canal/almacenamiento

Decodificador:
1. De JPEG (0.6654 bpp, 512x512)
2. A BMP (24 bpp, 512x512)

SC (JPEG+SR): Ver Tabla 4.19, columna SC (JPEG+SR), y Fig.4.10 (3ra desde arriba).

Codificador:
7. BMP (8 bpp, 512x512)
8. Sub-muestreo (8 bpp, 256x256)
9. JPEG (0.2760 bpp, 256x256)

Canal/almacenamiento

Decodificador:
7. JPEG (0.2760 bpp, 256x256)
8. Sobre-muestreo (0.5521 bpp, 512x512)
9. Deblurring (0.7811 bpp, 512x512)
4. BMP (8 bpp, 512x512)

4.3.2.3.2 JPEG2000 vs SC (JPEG2000+SR)

JPEG2000: Ver Tabla 4.20, columna JPEG2000, y Fig.4.10 (4to desde arriba).

Codificador:
1. De BMP (8 bpp, 512x512)
2. A JPEG2000 (2.9834 bpp, 512x512)

Canal/almacenamiento

Decodificador:
1. De JPEG2000 (2.9834 bpp, 512x512)
2. A BMP (8 bpp, 512x512)

SC (JPEG2000+SR): Ver Tabla 4.20, columna SC (JPEG2000+SR), y Fig.4.10 (abajo).

Codificador:

7. BMP (8 bpp, 512x512)
8. Sub-muestreo (8 bpp, 256x256)
9. JPEG2000 (0.8440 bpp, 256x256)

Canal/almacenamiento

Decodificador:

7. JPEG2000 (0.8440 bpp, 256x256)
8. Sobre-muestreo (1.6077 bpp, 512x512)
9. Deblurring (2.7121 bpp, 512x512)
4. BMP (8 bpp, 512x512)

Las siguientes tablas muestran las métricas vs los Algoritmos para ambos casos, i.e., JPEG y JPEG2000 vs Supercompresión.

TABLA 4.19
RODILLA (GRIS, 8 BPP, 512X512): JPEG VS SC (JPEG+SR)

Métricas	JPEG	SC (JPEG+SR)
MAE	1.0011	1.8971
MSE	3.7474	12.4744
PSNR	42.8743	39.7847
bpp	0.6654	0.2760
CR	8.9485	29.9845

TABLA 4.20
RODILLA (GRIS, 8 BPP, 512X512): JPEG2000 VS SC (JPEG2000+SR)

Métricas	JPEG2000	SC (JPEG2000+SR)
MAE	0.0790	1.3211
MSE	0.0688	7.1225
PSNR	61.0233	43.8743
bpp	2.9834	0.8440
CR	3.0005	10.9443

4.3.2.3.3 Interpretación de los resultados

La Fig.4.10 (arriba) nos muestra la imagen original de **Rodilla**, la segunda muestra la imagen recuperada del proceso de codificado y decodificado con JPEG, mientras que la tercera hace lo propio con JPEG+SC. Como puede observarse, no existen diferencias visuales entre las tres primeras imágenes considerando – no obstante – una diferencia importante en las métricas cuantitativas de la Tabla 4.19. El segundo paquete de simulaciones con respecto a **Rodilla** lo representan la cuarta y quinta imagen de la Fig.4.10, las cuales representan a la codificación y decodificación con JPEG2000, y lo mismo pero para la codificación y decodificación con JPEG2000+SC, respectivamente. La Tabla 4.20 muestra cambios notables en las métricas comparativas de calidad de ambos ejemplos. Más allá de las mencionadas diferencias en las tablas mencionadas, un análisis del tipo *doble-ciego* con 44 participantes arrojó que los respectivos aciertos acerca de la selección entre la imagen original y la tercera y la original y la quinta, dieron los mismos porcentajes de acierto que entre la original y la segunda, y la original y la cuarta, es decir, aproximadamente el 50 %, o sea, el mismo porcentaje que estar adivinando, ergo, no hay diferencia visual.

Figura 4.10: Primero (arriba) imagen original, segunda (codificado y decodificado con JPEG), tercera (codificada y decodificada con JPEG+Supercompresión), cuarta (codificada y decodificada con JPEG2000), quinta (abajo, codificada y decodificada con JPEG2000+Supercompresión).

4.3.2.4 Archivo sar.bmp

4.3.2.4.1 JPEG vs SC (JPEG+SR)

JPEG: Ver Tabla 4.21, columna JPEG, y Fig.4.11 (2da desde arriba).

Codificador:
1. De BMP (8 bpp, 512x512)
2. A JPEG (0.6302 bpp, 512x512)

Canal/almacenamiento

Decodificador:
1. De JPEG (0.6302 bpp, 512x512)
2. A BMP (24 bpp, 512x512)

SC (JPEG+SR): Ver Tabla 4.21, columna SC (JPEG+SR), y Fig.4.11 (3ra desde arriba).

Codificador:
10. BMP (8 bpp, 512x512)
11. Sub-muestreo (8 bpp, 256x256)
12. JPEG (0.1100 bpp, 256x256)

Canal/almacenamiento

Decodificador:
10. JPEG (0.1100 bpp, 256x256)
11. Sobre-muestreo (0.23412 bpp, 512x512)
12. Deblurring (0.4417 bpp, 512x512)
4. BMP (8 bpp, 512x512)

4.3.2.4.2 JPEG2000 vs SC (JPEG2000+SR)

JPEG2000: Ver Tabla 4.22, columna JPEG2000, y Fig.4.11 (4to desde arriba).

Codificador:
1. De BMP (8 bpp, 512x512)
2. A JPEG2000 (3.9459 bpp, 512x512)

Canal/almacenamiento

Decodificador:
1. De JPEG2000 (3.9459 bpp, 512x512)
2. A BMP (8 bpp, 512x512)

SC (JPEG2000+SR): Ver Tabla 4.22, columna SC (JPEG2000+SR), y Fig.4.11 (abajo).

Codificador:

10. BMP (8 bpp, 512x512)
11. Sub-muestreo (8 bpp, 256x256)
12. JPEG2000 (0.8438 bpp, 256x256)

Canal/almacenamiento

Decodificador:

10. JPEG2000 (0.8438 bpp, 256x256)
11. Sobre-muestreo (1.6010 bpp, 512x512)
12. Deblurring (2.1118 bpp, 512x512)
4. BMP (8 bpp, 512x512)

Las siguientes tablas muestran las métricas vs los Algoritmos para ambos casos, i.e., JPEG y JPEG2000 vs Supercompresión.

TABLA 4.21
SAR (GRIS, 8 BPP, 540X552): JPEG VS SC (JPEG+SR)

Métricas	JPEG	SC (JPEG+SR)
MAE	1.0002	2.0031
MSE	3.9765	12.7473
PSNR	44.7843	39.9995
bpp	0.6302	0.1100
CR	10.6322	30.7237

TABLA 4.22
SAR (GRIS, 8 BPP, 540X552): JPEG2000 VS SC (JPEG2000+SR)

Métricas	JPEG2000	SC (JPEG2000+SR)
MAE	0.0554	1.2180
MSE	0.0450	6.8685
PSNR	60.8743	39.7347
bpp	3.9459	0.8438
CR	2.0655	9.0010

4.3.2.4.3 Interpretación de los resultados

La Fig.4.11 (arriba) nos muestra la imagen original de **SAR**, la segunda muestra la imagen recuperada del proceso de codificado y decodificado con JPEG, mientras que la tercera hace lo propio con JPEG+SC. Como puede observarse, no existen diferencias visuales entre las tres primeras imágenes considerando – no obstante – una diferencia importante en las métricas cuantitativas de la Tabla 4.21. El segundo paquete de simulaciones con respecto a **SAR** lo representan la cuarta y quinta imagen de la Fig.4.11, las cuales representan a la codificación y decodificación con JPEG2000, y lo mismo pero para la codificación y decodificación con JPEG2000+SC, respectivamente. La Tabla 4.22 muestra cambios notables en las métricas comparativas de calidad de ambos ejemplos. Más allá de las mencionadas diferencias en las tablas mencionadas, un análisis del tipo *doble-ciego* con 44 participantes arrojó que los respectivos aciertos acerca de la selección entre la imagen original y la tercera y la original y la quinta, dieron los mismos porcentajes de acierto que entre la original y la segunda, y la original y la cuarta, es decir, aproximadamente el 50 %, o sea, el mismo porcentaje que estar adivinando, ergo, no hay diferencia visual.

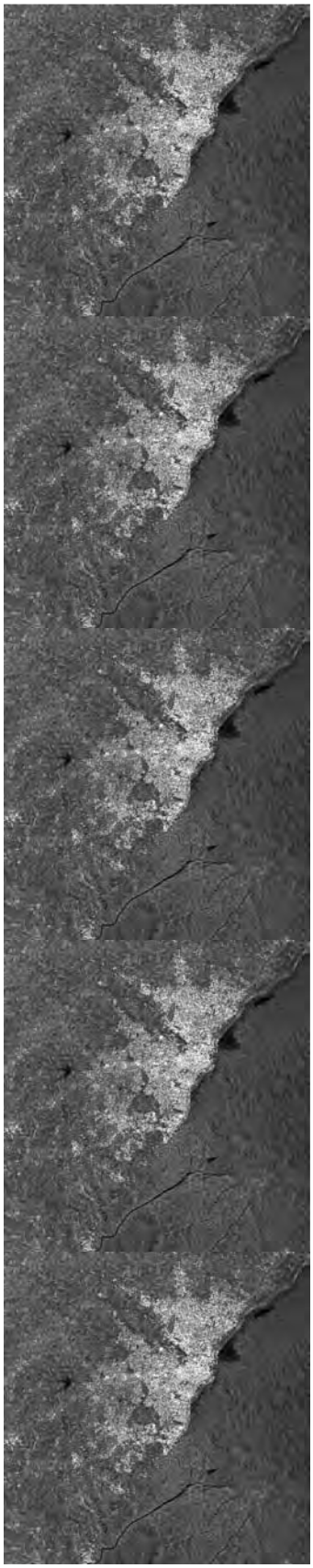


Figura 4.11: Primero (arriba) imagen original, segunda (codificado y decodificado con JPEG), tercera (codificada y decodificada con JPEG+Supercompresión), cuarta (codificada y decodificada con JPEG2000), quinta (abajo, codificada y decodificada con JPEG2000+Supercompresión).

4.4 Conclusiones del capítulo

En las 11 figuras del capítulo no se notan las diferencias entre las cinco componentes de cada uno, es decir, la versión original de cada imagen, su compresión en JPEG, JPEG2000, SR+JPEG y SR+JPEG2000, lo cual muestra la robustez y calidad visual del método propuesto.

4.4.1 Grupo color

Experimentos: JPEG vs SC (JPEG+SR)
En estos experimentos SC (JPEG+SR) tenemos a MAE, MSE y PSNR con prácticamente el mismo orden de magnitud que para el caso de JPEG solo, no obstante, bpp es 5 veces menor, y al mismo tiempo, CR es 5 veces mayor, ver Tablas 4.1, 4.3, 4.5, 4.7, 4.9, 4.11 y 4.13.

Como se muestra en las Figuras 4.1 a 4.7, la segunda imagen de *Angelina* (codificada y decodificada con JPEG) y la tercera (codificada y decodificada con JPEG+Super-compresión) ambas desde arriba, tienen la misma apariencia y calidad visual que la de arriba, i.e., la imagen original de Angelina.

Experimentos: JPEG2000 vs SC (JPEG2000+SR)
Hacemos aquí similares consideraciones para este experimento, que para el anterior, ver Tablas 4.2, 4.4, 4.6, 4.8, 4.10, 4.12, 4.14 y Figuras 4.1 a 4.7 (donde la cuarta codificada y decodificada con JPEG2000 solamente, y la quinta codificada y decodificada con JPEG2000+Supercompresión), no obstante, existe una diferencia notable entre JPEG y JPEG-2000 a la hora de comprimir este tipo de imágenes (comparemos los bpp y CR entre las Tablas de finalización impar y par de la 4.1 a la 4.14, es decir, 4.1 y 4.2, 4.3 y 4.4, y así).

4.4.2 Grupo grises

Experimentos: JPEG vs SC (JPEG+SR)
En este experimento SC (JPEG+SR) tiene unos MAE, MSE y PSNR de prácticamente los mismos órdenes de magnitud que JPEG solo, no obstante, bpp es 5 veces menor, al mismo tiempo, CR es 5 veces mayor, ver Tablas 4.15, 4.17, 4.19 y 4.21, idem Experimentos del grupo color.

Como podemos apreciar en las Figuras 4.8 a 4.11, la segunda (codificada y decodificada con JPEG) y la tercera (codificada y decodificada con JPEG+Super-compresión) desde arriba, tienen la misma apariencia y calidad visual que la de arriba, i.e., la imagen original de *Lena*.

Experimentos: JPEG2000 vs SC (JPEG2000+SR)
Idénticas consideraciones a las formuladas para los Experimentos en color son necesarias, ver Tablas 4.16, 4.18, 4.20 y 4.22 y Figuras 4.8 a 4.11, con las mismas conclusiones respecto a la diferencia entre JPEG y JPEG-2000 para comprimir este tipo de imágenes (compare bpp y CR entre las Tablas de finalización impar y par de la 4.15 a la 4.22, es decir, 4.15 y 4.16, 4.17 y 4.18, y así).

4.4.3 Para ambos grupos

Usamos la memoria de Texturas de la GPGPU en el interior del STB [294] para una implementación computacional eficiente de la máscara convolutiva bidimensional del módulo de deblurring, permitiéndonos alcanzar tiempos televisivos, i.e., un cuadro cada 40 milisegundos.

Finalmente, todas las técnicas de supercompresión de video fueron previamente implementadas en MATLAB® R2011a (Mathworks, Natick, MA) [319] y el middleware Jacket® 1.7 (Accelereyes) sobre una Notebook con un procesador Intel® Core(TM) i5 CPU M 520 @ 2.40 GHz y 6 GB RAM sobre Microsoft® Windows 7© 64 bits, y posteriormente sobre una desktop en OpenCL y CUDA© de NVIDIA® [300] sobre NVIDIA® Quadro SDI Capture, Quadro 6000 y Quadro SDI Output GPUs para el codificador, e idéntica configuración para el decodificador, como se muestra en la Fig.3.14 del Capítulo 3.

Mientras que para las simulaciones de imágenes fijas se empleó también MATLAB R2011a junto al middleware Jacket 1.7, con una GPGPU incorporada a una notebook DELL Alienware M17x Model Alienware M17x (m17x10-1847DSB) Part# m17x10-1847DSB con procesadores Intel Core 2 Extreme Quad QX9300 2.53GHz y una tarjeta gráfica NVIDIA GeForce GTX 260M, 4GB de RAM, con 112 CUDA cores a 462 Gigaflops c/u.

Capítulo 5

Conclusiones y futuros trabajos

Este capítulo atañe a las conclusiones formuladas a partir de las distintas aplicaciones en las que se ha probado el concepto de supercompresión y de sus futuras posibilidades.

Las aplicaciones propuestas en el Capítulo 3, las cuales fueron llevadas a cabo en su totalidad exitosamente nos demuestran lo siguiente:

1. La ductilidad de la innovación para adaptarse a distintos escenarios de compresión de señales de TV Digital en las diferentes configuraciones propuestas, así como para la *compresión adicional* en el caso de Cine Digital, sin dejar de mencionar sus potenciales aplicaciones en imágenes médicas del tipo tomografía computada o resonancias magnéticas del tipo seriadas bajo el protocolo DICOM 3.0 [320].

2. La innovación no es esencialmente un compresor, sino más bien, un catalizador de compresión (o precursor de compresión) el cual permite al sistema de compresión en su conjunto alcanzar tasas de compresión antes no conocidas para este nivel de calidad visual y de reconstrucción. Es por esta razón que en el apartado anterior no hablamos de compresión a secas, sino de *compresión adicional*.

3. Consecuente con lo dicho hasta aquí, la única forma de alcanzar tales niveles finales de compresión, manteniendo intacta la calidad de reconstrucción era tocando el último parámetro de las imágenes, es decir, la resolución. Razón por la cual, la innovación pasa a formar parte del arsenal de herramientas que constituyen el nuevo Sistema Argentino de TV Digital y conocido como Codificador por Control de Resolución (en inglés, Resolution, Control Coding: RCC), que es como se presentará ante la Unión Internacional de Telecomunicaciones (ITU) [321].

4. Para que pudiéramos alcanzar el éxito, primero, y en estos niveles, segundo, se tuvieron que alinear varios planetas, a saber:

    4.1. El aporte posee una muy baja complejidad computacional, por lo cual no incrementa sensiblemente la complejidad computacional del proceso completo en su conjunto.

    4.2. El aporte es fácilmente paralelizable, lo que permite su inmediata codificación en las placas GPGPUs.

4.3. Todos los algoritmos de compresión son esencialmente asimétricos en lo que respecta al balance de carga de costo computacional. El aporte acentúa ese proceso, pero dado que el mayor insumo de recursos computacionales se da en el codificador, permite su uso práctico en el mercado en cualquier escenario, en general, y TV Digital en particular, dado el bajo costo que implica embeber el decodificador en un chip simple y barato para ser alojado en el set-top-box de costo accesible.

5. Este trabajo puede ser perfeccionado, en la medida que se empleen mejores técnicas de filtrado antialiasing, super-resolución y compresión.

6. Los mayores aportes de este trabajo lo constituyen, a saber:

6.1. La introducción del concepto de catalizador de compresión, el cual no consiste en un proceso de compresión en sí mismo (como tradicionalmente se entiende), sino que se trata más bien del procedimiento mediante el cual se permite que los verdaderos algoritmos de compresión empleados compriman más y mejor.

6.2. La introducción del concepto de super-compresión = super-resolución + compresión. Este principio es el que permitirá alcanzar niveles de compresión combinada (producto de la multiplicación de las tasas individuales de compresión) como nunca antes pudieron alcanzar ningunos de los algoritmos mencionados en el Capítulo 1, sin degradar sensiblemente la calidad visual de la recuperación de la imagen reconstruida.

6.3. Una nueva y mejor máscara de realce, cuyos parámetros de diseño fueron estratégicamente obtenidos mediante el algoritmo genético del Capítulo 3, y que resulta en el mejor complemento visual de todo el proceso, con prestaciones superiores a todas las máscaras de realce conocidas hasta la fecha.

6.4. La combinación de todos los aportes anteriores da lugar a la primera transmisión de la historia de 1080p-3D en cualquier norma en la modalidad TV Digital terrestre, situando a este trabajo como un hito tecnológico en materia de TV Digital y sus potenciales aplicaciones.

6.5. Todos los logros del presente trabajo para TV Digital en modalidad terrestre son absolutamente reusables en transmisiones de TV Digital sobre redes ópticas del tupo de **Argentina Conectada**, permitiendo así transmisiones de TV Digital de Ultra Alta Resolución, Cine Digital y Telemedicina, aprovechando de forma dramática el ancho de banda de dicha red.

Apéndice A

Algunos elementos de compresión de imágenes con pérdidas mediante transformada

A.1 Introducción

En este apéndice se desarrollan las herramientas relativas a la compresión de imágenes con pérdidas en base a la codificación genérica por transformada en forma complementaria a las desarrolladas en el Capítulo 1. Esto quiere decir, que se analizarán las que en el mismo quedaran pendientes, a saber:

• Una noción de codificación genérica por transformada, sus aspectos más relevantes y propiedades
• Cuantización, sus variantes y la versión más apropiada para esta tesis
• Compresión entrópica, en particular de Huffman y Aritmética.

A.2 Elementos constitutivos

A.2.1 Codificación genérica por transformada

El sistema de compresión de imágenes con pérdidas no recupera la imagen original pixel a pixel. En su lugar toma ventaja de las limitaciones del ojo humano mediante las cuales el sistema visual tiende a aproximar la imagen recuperada a la original. Estos métodos pueden alcanzar tasas de compresión bastamente superiores que los métodos sin pérdidas, no obstante, los mismos deben ser utilizados con cuidado [322, 323]. Las técnicas de compresión con pérdidas generalmente solo trabajan bien con fotografías de la vida real; las mismas dan frecuentemente pésimos resultados con otros tipos de imágenes tales como las binarizadas, o texto. Sometiendo una imagen a través de varios ciclos de compresión-descompresión con pérdidas sucederá que la imagen se degradará más allá de los estándares aceptables. Por lo tanto, una compresión con pérdidas debería ser usada exclusivamente luego de que todos los otros eventuales procesos aplicados a la imagen ya han tenido lugar, es decir, no debería ser empleado como un formato de almacenamiento intermedio. Solo el ojo humano les asignará una apariencia similar a la imagen recuperada y a la original. En cambio, si una computadora posee un sistema de reconocimiento de imágenes, entonces, la misma reconocerá las diferencias entre ambas [324]. En otras palabras, considere el codificador de transformada genérica de la Fig.A.1 el cual consiste de una transformada bi-dimensional, un cuantizador, y un codificador entrópico (siendo oportuno aclarar en

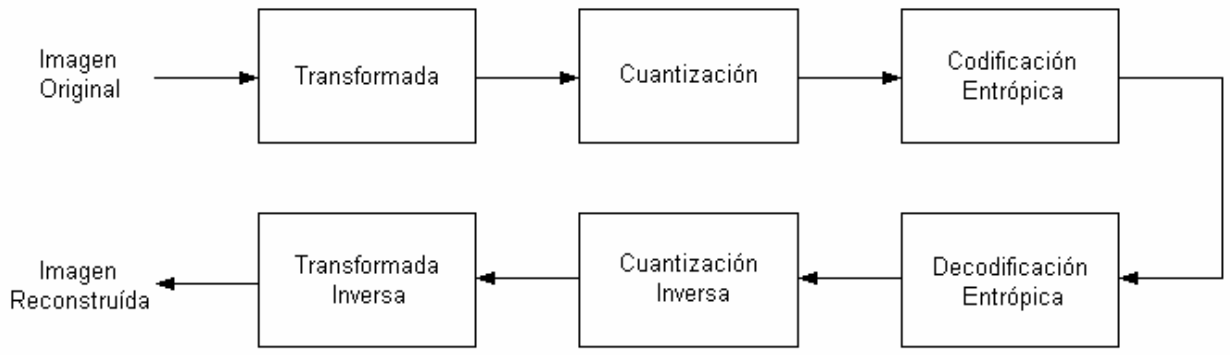


Figura A.1: Codificación genérica por transformada, para imágenes digitales.

este punto que entre la transformada y la cuantización se descartan coeficientes poco significativos con algún criterio de agresividad). Nosotros podemos observar en este punto que las pérdidas ocurren durante la cuantización y luego de la transformada. Por lo tanto, en orden a conducir a buen puerto nuestro análisis, debemos repetir la transformada para retornar al estado donde las pérdidas ocurren y examinar los efectos de la cuantización sobre los coeficientes de la transformación [325, 326].

A.2.1.1 Esquema de compresión

La codificación por transformada es un esquema de compresión donde se aplica una transformación $\boldsymbol{T}$ a un conjunto de datos dados $\{\boldsymbol{x}_i\}_{i=1\ldots n} \subset \Re^m$ en orden a obtener una nueva representación $\{\boldsymbol{y}_i\}_{i=1\ldots n} \subset \Re^m$ que resulta tener mejores propiedades con respecto a la cuantización y codificación:

$$\boldsymbol{y} = \boldsymbol{T}\,\boldsymbol{x}, \quad \boldsymbol{y} = \{\boldsymbol{y}_i\}_{i=1\ldots n}, \quad \boldsymbol{x} = \{\boldsymbol{x}_i\}_{i=1\ldots n}, \tag{A.1}$$

La compresión de imágenes que emplea codificación por transformada se constituye usualmente como un esquema de compresión con pérdidas. Esto significa que la imagen original puede ser reconstruida a partir de su código solo con cierta exactitud. El error de reconstrucción recibe el nombre de *distorsión*. Como se muestra en la Fig.A.2, la codificación por transformada consiste básicamente de los siguientes pasos:

1. la imagen es segmentada en mosaicos, cada mosaico es representado por un vector resultante en la representación de la imagen $\boldsymbol{x}$.
2. Una transformación $\boldsymbol{T}$ es aplicada a los datos vectoriales.
3. Los coeficientes resultantes de la transformación son cuantizados usando cuantización escalar (la cual se desarrollará en detalle en este apéndice), es decir, el codificador con pérdidas $\boldsymbol{Q}_e$ es aplicado sobre la transformada, el cual resulta en una representación discreta $\boldsymbol{Q}_e(\boldsymbol{Tx})$.
4. La transformada discretizada es codificada mediante una función $\boldsymbol{C}$ en una cadena de bits, es decir, el código. La representación de la imagen $\boldsymbol{C}(\boldsymbol{Q}_e(\boldsymbol{Tx}))$ puede entonces ser almacenada o transmitida a un receptor.
5. Para reconstruir una imagen desde su código, la transformada discretizada $\boldsymbol{Q}_e(\boldsymbol{Tx})$ es reobtenida desde el decodificador.
6. La función de decodificación $\boldsymbol{Q}_d$ del cuantizador es aplicada, reconstruyendo la transformada distorsionada $\boldsymbol{Q}_d(\boldsymbol{Q}_e(\boldsymbol{Tx})) = \boldsymbol{Q}(\boldsymbol{Tx})$.
7. La transformación inversa $\boldsymbol{T}^{-1}$ reconstruye el vector de representación distorsionado de la imagen $\tilde{\boldsymbol{x}} = \boldsymbol{T}^{-1}(\boldsymbol{Q}(\boldsymbol{Tx}))$.
8. Finalmente, la imagen es reconstruida desde su vector de representación.

En lugar de la aproximación de codificación de transformada presentada, la que generalmente es la mejor forma de codificar los datos vectoriales (sin pérdidas), es decir, los mosaicos de la imagen, debería ser en general una codificación vectorial o esquema de cuantización vectorial, ver la Sección 1.3.3 del Capítulo 1. No obstante, dado que la dimensión de su codebook (tabla de codificación) sería inmensa y dado que este resulta necesario para representar una imagen y teniendo en cuenta que la complejidad computacional de la búsqueda en el codebook es inaceptable, el método descrito resulta ser inaplicable en la práctica [324]. Entonces, el método de

compresión de imágenes con pérdidas más ampliamente usado (usado por ejemplo en el formato JPEG estándar [17-24], el cual es desarrollado en el Apéndice B) resulta ser el de aplicar una transformación y – más o menos – independientemente cuantización escalar y compresión entrópica sobre los componentes resultantes.

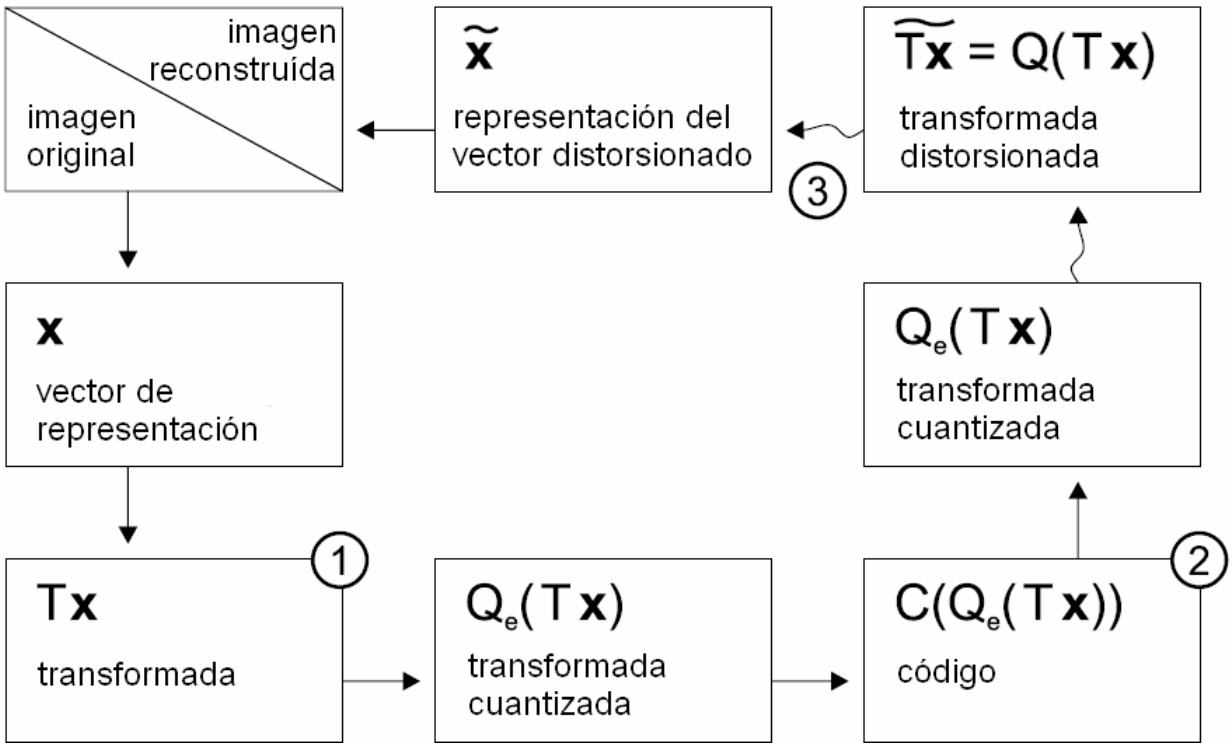


Figura A.2: Pasos de la codificación por transformada.

En la mayoría de los casos, las transformaciones lineales son usadas para reducir la cantidad de redundancia en la representación de la señal, por ejemplo, la TDKL, la TDC o la TDO, siendo estas dos últimas independiente de los datos. Esto resulta en un código más corto que la representación canónica que si nosotros aplicamos cuantizadores escalares independientes a los coeficientes. Hace varios años, el objetivo de la codificación por transformada se orientó a la recorrelación de los datos en pos a reducir las redundancias. Más tarde se demostró que la decorrelación, de hecho, conduce a la transformación óptima – al menos en el caso de señales de entrada Gaussianas en un límite de resolución elevado [325]. No obstante, la recorrelación no es ni necesaria ni suficiente para la codificación de transformada óptima en general.

Como se describió en el *paso 1*, en función de poder realizar la transformación sobre los datos de la imagen, vamos a segmentar a la misma en mosaicos de *nfm* x *ncm* píxeles y reordenar la intensidad de los valores de los píxeles para obtener una representación vectorial $\boldsymbol{x} \in \Re^{\boldsymbol{m}}$, donde $m = nfm \times ncm$ (ver Fig. A.3). Inversamente, podemos reconstruir la imagen a partir de su vector de representación, dada la dimensión de la imagen, es decir, el número de filas y columnas de los mosaicos de la imagen. La performance del esquema de compresión depende principalmente de la transformada usada. Optimizando la transformación *T* para compresión de imágenes, se minimiza la longitud del código, es decir, de la transformada cuantizada codificada

$$\boldsymbol{y}_{CQ} = \boldsymbol{C}(\boldsymbol{Q}(\boldsymbol{T}\boldsymbol{x})) \tag{A.2}$$

Mientras que al mismo tiempo se minimiza la distorsión

$$\boldsymbol{D} = \boldsymbol{d}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \ \boldsymbol{para} \ \tilde{\boldsymbol{x}} = \boldsymbol{T}^{-1}\left(\boldsymbol{Q}^{-1}\left(\boldsymbol{C}^{-1}\boldsymbol{Y}\right)\right) \tag{A.3}$$

donde *C* representa al codificador, *Q* al cuantizador y *d* es la medida de la distancia. Un paso de cuantización es en la mayoría de los casos necesario dado que en general la transformación retorna valores no-enteros, incluso tratándose de fuentes discretas. El codificador *C* convierte los valores entregados por el cuantizador a una cadena binaria de bits decodificable en forma única. Se debe notar que la distorsión resulta del paso de cuantización y algunas veces de una transformación no estrictamente invertible. Discutiremos tanto la cuantización como la codificación más adelante en el presente apéndice.

Dependiendo de diferentes suposiciones acerca de la cuantización y la subsecuente codificación de los coeficientes de la transformada, se pueden presentar diferentes criterios de performance para las transformaciones [326], mediante los cuales se puede realizar la transformación sin especificar cuantizadores y codificadores concretos. Esto significa que este tipo de análisis se realiza directamente sobre las entradas transformadas (*paso 1* en la Fig.A.2)

Existen muchas alternativas a los efectos de implementar diferentes cuantizadores, así como las medidas de la longitud del código resultante. Por lo tanto, podemos analizar la representación de la imagen en el *paso 2* de la Fig.A.2. Para una longitud de código dado, es decir, una limitación directa de la exactitud de cuantización, compararemos entonces la distorsión de las imágenes reconstruidas de las transformadas probadas.

A.2.1.2 Propiedades de las transformadas

Como se puede apreciar en la Fig.A.3 la codificación por transformada implica la aplicación de un proceso multi-etapas el cual se encuentra sujeto a una serie de propiedades. No obstante, dicho

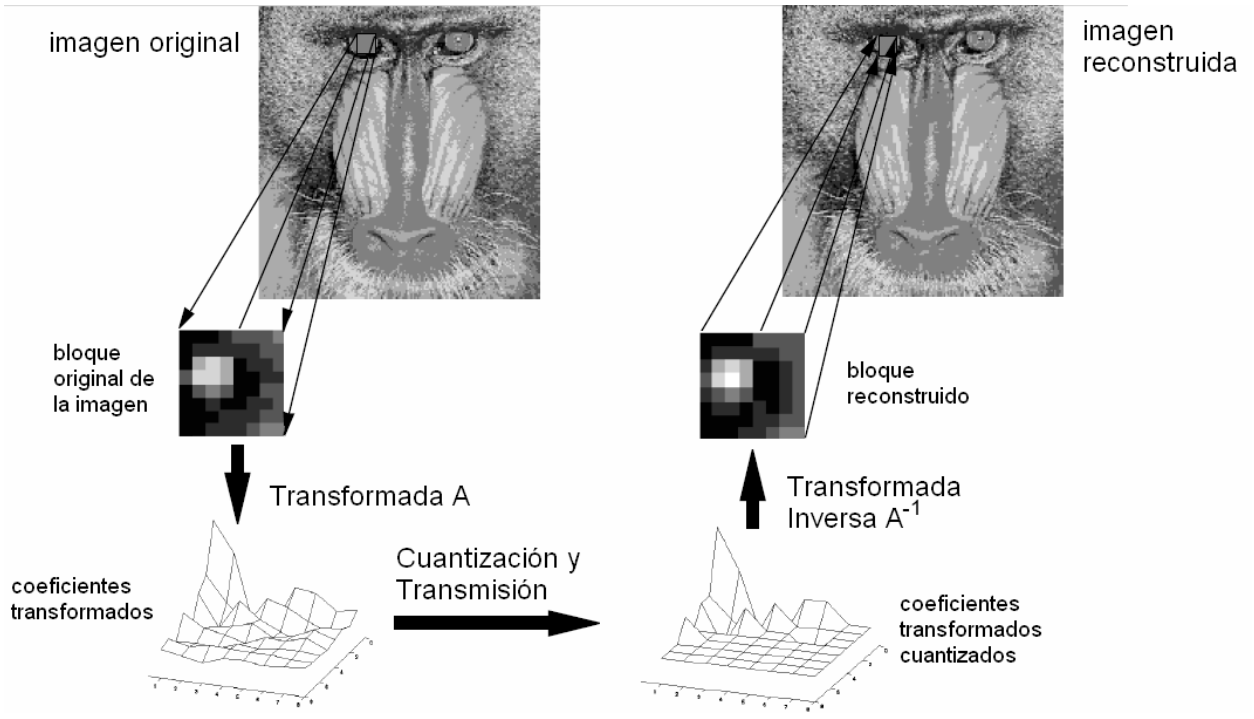


Figura A.3: Codificación por transformada.

proceso se inaugura con una transformación directa del tipo

$$\boldsymbol{y} = \boldsymbol{A}\,\boldsymbol{x} \tag{A.4}$$

donde las dimensiones involucradas (según el Capítulo 1) serán: $\boldsymbol{y} \in \Re^{nm\times(nfm\times ncm)}$, $\mathbf{A} \in \Re^{nm\times nm}$ y $\boldsymbol{x} \in \Re^{nm\times(nfm\times ncm)}$. Siendo los elementos constitutivos de $\boldsymbol{x}$, los bloques en los cuales se sub-dividió a la imagen original $\boldsymbol{I}$ organizados como columnas.

EL proceso se completa con una transformación inversa del tipo [327]

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\ \boldsymbol{y} = \boldsymbol{A}^{T}\,\boldsymbol{y} \tag{A.5}$$

De más está decir, que se cumple estrictamente con el principio de linealidad, mediante el cual $\boldsymbol{x}$ es representado como una combinación lineal de las *funciones base*.

Por otra parte, el Teorema de Parseval sostiene que una transformada es una rotación del vector señal alrededor del origen de un espacio vectorial – en este caso y en función de las dimensiones involucradas, de dimensiones – (*nfI/nfm*) x (*ncI/ncm*) [2].

Una transformación lineal es separable, si la transformada de un bloque de la señal de dimensiones (*nfI/nfm*) x (*ncI/ncm*) puede ser expresada por

$$\boldsymbol{y} = \boldsymbol{A}\,\boldsymbol{x}\,\boldsymbol{A}^{T} \tag{A.6}$$

siendo $\boldsymbol{y}$ la matriz de coeficientes transformados y $\boldsymbol{A}$ la matriz de transformación ortonormal.

La transformación inversa será entonces

$$\boldsymbol{x} = \boldsymbol{A}^{T}\ \boldsymbol{y}\,\boldsymbol{A} \tag{A.7}$$

Es de vital importancia en la práctica el hecho de que la transformada requiere dos multiplicaciones de matrices de dimensiones (*nfI/nfm*) x (*ncI/ncm*) en lugar de una única multiplicación de un vector de dimensión *1* x ((*nfI/nfm*)x(*ncI/ncm*)) con una matriz de dimensión ((*nfI/nfm*)x(*ncI/ncm*)) x ((*nfI/nfm*)x(*ncI/ncm*)) obteniendo se así una reducción en la complejidad de O((*nfI/nfm*)x(*ncI/ncm*)x(*nfI/nfm*)x(*ncI/ncm*)) a O((*nfI/nfm*)x(*ncI/ncm*)x(*nfI/nfm*)), asumiendo para este último caso que *nfI = ncI* y *nfm = ncm.*

Otro aspecto importante es que la transformada bidimensional puede ser implementada como dos transformadas unidimensionales a lo largo de las filas y columnas del bloque de señal (Fig.A.4).

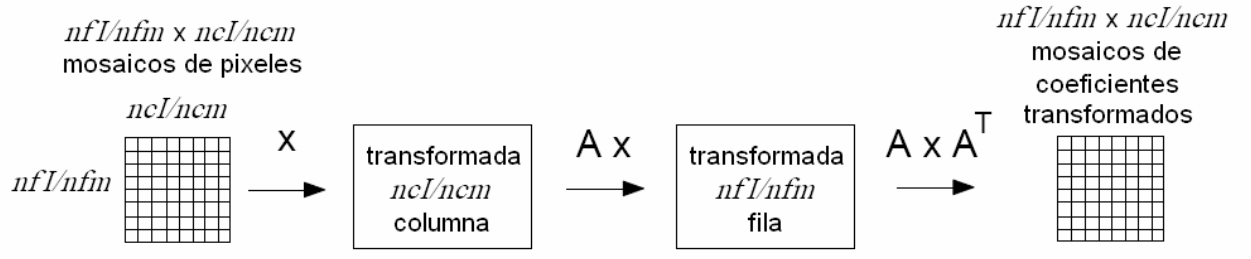


Figura A.4: Transformada ortonormal separable.

Las ecuaciones A.4 a A.7 constituyen las propiedades de las transformadas involucradas en el presente trabajo.

Los criterios de selección de una transformada particular son:

- Decorrelación, concentración de la energía (por ejemplo, TDKL, TDC, otras)

- Funciones base visualmente agradables (por ejemplo, ruido pseudo-aleatorio, *m*-secuencias, transformadas solapadas). Ver. Fig.A.5 [322, 328-335], donde TH es la Transformada de Haar, TWH es la Transformada de Walsh-Hadamard y TS es la transformada de Slant [2]. Para todos los casos las funciones base son consideradas en relación a bloques de 8 x 8.

- Baja complejidad computacional.

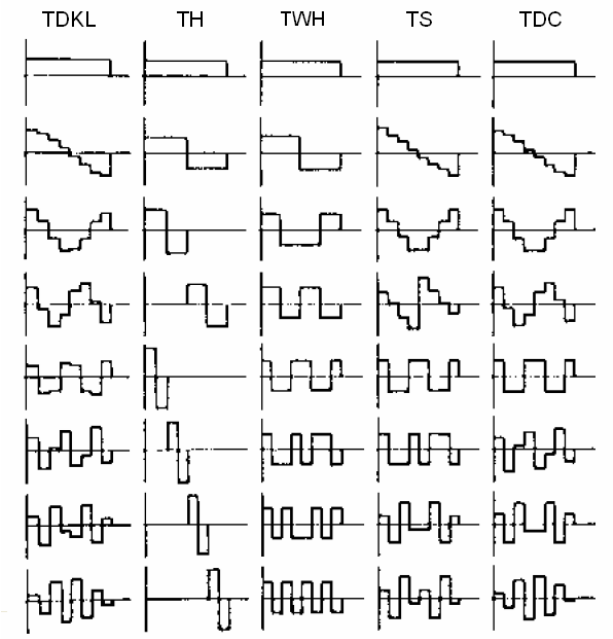


Figura A.5: Bases de varias transformadas.

La concentración de la energía medida para imágenes naturales típicas, para bloques de medida 1x32 [2], puede observarse en la Fig.A.6 donde la TDKL es óptima y la TDC solo es superada por la TDKL.

Finalmente en la codificación por transformada adaptativa (Fig.A.7) la cuantización y la codificación entrópica son optimizadas separadamente para cada clase.

Las clases típicas serán:
• Bloques sin detalles
• Estructuras horizontales
• Estructuras verticales
• Diagonales
• Texturas sin una orientación predilecta

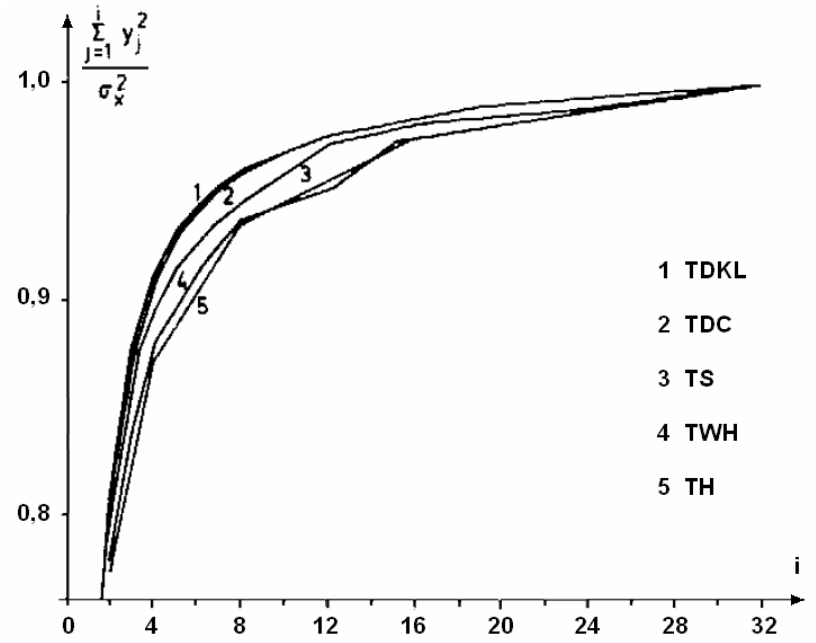


Figura A.6: Concentración de energía medida para varias transformadas.

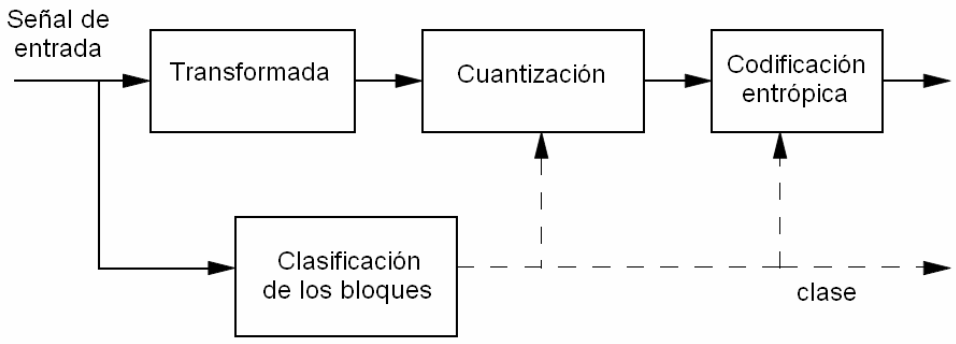


Figura A.7: Codificación por transformada adaptativa.

A.2.2 Paso de cuantización en compresión de imágenes con pérdidas.

La cuantización reduce la profundidad en bits de los coeficientes de los bloques a expensas de la precisión. Dos diferentes procedimientos de cuantización están permitidos en estos casos:

a) Cuantización escalar, y
b) Cuantización basada en el código de Trellis

La cuantización escalar consiste de un simple truncamiento de los bits menos significativos, frecuentemente obtenidos por un desplazamiento a la derecha de la magnitud de los coeficientes de los bloques [336]. Los coeficientes difieren solo en los dígitos que se eliminaron siendo esto indistinguible luego de la decuantización [336]. Esto resulta en una función truncada de a pasos como la mostrada en la Fig.A.8. La cuantización escalar sufre del así llamado problema de la *zona-muerta* [336]. A saber, si el paso de cuantización Δ_b es usado por un bloque *b*, los coeficientes cuya magnitud cae por debajo del umbral son convertidos a cero. Entonces la información acerca de los coeficientes en una esfera de radio Δ_b alrededor de cero se perderá completamente,

ver Fig.A.8. Siendo $p[n]$ el número de desplazamientos a la derecha para el coeficiente *n-ésimo*, $s[n]$, en el *b-ésimo* bloque, entonces, la cuantización escalar puede ser expresada por la fórmula:

$$\boldsymbol{q}^{p[n]}[\boldsymbol{n}] = \boldsymbol{signo}(\boldsymbol{s}[\boldsymbol{n}]) \times \left\lfloor \frac{|\boldsymbol{s}[\boldsymbol{n}]|}{\Delta_{\boldsymbol{b}}^{p[n]}} \right\rfloor \tag{A.8}$$

donde:

$\lfloor \bullet \rfloor$ significa "*redondear hacia abajo*", y

$$\Delta_{\boldsymbol{b}}^{p[n]} = 2^{p[n]} \times \Delta_{\boldsymbol{b}} \tag{A.9}$$

La ecuación de decuantización será:

$$\hat{\boldsymbol{s}}[\boldsymbol{n}] = \begin{cases} \left\lceil \left(\boldsymbol{q}^{p[n]}[\boldsymbol{n}] - \boldsymbol{r} \times 2^{M_b - p[n]} \right) \times \Delta_{\boldsymbol{b}}^{p[n]} \right\rceil & \boldsymbol{q}^{p[n]}[\boldsymbol{n}] < 0 \\ 0 & \boldsymbol{q}^{p[n]}[\boldsymbol{n}] = 0 \\ \left\lfloor \left(\boldsymbol{q}^{p[n]}[\boldsymbol{n}] + \boldsymbol{r} \times 2^{M_b - p[n]} \right) \times \Delta_{\boldsymbol{b}}^{p[n]} \right\rfloor & \boldsymbol{q}^{p[n]}[\boldsymbol{n}] > 0 \end{cases} \tag{A.10}$$

donde:

$\lceil \bullet \rceil$ significa "*redondear hacia arriba*", y

M_b es el número máximo de bits en la magnitud de los coeficientes y $\boldsymbol{r} \in [0,1)$ es usualmente establecido como $\boldsymbol{r}$ = 0.5. La cuantización basada en el código de Trellis es mucho más complicada pero usualmente da mejores resultados y no sufre del problema de la *zona-muerta*. Para más detalles acerca de esta cuantización se puede recurrir a [336].

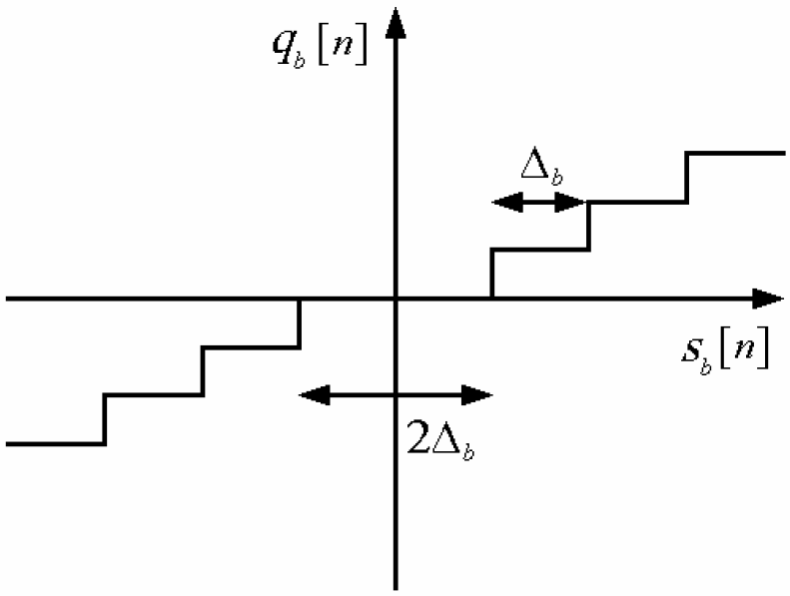


Figura A.8: Cuantización escalar: Una *zona-muerta* de $2 \times \Delta_{\boldsymbol{b}}$ es generada entorno de cero. $s_b[n]$ indica el valor de entrada, mientras que $q_b[n]$ es la señal cuantizada de salida.

A.2.3 Compresión entrópica o remoción de la redundancia

La compresión entrópica no produce pérdidas en el esquema de compresión planteado hasta el momento. Por otra parte, la compresión entrópica asigna a cada símbolo un código de longitud promedio igual a la entropía de la fuente [337]. Los compresores entrópicos constan de dos partes fundamentales [337] cuya interacción se describe gráficamente en la Figura 9:

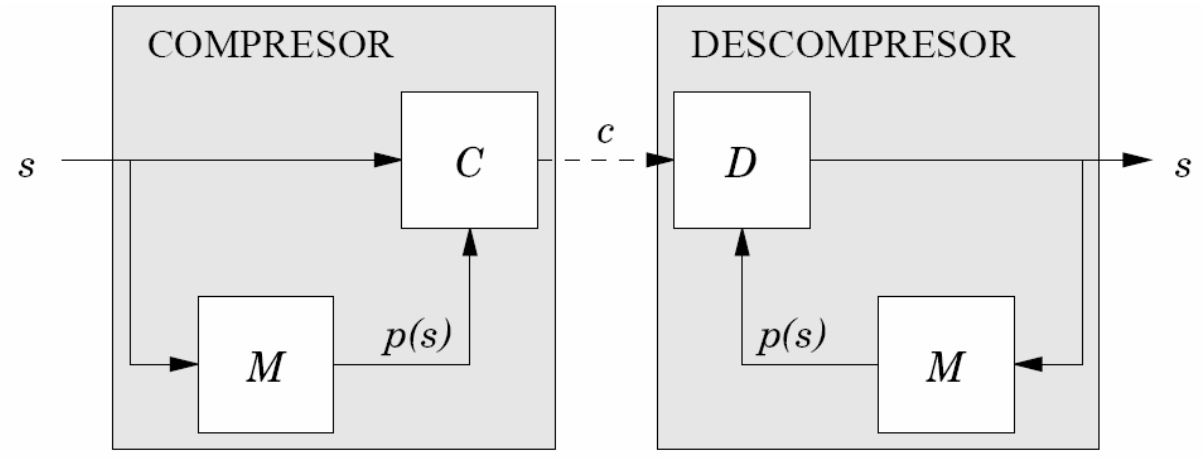


Figura A.9: Modelo de codificación/decodificación entrópica.

- Un modelo probabilístico *M* que asigna a cada símbolo de entrada *s* una probabilidad de ocurrencia *p*(*s*), bajo ciertas restricciones [337], es decir, dado que el codificador tratará de asignar al símbolo-fuente *s* codificado $-\log_2 p(s)$ bits de código. Para que la codificación sea posible, debe ocurrir que $0 < p(s) < 1$:

  1. Si $p(s) = 0$, el símbolo-fuente sería codificado con una palabra-código de longitud infinita. Además, *s* no pertenecería al alfabeto-fuente.

  2. Si $p(s) = 1$, la palabra-código generada obtendría una longitud de 0 bits y el decodificador sería incapaz de detectarla. Por otra parte, el alfabeto fuente sólo contendría un símbolo-fuente.

- Un codificador *C* que realiza las funciones de traductor, capaz de asignar a cada símbolo de entrada *s* un código *c* de longitud ideal igual a la expresada por

  $$\boldsymbol{I}_s(\boldsymbol{c}_i) = -\log_s \boldsymbol{p}(\boldsymbol{a}_i) \qquad \text{(A.11)}$$

  Donde las longitudes de las palabras-código tienen que ser igual a la información asociada al símbolo-fuente representado. En otras palabras, la longitud debe ser igual a la entropía de la fuente expresada en bits de código/símbolo. El traductor inverso o decodificador *D* recupera la representación original de los símbolos, conociendo la misma información de contexto que el traductor directo, que es proporcionada por un modelo probabilística idéntico al del descompresor.

La clave para obtener el máximo rendimiento en un compresor entrópico es la estimación exacta de las probabilidades de los símbolos. Si es demasiado baja, el código asignado aumenta de longitud y como el símbolo asociado ocurre en más ocasiones de las esperadas, la longitud de la

secuencia-código aumenta. Por otra parte, si la probabilidad es demasiado alta, estaremos codificando un símbolo que no es tan frecuente como creemos mediante un código muy corto, a costa de alargar la longitud de los códigos asignados al resto de símbolos que tendrán asignada una probabilidad menor que la real.

El rendimiento de los compresores entrópicos depende tanto de los codificadores de longitud variable como de los modelos probabilísticos usados, pero su independencia es tal que pueden ser estudiados por separado. Por otra parte, los compresores entrópicos en general son más lentos que los basados en diccionarios. Esto se debe a que ahora se busca la máxima compresión posible, incluso a costa de desarrollar algoritmos pesados que permitan aprovechar ese "grado de optimalidad" [337], y que los compresores basados en diccionarios no explotan sobre secuencias de longitud finita.

Los conceptos e ideas introducidos en esta sección son fundamentales para el diseño de los compresores de imágenes reversibles que serán analizados en el siguiente capítulo. Sin embargo, en éste se analizarán aquellos métodos de compresión que pueden ser considerados como el "estado del arte" en compresión de texto [338-358].

A.2.3.1 Capacidad de un canal discreto sin ruido

En la teoría de la información, el canal de comunicación constituye el enlace entre el codificador del emisor y el decodificador del receptor. Un canal discreto transmite símbolos-código (bits de código cuando sea binario, como casi siempre ocurre). Si además no existe ruido de transmisión, a la salida tenemos los mismos símbolos-código y en la misma secuencia (orden) que a la entrada, transcurrido un cierto tiempo. Puesto que no existen alteraciones durante la transmisión, podemos afirmar que la cantidad de información de salida es igual a la de entrada.

En el contexto de la compresión de datos, el canal simboliza el dispositivo de transmisión o de almacenamiento. Por tanto, la compresión de datos va a ser óptima si aprovechamos al máximo la capacidad de transmisión del canal.

Se define la capacidad de un canal como la máxima cantidad de información que puede transmitirse en cierto intervalo de tiempo. Todos los canales, incluso los ruidosos, pueden transmitir una cantidad de información infinita durante un periodo de tiempo ilimitado, así que lo normal es hablar de cantidad de información transmitida por unidad de tiempo. Este es el concepto de capacidad de un canal manejado en muchos ámbitos de transmisión de datos, independientemente del tipo de canal (analógico o digital).

Sin embargo, para definir la capacidad de un canal discreto, se puede hablar de capacidad en términos de la cantidad de información transmitida por símbolo-código o palabra código enviada. Cuando la duración o longitud de los símbolos-código es constante y tenemos un total de s símbolos-código, la capacidad de un canal puede calcularse como

$$\boldsymbol{W} = \log_2 \boldsymbol{s} \qquad \text{(A.12)}$$

cantidad medida en bits de información/bits de código. En el caso de trabajar con códigos y fuentes binarias, la capacidad de un canal (sin ruido) es de 1 bit de información/bit de código transmitido. En este caso, se está utilizando un código óptimo desde el punto de vista de la compresión de datos.

Es interesante resaltar que las dimensiones de la capacidad de un canal y de la eficiencia de un código coinciden.

A.2.3.2 Un codificador entrópico universal

Antes de comenzar a estudiar los diferentes codificadores entrópicos usados en compresión de datos, vamos a adelantar la idea que es la esencia de todos ellos. Esto nos va a permitir comprender con más facilidad cómo funcionan, qué esperar de ellos y en qué se parecen todos los compresores entre si.

Como fue expresado en la sección anterior, un código no es redundante si cada bit de información es representado exactamente por un bit de código. Los compresores basados en diccionarios no estiman las probabilidades de los símbolos y por lo tanto es difícil entender que la afirmación anterior también es cierta en este caso, pero así es. Cuando las cadenas son sustituidas por índices, lo que en realidad se hace es codificar un gran símbolo-fuente compuesto por varios caracteres usando un código de longitud constante de al menos 1 bit.

El mismo efecto puede obtenerse en el caso de la codificación entrópica porque cada carácter se codifica usando un código de longitud variable. Ahora estimamos las probabilidades de los símbolos (caracteres) y construimos un código de longitud igual al número de bits de información correspondientes.

Por lo tanto, el objetivo de todo codificador es encontrar una representación lo más compacta posible para un determinado símbolo *s*. En teoría, debería ser posible asignar fracciones de bit de código dependiendo de su probabilidad de ocurrencia. La ventaja principal de los compresores basados en diccionarios sobre los entrópicos es que al extender la fuente no es necesario trabajar con fracciones de bits lo que produce el correspondiente aumento en velocidad. Bajo estas condiciones, sean los siguientes algoritmos: Algoritmo 1 de codificación y Algoritmo 2 de decodificación:

Algoritmo A.1

Mientras *s* no esté determinado sin incertidumbre (por el decodificador):

(a) Realizar una afirmación sobre *s* que permita al decodificador averiguar algo acerca de la identidad de *s*. Intentar que dicha afirmación tenga tantas posibilidades de ser cierta como de ser falsa.

(b) Emitir un bit de código indicando el resultado de la afirmación.

El decodificador asociado debería constar de los siguientes pasos:

Algoritmo A.2

Mientras *s* no esté determinado sin incertidumbre:

(a) Realizar la misma afirmación que el decodificador.

(b) Recibir un bit de código indicando el resultado de la información.

No es difícil apreciar que si se cumple la premisa de que la afirmación sea equiprobable, estaremos encontrando un código 100% eficiente. Sin embargo, lo que ocurre en la práctica es que formular afirmaciones ciertas exactamente en un 50% es muy complicado y por tanto el código contiene cierto nivel de redundancia.

Planteemos una implementación sencilla de los anteriores algoritmos. Supongamos que trabajamos con un alfabeto de 256 símbolos y que conocemos sus probabilidades ya que existe un modelo que las calcula. El paso no trivial que hay que describir es el de la formulación de la afirmación equiprobable. Supongamos que ordenamos los símbolos por su probabilidad de forma decreciente. Una afirmación lo más equiprobable posible sería: "el símbolo a codificar pertenece al conjunto formado por uno o más símbolos que se forma cuando recorremos la lista de símbolos y la suma de las probabilidades de todos ellos alcanza el valor más próximo a 0.5".

El decodificador puede hacer exactamente la misma predicción porque para formarla no se usa el símbolo codificado. Así, con la llegada del bit, el decodificador conoce si el símbolo está en el primer conjunto o en el segundo. El símbolo es encontrado cuando en el conjunto seleccionado sólo existe un símbolo.

Una forma eficiente de realizar las particiones (binarias) en nuestra lista ordenada es representarla en como un árbol binario en el que los símbolos más probables estén más cerca de la raíz y los más improbables estén lo más alejados de ésta como sea posible. En realidad, justamente esto es lo que hace la codificación de Huffman.

A.2.3.3 La codificación de Huffman

El código de Huffman [338] fue inventado por David A. Huffman en 1952 y desde entonces ha sido intensivamente utilizado en compresión de datos, porque genera un código de longitud entera instantáneo y óptimo.

A.2.3.4 El codificador

Huffman ideó un método para construir un árbol binario donde cada hoja representa a un símbolo, con la propiedad de que la distancia de una hoja s a la raíz del árbol es exactamente

$$\lceil -\log_2 \boldsymbol{p}(\boldsymbol{s}) \rceil \tag{A.13}$$

A continuación asignó un dígito binario a cada rama del árbol y diseño así un código de longitud variable que representa a los símbolos más probables con un código más corto y viceversa. El algoritmo de construcción del árbol de Huffman es el siguiente:

1. Crear una lista con tantos nodos como símbolos diferentes vayamos a codificar. Cada nodo representa a un símbolo diferente y almacena su probabilidad.

2. Mientras existan al menos dos nodos en la lista:

   (a) Extraer de la lista los dos nodos con menor probabilidad. Si existen más de dos, la elección es irrelevante.

   (b) Insertar en la lista un nodo que sea padre de los dos nodos extraídos, formando un árbol binario equilibrado de tres nodos. Este nuevo nodo tiene una probabilidad que es la suma de las probabilidades de los hijos y no representa a un símbolo en concreto.

La Fig.A.10 muestra un ejemplo de construcción de un árbol de Huffman. Por comodidad trabaja-remos con pesos enteros y no con probabilidades reales. Supongamos que codificamos 5 símbolos representados por A, B, C, D y E con pesos 15, 7, 6, 6 y 5 respectivamente. El primer

paso (Fig.A.10-a) consiste en crear una lista de 5 nodos. Gráficamente esta lista estaría formada por los nodos que queden en el nivel superior.

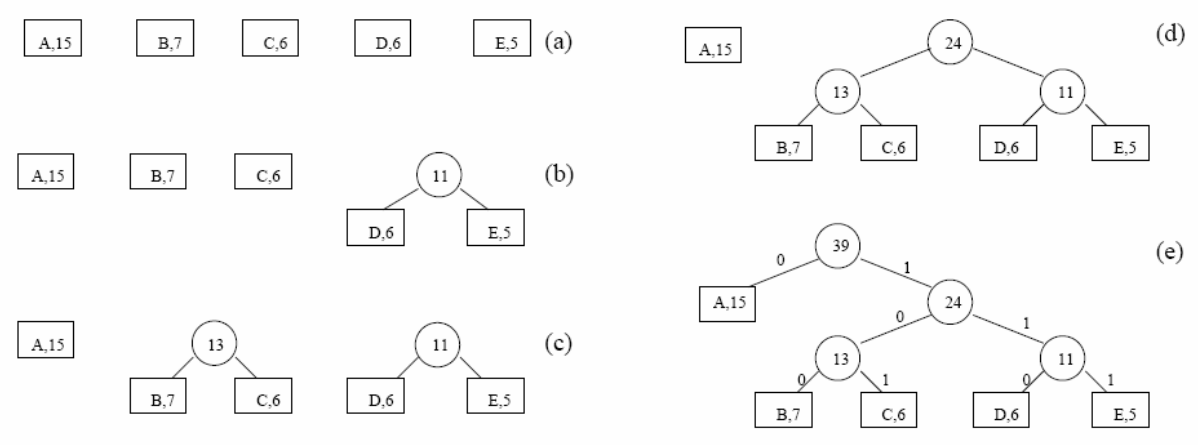


Figura A.10: Ejemplo de construcción de un árbol de Huffman.

A continuación extraemos los nodos con símbolos D y E, que son los de menor peso. También podríamos seleccionar el nodo C y el árbol de Huffman resultante sería distinto pero equivalente desde el punto de vista de la compresión de datos. Los pesos de D y E son 6 y 5 y por lo tanto, insertamos un nuevo nodo en la lista con peso 11 (Fig.A.10-b). Los dos siguientes nodos utilizados para construir un nuevo subárbol son el B y el C (Fig.A.10-c). A continuación, los dos subárboles forman un árbol mayor (Fig.A.10-d) y por último, el nodo A se procesa quedando el árbol de Huffman completo (Fig.A.10-e). El número de iteraciones necesarias para crear un árbol de Huffman que codifique N símbolos es N, ya que en cada iteración extraemos dos nodos de la lista e insertamos uno.

Una vez empleada la información que proporciona el modelo para construir el árbol es posible codificar y descodificar. La codificación de un símbolo consiste en emitir el código formado por la concatenación de las etiquetas de cada una de las ramas que es necesario recorrer desde la raíz hasta la hoja que representa al símbolo. Por ejemplo, si las ramas izquierdas se etiquetan con el bit de código 0 y las derechas con el 1 (ver Fig.A.10-e), el código de Huffman calculado es c_A = 0, c_B = 100, c_C = 101, c_D = 110 y c_E = 111.

A.2.3.5 El decodificador

La descodificación es justamente el proceso inverso. Un código describe con los bits que lo forman un camino desde la raíz hasta una hoja del árbol de Huffman. El símbolo descodificado es el asociado a la hoja.

Por ejemplo, supongamos que codificamos la secuencia de símbolos ABACDE. La secuencia de códigos resultante es (teniendo en cuenta el árbol de Huffman anteriormente construido): 01000101110111. Para descodificarla, inicialmente nos situamos en la raíz del árbol. Como el primer bit es un 0, el primer símbolo descodificado es una A (ver Fig.A.10). Para decodificar el siguiente símbolo nos situamos de nuevo en la raíz y utilizamos los bits 100 para llegar hasta el símbolo B. Este proceso se repite hasta que no existen más bits de código que descodificar.

A.2.3.6 Fuentes de información

Una fuente de información discreta es aquella que produce mensajes como sucesiones de símbo-

los-fuente (a las que llamaremos en general secuencias-fuente) extraídos de entre una colección finita de símbolos-fuente que conforman un alfabeto-fuente. La notación

$$\boldsymbol{A} = \{\boldsymbol{a}_1, \cdots, \boldsymbol{a}_r\} \tag{A.14}$$

indica un alfabeto-fuente A compuesto de r símbolos-fuente a_i. A r se le llama base de la fuente de información discreta. Nótese que hablar de bits de información no significa tomar $r = 2$.

En el ámbito de la teoría de la información, la emisión de símbolos para la construcción de mensajes se considera gobernada por un proceso estocástico, lo que quiere decir que la fuente de información produce los símbolos-fuente de acuerdo con unas ciertas probabilidades

$$\boldsymbol{P} = \{\boldsymbol{p}(\boldsymbol{a}_1), \cdots, \boldsymbol{p}(\boldsymbol{a}_r)\} \tag{A.15}$$

La probabilidad asociada a un símbolo-fuente debe ser mayor que cero, para que el símbolo-fuente tenga razón de existir como tal en el alfabeto-fuente (o de lo contrario nunca formaría parte de ninguna secuencia-fuente) e inferior a uno, para que existan al menos dos símbolos-fuente diferentes en el alfabeto-fuente.

Además se cumple que, teniendo una muestra (mensaje) razonablemente grande de la fuente de información, dicha muestra es representativa (estadísticamente hablando) del conjunto de todas las cadenas o secuencias de Markov que pueden ser producidas por la fuente, y por esta razón, diremos que se trata de un "proceso ergódico". Una fuente de información se dice ergódica si una muestra suficientemente corta de ella es representativa (estadísticamente hablando) de toda la información que es capaz de emitir. Gracias a esta propiedad, el famoso experimento realizado por Shannon que consistió en sintetizar mensajes escritos en inglés utilizando una fuente de este tipo, tras haber analizado suficientes secuencias de texto escrito, fue un éxito.

Los procesos de Markov nos hablan acerca de fuentes de información que generan los símbolos-fuente con una probabilidad que puede depender o no de los símbolos anteriormente producidos. Cuando, la probabilidad de un símbolo-fuente no depende del previamente emitido, hablamos de fuentes sin memoria, de fuentes de memoria nula o de fuentes de Markov de orden 0. Cuando ocurre lo contrario, hablamos de fuentes con memoria o de fuentes de Markov de orden K, donde K es el número de símbolos-fuente que forman el contexto que determina la probabilidad de generación del siguiente símbolo-fuente.

A.2.3.7 Compresión adaptativa

La longitud de los códigos de Huffman depende de la correcta estimación de las probabilidades de los símbolos. Existen fuentes que emiten los símbolos cuyas probabilidades permanecen constantes (fuentes ergódicas, ver Sección anterior) y por lo tanto puede usarse un único árbol de Huffman para codificarla eficientemente. El modelo probabilístico se calcula una única vez o es conocido de antemano tal como ocurre en la compresión de imágenes [359-361]. Los modelos que permanecen inalterables durante el proceso de compresión se llaman modelos estáticos.

Cuando las distribuciones de probabilidad no son conocidas de antemano, la utilización de modelos estáticos requiere que la secuencia a comprimir sea recorrida dos veces. En la primera pasada se calculan las probabilidades de los símbolos y en la segunda se codifica. El modelo debe ser comunicado de forma explícita al descompresor, típicamente en forma de cabecera previa a la secuencia de códigos.

El otro tipo de modelos probabilísticos que existen se llaman modelos dinámicos o adaptativos. En general proporcionan mejores tasas de compresión que los estáticos debido fundamentalmente a que las fuentes de información no son ergódicas. Sin embargo, también provocan procesos de compresión y descompresión más lentos, pues hay que emplear recursos para gestionar el modelo.

La ventaja más interesante de usar modelos adaptativos es que pueden estimar las probabilidades de los símbolos sin necesidad de realizar dos pasadas (diremos en estos casos que la secuencia puede ser tratada como una corriente o stream), sin embargo, cuando la fuente es ergódica, los modelos estáticos son realmente eficientes, su rendimiento es comparable al de los no adaptativos [339-341].

La idea que hay detrás de la compresión entrópica adaptativa es la misma que la que usan los compresores basados en diccionarios: codificar usando sólo la información que hasta ese instante conoce el descompresor. El Algoritmo A.3 de compresión adaptativa realiza los siguientes pasos:

Algoritmo A.3

1. Inicialmente todos los símbolos son equiprobables y su probabilidad es mayor que 0.
2. Mientras existan símbolos que codificar:
   (a) Codificar el siguiente símbolo.
   (b) Actualizar la probabilidad del símbolo codificado.

Mientras que el Algoritmo A.4 de descompresión adaptativa consiste en:

Algoritmo A.4

1. Idéntico al paso 1 del compresor.
2. Mientras existan símbolos que decodificar:
   (c) Decodificar el siguiente símbolo.
   (d) Actualizar la probabilidad del símbolo decodificado.

A.2.3.8 Actualización en el árbol de Huffman

La actualización de un símbolo en el árbol provoca que la probabilidad del símbolo actualizado y de todos los nodos internos que son antecesores (hasta llegar a la raíz) del símbolo sean incrementadas. La modificación de los pesos de los nodos puede provocar que los nodos afectados tengan que ascender por el árbol en dirección a la raíz, lo que acarrea una modificación de su estructura.

Usar el algoritmo de construcción presentado en la Sección A.**2.3.3** es demasiado costoso pues su complejidad es de $O(n^2)$ si n es el número de símbolos contemplados. Para acelerar este proceso, el árbol se representa de forma ordenada atendiendo a los pesos de los nodos usando un *array* $a[]$ con $2 \times n \nwarrow 1$ elementos [342-345]. El nodo raíz se almacena en $a[0]$, su hijo de mayor peso en $a[1]$ y el de menor en $a[2]$. A continuación se hace lo mismo con los hijos, comenzando por el de más peso, siempre que los nodos no sean hojas. Como resultado el array se ordena de forma decreciente y siempre debería cumplir esta propiedad.

Usando *a*[], es sencillo comprobar si la actualización de un símbolo provoca una modificación de la estructura del árbol porque esto sólo ocurre cuando el array deja de estar ordenado. En dicho caso, el nodo cuyo peso supera al que está por encima de él en el array debe ser intercambiado con el actualizado. Para comprender mejor este proceso, en la Fig.A.11-(A) se muestra un ejemplo. Cuando incrementamos en una unidad el peso del símbolo *a* (lo que llamamos una actualización en un algoritmo de compresión adaptativo), tenemos que incrementar el peso también de su nodo padre, de su nodo abuelo y así sucesivamente hasta llegar a la raíz, tal y como se describe en la Fig.A.11-(B). Si incrementamos de nuevo el símbolo *a* se viola la propiedad de que se puedan recorrer los nodos en orden decreciente (se degenera el árbol). Por esta razón, antes de continuar con la actualización de los pesos hasta la raíz, los nodos 8 y 5 deben ser intercambiados (Fig.A.11-(C)). Después del intercambio el proceso de actualización continua, comprobándose en cada actualización el cumplimiento de la propiedad del recorrido ordenado. De esta forma se consigue el árbol de Huffman de la Fig.A.11-(D).

Hasta ahora, la forma del árbol no ha variado ya que el incremento del peso de una de las hojas es insuficiente para mover un nodo interno. Para que esto ocurra vemos que por ejemplo, el símbolo *a* debe ser incrementado al menos dos veces más. Tras el primer incremento las cosas quedan como se indica en la Fig.A.11-(E). Como puede verse no se incumple la propiedad del recorrido ordenado. Pero el segundo incremento de *a* provoca varios intercambios. En primer lugar, al acumular un peso igual a 5, debe ser intercambiado por el nodo almacenado en 4. Dicha circunstancia es la que se muestra en la Fig.A.11-(F). Tras el intercambio continua la propagación del incremento del nodo padre del símbolo *a* situado en la posición 2 (Fig.A.11-(G)). El incremento de este nodo provoca de nuevo la violación del recorrido ordenado y los nodos 1 y 2 deben ser intercambiados (Fig.A.11-(H)). Cuando éste ya se ha realizado, se produce el incremento del nodo raíz (Fig.A.11-(I)) y el árbol de Huffman queda construido (Fig.A.11-(J)). Debido a que normalmente se utilizan datos de tipo entero (con pocos bits de precisión) para almacenar la probabilidad de un símbolo, es necesario ejecutar algún proceso de escalado de todas las probabilidades de los nodos del árbol de Huffman y así evitar desbor-damientos. Dicho proceso consiste típicamente en dividir (truncando) todos los recuentos entre 2. Esta operación no modifica la forma del árbol y además tiene un efecto beneficioso. Es frecuente encontrar secuencias de símbolos en las que muchos de ellos aparecen en zonas determinadas de la secuencia, como consecuencia de que las fuentes no son realmente ergódicas. La operación de escalado provoca que debido a los truncamientos, las probabilidades de estos símbolos sean menores que si no se produjera escalado, y esto ocurre precisamente cuando no son utilizados. Visualmente lo que ocurre es que dichos símbolos se "hunden" en el árbol más rápidamente y dejan que los símbolos más usados puedan codificarse con menos bits.

A.2.3.9 Extensión de una fuente de información

La extensión de una fuente es el nombre que recibe el proceso de construir una fuente llamada fuente extendida a partir de una fuente de información, generando cada símbolo-fuente extendido como la concatenación de dos o más símbolos-fuente. Como consecuencia, la base de la fuente extendida es r^N, donde N es el orden de la extensión y r es el tamaño del alfabeto. El objetivo de extender una fuente es aumentar suficientemente el número de símbolos-fuente. Como veremos, haciendo esto es posible encontrar representaciones más eficientes para las secuencias-fuente originales. Por ejemplo, si el alfabeto-fuente $a = \{a_1, a_2, a_3\}$, el alfabeto-fuente extendido a^2 de orden 2 será

$$a^2 = \{a_1a_1, a_1a_2, a_1a_3, a_2a_1, a_2a_2, a_2a_3, a_3a_1, a_3a_2, a_3a_3\} \tag{A.16}$$

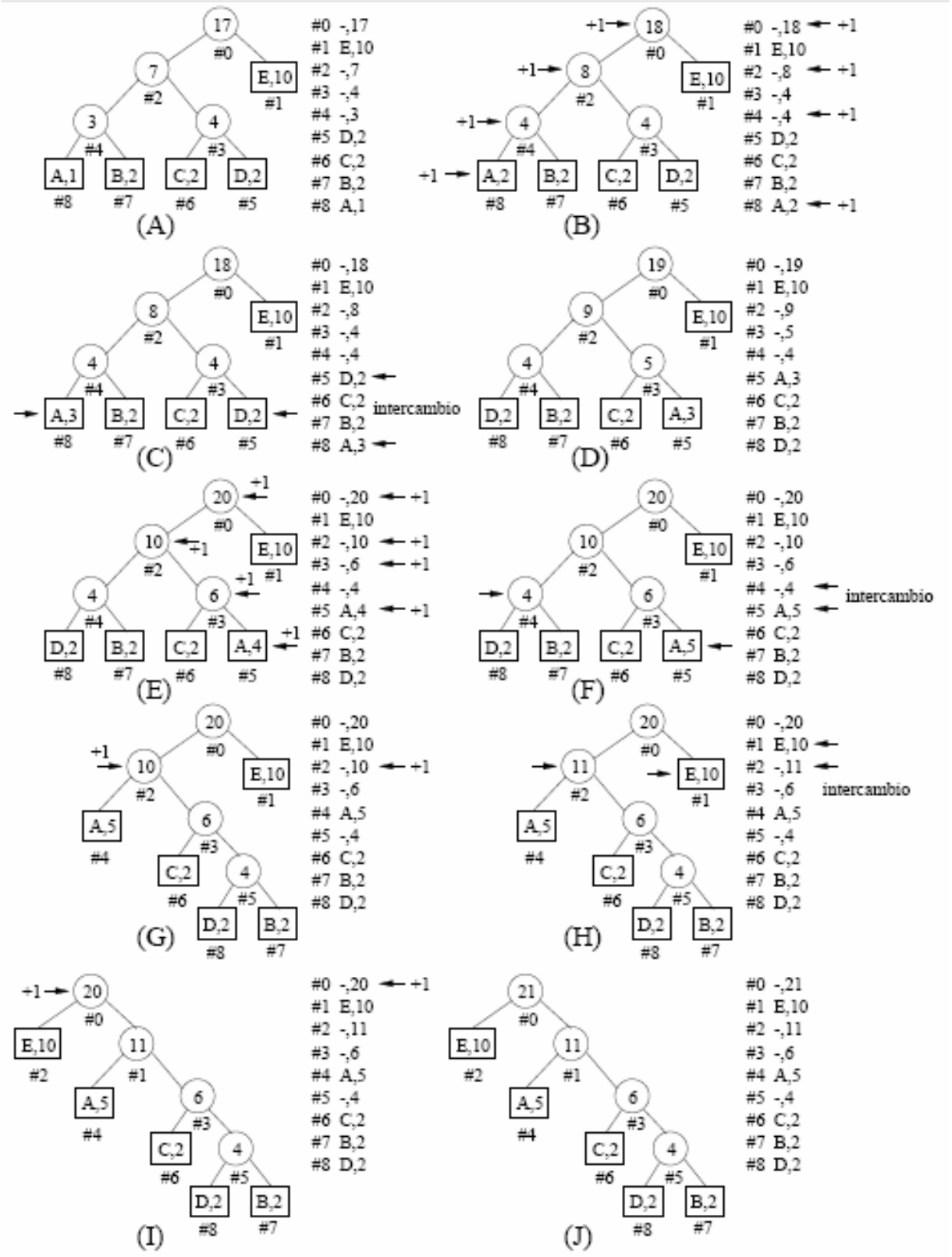


Figura A.11: Ejemplo de actualización de un árbol de Huffman.

A.2.3.10 La codificación aritmética

La codificación aritmética fue introducida por Abramson [346] en 1963 y desarrollada por Pasco [347] en 1976. Desde entonces, ha recibido una atención especial porque es una forma de

representación de la información muy eficiente y permite una independencia máxima entre el modelo probabilístico y el codificador.

Supongamos que necesitamos codificar una fuente de información binaria (por ejemplo, una imagen de fax) que emite símbolos con probabilidades desiguales. La codificación de Huffman es incapaz de expresar eficientemente esta fuente de información porque representaría cada símbolo con 1 bit de código al menos que es el tamaño de la representación actual. La solución dada a este problema durante mucho tiempo ha consistido en extender primero la fuente (ver la Sección anterior) y luego usar la codificación de Huffman sobre un alfabeto mayor. La eficiencia de la solución sólo es ideal cuando el orden de la extensión es infinito y por tanto, nunca podemos representar la fuente sin redundancia.

Abramson, Pasco y más tarde Rissanen y Landon [348, 349] desarrollaron un método de codificación que permitía codificar fuentes binarias sin necesidad de extender la fuente. El método se llamó codificación aritmética binaria. La idea es muy sencilla, aunque sobre su implementación software y hardware se han escrito decenas de artículos [350-361].

Consiste en asignar a cada posible secuencia de símbolos un subintervalo dentro del intervalo [0, 1) cuyo tamaño es igual al producto de todas las probabilidades de los símbolos que forman la secuencia. La posición del subintervalo dentro del intervalo real [0, 1) está determinado por la secuencia concreta de símbolos. El algoritmo básico de codificación consiste en emitir un número cualquiera perteneciente al subintervalo final. El número de bits de código necesarios para representar el código aritmético, es igual a la entropía de la secuencia de símbolos y por esta razón, la codificación aritmética se considera 100% eficiente.

A.2.3.11 La codificación aritmética binaria

Históricamente la codificación aritmética binaria ha recibido una atención especial debido a las siguientes razones:

- El cálculo del intervalo siguiente sólo afecta en una iteración a *L* (símbolo más bajo) o a *H* (símbolo más alto), pero nunca a ambos: si el símbolo a codificar es un 0, sólo hay que calcular *H* y viceversa. El tiempo de cálculo del intervalo siguiente se divide por tanto por la mitad aproximadamente.

- Se puede utilizar para codificar fuentes de información multisímbolo si construimos un árbol de Huffman en el que cada nodo tiene asociada una probabilidad de transición hacia su rama izquierda o derecha [362]. Se usa un árbol de Huffman porque esta estructura minimiza el número de decisiones para codificar un símbolo en función de su probabilidad. Cada vez que atravesamos un nodo interno, se codifica aritméticamente la probabilidad de la rama escogida que depende del nodo asociado.

- Los modelos probabilísticos para dos símbolos son muy sencillos ya que con una única probabilidad se describe a la fuente. Además, en el proceso de descodificación la búsqueda del subintervalo que contiene el código aritmético se trivializa porque sólo existen dos alternativas.

Estas circunstancias (y en especial la última) han provocado que la mayoría de las implementaciones físicas sean para codificadores y decodificadores binarios y el ejemplo más claro lo tenemos en el Q-Coder [363-366]. Sin embargo, cuando los alfabetos son multisímbolo (256 típicamente), el codificador aritmético multisímbolo supera en velocidad a la versión binaria.

Esto es especialmente cierto en el caso de una implementación software en la que es mucho más determinante el número de instrucciones ejecutadas que la complejidad aritmética de éstas. Incluso en el caso de una codificación de fuentes binarias, la menor velocidad de funcionamiento del algoritmo multisímbolo puede ser suplida procesando la fuente de forma extendida, incluso tratándose de implementaciones hardware [367].

A.3 Conclusiones del apéndice

En este apéndice se analizaron todos los elementos complementariamente y no desarrollados en el Capítulo 1 en función de los esquemas de compresión/descompresión basados en transformadas, así como cuantificación vectorial y escalar.

Apéndice B

Formatos JPEG y JPEG2000

B.1. Introducción

En este apéndice se verán en detalle los algoritmos de JPEG y JPEG2000 con los cuales se medirán en el Capítulo 4 (Métricas y evaluación de performance de reconstrucción) las técnicas propuestas en el Capítulo 3.

B.2. Formatos JPEG y JPEG2000

El formato de archivo gráfico conocido por sus siglas en inglés JPEG (Joint Photographic Experts Group) es actualmente el más conocido, diseminado y utilizado en la industria de las imágenes electrónicas. JPEG2000 es una versión posterior y mejorada que (entre otras cosas) incorpora la tecnología de onditas, a los efectos de aumentar dramáticamente la tasa de compresión. En el presente apéndice, se verán ambos en detalle.

B.2.1. Formato JPEG

B.2.1.1. Requerimientos y proceso de selección

El objetivo de JPEG ha sido desarrollar un método para compresión de imágenes de tono continuo que cumpla los siguientes requisitos:

1. estar en o cerca del estado del arte con respecto a la tasa de compresión y acompañando la fidelidad de la imagen, a lo largo de una amplia gama de métricas de calidad de la imagen y, especialmente, en el rango, donde la fidelidad al original se caracteriza como "muy buena" a "excelente"; también, el codificador debería ser parametrizable, a fin de que la aplicación (o usuario) pueda ajustar el compromiso deseado de compresión/calidad;

2. ser aplicable a prácticamente cualquier clase de imagen fuente digital de tono continuo (i.e. para propósitos más prácticos no estar restringido a imágenes de ciertas dimensiones, espacios de color, tasas de aspecto de pixel, etc.) y no estar limitada a cierta clase de imagenología con restricciones sobre el contenido de una escena, tal como complejidad, rango de colores, o propiedades estadísticas;

3. tenga una CC tratable, para hacer factible implementaciones en software de una performance viable sobre una gran variedad de CPU's, tan bien como las implementaciones de hardware con un costo viable para aplicaciones que requieren alta performance;

4. tenga los siguientes modos de operación:
   - Codificación Secuencial: cada componente de la imagen es codificada en una única exploración izquierda-a-derecha, arriba-a-abajo;

   - Codificación Progresiva: la imagen es codificada en múltiples exploraciones para aplicaciones en las cuales el tiempo de transmisión es largo, y el usuario prefiere mirar la imagen a crearse en múltiples pasos rústico-a-claro;

- Codificación sin pérdidas: la imagen es codificada para garantizar la exacta recuperación de los valores de los pixeles de cada imagen fuente (aunque el resultado sea una compresión baja comparada a los modos con pérdidas);

- Codificación Jerárquica: la imagen es codificada a múltiples resoluciones a fin de que puedan ser accedidas versiones de baja resolución sin tener que primero descomprimir la imagen en toda su resolución.

En Junio de 1987, JPEG condujo un proceso de selección basado en una evaluación a ciegas de la calidad subjetiva de una foto, y se redujo de doce métodos a tres. Tres grupos de trabajo informal formados para refinarlos, y en Enero de 1988, un segundo proceso de selección más riguroso [17] reveló que la propuesta conocida como TDC Adaptativa (TDCA) [18], basada en TDC de 8x8, produjo la mejor calidad de imagen.

Al momento de su selección, el método basado en TDC fue solo parcialmente definido para algunos de los modos de operación. De 1988 a 1990, JPEG emprendió la importante tarea de definir, documentar, simular, probar, validar, y simplemente llegar a un acuerdo sobre la plétora de detalles necesarios para una verdadera interoperabilidad y universalidad. Más detalles sobre la historia de los esfuerzos del JPEG pueden verse en [19-22].

B.2.1.2. Arquitectura del estándar propuesto

El estándar propuesto contiene los cuatro “modos de operación” previamente identificados. Para cada modo, uno o más codecs distintivos son especificados. Los codecs difieren en un modo de acuerdo a la precisión de las muestras de la imagen fuente que ellos pueden manejar o el método de codificación de entropía que ellos usan. Aunque la palabra codec (codificador/decodificador) es usada frecuentemente en esta tesis, no existe ningún requisito acerca de que las implementaciones deben incluir ambos, es decir, un codificador y un decodificador. Muchas aplicaciones tendrán sistemas o dispositivos que requieren exclusivamente una o la otra.

Los cuatro modos de operación y sus varios codecs han resultado del objetivo de JPEG de ser genéricos y de la diversidad de los formatos de imagen a través de las aplicaciones. Las piezas múltiples pueden dar la impresión de una complejidad indeseable, pero en realidad debería ser considerada como una "caja de herramientas" que pueden extenderse a una amplia gama aplicaciones de imágenes de tono continuo. Es poco probable que muchas implementaciones utilicen todas las herramientas - de hecho, la mayoría de las primeras implementaciones - ahora en el mercado (incluso antes de finalizada la aprobación ISO) - han aplicado sólo la secuencia de referencia del codec.

El codec secuencial de referencia es inherentemente un rico y sofisticado método de compresión el cual será suficiente para muchas aplicaciones. Obtener esta mínima capacidad del JPEG implementada apropiadamente y su interoperabilidad proporciona a la industria una importante capacidad inicial para el intercambio de imágenes a través de los proveedores y las aplicaciones

B.2.1.3. Pasos del procesamiento para una codificación basada en la TDC

Las Figuras B.1 y B.2 muestran los principales pasos de procesamiento los cuales son el corazón de los modos de operación basados en TDC. Estas figuran ilustran el caso especial de compresión de imágenes de componente única (escala de grises). El lector puede captar la esencia de la

compre-sión basada en TDC pensando esencialmente en una compresión de cadenas de bloques de 8x8 pixeles de la imagen en escala de grises. La compresión de una imagen a color consiste en transformar las componentes RGB (azul, verde y rojo; fuertemente correlacionadas) al espacio de componentes YIQ (espacio de color usado en TV a colores; fuertemente descorrelacionadas) y cada una de estas tres bandas se comprime como corresponde con diferentes criterios.

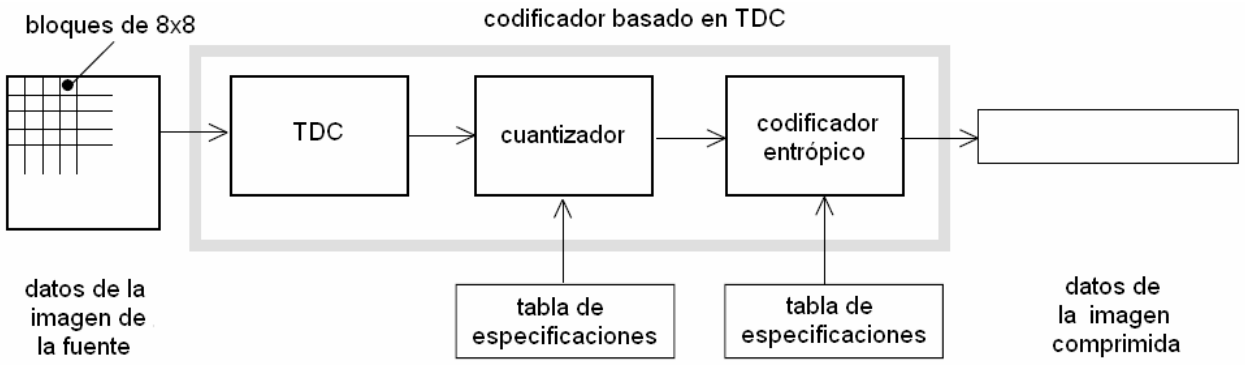


Figura B.1: Pasos de procesamiento del codificador basado en TDC.

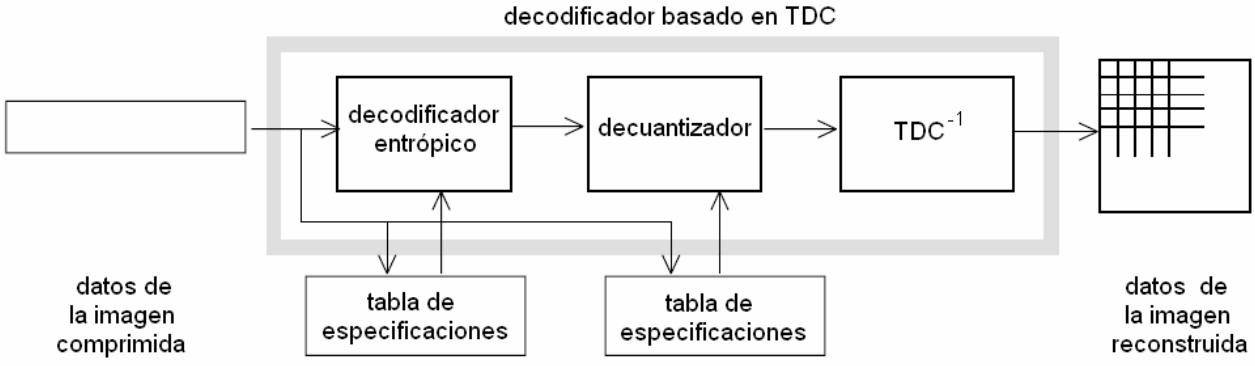


Figura B.2: Pasos de procesamiento del decodificador basado en TDC.

Para los codecs de TDC de modo secuencial, los cuales incluyen el codec secuencial de referencia, el diagrama simplificado indica cómo trabaja la compresión de componente única de una forma bastante completa. Cada bloque de 8x8 es una entrada, hace su camino a través de cada paso del proceso, y genera la salida en forma comprimida en la cadena de datos. Para los codecs TDC de modo progresivo, existe una imagen de reserve antes del paso de codificación entrópica, de manera que una imagen se puede almacenar y, a continuación, separar en múltiples exploraciones, sucesivamente, con una mejora sucesiva de la calidad. Para el modo de operación jerárquico, los pasos mostrados son usados para construir bloques en contexto mucho mayor.

B.2.1.3.1. TDC y TDC^{-1} para bloques de 8x8 pixeles

Como se vio en detalle en el Apéndice A, solo hay que considerar a la Fig.B.3 para bloques de 8x8 pixeles para poder ser aplicados en JPEG.

B.2.1.3.2. Cuantización

Como se vio en detalle en el Apéndice A, la cuantización considerada en JPEG será la del tipo escalar.

B.2.1.3.3. Codificación DC y secuencia zig-zag

Luego de la cuantización, el coeficiente DC es tratado separadamente de los 63 coeficientes AC. El coeficiente DC es una medida del valor promedio de los 64 pixeles del bloque. Dado que existe usualmente una fuerte correlación entre los coeficientes DC de los bloques 8x8 adyacentes, el coeficiente DC cuantizado es codificado como la diferencia del término DC del previo bloque en el orden codificado (definido a continuación), como se muestra en la Fig.B.3. Este tratamiento especial vale la pena, dado que los coeficientes DC contienen frecuentemente una fracción significativa del total de la energía de la imagen.

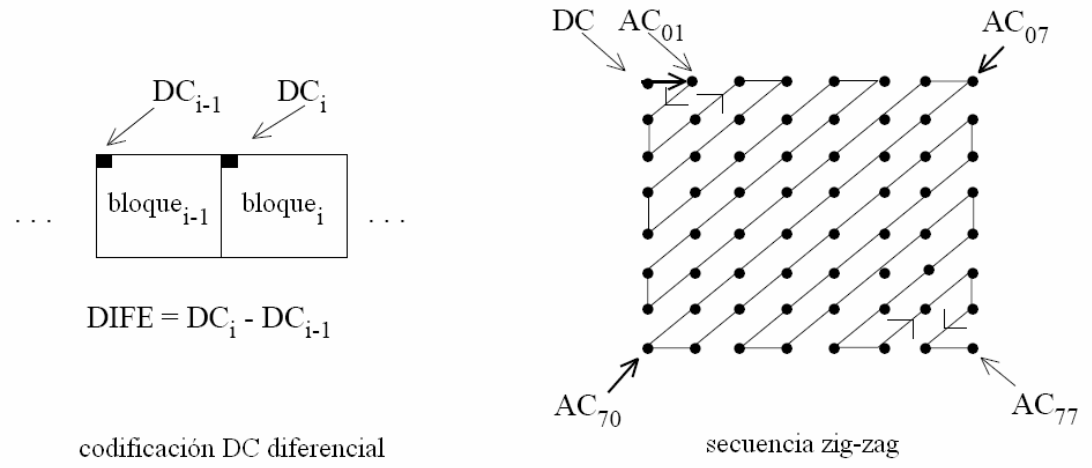


Figura B.3: Preparación de los coeficientes de cuantizado para la codificación entrópica.

Finalmente, la totalidad de los coeficientes cuantizados son ordenados en la secuencia “zig-zag” (vista en detalle en el Apéndice C), de frecuencias menores (mayor energía) a frecuencias mayores (menor energía) también mostrada en la Fig.B.3, lo cual, en rigor, constituye un criterio eurístico, no un ordenamiento. Este ordenamiento ayuda a facilitar la codificación de la entropía al poner los coeficientes de baja frecuencia (los cuales tienen más probabilidad de ser distintos de cero) antes de los coeficientes de alta frecuencia.

B.2.1.3.4. Codificación entrópica

Como se vio en detalle en el Apéndice A, la codificación entrópica la emplearemos previa al canal o bien al proceso de almacenamiento.

B.2.1.3.5. Compresión y calidad de la imagen

Para imagines color con escenas moderadamente complejas, todos los modos de operación basados en TDC producen típicamente los siguientes niveles de calidad de imagen para los rangos indicados de compresión. Estos niveles son simplemente una guía – la calidad y la compresión pueden variar significativamente de acuerdo a las características de la imagen fuente y el

contenido de la escena. (Las unidades “bits/pixel” significan aquí el número total de bits en la imagen comprimida – incluyendo las componentes de crominancia – dividido por el número de muestras en la componente de luminancia.)

• 0.25-0.5 bits/pixel: moderada a buena calidad, suficiente para algunas aplicaciones;

• 0.5-0.75 bits/pixel: buena a muy buena calidad, suficiente para muchas aplicaciones;

• 0.75-1/5 bits/pixel: excelente calidad, suficiente para la mayoría de las aplicaciones;

• 1.5-2.0 bits/pixel: usualmente indistinguible de la original, suficiente para la mayoría de las aplicaciones demandadas.

B.2.1.4. Pasos del procesamiento para un codificador predictivo sin pérdidas

Después de su selección de un método basado en TDC en 1988, JPEG descubrió que los modos sin pérdidas basados en TDC eran difíciles de definir como una práctica estándar en contra de los codificadores y decodificadores que podrían ser aplicados independientemente, sin imponer importantes limitaciones en las implementaciones de ambos codificador y decodificador.

JPEG, con el objeto de cumplir con su requisito de un modo de operación sin pérdidas, ha optado por un método predictivo simple el cual es totalmente independiente del procesamiento de la TDC descrito anteriormente. La selección de este método no fue el resultado de una evaluzación competitiva rigurosa como fue el caso del método basado en TDC. Sin embargo, el método sin pérdidas JPEG produce resultados que, a la luz de su sencillez, se hallan sorprendentemente cerca del estado del arte para la compresión sin pérdidas de tono continuo, como se indica en un reciente informe técnico [23].

La Fig.B.4 muestra los pasos principales del procesamiento para una imagen de componente única. Un predictor combina los valores de hasta tres muestras de vecinos (A, B, y C) para realizar una predicción de la muestra indicada por X en Fig.B.5. Esta predicción es entonces sustraida del valor actual de la muestra X, y la diferencia es codificada sin pérdida por algunos de los métodos de codificación entrópica - Huffman o Aritmético. Pueden ser usados cualquiera de los ocho predictores listados en la Tabla B.1 (bajo “valor de selección”).

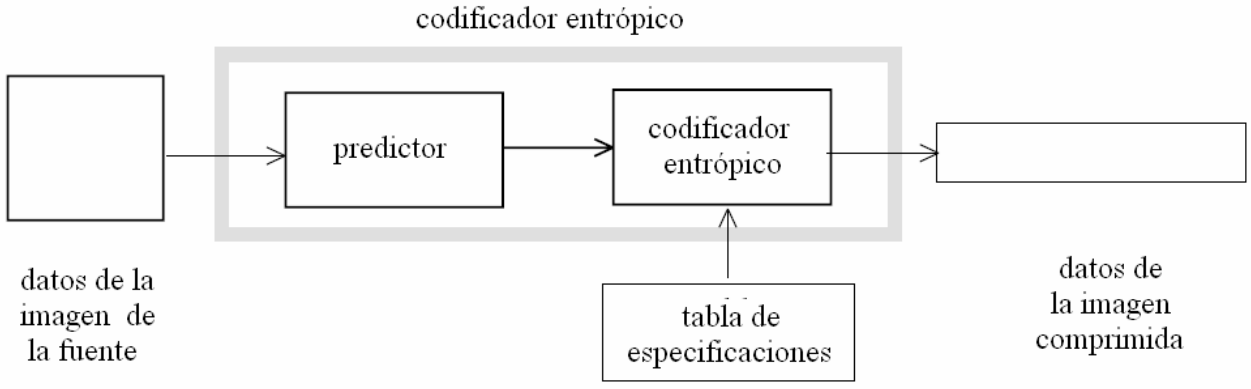


Figura B.4: Pasos de procesamiento del codificador en modo sin pérdidas.

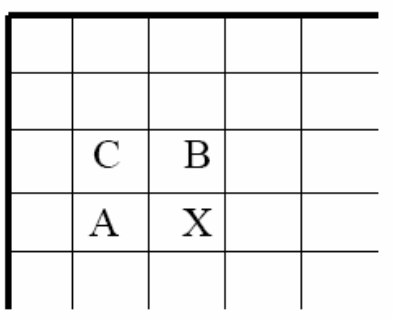


Figura B.5: Vecindario de predicción de 3-muestras.

Tabla.B.1: Predictores para codificación sin pérdida.

Valor de selección	Predicción
0	Sin predicción
1	A
2	B
3	C
4	A+B-C
5	A+((B-C)/2)
6	B+((A-C)/2)
7	(A+B)/2

Las selecciones 1, 2, y 3 son predictores uni-dimensionales y las selecciones 4, 5, 6 y 7 son predoctores bi-dimensionales. La selección-valor 0 puede ser usada solamente para codificación diferencial en el modo de operación jerárquico. La codificación entrópica es casi idéntica a la usada para el coeficiente DC descripto en la Sección B.2.1.6.1. (para codificación de Huffman).

Para el modo de operación sin pérdidas, son especificados dos diferentes codecs – uno por cada método de codificación entrópica. Los codificadores pueden usar cualquier precisión de imagen fuente de 2 a 16 bits/muestra, y puede usar cualquiera de los predictores excepto el de valor 0 de selección. Los decodificadores deben manejar cualquiera de las precisiones de muestra y cualquiera de los predictores. Los codificadores sin pérdidas producen típicamente una compresión de alrededor de 2:1 para imágenes color con escenas moderadamente complejas.

B.2.1.5. Imágenes de múltiples componentes

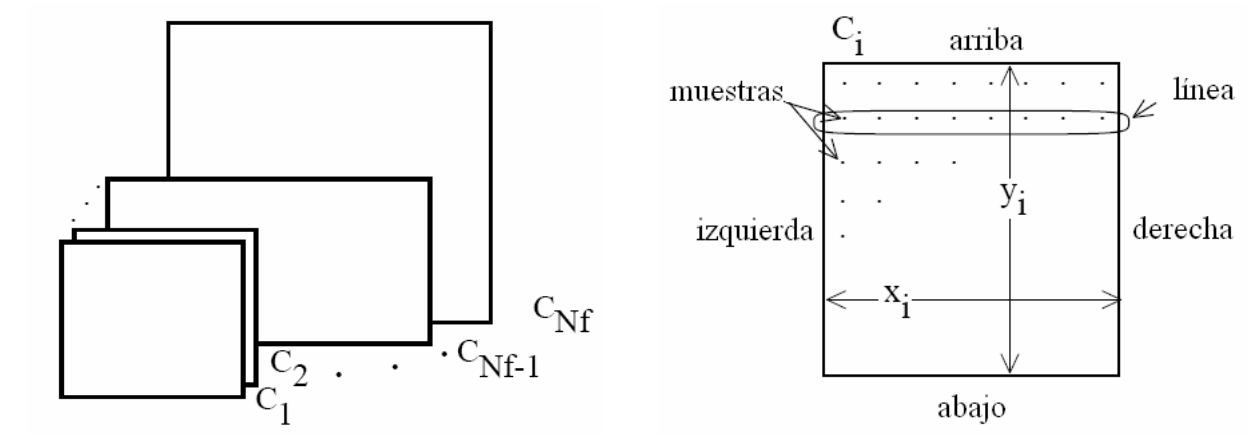
En las secciones previas se discutieron los pasos principales de los codecs basados en TDC y predictivos sin pérdidas para el caso de imágenes fuente de componente única. Estos pasos logran la compresión de los datos de la imagen. Pero una buena parte de la propuesta JPEG también se ocupa de la manipulación y el control del color (o dicho de otra forma) imágenes con múltiples componentes. El objetivo de JPEG para un estándar de compresión genérica requiere su propuesta para dar cabida a una variedad de formatos de imagen fuente.

B.2.1.5.1. Formatos de imagen de fuente

El modelo de imagen fuente usado en la propuesta JPEG es una abstracción de una variedad de tipos de imágenes y aplicaciones, y consiste de solo lo que es necesario para comprimir y recons-

truir los datos de la imagen digital. El lector debería reconocer que el formato de datos comprimidos JPEG no codifica suficiente información para servir como una representación completa de la imagen. Por ejemplo, JPEG no especifica o codifica cualquier información sobre tasas de aspecto s de pixeles, espacio de color, o características de adquisición de la imagen.

La Fig.B.6 ilustra el modelo de imagen fuente JPEG. Una imagen fuente contiene de 1 a 255 componentes de la imagen (Nf), algunas veces llamado color o bandas espectrales o canales. Cada componente consiste en un arreglo rectangular de muestras. Una muestra se define para ser un entero sin signo con una precisión de *P* bits, con cualquier valor en el rango [0, 2^P-1]. Todas las muestras de todos los componentes en la misma imagen fuente debe tener la misma precisión *P*. *P* puede ser 8 o 12 para codecs basados en TDC, y 2 a 16 para codecs predictivos.



(a) Imagen fuente con múltiples componentes. (b) Características de una componente de imagen.

Figura B.6: Modelo de imagen de fuente JPEG.

La *i*-ésima componente tiene las dimensiones de la muestra x_i por y_i. Para acomodar los formatos en los cuales algunas componentes de la imagen son muestreadas a diferentes tasas que otras, componentes pueden tener diferentes dimensiones. Las dimensiones deben tener un relación integral mutua definida por H_i y V_i, los factores de muestreo relativo horizontal y vertical, el cual debe ser especificado para cada componente. Las dimensiones totales de la imagen *X* e *Y* están definidas como el máximo x_i e y_i para todas las componentes en la imagen, y pueden ser cualquier número hasta 2^{16}. Para *H* y *V* están permitidos solo los valores enteros 1 a 4. Los parámetros codificados son *X*, *Y*, y H_is y V_is para cada componente. El decodificador reconstruye las dimensiones x_i e y_i para cada componente, de acuerdo a las siguientes relaciones que muestra la Ecuación D.1:

$$\boldsymbol{x_i} = \left\lceil \boldsymbol{X} \times \frac{\boldsymbol{H_i}}{\boldsymbol{H}_{\max}} \right\rceil$$
$$\boldsymbol{y_i} = \left\lceil \boldsymbol{Y} \times \frac{\boldsymbol{V_i}}{\boldsymbol{V}_{\max}} \right\rceil \qquad \text{(B.1)}$$

donde $\lceil\ \rceil$ es la función “pasa al entero mayor más próximo” .

B.2.1.5.2. Orden de codificación e entrelazado

Un estándar de compresión práctico de imagines debe abordar como los sistemas necesitarán manejar los datos durante el proceso de descompresión. Muchas aplicaciones necesitan vehiculizar el proceso de mostrar e imprimir imágenes de múltiples componentes en paralelo con el proceso de descompresión. Para muchos sistemas, esto solo es posible si los componentes son intercalados junto con la cadena de datos comprimidos.

Para hacer el mismo mecanismo de entrelazado aplicable a ambos codecs: basado en TDC y predictivo, la propuesta JPEG ha definido el concepto de "unidad de datos." Una unidad de datos es una muestra en codecs predictivos y un bloque 8x8 de muestras en codecs basados en TDC.

El orden en el cual las unidades de datos comprimidas tienen lugar en la cadena de datos comprimidos es una generalización del orden de exploración por filas. Generalmente, las unidades de datos son ordenadas de izquierda-a-derecha y de arriba-a-debajo de acuerdo a la orientación mostrada en la Fig.B.6. (Es responsibilidad de las aplicaciones definir cuales bordes de la imagen fuente son arriba, abajo, izquierda y derecha.) Si una componente de imagen es no-entrelazada (i.e., comprimida sin ser entrelazada con otras componentes), las unidades de datos comprimidas están ordenadas una exploración de raster pura como muestra la Fig.B.7.

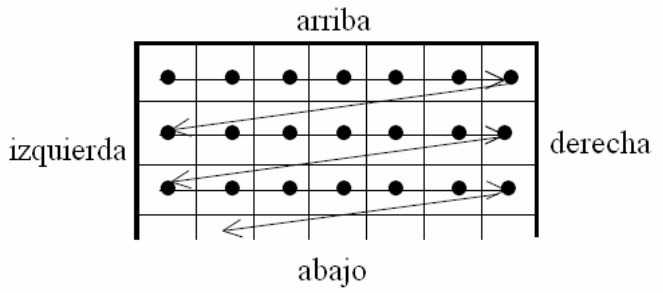


Figura B.7: Ordenamiento no-entrelazado de los datos.

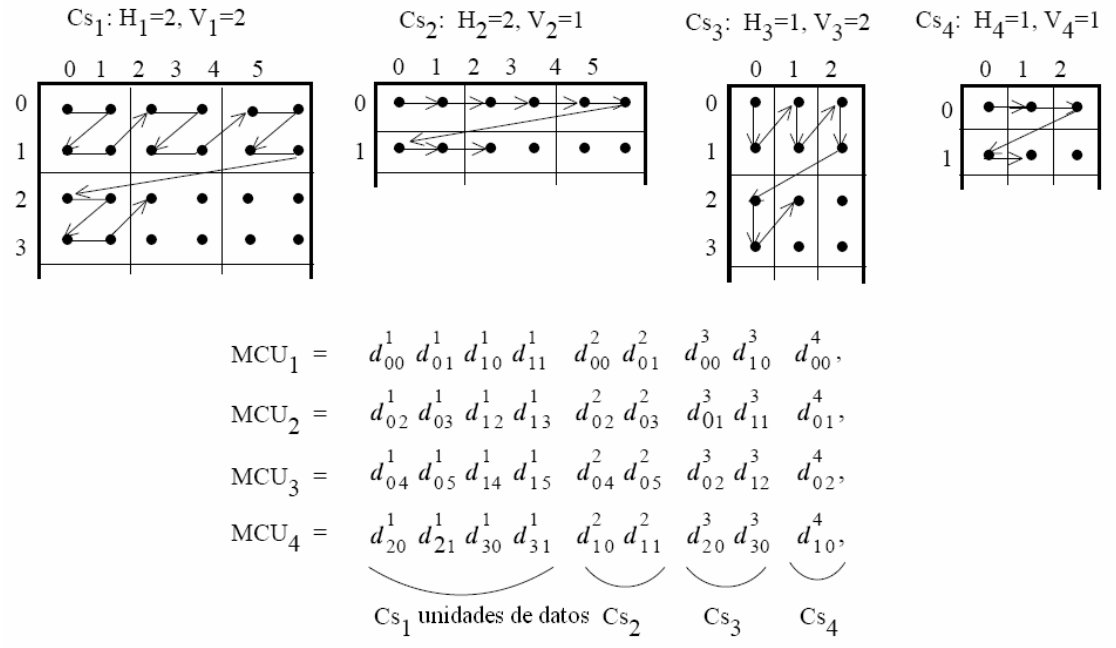


Figura B.8: Ejemplo de ordenamiento entrelazado de los datos.

Cuando dos o más componentes están entrelazados, cada componente C_i es particionada en regiones rectangulares de H_i por V_i unidades de datos, como se muestra en el ejemplo generalizado de la Fig.B.8. Las regiones están ordenadas en una componente de izquierda-a-derecha y arriba-a-abajo, y en una región, unidades de datos son ordenados de izquierda-a-derecha y arriba-a-abajo. La propuesta JPEG define el término Minimum Coded Unit (MCU, unidad de codificado mínimo) a ser el grupo más pequeño de unidades de datos entrelazados. Para el ejemplo mostrado, MCU1 consiste de unidades dato que toman primero de la mayoría de las regiones arriba-izquierda de C_1, seguido por las unidades de datos de la misma región de C_2, y asimismo para C_3 y C_4. MCU2 continúa el patrón mostrado.

Entonces, el dato entrelazado es una secuencia ordenada de MCUs, y el número de unidades dato contenida en un MCU es determinado por el número de componentes entrelazados y sus factores de muestreo relativo. El número máximo de componentes los cuales pueden ser entrelazados es 4 y el número máximo de unidades dato en un MCU es 10. La última restricción es expresada como se muestra en la Ecuación (B.2), donde la sumatoria se realiza sobre las componentes entrelazadas:

$$\sum_{\substack{\textit{todos los i en}\\ \text{int}\textit{erlaceado}}} \boldsymbol{H}_i \times \boldsymbol{V}_i \leq 10 \qquad \text{(B.2)}$$

Debido a esta restricción, no todas las combinaciones de 4 componentes las cuales pueden ser representadas en un orden no-entrelazado en una imagen comprimida JPEG son permitidas para ser entrelazadas. Además, nótese que la propuesta JPEG permite algunos componentes ser entrelazados y algunos ser no-entrelazados en la misma imagen comprimida.

B.2.1.5.3. Tablas múltiples

En adición al control de entrelazado discutido previamente, los codecs JPEG que deben controlar la aplicación de la tabla de datos propia para los componentes propios. La misma tabla de cuantización y la misma tabla de codificación entrópica (o conjunto de tablas) deben ser usadas para codificar todas las muestras en un componente.

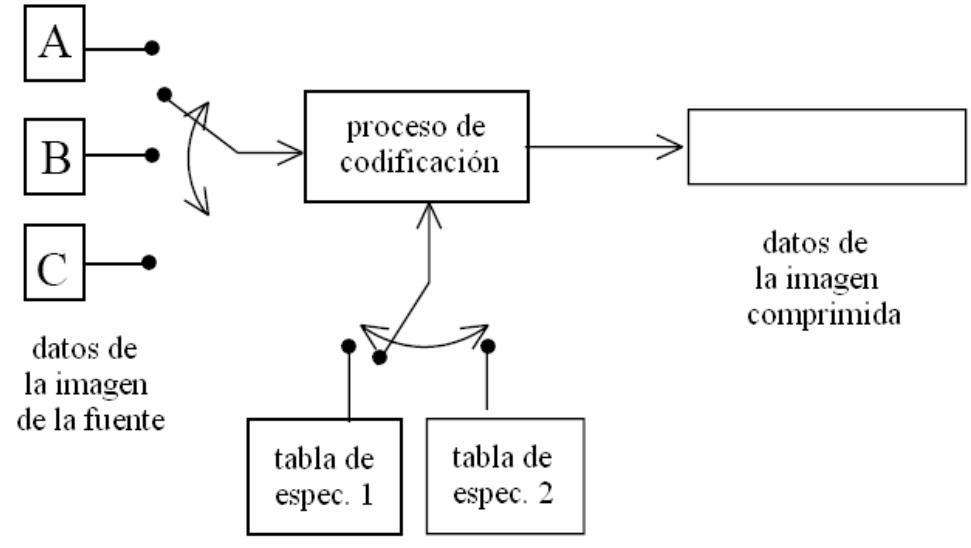


Figura B.9: Componente entrelazado y control de selección de tabla.

Los decodificadores JPEG pueden almacenar hasta 4 diferentes tablas de cuantización y hasta 4 diferentes (conjuntos de) tablas de codificación entrópica simultáneamente. (El decodificador secuencial de referencia es la excepción; dado que puede almacenar solo hasta 2 conjuntos tablas de codificación entrópica.) Esto es necesario para conmutación entre diferentes tablas durante la descompresión de una exploración conteniendo múltiples (entrelazados) componentes, en función de aplicar la tabla propia para la componente propia. (Las tablas no pueden ser cargadas durante la descompresión de una exploración.) La Fig.B.9 ilustra el control de conmutación de tabla que debe ser manejado en conjunción con el entrelazado de componentes múltiples para el lado del codificador. (Esta visión simplificada no distingue entre cuantización y tablas de codificación entrópica.)

B.2.1.6. Referencia y otros codificadores TDC secuenciales

El modo de operación secuencial TDC consiste de los pasos de TDC y cuantización de la Sección B.2.1.3.2, y el control de componente múltiple de la Sección B.2.1.5.3. En adición al codec secuencial de referencia, otros codecs secuenciales TDC están definidos para acomodar las dos precisiones de muestras diferentes (8 y 12 bits) y los dos diferentes tipos de métodos de codificación entrópica (Huffman y Aritmético).

La codificación secuencial de referencia es para imágenes con 8-bit por muestra y usa codificación de Huffman exclusivamente. Esto difiere de los codecs de TDC secuencial en que su decodificador puede almacenar solo dos conjuntos de tablas de Huffman (una tabla AC y tabla DC por conjunto). Esta restricción significa que, para imagines con tres o cuatro componentes entrelazados, al menos un conjunto de tablas de Huffman deben ser compartidas por dos componentes. Esta restricción no plantea ninguna limitación en todos los componentes no-entrelazados; un nuevo conjunto de tablas puede ser cargado en el decodificador antes de que comience la descompresión de componentes no-entrelazados. Para muchas aplicaciones las cuales necesitan entrelazar tres componentes de color, esta restricción es una dura limitación en todos. Los espacios de color (YUV, CIELUV, CIELAB, y otros) los cuales representan la información cromática (‘‘color’’) en dos componentes y la información acromática (‘‘escala de grises’’) en una tercera son más eficientes para la compresión que los espacios tales como RGB [306]. Un conjunto de tablas de Huffman puede ser usado para la componente acromática y una para las componentes cromáticas. La estadística de los coeficientes TDC son similares para las componentes de crominancia de la mayoría de las imágenes, y un conjunto de tablas de Huffman pueden codificar ambos al menos tan óptimamente como dos.

El comité consideró también que la pronta disponibilidad de implementaciones en un único chip a precios accesibles animaría la pronta aceptación de la propuesta JPEG en una variedad de aplicaciones. En 1988 cuando el secuencial de referencia fue definido, el comité de expertos de VLSI consideró que la actual tecnología permitió la agrupación de cuatro tablas cargadas de Huffman – en adición a cuatro conjuntos de tablas de cuantización - sobre un único chip de codec a precio accesible era una propuesta arriesgada. Los pasos de procesamiento: TDC, cuantización, diferenciación de DC, y el ordenamiento zig-zag para el codec secuencial de referencia que acaba de proceder, se describen en la Sección 2.1.3. Antes de la codificación entrópica, usualmente hay pocos coeficientes distintos de cero y muchos ceros. La tarea de la codificación entrópica consiste en codificar estos pocos coeficientes eficientemente. La descripción de la codificación entrópica secuencial de referencia está dada en dos pasos: la conversión de los coeficientes TDC cuantizados en una secuencia de símbolos y un cesión de códigos de longitud variable para los símbolos.

B.2.1.6.1. Representaciones de codificación entrópica intermedia

En la secuencia de símbolos intermedia, cada coeficiente AC distinto de cero es representado en combinación con la ‘‘longitud de la corrida’’ (número consecutivo) los coeficientes AC de valor cero los cuales preceden en la secuencia zig-zag. Cada una de esas combinaciones de coeficientes de longitud de corrida distinta de cero es (usualmente) representada por un par de símbolos:

símbolo-1	símbolo-2
(LONGITUD de la CORRIDA, DIMENSION)	(AMPLITUD)

El símbolo-1 representa dos piezas de información, LONGITUD de la CORRIDA y DIMENSION. El símbolo-2 representa una pieza única de información designada AMPLITUD, la cual es simplemente la amplitud de los coeficientes AC distintos de cero. La LONGITUD de la CORRIDA es el número de coeficientes AC cero consecutivos en una secuencia zig-zag precediendo al coeficiente AC distinto de cero representado. La DIMENSION es el número de bits usados para codificar AMPLITUD – es decir, para codificar el símbolo-2, por la codificación d eentero con signo usada con el método particular de JPEG de codificación de Huffman. La LONGITUD de la CORRIDA representa los ceros corridos de longitud 0 a 15. En este punto los ceros corridos en la secuencia zig-zag pueden ser más grandes que 15, así como el valor del símbolo-1 (15, 0) es interpretado como el símbolo extensión con *longituddelacorrida* = 16. Puede ser de tres extensiones consecutivas (15, 0) antes la terminación del símbolo-1 cuyo valor de LONGITUD de la CORRIDA completa la actual *longituddelacorrida*. La terminación del símbolo-1 es siempre seguida por un único símbolo-2, excepto para el caso en el cual la última corrida de ceros incluye los últimos coeficientes AC (63d). En este caso frecuente, el valor especial del símbolo-1 (0,0) significa EOB (end of block, fin de bloque), y puede ser visto como un símbolo de ‘‘escape’’ el cual finaliza el bloque de 8x8 muestras. Entonces, para cada bloque de 8x8 muestras, la secuencia zig-zag de 63 coeficientes AC cuantizados se representa como una secuencia de pares de símbolos símbolo-1, símbolo-2, aunque cada uno ‘‘par’’ puede tener repeticiones de símbolo-1 en el caso de una gran *longituddelacorrida* o solo un símbolo-1 en el caso de un EOB. El rango posible de coeficientes AC cuantizados determina el rango de valores en el cual ambos, la información de AMPLITUD y de la DIMENSION deben ser representados. Un análisis numérico de la ecuación de la TDC para 8x8 muestra que, si el punto-64 (bloque 8x8) de la señal de entrada contiene N-bits enteros, entonces la parte no-fraccional de los números de salida (coeficientes TDC) pueden crecer en más de tres bits. Esta es también la mayor dimensión posible de un coeficiente TDC cuantizado cuando su dimensión de paso de cuantizado tiene valor entero 1. El secuencial de referencia tiene muestras de fuente enteras de 8-bits en el rango $[-2^7, 2^7-1]$, así las amplitiudes del coeficiente AC cuantizado están cubiertos por enteros en el rango $[-2^{10}, 2^{10}-1]$. La codificación de entero con signo usa símbolo-2 con AMPLITUD de codificado de 1 a 10 bits de longitud (así la DIMENSION también representa valores desde 1 a 10), y la LONGITUD de la CORRIDA representa valores de 0 a 15 como se discutió previamente. Para los coeficientes, la estructura de las representaciones intermedias de los símbolo-1 y símbolo-2 se ilustra en las Tablas B.2 y B.3, respectivamente.

Tabla.B.2: Codificación en base de Huffman, estructura del símbolo-1

	DIMENSION					
	0	1	2	…	9	10
LONGITUD de la CORRIDA	0 EOB					
	. X					
	. X	valores de la				
	. X	DIMENSION de la CORRIDA				
	15 ZRL					

Tabla.B.3: Base de codificación entrópica, estructura del símbolos-2

DIMENSION	AMPLITUD
1	-1, 1
2	-3, -2, 2, 3
3	-7.. -4, 4.. 7
4	-15..-8, 8,..15
5	-31..-16, 16,..31
6	-63..-32, 32..63
7	-127 .. -64, 64 .. 127
8	-255 .. -128, 128 .. 255
9	-511 .. -256, 256 .. 511
10	-1023 .. -512, 512 .. 1023

La representación intermedia para un coeficiente DC diferencial del bloque de muestra de 8x8 se estructura en forma similar. El símbolo-1, no obstante, representa solo la información de DIMENSION; el símbolo-2 representa la información AMPLITUD como vimos antes:

símbolo-1	símbolo-2
(DIMENSION)	(AMPLITUD)

Dado que el coeficiente DC se codifica diferencialmente, esta cubierto por el doble de valores, $[-2^{11}, 2^{11}-1]$ como los coeficientes AC, de manera que un nivel adicional debe añadirse a la parte inferior de la Tabla B.3 para los coeficientes DC. El símbolo-1 para los coeficientes DC entonces representa un valor de 1 a 11.

B.2.1.6.2. Codificación entrópica de longitud variable

Una vez que el dato del coeficiente cuantizado para un bloque de 8x8 es representado en la secuencia de símbolos intermedia descrita arriba, se asignan los códigos de longitud variable. Para cada bloque de 8x8, la representación del par símbolo-1 y símbolo-2 para el coeficiente DC es codificada y enviada en primer lugar.

Para ambos coeficientes DC y AC, cada símbolo-1 es codificado con un código de longitud variable (VLC, variable-length code) del conjunto de tablas de Huffman asignado a la componente del bloque de 8x8. Cada símbolo-2 es codificado con un código "entero de longitud variable" (VLI, variable-length integer) cuya longitud en bits está dada en la Tabla B.3. Los VLCs y los VLIs son codificados con longitudes variables, pero los VLIs no son códigos de Huffman. Una importante distinción es que la longitud de un VLC (código de Huffman) no es conocido hasta que es decodificado, pero la longitud de un VLI es almacenada en su precedente VLC. El código de Huffman (VLCs) debe ser especificado externamente como una entrada a los codificadores JPEG. (Nótese que la forma en la cual las tablas de Huffman son representadas en la cadena de datos es una especificación indirecta en la cual el decodificador debe construir las tablas mismas antes de la descompresión.) La propuesta JPEG incluye un conjunto de ejemplo de tablas de Huffman en sus anexos de información, pero dado que ellos son aplicaciones específicas, no especifica ninguno para el uso requerido. En contraste, los códigos VLI están "incorporados" en la propuesta. Esto es apropiado, dado que los códigos VLI son mucho más numerosos, pueden ser calculados más que almacenados, y no se ha mostrados que sean apreciablemente más eficientes cuando son implementados como códigos de Huffman.

B.2.1.6.3. Ejemplo de codificación por referencia

Esta sección da un ejemplo de compresión por referencia y codificación de un único bloque de muestras de 8x8. Nótese que aquí omitimos una buena parte de la operación de un codificador complete de JPEG por referencia, incluyendo la creación de el Formato de Intercambio de información (parámetros, encabezados, cuantización y tablas de Huffman), byte de relleno, relleno para el byte de los límites antes de la creación del código, y otras operaciones principales. Sin embargo, este ejemplo debería ayudar a concretar gran parte de la explicación anterior. La Fig.B.10(a) es un bloque 8x8 de 8-bit por muestra, extraídas arbitrariamente de una imagen real. Las pequeñas variaciones de muestra a muestra indican la predominancia de baja frecuencia espacial. Luego de sustraer 128 de cada muestra para el nivel de desplazamiento requerido, el bloque de 8x8 es una entrada para la TDC FDCT, la Ecuación (1). La Fig.B.10(b) muestra (para un lugar decimal) los coeficientes TDC resultantes. Excepto para unos pocos coeficientes de las frecuencias más bajas, las amplitudes son muy pequeñas.

139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	162	160	160	159	159	159
159	160	161	162	162	155	155	155
161	161	161	161	160	157	157	157
162	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

(a) muestras de la imagen de la fuente

235.6	-1.0	-12.1	-5.2	2.1	-1.7	-2.7	1.3
-22.6	-17.5	-6.2	-3.2	-2.9	-0.1	0.4	-1.2
-10.9	-9.3	-1.6	1.5	0.2	-0.9	-0.6	-0.1
-7.1	-1.9	0.2	1.5	0.9	-0.1	0.0	0.3
-0.6	-0.8	1.5	1.6	-0.1	-0.7	0.6	1.3
1.8	-0.2	1.6	-0.3	-0.8	1.5	1.0	-1.0
-1.3	-0.4	-0.3	-1.5	-0.5	1.7	1.1	-0.8
-2.6	1.6	-3.8	-1.8	1.9	1.2	-0.6	-0.4

(b) coeficientes de la TDC

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

(c) tabla de cuantización

15	0	-1	0	0	0	0	0
-2	-1	0	0	0	0	0	0
-1	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(d) coeficientes cuantizados normalizados

240	0	-10	0	0	0	0	0
-24	-12	0	0	0	0	0	0
-14	-13	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(e) coeficientes cuantizados desnormalizados

144	146	149	152	154	156	156	156
148	150	152	154	156	156	156	156
155	156	157	158	158	157	156	155
160	161	161	162	161	159	157	155
163	163	164	163	162	160	158	156
163	164	164	164	162	160	158	157
160	161	162	162	162	161	159	158
158	159	161	161	162	161	159	158

(f) muestras de la imagen reconstruida

Figura B.10: TDC y ejemplos de cuantización.

La Fig.B.10(c) es el ejemplo de la tabla de cuantización para los componentes de luminancia (escala de grises) incluidos en el anexo de información del borrador del JPEG estándar parte 1 [18]. La cuantización se aplica sobre los coeficientes resultantes de la TDC. Luego de dicha cuantización se podan los decimales resultantes. La Fig.B.10(d) muestra los coeficientes TDC cuantizados, normalizados por sus entradas de la tabla de cuantización. En el decodificador estos números son "desformalizados" de acuerdo a la entrada a la TDC^{-1}. Finalmente, la Fig.B.10(f) muestra los valores de las muestras reconstruidas, marcadamente similares a los originales en la Fig.B.10(a). Desde luego, los números en la Fig.B.10(d) deben ser codificados de Huffman antes de la transmisión hacia el decodificador. El primer número del bloque a ser codificado es el término DC, el cual debe ser diferencialmente codificado. Si el término DC cuantizado del bloque

previo es, por ejemplo, 12, entonces la diferencia es +3. Entonces, la representación intermedia es (2)(3), para DIMENSION = 2 y AMPLITUD = 3.

A continuación, los coeficientes AC cuantizados son codificados. Seguimos el orden zig-zag, el primer coeficiente distinto de cero es -2, precedido por una corrida de cero de 1. Esto resulta en una representación intermedia de (1,2)(-2). Los siguientes encontrados en el orden zig-zag son los tres distintos de cero consecutivos de amplitud -1. Cada uno es precedido por una corrida de cero de longitud cero, para símbolos intermedios (0,1)(-1). El ultimo coeficiente distinto de cero es -1 precedido por dos ceros, para (2,1)(-1). Dado que este es el último coeficiente distinto de cero, el símbolo representando este bloque de 8x8 es EOB, o (0,0). Entonces, la secuencia intermedia de símbolos para este ejemplo del bloque 8x8 es:

(2)(3), (1,2)(-2), (0,1)(-1), (0,1)(-1), (0,1)(-1), (2,1)(-1), (0,0)

A continuación deben ser asignados los códigos. Para este ejemplo, los VLCs (códigos de Huffman) del anexo de información [24] que usaremos. El diferencial-DC VLC para este ejemplo es:

(2) 011

Los VLCs AC de luminancia para este ejemplo son:

(0,0) 1010
(0,1) 00
(1,2) 11011
(2,1) 11100

Los VLIs especificados en [24] son relativos a la representación complementaria de los dos. Ellos son:

(3) 11
(-2) 01
(-1) 0

Entonces, la cadena de bits para este bloque 8x8 de ejemplo es como sigue. Note que son requeridos 31 bits para representar 64 coeficientes, los cuales logran la compresión de algo menos de 0.5 bits/muestra:

01111110110100000000011100010 10

B.2.1.6.4. Otros codificadores TDC secuenciales

La estructura del codec secuencial TDC 12-bit con codificación de Huffman es una simple esténsión del método de codificación entrópica descrita previamente. Los coeficientes TDC cuantizados pueden ser 4 bits mas grandes, de manera tal que la información de DIMENSION y AMPLITUD se extiende en consecuencia. La TD secuencial con codificación Aritmética se describe en detalle en [24].

B.2.1.7. Modo TDC progresivo

El modo de operación TDC progresivo consiste de los mismos pasos de TDC y cuantización (de la Sección B.2.1.3) que son usados para el modo TDC secuencial. La diferencia principal reside en que cada componente de imagen es codificada en múltiples exploraciones más que en una única exploración. La primera exploración(es) codifica una rústica pero reconocible versión de la

imagen la cual puede ser transmitida rápidamente en comparación al tiempo de transmisión total, y es refinado por sucesivas exploraciones hasta alcanzar un nivel de calidad de imagen que fue establecida por las tablas de cuantización. Para lograr esto se requiere la adición de un buffer de memoria de una capacidad similar a la de la imagen a la salida del cuantizador, antes de la entrada al codificador entrópico. La memoria del buffer debe ser de tamaño suficiente para almacenar la imagen así como los coeficientes TDC cuantizados, cada uno de los cuales (si se almacena directamente) es 3 bits más grande que las muestras de la imagen fuente. Después de que cada bloque de coeficientes TDC es cuantizado, es almacenado en el buffer de memoria para coeficientes. Los coeficientes almacenados son entonces parcialmente codificados en cada exploración múltiple.

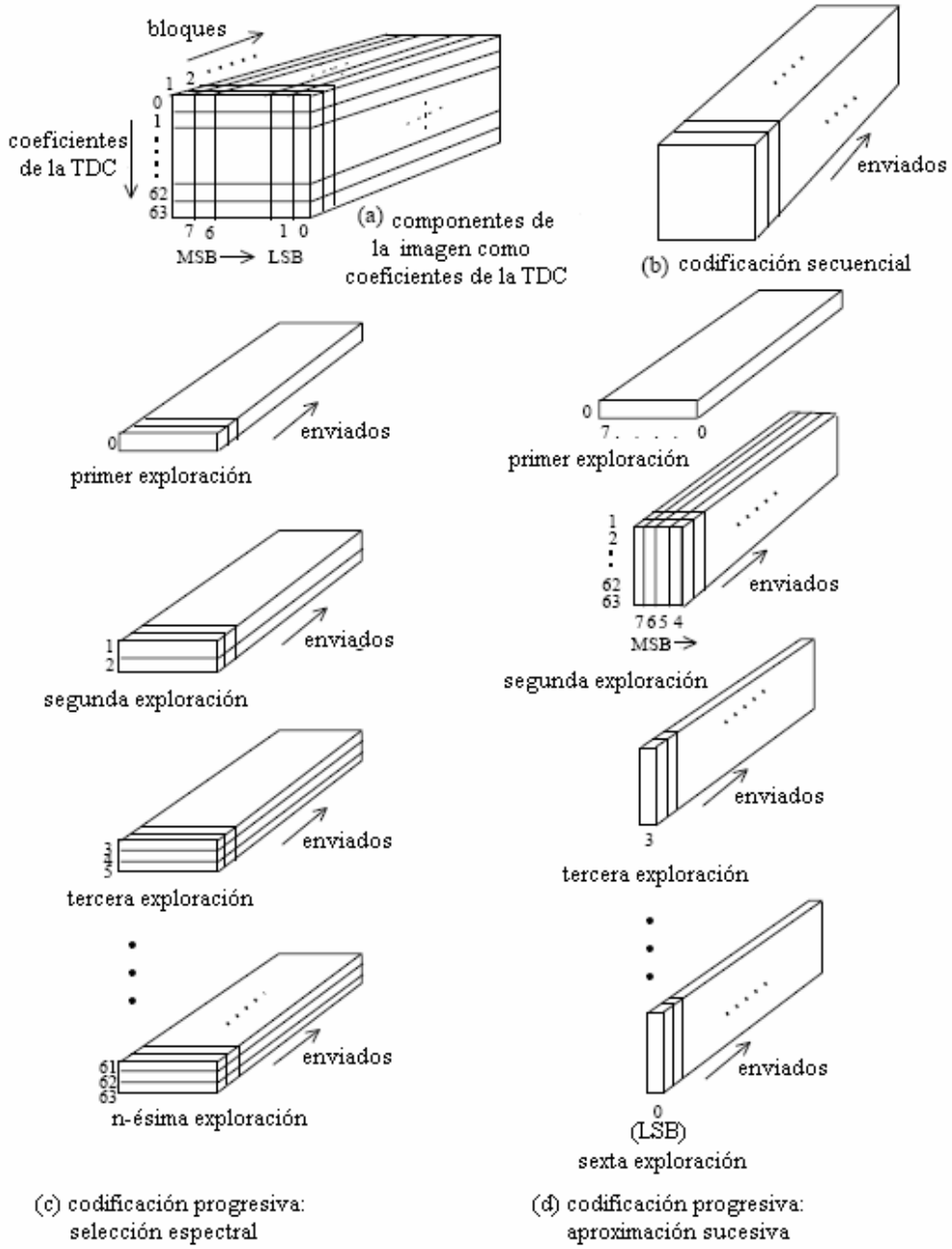


Figura B.11: Selección espectral y métodos de aproximaciones sucesivas de codificación progresiva.

Hay dos métodos complementarios mediante los cuales un bloque de coeficientes TDC cuantizados puede ser parcialmente codificado. En primer lugar, solo una "banda" de coeficientes especificados de la secuencia zig-zag necesita ser codificado en una exploración dada. Este procedimiento se llama "selección espectral," dado que cada banda contiene típicamente coeficientes los cuales ocupan una parte más baja o más alta del espectro espacial de frecuencias para ese bloque 8x8. En segundo lugar, los coeficientes en la actual banda no necesitan ser codificados a su completa exactitud (cuantizados) en una exploración dada. Tras la primera codificación del coeficiente, los N bits más significativos pueden ser codificados primero, donde N es especificable. En subsecuentes exploraciones, los bits menos significativos pueden entonces ser codificados. Este procedimiento se llama ''aproximación sucesiva.'' Ambos procedimientos pueden ser usados separadamente, o intercalados en combinaciones flexibles. Alguna intuición para la selección espectral y la aproximación sucesiva pueden ser obtenidas de la Fig.B.11. La información de los coeficientes TDC cuantizados puede ser vista como un rectángulo para el cual los ejes son los coeficientes TDC (en orden zig-zag) y sus amplitudes. La selección espectral secciona la información en una dimensión y la aproximación sucesiva en la otra.

B.2.1.8. Modo jerárquico de operación

El modo jerárquico provee una codificación "piramidal" de una imagen en resoluciones múltiples, cada una difiriendo en resolución de su codificación adyacente por un factor de dos en la dimensión horizontal o vertical o ambas. El procedimiento de codificación se puede resumir así:

1. Filtrar y sub-muestrear la imagen original por el número deseado en múltiplos de 2 en cada dimensión.

2. Codificar esta imagen de dimensión reducida usando uno de los TDC secuenciales, TDC progresivos, o codificadores sin pérdidas como los descritos previamente.

3. Decodificar esta imagen de dimensión reducida y entonces interpolar y sobre-muestrearla por 2 horizontalmente y/o verticalmente, usando el filtro de interpolación idéntico el cual debe ser usado en el receptor.

4. Usar esta imagen sobre -muestreada como una predicción de la original en esta resolución, y codificar la imagen diferencia usando uno de los TDC secuencial, TDC progresivo, o codificadores sin pérdidas descritos previamente.

5. Repetir los pasos 3) y 4) hasta que la resolución completa de la imagen ha sido codificada.

La codificación en los pasos 2) y 4) se debe hacer usando solo procesos basados en TDC, solo los procesos sin pérdidas, o los procesos basados en TDC con un proceso final sin pérdidas para cada componente. La codificación jerárquica es útil en aplicaciones en las cuales se requiere que una imagen de muy alta resolución sea expuesta a una resolución más baja. Un ejemplo de esto es una imagen escaneada y comprimida en alta resolución para una impresora de muy alta calidad, donde la imagen debe ser también expuesta sobre la pantalla de una PC de baja resolución.

B.2.1.9. Otros aspectos de JPEG

Algunos aspectos principales del estándar propuesto solo pueden ser mencionados brevemente. Entre estos están los puntos concernientes a la representación codificada para los datos de la imagen comprimida especificada en adición a los procedimientos de codificación y decodificación.

Lo que es más importante, una sintaxis *formato de intercambio* es especificada la cual asegura que una imagen JPEG comprimida puede ser intercambiada satisfactoriamente entre diferentes ambientes de aplicaciones. El formato es estructurado en una forma consistente para todos los modos de operación. El formato de intercambio siempre incluye todas las tablas de cuantización codificación entrópica las cuales fueron usadas para comprimir la imagen.

La aplicaciones un estándar específico de una de aplicación) son los "usuarios" del estándar JPEG. El estándar JPEG no impone ninguna exigencia que, en un ambiente de aplicación, todo o aún cualquiera de las tablas debe ser codificada con los datos de la imagen comprimidos durante el almacenamiento o la transmisión. Esto deja a las aplicaciones la libertad para especificar la falta o no de tablas referenciadas si ellas son consideradas apropiadas. También les deja la responsabilidad para asegurar que los decodificadores JPEG acordes usados en su ambiente consiguen cargarse con tablas propias al propio tiempo, y que las tablas propias están incluidas en el formato de intercambio cuando una imagen comprimida es "exportada" fuera de la aplicación. Algunas de las aplicaciones importantes que están listas en el proceso de adoptar la compresión JPEG o han declarado su interés en hacerlo así son el lenguaje PostScript de Adobe para sistemas de impresión [15], la porción del contenido tramado del la arquitectura y formato de intercambio de documentos Office ISO [16], el futuro estándar de facsimil color CCITT, y el estándar de videotexto europeo ETSI [17].

B.2.2. Formato JPEG2000

B.2.2.1. Pre-procesamiento de los datos

JPEG2000 [306] es un joven estándar para compresión y transmisión de imágenes, el cual puede comprimir eficientemente tanto imágenes de tono continuo como indexadas (ver Sección B.2.2.5) en los modos con y sin pérdidas. La resolución y la escalabilidad de calidad, el bajo error de recuperación, y la codificación de regiones-de-interés son solo unas pocas de sus características. JPEG2000 emplea un número de mecanismos los cuales son considerados como el estado del arte en compresión de imágenes con y sin pérdidas. A diferencia de la arquitectura de JPEG (ver Fig.B.12), el diagrama del JPEG2000 está en el espíritu del descripto en la Sección B.2.1 y se muestra en la Fig.B.13. El primer paso del codificador JPEG2000 es la transformación del color. Usualmente las imágenes a color están almacenadas como una serie en la triada RGB. No obstante, el espacio de color RGB no es un color consistente, i.e. una leve variación en la componente de color de un pixel podría conducir a problemas en los bordes sensibles en las otras componentes de color (como blur o deterioro de los bordes). Existen otros espacios de color equivalentes a RGB en los cuales esto no es cierto. El paso inicial en la transformación del color es entonces realizado para convertir imágenes RGB en alguna otra representación más conveniente. La transformación del color no tiene gran impacto sobre la compresión sin pérdidas dado que el dato de la imagen es restaurado exactamente por el decodificador. Por el contrario, es un paso crucial en el modo con pérdidas. La performance de compresión puede beneficiarse también de la independencia de componentes del espacio de color transformado, dado que permite al usuario dejar el canal de luminancia sin cambios, y sub-muestrar las componentes de color sin ninguna pérdida perceptual (o apenas sensible) en la imagen reconstruida.

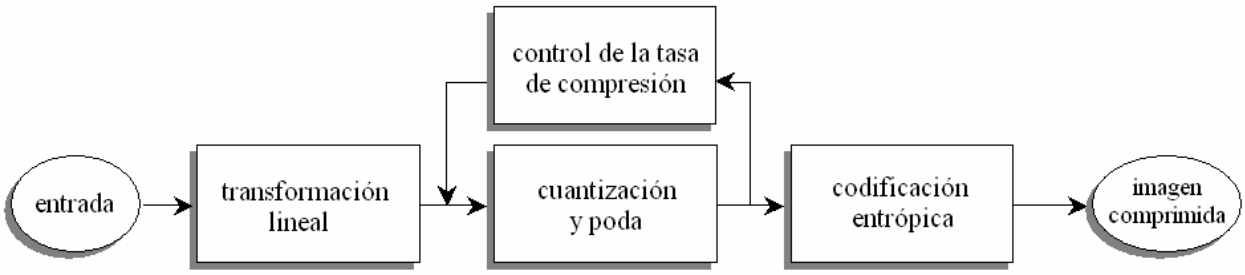


Figura B.12: Esquema común de compresión de imágenes (en el espíritu de JPEG).

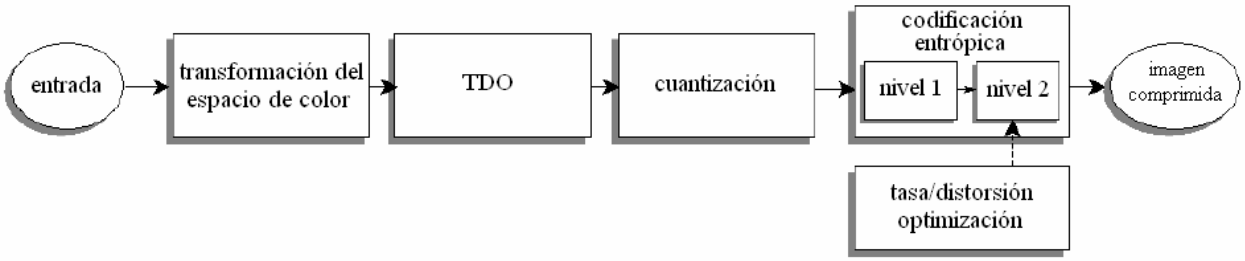


Figura B.13: Esquema de compresión de imágenes JPEG2000.

Los planos de color son codificados separadamente como si ellos fueran imágenes en niveles de grises. Ellos están usualmente divididos en bloques de de igual tamaño llamados mosaicos. Para cada mosaico se genera una cadena de código disjunta usando el mismo esquema. Esto se hace principalmente para mantener baja la carga computacional, y para accesos relativamente aleatorios en el dominio comprimido [307-318].

La transformada de onditas se aplica a los mosaicos antes de la codificación entrópica. El beneficio de emplear la transformada de onditas es que los datos transformados exhiben usualmente una entropía más baja por lo que son más "comprimibles". En particular, dado que la transformada de onditas obtiene cuatro sub-bandas a partir de un mosaico, el modelado de la fuente puede ser mosaiqueado para cada sub-banda. De hecho, dado que los filtros de onditas están diseñados para almacenar frecuencias diferentes en cada sub-banda (ver Fig.B.14), las sub-bandas exhiben características peculiares las cuales son "capturadas" por el modelador fuente JPEG2000. La Fig.B.15 muestra los efectos del filtrado en frecuencia sobre una imagen en niveles de grises. Un número de filtros de onditas diferente es permitido por el estándar para ambos tipos de compresión con y sin pérdidas. Los filtros con pérdidas dan usualmente mejores resultados pero ellos involucran operaciones de punto flotante. Debido a la aproximación de punto flotante, la correcta reconstrucción de las señales de entrada no esta garantizada usando estos filtros. Estos filtros implican solo operaciones enteras que además son permitidas para superar este problema.

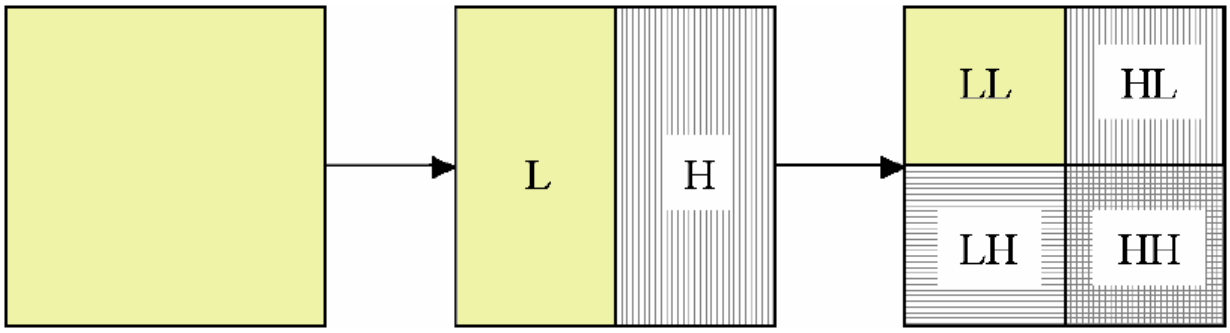


Figura B.14: Transformada de onditas: División en sub-bandas y filtrado en frecuencia.

Figura B.15: Transformada de onditas: efecto del filtrado en frecuencia sobre una imagen en escala de grises.

La transformada de onditas es seguida por un paso de cuantización en la compresión con pérdidas. Ver Apéndice A.

B.2.2.2. EBCOT – Nivel 1

El núcleo de JPEG2000 es su motor de codificación, el Embedded Block Coding with Optimised Truncation (EBCOT, codificación por bloque embebido con truncamiento optimizado) [368]. El algoritmo EBCOT esta conceptualmente dividido en dos capas llamadas niveles. El nivel 1 es responsable por el modelizado de la fuente y la codificación entrópica, mientras que el nivel 2 genera la cadena de salida. La Fig.B.16 muestra una representación esquemática del nivel 1 del EBCOT. Las sub-bandas de onditas son particionadas en pequeños bloques de código (para no

confundirlos con los mosaicos) los cuales a su tiempo son codificados en planos de bits, i.e. bits del coeficiente del mismo orden son codificados juntos (ver Fig.B.17). Los bits más significativos se codifican primero, entonces los bits de orden inferior son codificados en orden descendiente. Cada plano de bits es codificado separadamente como si fuera una imagen de solo dos niveles, excepto por el hecho de que la formación de contexto es guiada no solo por los bits previamente codificados en el mismo plano de bits, sino además porque ha sido visto en el plano de bits de orden más alto. Los planos de bits son además particionados en tres pasos de codificación. Cada paso de codificación constituye una unidad atómica de código, llamada trozo. Los trozos de palabra-código son agrupados en capas de calidad y pueden ser transmitidas en cualquier orden, a condición de que aquellos trozos que pertenecen al mismo bloque de código sean transmitidos en su orden relativo.

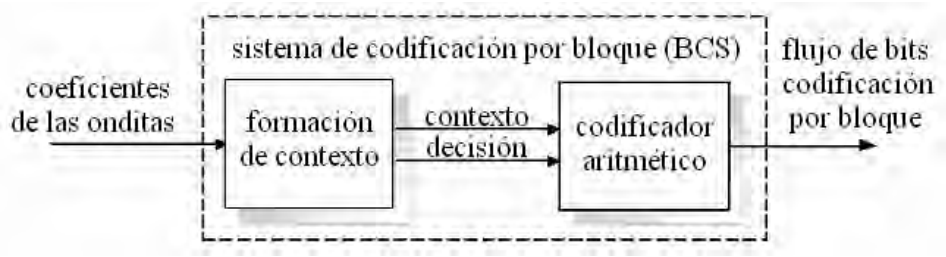


Figura B.16: Representación esquemática del nivel 1 del EBCOT.

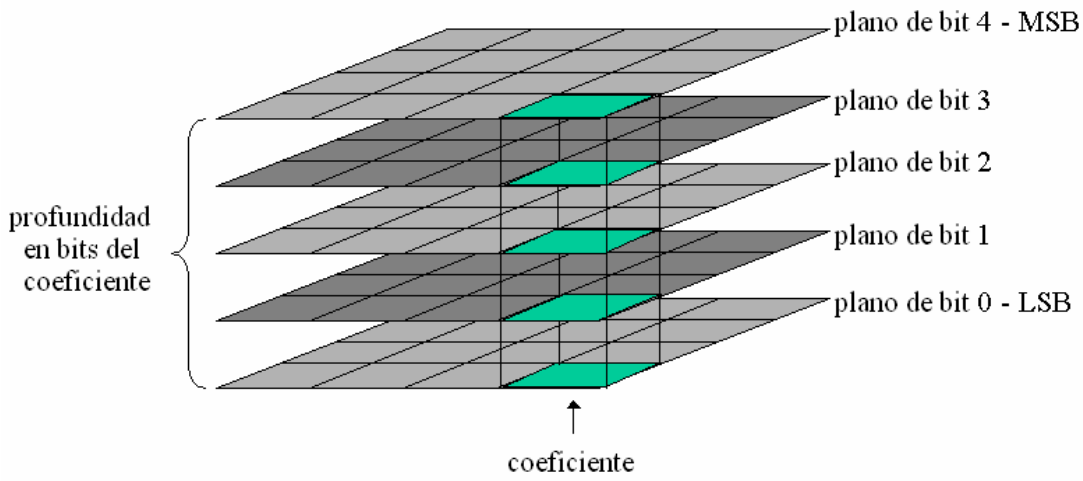


Figura B.17: Codificación por plano de bits: Un ejemplo de codificación por plano de bits de coeficientes onditas con un nivel de profundidad de bits de 5.

Los trozos constituyen puntos de truncamiento válidos, i.e. la palabra-código puede ser truncada al final del trozo sin comprometer la correcta reconstrucción de los datos comprimidos (el particionado es consistente con la sintaxis de la palabra-código). Ambos, el codificador y el decodificador pueden truncar la cadena de bits en correspondencia de un punto de truncado válido para recuperar los datos originales hasta una calidad objetivo. La Fig.B.18 muestra un ejemplo de cadena de código particionada en trozos y capas. La contribución de bloques de código para capas puede ser extremadamente variable y algunos bloques de código no deberían contribuir en todo a algunas capas. Este comportamiento depende de la información contenida en cada trozo, mientras que la contribución de los bloques de código de las capas es requerida para reflejar que tan “útil” es esta información para alcanzar la calidad objetivo dada. Este concepto quedará más claro posteriormente en esta sección.

La cadena de salida es acondicionada por un número de paquetes conteniendo trozos de los bloques de código en una sub-banda dada y pertenencia a la misma capa (i.e. una fila en la Fig.B.18). Dado que la información del encabezado del paquete es altamente redundante, el encabezado mismo emplea técnicas de compresión conocidas para reducir el sobre-encabezado del paquete (ver Sección B.2.2.3).

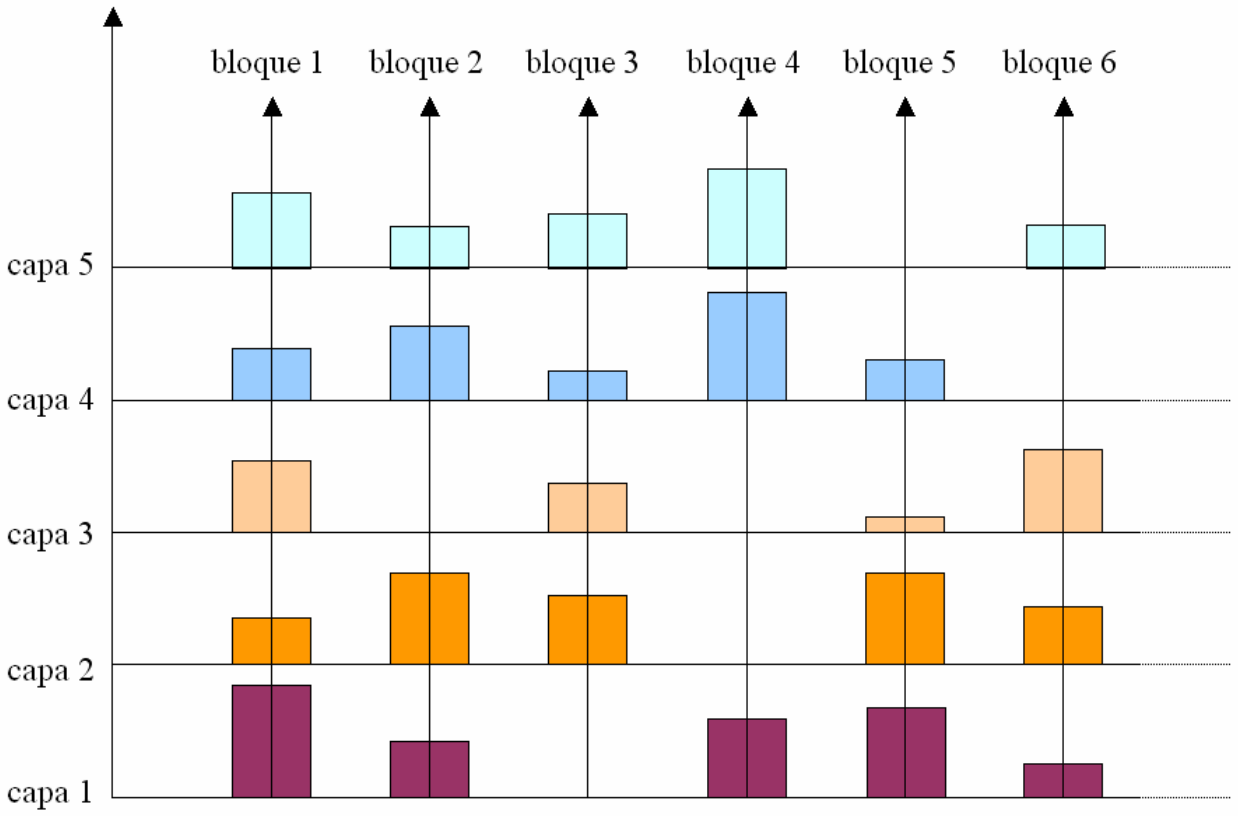


Figura B.18: Partición de la cadena de código: se muestran cinco capas y seis bloques de código. Cada rectángulo representa los trozos de un bloque de código el cual ha sido incluido en la capa relativa. Los colores codifican las capas. Los datos codificados para un bloque se representan como un trozo falso sobre la correspondiente flecha vertical, mientras la cadena de bits comprimida es transmitida de una forma de a capa (es decir, primero los niveles más bajos).

B.2.2.2.1. Modelización de contexto

La modelación del contexto en JPEG2000 es un poco difícil por su complejidad conceptual y es compensada por su impresionante performance de compresión. La modelización de contexto más ampliamente usada es la JPEG-LS [23]. El bit actual (la decisión en la Fig.B.16) es codificado entrópico usando un codificador aritmético cuyos estimados de probabilidad son actuali-zados usando un autómata con 46 estados. La probabilidad estimada se actualiza separadamente para cada una de las 18 diferentes clases usando contextos separados (ver Fig.B.16). Para entender como trabaja la modelización de contexto de JPEG2000 deben quedar claros tres conceptos.

Paso de Codificación – Una exploración completa del plano de bits actual agrupando juntos bits con similares propiedades estadísticas (ver Fig.B.19). Generalmente un paso de codificación no compromete a todos los bits en el plano de bits actual.

Primitiva de Codificación – Un pequeño conjunto de operaciones implica la actualización del estado de los coeficientes siendo codificados y la selección del contexto más apropiado para codificar el bit actual, dado la información estadística disponible sobre los pixeles vecinos.

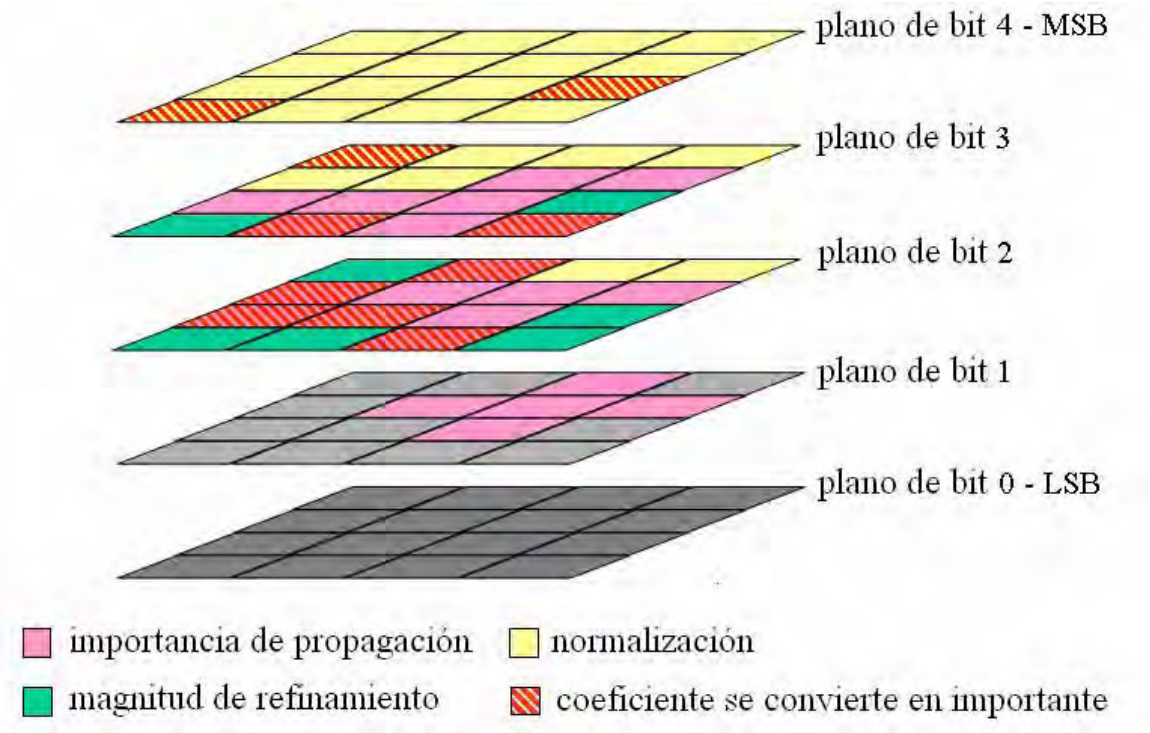


Figura B.19: Paso de codificación: Un ejemplo de codificación con 5 bits de profundidad.

Contexto – Se refiere a toda la información necesaria para mantener la pista del estado actual del autómata finito el cual guía la actualización estadística de cada clase de código. Los contextos están determinados por las primitivas de codificación, las estadísticas de los vecinos, y las propiedades de frecuencia de las sub-bandas.

Son usados tres diferentes pasos de codificado (ver la Fig.B.19): la importancia de la propagación, la magnitud del refinamiento, y la normalización (limpieza). Cada paso es requerido para estimar las probabilidades de los símbolos de una de las tres clases diferentes de bits. Ellos son ejecutados en el orden en el cual son presentados aquí. Cada bit es codificado en uno y solo uno de tres pasos. Las probabilidades de los símbolos son estimadas teniendo en cuenta que ha sido codificado en los planos de bits previos y en los bits vecinos. La vecindad se define como los ocho bits inmediatamente adyacentes al bit actual en el plano de bits actual. Se dice que un coeficiente es significativo si hay al menos un bit no nulo entre sus correspondientes bits codificados en los planos de bits previos. Un coeficiente que no es significativo se llama no-significativo.

El refinamiento de magnitud es usado para refinar la magnitud del coeficiente correspondiente al bit siendo codificado en uno o más bits (el actual). Un bit es codificado en este paso si pertenece a un coeficiente que ya es significativo. La importancia de la propagación es usada para propagar la información importante. Cuando un coeficiente se convierte en significativo, la probabilidad de que sus coeficientes vecinos se conviertan en significativos se hace más y más alta como los beneficios de la codificación. Esto es porque esos bits son codificados usando diferentes estimados de probabilidad. De ahí, la importancia de propagación junta a todos los bits que pertenecen a

los coeficientes no-significativos teniendo una vecindad preferida (i.e. al menos un coeficiente en la vecindad del coeficiente actual es significativo). Finalmente, todos los bits pertenecen a coeficientes no-significativos no teniendo coeficiente significativo en su vecindad que sea codificado en el paso de normalización. Nuevamente, esto se hace porque son usados diferentes estimados de probabilidad dado que es altamente probable que esos coeficientes pertenezcan a áreas homogéneas de imagen. Este procedimiento de particionado de planos de bits es representado en Fig.B.20. Los pasos de codificación, más que proveer una modelización exacta de fuente, da un particionamiento fino de planos de bits en tres conjuntos. En realidad su propósito es principalmente permitir más puntos de truncamiento válidos en la cadena de código comprimido (ver Sección B.2.2.3).

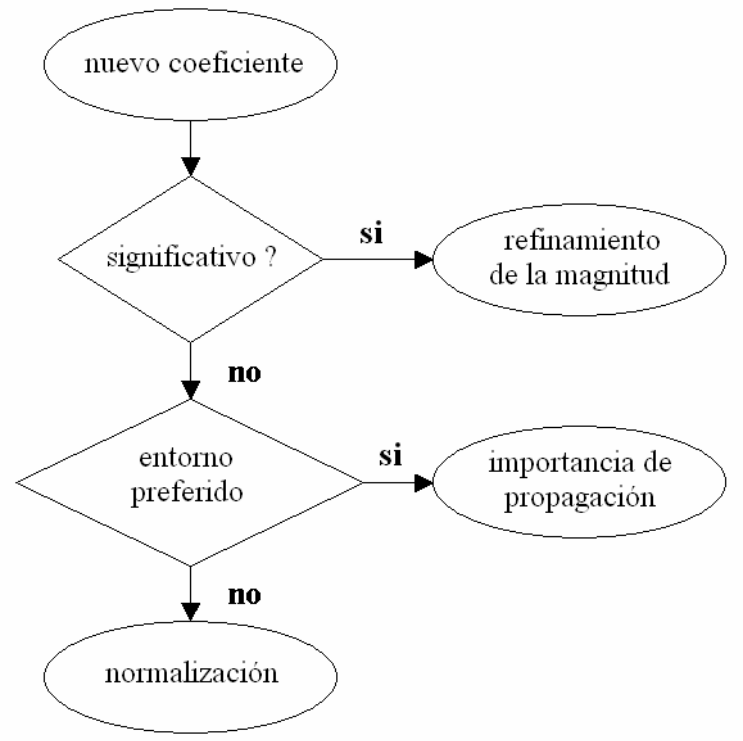


Figura B.20: Diagrama de flujo de la partición de plano de bits.

Las primitivas de codificación son empleadas para obtener una modelización más fina. Ellas son responsables de encontrar el contexto más apropiado para codificar al bit actual. Cuatro primitivas son implementadas: Codificación Cero (CC), Codificación por Longitud de Corrida (CLC), Codificación por Signo (CS), y Refinamiento de Magnitud (RM) (para no confundir con el paso de refinamiento de magnitud). La primitiva CC es ejecutada sobre coeficientes no-significativos. Se puede elegir entre nueve contextos dependiendo de la vecindad de los coeficientes (los vecinos verticales, horizontales, y diagonales están diferenciados) y la sub-banda a la cual pertenece el bit que está siendo codificado. CLC es usada en lugar de CC cuando se codifican grandes áreas homogéneas. Si son encontrados (exactamente) cuatro coeficientes consecutivos no-significativos sin vecindad preferida, solo un bit es codificado para la secuencia entera de cuatro bits para indicar si uno de esos cuatro coeficientes se convertirá en significativo en el actual plano de bits. Entonces un índice adicional de dos bits que indica el primero de ellos es transmitido. Esta primitiva es usada para reducir el número de bits a ser codificados por el codificador entrópico cuando secuencias largas de coeficientes no-significativos son encontradas.

Un contexto particular es reservado para la primitiva CLC. La CS es invocada al menos una vez por coeficiente tan pronto se convierte en significativo, al codificar su signo. La primitiva RM es usada para refinar la magnitud de un coeficiente significativo, para la cual el signo ya ha sido codificado en un plano de bits previo. Se elige el contexto correcto teniendo en cuenta la vecindad del coeficiente. Además, son elegidos diferentes contextos si el RM está siendo ejecutado por primera vez o no sobre el coeficiente actual. Cada paso de codificación emplea estas primitivas para codificar los bits en el plano de bits actual. El paso de propagación de importancia ejecuta un CC por cada bit siendo codificado y un CS por cada bit el cual se convierte en significativo en el plano de bits actual. El paso de normalización tiene el mismo comportamiento excepto para la oportunidad de aplicar CLC en lugar de CC donde sea posible. Finalmente, el paso de refinamiento de magnitud puede llamar solo a la primitiva RM. El símbolo a ser codificado en el contexto relativo es pasado al codificador aritmético, el cual "carga" el estado relativo actual para el contexto dado y lo usa para codificar el símbolo dado. Entonces el estado es actualizado para refinar los estimados de probabilidad para el contexto actual.

B.2.2.2.2. Codificación Aritmética: Codificador MQ

El motor de codificación Aritmética (ver Apéndice A) en JPEG2000 (codificador-MQ) fue desarrollado por el estándar JBIG (Joint Bi-level Image Experts Group, Comité de Grupos de Expertos en Imágenes de dos niveles) para la compresión de imágenes de dos niveles [23]. Es ligeramente diferente de los codificadores aritméticos estándar. Por lo tanto, necesita una corta explicación. El codificador MQ es un codificador entrópico el cual, más que codificar los valores binarios de los símbolos de entrada (el bit actual), codifica una información binaria la cual indica si el símbolo que es codificado es el que esperamos (el símbolo más probable, o MPS) o no (el símbolo menos probable, o LPS). En particular, dos características peculiares del codificador MQ serán discutidas brevemente: la renormalización, y el cambio condicional.

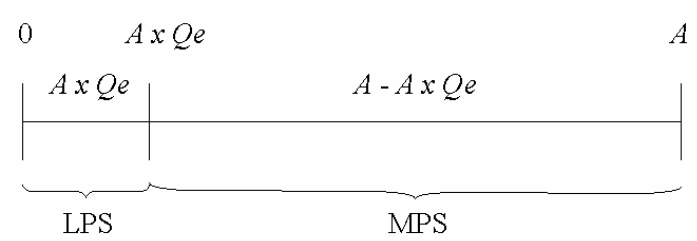


Figura B.21: Intervalo de subdivisión en el codificador MQ.

Recalcamos que en cada paso de la codificación aritmética de una secuencia la sub-secuencia actual es representada como un intervalo en [0, 1]. La Fig.B.21 muestra cómo se particiona el intervalo actual en un codificador MQ. Siguiendo la notación de [306], ***C*** y ***A*** se refieren a la base y la longitud del intervalo actual, mientras que ***Qe*** es la probabilidad estimada del LPS. Las ecuaciones clásicas de actualización pueden entonces ser re-escritas como siguen.

Codificación MPS:

$$C = C + A \times Qe \quad \text{(B.3)}$$

$$A = A - A \times Qe \quad \text{(B.4)}$$

Codificación LPS:

$$C : \text{sin cambio} \quad \text{(B.5)}$$

$$A = A \times Qe \quad \text{(B.6)}$$

Para prescindir de cálculos complejos, se usa un truco para simplificar las ecuaciones. La longitud del intervalo actual se limita a mentir en $\Re = [0.75, 1.5)$. Cuando A cae al mínimo se deja hasta pasado un $A \in \Re$ (renormalización). Con éste simple ajuste $A \approx 1$ es siempre verdadero. Las ecuaciones de arriba pueden entonces ser aproximadas estableciendo $A = 1$ en las multiplicaciones sobre el lado derecho de las Ecuaciones (B.3) (B.4) y (B.6).

Codificación MPS:

$C = C + Qe$ (B.7)

$A = A - Qe$ (B.8)

Codificación LPS:

C : *sin cambio* (B.9)

$A = Qe$ (B.10)

El cálculo de las Ecuaciones (B.7), (B.8), y (B.10) es mucho más eficiente que el cálculo de las ecuaciones relativas no-aproximadas dado que no involucran multiplicaciones, las cuales pueden ser costosas incluso en las implementaciones de hardware. Además, puede ser usada también aritmética de punto fijo para llevar a cabo los cálculos. Esta implementación no da lugar a ninguna una pérdida de eficiencia de codificación, dado que no reduce el espacio de codificación (i.e. la longitud de los dos sub-intervalos generados es todavía la longitud del intervalo actual antes de la división sub-intervalo). No obstante, cuando ***Qe*** es cercano a 0.5 y ***A*** es pequeño, puede suceder que el sub-intervalo MPS sea más pequeño que el sub-intervalo LPS. Como un ejemplo, considere la situación: $A = 0.8$, $Qe = 0.43$. De las Ecuaciones B.8 y B.10 se desprende que $A_{MPS} = 0.37$ y $A_{LPS} = 0.43$, i.e. $A_{MPS} < A_{LPS}$. A los efectos de esquivar tal inversión, es esos casos el intervalo asignado es invertido como se muestra en la Fig.B.22 (cambio condicional). Los algoritmos para codificar MPS y LPS son expresados aquí.

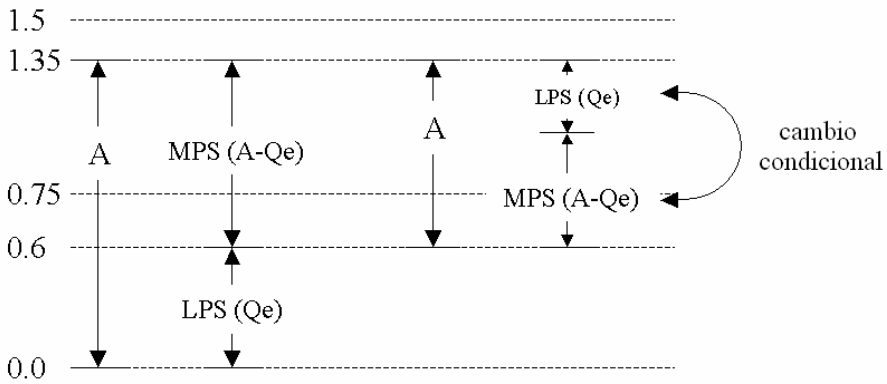


Figura B.22: Cambio condicional.

El proceso de estimación de la probabilidad se basa en un ordenamiento del contador de los límites aproximados de los MPSs por las renormalizaciones del registro ***A***. Luego de cada renormalización el contador se pone a cero y se obtiene un nuevo estimado de ***Qe*** del autómata finito. Si el valor de ***Qe*** es también alto MPS la renormalización es más probable. Esto a su vez, causa que ***Qe*** se fije a un valor más pequeño. Por otro lado, si ***Qe*** es muy pequeño la probabilidad de renormalización de LPS crece y como una consecuencia de ello ***Qe*** tiende aincrementarse. Si la identidad de MPS es errónea ***Qe*** se incrementará hasta aproximadamente 0.5 causando una conmutación de la identidad de MPS.

Algoritmo B.5 Codificación MPS

A = A-Qe	/* Calcular el sub-intervalo MPS */
si A < min	/* Si la renormalización es necesaria */
si A < Qe	/* Si la dimensión del intervalo esta invertida */
A = Qe	/* Poner el intervalo a sub-intervalos LPS */
sino	/* De otra forma */
C = C+Qe	/* Punto a base del intervalo MPS */
fin si	
renormalizar A y C	/* renormalizar */
sino	/* si no renormalización es necesaria */
C = C+Qe	/* Punto a base del intervalo MPS */
fin si	

Algoritmo B.6 Codificación LPS

A = A-Qe	/* Calcular el sub-intervalo MPS */
si A < Qe	/* Si la dimensión del intervalo esta invertida */
C = C+Qe	/* Punto a base del intervalo MPS */
sino	/* De otra forma */
A = Qe	/* Poner el intervalo a sub-intervalos LPS */
fin si	
renormalizar A y C	/* renormalización siempre que sea necesaria */

El pseudo-código para la renormalización está dado abajo. Aquí, ***CX*** es el contexto actual como determinado por el modelador de fuente, y ***I*** es el índice del estado actual para ***CX***. NMPS y NLPS indican respectivamente el próximo estado si un MPS o n LPS es encontrado.

Algoritmo B.7 Renormalización MPS

I = indice(CX)	/* Indice actual para el contexto CX */
I = NMPS(I)	/* Nuevo índice para el contexto CX */
Indice(CX) = I	/* Salvar este índice en el contexto CX */
Qe(CX) = valor_Qe(I)	/* Nueva probabilidad estimada para CX */

Algoritmo B.8 Renormalización LPS

I = indice(CX)	/* Indice actual para el contexto CX */
si selección(I) = I	/* Si ocurre un cambio condicional */
MPS(CX) = 1-MPS(CX)	/* Cambio en el sentido MPS */
fin si	/* (1 -> 0 o 0 -> 1) */
I = NLPS(I)	/* Nuevo índice para el contexto CX */
Indice(CX) = I	/* Salvar este índice en el contexto CX */
Qe(CX) = valor_Qe(I)	/* Nueva probabilidad estimada para CX */

B.2.2.3. EBCOT – Nivel 2

La cadena de bits de JPEG2000 es escalable con respecto a la resolución (gracias a la descomposición de onditas) y la calidad, y provee un acceso aleatorio razonable para bloques de imágenes en el dominio comprimido. La escalabilidad de la calidad se obtiene por medio del plano de bits y de la codificación del paso de particionado. Todos los bits codificados para el paso de codificación constituyen una unidad atómica y son transmitidos como bloque único. No obstante, los trozos comprimidos necesitan ser transmitidos solo hasta un número mínimo para alcanzar una

calidad definida por el usuario. Es una especie de cuantización desigual post-compresión la cual usualmente da mejores resultados que los métodos de cuantización estándar, en función de la calidad de la imagen reconstruida. Con el fin de que el usuario pueda interrumpir la cadena hasta la resolución y/o calidad deseadas, o visualizar un área particular de la imagen, la estructura de cadena de bits debe ser cuidadosamente organizada. Esta tarea es llevada a cabo por el nivel 2 del algoritmo EBCOT. La cadena es seccionada en paquetes conteniendo la información del encabezado en adición a la cadena misma. En particular, los encabezados de los paquetes contienen información útil para recuperar los trozos de la palabra de código en la cadena de salida, e.g. si cualquier bloque de una determinada sub-banda de ondita es incluida en el paquete o no, la longitud de la palabra código, y así sucesivamente.

B.2.2.3.1. Puntos de truncado óptimo

La salida del nivel 1 es una serie de palabras código una por cada bloque de código) con una indicación sobre los puntos de truncado válidos. Esta información complementaria es usada para encontrar los puntos de truncado óptimos en un sentido tasa/distorsión, dando una tasa de bits objetivo. El conjunto de puntos de truncado los cuales dan juntos el mejor compromiso entre longitud de palabra código y la cantidad correspondiente de distorsión con respecto a la imagen original (i.e. calidad), es seleccionada para cada bloque de código incluido en el paquete. Encontrar los puntos de truncado óptimo implica la minimización de la relación tasa/distorsión. Indiquemos con $\boldsymbol{R_i^j}$, $\boldsymbol{j} = 1,2,\ldots$ la secuencia válida de puntos de truncado de la palabra código correspondientes al $\boldsymbol{i}$-ésimo bloque de código, y con $\boldsymbol{D_i^j}$, $\boldsymbol{j} = 1,2,\ldots$ las distorsiones asociadas. Si indicamos con $\boldsymbol{n_i}$ el punto de truncado óptimo para la $\boldsymbol{i}$-ésima palabra código, la longitud total de la cadena de bits puede ser escrita como $\boldsymbol{R} = \sum_i \boldsymbol{R_i^{n_i}}$ y $\boldsymbol{D} = \sum_i \boldsymbol{D_i^{n_i}}$ es la distorsión global. Los puntos de truncado óptimo $\boldsymbol{n_i}$ son encontrados en correspondencia de las longitudes de las palabras código minimizando la distorsión global. Dada una tasa de bits objetivo $\boldsymbol{R_{max}}$, queremos minimizar la distorsión global $\boldsymbol{D}$ sujeta a las restricciones $\boldsymbol{R} = \boldsymbol{R_{max}}$. Un método bien conocido para hacer el trabajo es el de los multiplicadores de Lagrange para problemas de optimización con restricciones. En nuestra configuración tenemos que encontrar un valor λ el cual minimize la función $\boldsymbol{D} + \lambda \boldsymbol{R}$. De todos modos, dado que existen un gran número finito de puntos de truncado no podemos garantizar que $\boldsymbol{R} = \boldsymbol{R_{max}}$. Esta condición es entonces relajada y se convierte en $\boldsymbol{R} \leq \boldsymbol{R_{max}}$. Por lo tanto, buscamos la longitud del código $\boldsymbol{R}$ más cercano a $\boldsymbol{R_{max}}$ el cual minimiza nuestra función costo. Se puede profundizar acerca de este tema en [307].

B.2.2.3.2. Encabezado del paquete

Dada una tasa de bits objetivo, los puntos de truncado óptimo definen el número de bits que aparecen en una capa dada para cada bloque de código. Dependiendo de su contenido de información, un código bloque dado puede estar completamente incluido en una capa o, en el otro lado, puede no aparecer en absoluto en la capa. Por lo tanto, el encabezado del paquete debería codificar la inclusión de información. Otra información importante contenida en el encabezado del paquete es la longitud de la palabra código para cada bloque de código incluido y el número de los planos de bits nulos más significativos, el cual es útil para eludir el gasto de tiempo y bits para codificar secuencias largas de ceros. Todo en estas tres piezas de información muestra alta redundancia, en ambos casos, dentro de la misma capa y entre diferentes capas (ver [306-318]). Una estructura de árbol particular, el árbol etiquetado, es adoptada para exhibir esta redundancia para obtener una representación más compacta de los encabezados de los paquetes. A continuación, serán introducidos los árboles etiquetados y se mostrará como son empleados eficiente-

mente para codificar información del encabezado de los paquetes.

B.2.2.3.2.1. Arboles etiquetados

Un árbol etiquetado es una estructura particular de árbol concebida para la codificación eficiente de matrices bi-dimensionales altamente redundantes. La información a ser codificada esta asociada a las hojas. Su estructura se asemeja a un árbol cuaternario. No obstante, ellos tienen dos diferencias principales con respecto a los árboles cuaternarios convencionales: los árboles etiquetados no necesitan estar valuados en forma binaria, y la información en un árbol etiquetado puede ser codificada en cualquier orden, a condición de que el decodificador siga el mismo orden. La Fig.B.23 muestra un ejemplo de un árbol etiquetado.

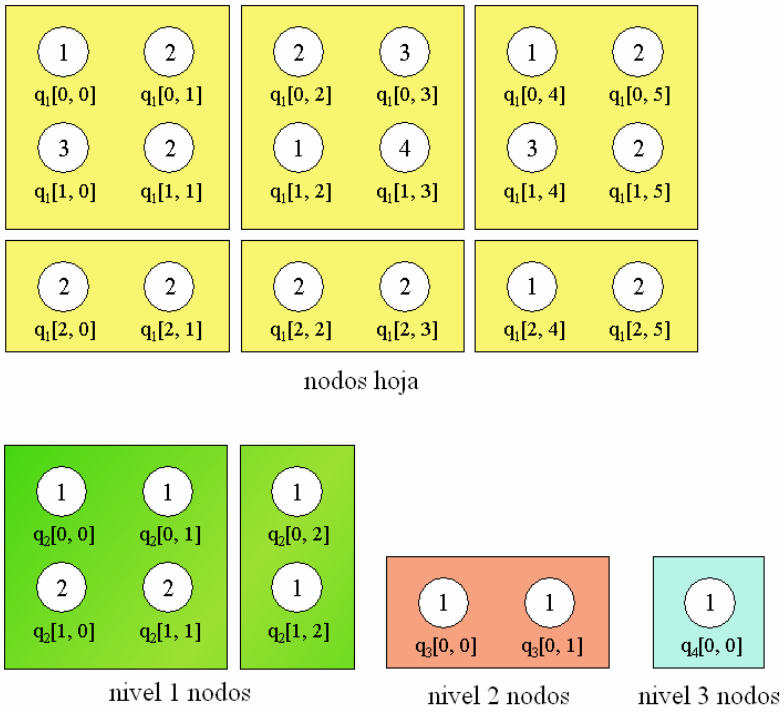


Figura B.23: Un ejemplo de árbol etiquetado.

$q_1[m, n]$ denota un arreglo de cantidades (enteros no-negativos) que nos gustaría representar a través de árboles etiquetados. Asociamos estas cantidades con los nodos hoja del árbol. Los nodos de los próximos niveles son rotulados como $q_2[m, n]$, $q_3[m, n]$, y así sucesivamente. Cada nodo interno está asociado con un bloque de 2×2 de nodos en el nivel más bajo, y contiene el mínimo entre los valores de sus cuatro hijos. La raíz contiene el mínimo de toda la estructura. El árbol etiquetado codifica eficientemente la información $q_1[\overline{\boldsymbol{m}}, \overline{\boldsymbol{n}}] \geq t$, donde $q_1[\overline{\boldsymbol{m}}, \overline{\boldsymbol{n}}]$ es un valor en la matriz de entrada y t e el valor de un umbral. El algoritmo para construer el árbol etiquetado es usualmente expresado como un procedimiento $\boldsymbol{T}(m, n, t)$ el cual codifica la cadena de bits de longitud mínima para indicar si $q_1[\overline{\boldsymbol{m}}, \overline{\boldsymbol{n}}] \geq t'$, $\forall\, t'$: $1 \leq t' \leq t$, dada la información la cual ha sido codificada en llamadas previas. A saber, el algoritmo evita codificar de nuevo la información codificada en previas llamadas para los nodos intermedios. Esta información se mantiene por medio de una variable de estado $t_k[m, n]$, la cual asegura que suficiente información ha sido codificada para saber si $q_1[\overline{\boldsymbol{m}}, \overline{\boldsymbol{n}}] \geq t'$, $t' = 1, 2 \ldots t_k[m, n]$. Es inicializado a cero, dado que la información no ha sido codificada en el comienzo.

Algoritmo B.9 Procedimiento de codificación del árbol etiquetado ($T[m, n, t]$).

(1) $k = K$	/* comenzar en el nodo raíz */
(2) $t_{mín} = 0$	/* se usa para propagar el conocimiento a los descendientes */
(3) $\boldsymbol{m}_k = \left\lfloor \frac{\overline{\boldsymbol{m}}}{2^{k-1}} \right\rfloor, \quad \boldsymbol{n}_k = \left\lfloor \frac{\overline{\boldsymbol{n}}}{2^{k-1}} \right\rfloor$	/* $[\boldsymbol{m}_k, \boldsymbol{n}_k]$ es la ubicación del nodo del ancestro hoja en el nivel k */
(4) si $\boldsymbol{t}_k[\boldsymbol{m},\boldsymbol{n}] < \boldsymbol{t}_{\min}$ hacer $\boldsymbol{t}_k[\boldsymbol{m},\boldsymbol{n}] = \boldsymbol{t}_{\min}$	/* actualizar la variable de estado */
(5) si $\boldsymbol{t} \le \boldsymbol{t}_k[\boldsymbol{m},\boldsymbol{n}]$ (5.1) si $k = 1$ se realizan (5.2) de otra manera - poner $\boldsymbol{t}_{\min} = \min\{\boldsymbol{t}_k[\boldsymbol{m}_k,\boldsymbol{n}_k], \boldsymbol{q}_k[\boldsymbol{m}_k,\boldsymbol{n}_k]\}$ - decrecer k e ir al paso (3)	/* Hemos alcanzado la hoja. Ha sido codificada suficiente información */
(6) de otra forma (i.e. si $\boldsymbol{t} > \boldsymbol{t}_k[\boldsymbol{m},\boldsymbol{n}]$) - si $\boldsymbol{q}_k[\boldsymbol{m}_k,\boldsymbol{n}_k] > \boldsymbol{t}_k[\boldsymbol{m}_k,\boldsymbol{n}_k]$ emite un bit 0 - sino $(\boldsymbol{q}_k[\boldsymbol{m}_k,\boldsymbol{n}_k] = \boldsymbol{t}_k[\boldsymbol{m}_k,\boldsymbol{n}_k])$ emite un bit 1 - incrementar $\boldsymbol{t}_k[\boldsymbol{m}_k,\boldsymbol{n}_k]$ e ir al paso (5)	/* enviar suficiente información para incrementar $\boldsymbol{t}_k[\boldsymbol{m}_k,\boldsymbol{n}_k]$ */

Algoritmo B.10 Procedimiento de codificación del árbol etiquetado.

(1) $k = K$	/* comenzar en el nodo raíz */
(2) $t_{mín} = 0$	/* se usa para propagar el conocimiento a los descendientes */
(3) $\boldsymbol{m}_k = \left\lfloor \frac{\overline{\boldsymbol{m}}}{2^{k-1}} \right\rfloor, \quad \boldsymbol{n}_k = \left\lfloor \frac{\overline{\boldsymbol{n}}}{2^{k-1}} \right\rfloor$	/* $[\boldsymbol{m}_k, \boldsymbol{n}_k]$ es la ubicación del nodo del ancestro hoja en el nivel k */
(4) si $\boldsymbol{q}_k[\boldsymbol{m},\boldsymbol{n}] < \boldsymbol{t}_{\min}$ hacer $\boldsymbol{q}_k[\boldsymbol{m},\boldsymbol{n}] = \boldsymbol{t}_{\min}$	/* actualizar la variable de estado */
(5) si $\boldsymbol{s}_k[\boldsymbol{m}_k,\boldsymbol{n}_k]$ es verdadero - poner $\boldsymbol{t}_{\min} = \boldsymbol{q}_k[\boldsymbol{m}_k,\boldsymbol{n}_k]$ - decrecer k e ir al paso (3)	/* suficiente información ha sido codificada para el nivel actual */ /* actualizar $\boldsymbol{t}_{\min}$ para propagar el conocimiento */ /* y proceder al próximo nivel */
(6) de otra forma ($\boldsymbol{s}_k[\boldsymbol{m}_k,\boldsymbol{n}_k]$ es falso) - si $\boldsymbol{t} \le \boldsymbol{q}_k[\boldsymbol{m}_k,\boldsymbol{n}_k]$ - si $k = 1$ se realizan - de otra forma - poner $\boldsymbol{t}_{\min} = \boldsymbol{q}_k[\boldsymbol{m}_k,\boldsymbol{n}_k]$ - decrecer k e ir al paso (3) - de otra forma (i.e. si $\boldsymbol{t} > \boldsymbol{q}_k[\boldsymbol{m}_k,\boldsymbol{n}_k]$) - sea b el próximo símbolo de entrada - si $b = 0$ - incrementar $\boldsymbol{q}_k[\boldsymbol{m}_k,\boldsymbol{n}_k]$ - sino poner $\boldsymbol{s}_k[\boldsymbol{m}_k,\boldsymbol{n}_k]$ a verdadero - ir al paso (5)	/* hemos alcanzado la hoja. Suficiente información ha sido codificada */ /* actualizar $\boldsymbol{t}_{\min}$ para propagar el conocimiento sin enviar nuevos bits */ /* leer suficiente información para actualizar el valor de $\boldsymbol{q}_k[\boldsymbol{m}_k,\boldsymbol{n}_k]$ */ /* Un 0 significa que puede */ /* actualizar nuestro conocimiento de $\boldsymbol{q}_k[\boldsymbol{m}_k,\boldsymbol{n}_k]$ */ /* 1 significa que no es necesaria más información para codificar este nodo a partir de ahora */

El algoritmo de codificación se muestra en el Algoritmo B.9. Como un ejemplo, considerar el árbol etiquetado en la Fig.B.23 y suponer que queremos codificar suficiente información para un umbral $t = 1$, usando el Algoritmo B.9.

Comenzamos con $q_1[0, 0]$ y procedemos en un orden por barrido (i.e. de izquierda a derecha, y luego la fila de abajo). Como la prueba $t \leq t_4[0, 0]$ falla en el paso (5) (i.e. no ha sido codificada suficiente información todavia para la raíz) nos vemos impulsados al paso (6), donde $t_4[0, 0]$ es incrementado y un bit 0 es emitido. Luego se conduciría de nuevo al paso (5).

Esta vez la prueba es satisfactoria por lo que t_{min} es actualizado en el paso (5.2) y pasamos al nivel inferior del árbol. Para cada nivel subsecuente del árbol etiquetado nada es codificado dado que la información reunida ya es suficiente. Finalmente, Acabamos con un único bit 0 a codificar si $q_1[0, 0] \geq 1$. No se necesita información adicional a codificar si $q_1[m, n] \geq 1, \forall\, m, n$ dado que sabemos que la raiz contiene el mínimo del árbol completo, y este valor no es menor que el umbral dado. La Fig.B.24 muestra los bits codificados para el árbol etiquetado incrementando el valor del umbral a dos. Observe que la secuencia de los umbrales utilizados para codificar debe ser conocida por el decodificador. De hecho, el decodificador puede reconstruir los datos originales solo si la secuencia es decodificada en el mismo orden que fue codificada. Por lo tanto, los valores de la Fig.B.24 son válidos solo si el procedimiento de codificación ha sido previamente ejecutado con umbral $t = 1$.

1111	0	10	0	111	0
0	0	1	0	0	0
0	-	0	-	11	0

Figura B.24: Bits codificados para el árbol etiquetado de la Fig.B.23. El valor del umbral se establecido en 2. El procedimiento de codificado ha sido previamente ejecutado con un umbral de $t = 1$.

El algoritmo de decodificado sigue la misma idea general. Dadas las dimensiones de la matriz original y un umbral, se pueden extraer los datos originales al propagar en niveles más bajos la información ya conocida por los niveles superiores. Un pseudocódigo para el decodificador esta dado en el Algoritmo B.2. Una variable de estado Booleana adicional, $s_k[m_k, n_k]$, es usada en lugar de la variable de estado $t_k[m, n]$ del procedimiento de codificado para indicar cuando la información suficiente ha sido decodificada para determinar los valores de $q_k[m_k, n_k]$. Cuando $s_k[m_k, n_k]$ se convierte en verdad no mas bits serán decodificados para $q_k[m_k, n_k]$.

B.2.2.3.2.2. Codificando la información del encabezado del paquete

El árbol etiquetado encuentra una aplicación natural en la información del encabezado del paquete de codificado JPEG2000. En particular, son usados para codificar dos campos de cada cuatro. Los valores contenidos en el encabezado del paquete para cada bloque de código son:

- información de inclusión,
- máximo número de bits en la mantisa de los coeficientes del bloque código (i.e. el número de planos de bits nulos más significativos).
- número de pasos nuevos incluidos (determinado como una función de los puntos de truncado óptimo),
- longitud del código de palabra

La información de inclusión es codificada usando un árbol etiquetado para indicar la primera capa en la cual se incluye un código de bloque dado, entonces se usa un único bit para más capas a los efectos de indicar si el bloque de código se incluye o no. Los árboles etiquetados son eficientemente codificados para esta clase de información dado que la vecindad de los bloques de código tienden a aparecer en capas cercanas unas de otras. La misma observación se mantiene verdadera para el número de planos de bits nulos más significativos. El número de pasos de codificación incluidos usan un esquema de codificación diferente, similar a la codificación de Huffman en esos valores bajos (altamente probable) obtiene menos bits que los valores altos (menos probable). Finalmente, la longitud de la palabra código es codificada como un código de dos partes en el cual la primera parte indica la longitud de la segunda. Esto a su vez indica la longitud de la palabra código. En particular, el número de bits en la segunda parte es calculada como $\boldsymbol{n}_{bits} = \lfloor \log_2(\boldsymbol{p}_i) \rfloor$, donde $\boldsymbol{p}_i$ es el número de nuevos pasos de codificación incluidos en el paquete actual. Este delicado esquema se destina como modelo del hecho que la longitud de la palabra código de alguna manera proporcional al número de pasos de codificación incluidos.

La estructura orientada a los paquetes de la cadena comprimida de JPEG2000 permite alta escalabilidad en el dominio comprimido. La progresión del orden de decodificación se determina al colocar adecuadamente los paquetes en la cadena de salida.

B.2.2.4. Resumen del algoritmo JPEG2000

Para tener una idea global del algoritmo JPEG2000, resumimos aquí las operaciones llevadas acabo por el codificador (referentes a la Fig.B.13). La imagen de entrada se somete a una transformación de color preliminar en un espacio de color separable (para la cual no haya correlación mutua entre las bandas como es el caso de RGB). Los planos de color (aparte del plano de luminancia) son posiblemente sub-muestreados (compresión con pérdidas). Entonces, la imagen es particionada en grandes bloques llamados mosaicos. Se aplica la descomposición de onditas a los mosaicos separadamente. Bloques pequeños (posiblemente cuantizados) de coeficientes de onditas son procesados separadamente usando el algoritmo EBCOT (Fig.B.16). En el nivel 1 tienen lugar el modelado de la fuente y la compresión entrópica. El modelado de la fuente se lleva a cabo en un nivel de plano de bits y es impulsado por la información estadística sobre los coeficientes de la vecindad. Su salida es un contexto en el cual ayuda el codificador aritmético para actualizar correctamente las probabilidades de los símbolos. Luego de que ha sido codificado cada bloque en un mosaico, el nivel 2 lo posiciona convenientemente en la cadena de bits de salida comprimida. La información del encabezado es comprimida así por medio de una estructura particular de árbol llamada árbol etiquetado. Una cuantización apropiada puede llevarse a cabo en esta fase de ambos lados, codificador y decodificador al truncar la cadena de bits hasta un punto de truncado óptimo. El decodificador realiza los mismos pasos en orden invertido.

B.2.2.5. Codificación de imágenes indexadas

Cuando la compresión de imágenes muestra un número bajo de colores distintos, se adopta usualmente el esquema de compresión de imágenes indexadas. Aquí la observación reside en que el número de colores en una imagen no puede exceder nunca el número de pixeles, y es usualmente mucho más baja. Entonces, podemos asignar un índice para cada color y almacenar este en lugar del color relativo. Una tabla de color (paleta) almacena la triada RGB correspondiente a las imágenes indexadas. Si el número de colores es lo suficientemente bajo (i.e. la representación de los índices es corta), los costos generales pagados para almacenar la paleta son compensados por la ganancia en la representación de los pixeles. La matriz de indices puede ser comprimida

usando cualquier método genérico de compresión sin pérdidas. Este esquema es muy efectivo cuando el número de colores es bajo, mientras su eficiencia se degrada a medida que el número de colores se hace más grande. Esto, por supuesto, depende de la dimensión de la tabla de colores. Los métodos de compresión de imagines indexadas se comportan bien con imagines de pocos colores, tal como líneas dibujadas, imágenes en blanco y negro y texto pequeño con solo unos pocos pixeles de alto. Por el contrario, la performance de codificación de tales algoritmos para imágenes de tono continuo, tales como fotos, puede ser muy pobre.

JPEG2000 puede ser usado para comprimir imágenes indexadas tan bien como aquellas de tono continuo. Datos experimentales muestran que JPEG2000 puede obtener mejores resultados que GIF o PNG si es elegida una apropiada indexación de color. La idea es suavizar las transiciones de índices en la imagen indexada. Desafortunadamente, dada una imagen $\boldsymbol{I}$ con $\boldsymbol{M}$ colores distintos, el número posible de colores indexados es $\boldsymbol{M!}$. La obtención de un esquema de re-indexado óptimo se dificulta al ser un problema intrínsicamente difícil (NP-completo). Entonces solo los métodos heurísticos pueden escoger un ordenamiento óptimo de los muchos posibles que se conocen. En esta sección revisaremos dos de ellos [306,312].

Ambas aproximaciones presentarán una recopilación de las estadísticas de adyacencia del color usando una matriz $\boldsymbol{C}$ llamada matriz de co-ocurrencia. Cada entrada $\boldsymbol{C}(i, j)$ reporta el número de veces que el par de símbolos de color vecinos (c_i, c_j) ha sido observado sobre la imagen indexada. $\boldsymbol{C}$ es usada para ponderar los pares de índices para guiar el proceso de re-indexado. La matriz de co-ocurrencia podría ser re-organizada en función del perfil del esquema predictivo particular adoptado por el motor de compresión. Ambos algoritmos construyen una cadena de colores indexados y entonces deriva un simple re-indexado mediante la asignación de índices consecutivos para colores consecutivos. El primer método que presentamos fue propuesto por Zeng *et al* [306] durante el proceso de estandarización de JPEG2000. Es un ambicioso algoritmo iterativo basado en la maximización de la función potencial definida como $\boldsymbol{w}_{N,j} = \log_2\left(1+\frac{1}{\boldsymbol{d}_{N,j}}\right)$, donde $\boldsymbol{d}_{N,j}$ es la distancia del j-ésimo índice desde el fin de la cadena de índices actual. $\boldsymbol{N}$ indica la longitud de la cadena. En cada paso se adiciona un índice de color a uno de los de los criterios de evaluación de la cadena actual al maximizar la función $\boldsymbol{w}_{N,j}$. La cadena es inicializada al adicionar el color c_i maximizando $\sum_{j\neq i}\boldsymbol{C}(\boldsymbol{i},\boldsymbol{j})$.

En la aproximación introducida por Battiato *et al* [312] el problema del re-indexado es mapeado sobre un problema de optimización para un camino Hamiltoniano en un grafo ponderado. Dada la matriz de co-ocurrencias $\boldsymbol{C}$ calculada como se indica arriba, un grafo $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ se define como sigue:

$\boldsymbol{V} = \{ c_1, c_2, \ldots c_M\}$
$\boldsymbol{E} = \boldsymbol{V} \times \boldsymbol{V}$
$\boldsymbol{w} : \boldsymbol{E} \rightarrow \boldsymbol{N}$ donde $\boldsymbol{w}$ se define como

$$\boldsymbol{w}(\boldsymbol{i},\boldsymbol{j}) = \begin{cases} \boldsymbol{C}(\boldsymbol{i},\boldsymbol{j}) + \boldsymbol{C}(\boldsymbol{j},\boldsymbol{i}) & \boldsymbol{i} \neq \boldsymbol{j} \\ 0 & \boldsymbol{i} = \boldsymbol{j} \end{cases}$$

Un re-indexado es encontrado al buscar por el camino Hamiltoniano más pesado en $\boldsymbol{G}$, i.e. una solución para el Problema de Viajante de Comercio (Travelling Salesman Problem, TSP), el cual

es un bien conocido problema NP-Completo. Los autores sugieren una solución muy eficiente para resolver el TSP de una manera aproximada.

Los resultados experimentales muestran que los dos algoritmos presentados son comparables en función de las tasas de compresión, eficiencia computacional, y consumo de memoria.

B.3 Conclusiones del apéndice

En este apéndice se presentaron los dos algoritmos con los cuales confrontaremos las soluciones propuestas, es decir, JPEG y JPEG2000, en el Capítulo 4 (Simulaciones computacionales).

Apéndice C

Tipos de exploración de los mosaicos

C.1 Introducción

Este apéndice contiene los distintos métodos de exploración de los bloques. Mas allá de cómo se generen los mosaicos, el procedimiento consiste en explorar la imagen por bloques y recolectar los mismos de manera consecuente con una mayor o menor afinidad entre ellos según un determinado criterio en el contexto de su uso.

C.2 Tipos de exploración

La Fig.C.1 muestra seis diferentes tipos de exploración espacial de los bloques [10]. En el presente trabajo usaremos los métodos (a), (d) y (f).

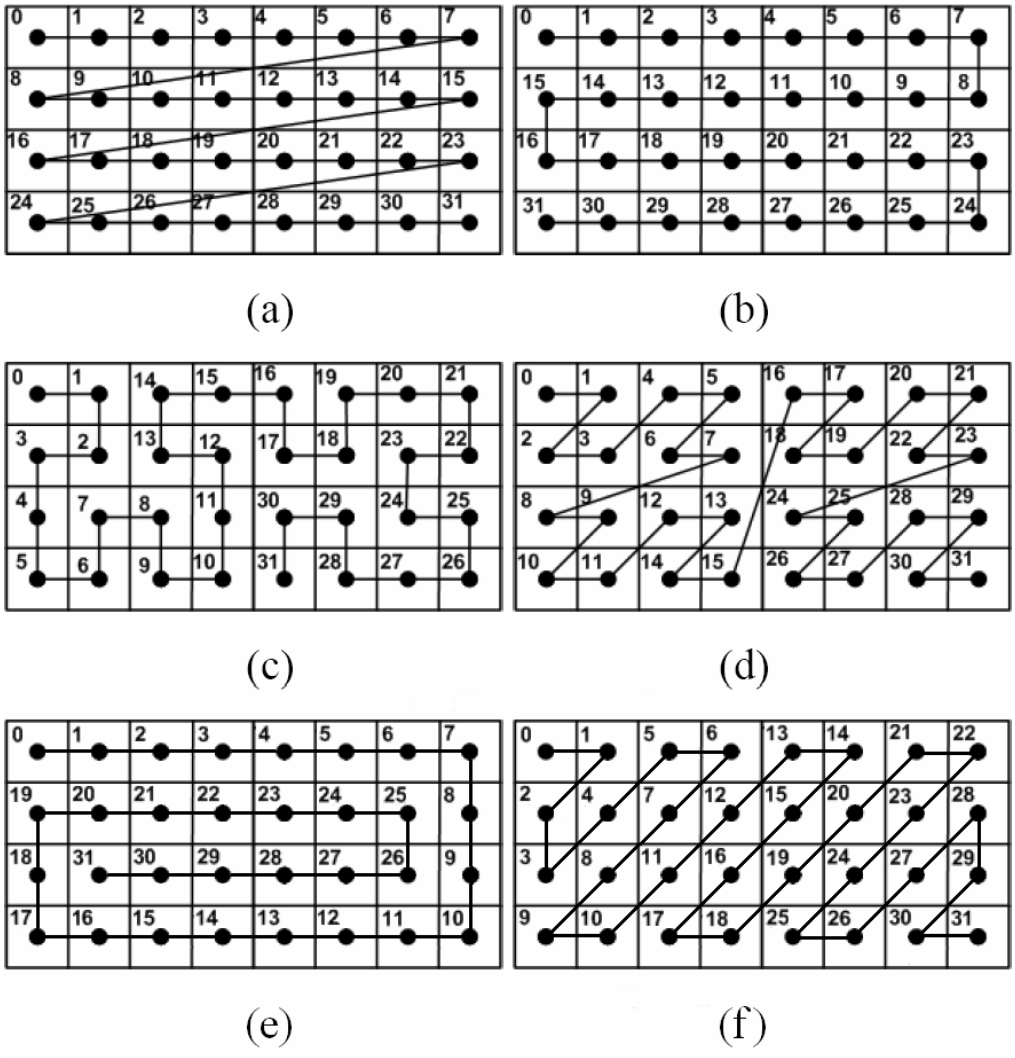


Figura C.1: Diferentes métodos de exploración de los bloques. a) Orden de barrido por filas, b) orden de filas primas, c) orden de Peano-Hilbert, d) orden Z-Morton, e) orden espiral, y f) orden en zig-zag.

En la Fig.C.1 cada celda numerada representa un sub-bloque (por ejemplo en el ámbito del dominio de una transformada que trabaje con bloques o mosaicos) la cual debe estar espacialmente ordenada (en orden creciente) en una matriz tridimensional antes de aplicar a esta última la TDKL, ver Fig.C.2.

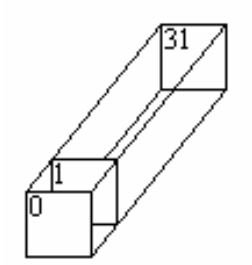


Fig.C.2: Construcción de una matriz-3D con los sub-bloques en orden creciente.

Como se pudo observar en la Fig.C.1, los pixeles, los cuales pueden ser tratados con cualquier transformada lineal, se encuentran concentrados formando lo que llamamos bloques o mosaicos. En los bloques se agrupan cantidades de 2×2, 4×4, 8×8 … pixeles, los cuales pueden ser fácilmente extraídos, dado que los pixeles en estos bloques son transmitidos uno a continuación del otro (el ordenamiento por filas no posee esta importante característica porque los pixeles son transmitidos serialmente fila tras fila). Esta característica puede ser aprovechada para procesamiento espacial de imágenes, tal como es el caso de una reducción en la resolución. En orden a reducir la resolución de la imagen por un factor de dos, hay que calcular la media de cuatro píxeles (un bloque de 2×2 pixeles). Con estos ordenamientos (Morton y barrido por filas), se puede realizar de una manera simple y sencilla, sin requerir múltiples elementos de almacenamiento. Estos cálculos pueden ser extensivos a bloques de dimensiones 4×4, 8×8, etc.

Finalmente, el tipo de barrido de bloques por zig-zag se empleará en el Apéndice B, en la forma de un barrido de pixeles por zig-zag, el cual es empleado en los formatos de compresión JPEG y JPEG2000.

C.3 Conclusiones del apéndice

En este apéndice se mostraron los tipos de exploración de bloques más frecuentemente usados en la práctica, así como los más recomendables para el Apéndice B.

Apéndice D

Revisión de Álgebra Lineal

D.1 Introducción

La matemática de las onditas depende en gran medida de las ideas fundamentales del álgebra lineal [369-392]. En este apéndice se examinan algunas ideas importantes.

D.2 Algebra Lineal

D.2.1 Espacio vectorial

El punto de partida para álgebra lineal es la noción de un *espacio vectorial*. Un espacio vectorial (sobre los reales) puede ser vagamente definido como una colección de elementos de V donde

1. Para todo $a, b \in \Re$ y para todo $u, v \in \boldsymbol{V}$, $au + bv \in \boldsymbol{V}$.

2. Existe un único elemento $\boldsymbol{0} \in \boldsymbol{V}$ tal que
   - para todo $u \in \boldsymbol{V}$, $0u = \boldsymbol{0}$, y
   - para todo $u \in \boldsymbol{V}$, $\boldsymbol{0} + u = u$.

3. Otros axiomas (omitidos aquí) mantienen esta verdad, la mayoría de las cuales son necesarias para garantizar que la multiplicación y la adición se comportan como se esperaba.

Los elementos de un espacio vectorial V se denominan vectores, y el elemento 0 es el llamado vector cero. Los vectores pueden ser vectores geométricos, o pueden ser funciones, como es el caso cuando se habla de onditas y el análisis de multiresolución.

D.2.2 Bases y dimensión

Se dice que una colección de vectores $u_1, u_2, \ldots$ en un espacio vectorial V son *linealmente independientes* si

$$\boldsymbol{c}_1\boldsymbol{u}_1 + \boldsymbol{c}_2\boldsymbol{u}_2 + \ldots = 0 \quad \text{si y solo si} \quad \boldsymbol{c}_1 = \boldsymbol{c}_2 = \ldots = 0 \tag{D.1}$$

Una colección $u_1, u_2, \ldots \in \boldsymbol{V}$ de vectores linealmente independientes constituye una *base* para V si cada $v \in \boldsymbol{V}$ puede ser expresado como

$$\boldsymbol{v} = \sum_i \boldsymbol{c}_i \boldsymbol{u}_i \tag{D.2}$$

para algunos números reales $c_1, c_2, \ldots$. Se dice que los vectores en una base para V son *generados* en V. Hablando intuitivamente, la independencia lineal significa que los vectores no son redundantes, y una base consiste en un conjunto de vectores mínimo y completo.

Si una base para V tiene un número finito de elementos $u_1, \ldots, u_m$, entonces V es *dimensional-finita* y su dimensión en m. De otra forma, se dice que V es *dimensional-infinito*.

Ejemplo: $\Re^3$ es un espacio tridimensional, y $e_1 = (1, 0, 0)$, $e_2 = (0, 1, 0)$, $e_3 = (0, 0, 1)$ es una base para el mismo.

Ejemplo: El conjunto de todas las funciones continuas sobre [0, 1] es un especio vectorial dimensional-infinito. Llamaremos a este especio $C[0, 1]$.

D.2.3 Productos internos y ortogonalidad

Cuando se trata de los vectores geométricos del espacio vectorial $\Re^3$, la operación"producto punto" posee un número de usos. La generalización del producto punto para espacios vectoriales arbitrarios es llamado *producto interno* [369-371]. Formalmente, un producto interno $\langle \bullet \mid \bullet \rangle$ sobre un espacio vectorial V es todo mapeo desde V x V a $\Re$ de la que es

1. simetría: $\langle \boldsymbol{u} \mid \boldsymbol{v} \rangle = \langle \boldsymbol{v} \mid \boldsymbol{u} \rangle$,

2. bilineal: $\langle a\boldsymbol{u} + b\boldsymbol{v} \mid \boldsymbol{w} \rangle = a\langle \boldsymbol{u} \mid \boldsymbol{w} \rangle + b\langle \boldsymbol{v} \mid \boldsymbol{w} \rangle$, y

3. definida positiva: $\langle \boldsymbol{u} \mid \boldsymbol{u} \rangle > 0$ para todo $\boldsymbol{u} \neq 0$.

A un espacio vectorial junto con un producto interno se lo conoce como *espacio de producto interno*.

Ejemplo: Es sencillo demostrar que el producto interno en puntos sobre $\Re^3$ está definido por

$$\langle (a_1, a_2, a_3) \mid (b_1, b_2, b_3) \rangle = a_1 b_1 + a_2 b_2 + a_3 b_3 \tag{D.3}$$

lo que satisface los requerimientos de producto interno.

Ejemplo: El siguiente producto interno "estándar" sobre $C[0, 1]$ juega un rol central en la mayoría de las formulaciones del análisis multirresolución:

$$\langle f \mid g \rangle = \int_0^1 f(x) g(x) dx \tag{D.4}$$

El producto interno estándar puede ser generalizado también para incluir una función de ponderación positiva $w(x)$:

$$\langle f \mid g \rangle = \int_0^1 w(x) f(x) g(x) dx \tag{D.5}$$

Uno de los más importantes usos del producto interno es el de formalizar la idea de ortogonalidad. Se dice que dos vectores u, v en un espacio de producto interno son *ortogonales* si $\langle \boldsymbol{u} \mid \boldsymbol{v} \rangle = 0$. No

es dificil demostrar que una colección u_1, u_2, … de vectores mutuamente ortogonales deben ser linealmente independientes, sugiriendo que la ortogonalidad es una forma fuerte de independencia lineal. Una *base ortogonal* es aquella que consiste de vectores mutuamente ortogonales.

D.2.4 Normas y normalización

Una *norma* es una función que mide la longitud de la vectores. En un espacio vectorial dimensional finita, típicamente usamos la norma $\|\boldsymbol{u}\| = \langle \boldsymbol{u} \mid \boldsymbol{u} \rangle^{1/2}$. Si estamos trabajando con una espacio funcional tal como $C[0, 1]$, nosotros comúnmente usamos una de las normas L^p, definida como

$$\|\boldsymbol{u}\|_p = \left(\int_0^1 |\boldsymbol{u}(\boldsymbol{x})|^p \, d\boldsymbol{x} \right)^{1/p} \tag{D.6}$$

En el límite, como p tiende a infinito, obtenemos aquello que es conocido como la *norma-máxima*:

$$\|\boldsymbol{u}\|_\infty = \max_{\boldsymbol{x} \in [0,1]} |\boldsymbol{u}(\boldsymbol{x})| \tag{7}$$

Aún más frecuentemente utilizada es la norma L^2, la cual también puede ser escrita como $\|\boldsymbol{u}\|_2 = \langle \boldsymbol{u} \mid \boldsymbol{u} \rangle^{1/2}$ si nosotros estamos usando el producto interior estándar.

Un vector u con $\|\boldsymbol{u}\| = 1$ se dice que está *normalizado*. Si tenemos una base ortogonal compuesta de vectores que están normalizados en la norma L^2, la base es llamada *ortonormal*. Dicho de forma concisa, una base u_1, u_2, … es ortonormal si

$$\langle \boldsymbol{u}_i \mid \boldsymbol{u}_j \rangle = \delta_{ij} \tag{D.8}$$

donde a δ_{ij} se lo conoce como delta de Kronecker y se define para ser 1 si $i = j$, y 0 en cualquier otro caso.

Ejemplo: Los vectores $e_1 = (1, 0, 0)$, $e_2 = (0, 1, 0)$, $e_3 = (0, 0, 1)$ forman una base ortonormal para el espacio de producto interno $\Re^3$ dotado con el *producto punto* de la Ecuación (3).

D.3 Conclusiones del apéndice

En este apéndice se han expuesto los conceptos mínimos imprescindibles de Álgebra Lineal a los efectos de constituir al presente trabajo en una unidad autosuficiente.

Apéndice E

Concepto de resolución de una imagen

E.1 Resolución de imagen

La **resolución de una imagen** indica cuánto detalle puede observarse en esta. El término es comúnmente utilizado en relación a imágenes de fotografía digital, pero también se utiliza para describir cuán nítida (como antónimo de granular) es una imagen de fotografía convencional (o fotografía química). Tener mayor resolución se traduce en obtener una imagen con más detalle o calidad visual.

Para las imágenes digitales almacenadas como mapa de bits, la convención es describir la resolución de la imagen con dos números enteros, donde el primero es la cantidad de columnas de píxeles (cuántos píxeles tiene la imagen a lo ancho) y el segundo es la cantidad de filas de píxeles (cuántos píxeles tiene la imagen a lo alto).

Es bueno señalar que si la imagen aparece como granular se le da el nombre de **pixelada** o **pixelosa**.

La convención que le sigue en popularidad es describir el número total de píxeles en la imagen (usualmente expresado como la cantidad de megapíxeles), que puede ser calculado multiplicando la cantidad de columnas de píxeles por la cantidad de filas de píxeles. A continuación se presenta una ilustración sobre cómo se vería la misma imagen en diferentes resoluciones.

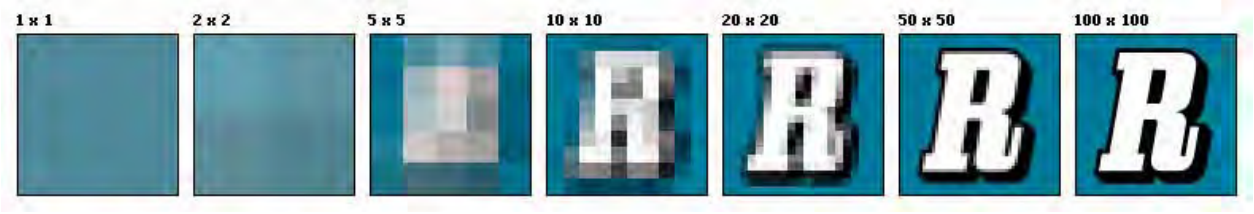


Figura E.1: Diferentes resoluciones de una misma imagen.

Para saber cuál es la resolución de una cámara digital debemos conocer los píxeles de ancho x alto a los que es capaz de obtener una imagen. Así una cámara capaz de obtener una imagen de 1600 x 1200 píxeles tiene una resolución de 1600x1200=1.920.000 píxeles, es decir 1,92 megapíxeles.

Además, hay que considerar la resolución de impresión, es decir, los puntos por pulgada (ppp) a los que se puede imprimir una imagen digital de calidad. A partir de 200 ppp podemos decir que la resolución de impresión es buena, y si queremos asegurarnos debemos alcanzar los 300 ppp

porque muchas veces la óptica de la cámara, la limpieza de la lente o el procesador de imágenes de la cámara digital disminuyen la calidad.

Para saber cuál es la resolución de impresión máxima que permite una imagen digital hay que dividir el ancho de esa imagen (por ejemplo, 1600 entre la resolución de impresión 200, 1600/200 = 8 pulgadas). Esto significa que la máxima longitud de foto que se puede obtener en papel para una foto digital de 1600 píxeles de largo es de 8 pulgadas de largo en calidad 200 ppp (1600/300=5.33 pulgadas en el caso de una resolución de 300 ppp). Una pulgada equivale a 2,54 centímetros.

E.2 La resolución, como cantidad de pixeles

La resolución de la imagen, es la cantidad de píxeles. La resolución se utiliza también para clasificar casi todos los dispositivos relacionados con las imagen digital ya sean pantallas de ordenador o televisión, impresoras, escáneres, cámaras, etc.

La resolución expresa el número de píxeles que forman una imagen de mapa de bits. La calidad de una imagen, también depende de la resolución que tenga el dispositivo que la capta. El número de píxeles que contenga una imagen dependen de cuántos píxeles utilice el sensor CCD de la cámara para captar la imagen.

E.3 Expresión de la resolución de una imagen

La resolución de una imagen digital se expresa multiplicando su anchura por la altura en pantalla. Por ejemplo la imagen de 1200 x 1200 píxeles = 1.440.000 píxeles, expresado en Mp megapixel es igual a 1,4 Mp. Conviene tener en cuenta que 1 Megapíxels = 1024 píxeles.

Figura E.2: Resolución de una imagen como un área pixelada.

E.4 Resolución: la medida de una imagen

Cuando nos decidimos a dar el paso de la película química al formato digital nos enfrentamos a una nueva manera de hacer fotografía. Quizás no en el momento de disparar, pero sí en todo lo que viene a continuación. La imagen digital es muy versátil y acabará dándonos muchas satisfacciones en cuanto asimilemos un par de conceptos que, en un principio, nos pueden liar. Uno de ellos es la resolución, un parámetro más simple de lo que el uso indiscriminado de esta palabra mágica hace parecer.

En un sentido amplio, resolución se refiere a la capacidad de una tecnología o un mecanismo para reflejar los detalles de una imagen. La forma de traducir una fotografía en bits para poder manejarla como archivo informático es dividirla según una malla de filas y columnas. A las unidades resultantes se les llama píxeles: son todos del mismo tamaño y representan áreas cuadradas de la imagen original. Si dividimos la imagen en pocos píxeles, podremos codificarla con poca información, pero seguramente perderemos mucho detalle, por lo que decimos que tiene poca resolución. Si la dividimos en muchas más partes, éstas llegarán a ser tan pequeñas que no las distinguiremos. La visión de la imagen será mucho mejor y más detallada, pero también mucho más costosa en bits. Un aspecto importante es que, salvo limitaciones en la tecnología que utilicemos, el tamaño y la frecuencia de los píxeles siempre son a voluntad nuestra. Los frecuentes equívocos en el uso de la palabra resolución se resuelven distinguiendo en la imagen tres tipos de tamaño: en píxeles, informático y superficial.

Figura E.3: Cada unidad debe ser una zona homogénea, para anotar sólo su color.

E.5 Tamaño en pixeles

Obviamente, indica en cuántas filas y columnas se ha dividido la imagen, o bien cuál es el número total de píxeles.

Por ejemplo, decimos que una foto tiene 1600 x 1200 píxeles. También podemos decir que tiene

1.920.000 píxeles, o redondear diciendo que es una foto de 2 megapíxeles. Se redondea tanto que no se tiene en cuenta que nos referimos a un sistema binario, en el que kilo no significa 1000, sino 1024 (la décima potencia de 2) y mega no significa 1.000.000, sino 1.048.576.

Figura E.4: De las dimensiones en píxeles depende el detalle de la imagen. Aquí vemos la misma foto dividida en 4 x 4, 12 x 12, 30 x 30 y 150 x 150 píxeles.

E.6 Tamaño informático

Se cuenta en unidades de información como bytes, kilobytes o megabytes, y depende directamente de dos cosas: del número de píxeles y de la cantidad de bytes que gastamos para definir cada píxel.

La profundidad de bits permite diferenciar y aplicar un número más o menos grande de colores. La mayoría de las cámaras digitales utilizan la profundidad de 24 bits del modo RGB, por lo que cada píxel se anota con 3 bytes. Se calcula rápidamente que cada megapíxel ocupará en memoria 3 megabytes (algo menos, porque la máquina no redondea como nosotros). En las tarjetas de memoria suele ocupar mucho menos, porque los datos se guardan comprimidos.

E.7 Tamaño superficial de salida

Es lo que ocupará la foto si la imprimimos. Los píxeles son realmente información numérica, así que este tamaño lo decidimos nosotros, indicando cuántos píxeles queremos imprimir en cada centímetro o pulgada de papel.

Todo sería mucho más simple si reservásemos el término "resolución" para expresar esta relación: número de píxeles por unidad de medida lineal.

Podemos cambiarla sin modificar en absoluto la información de imagen. Simplemente, indicando menos resolución (menos píxeles por pulgada) la foto se imprimirá más grande, e indicando más resolución se imprimirá en menos papel porque los píxeles serán más pequeños y concentrados.

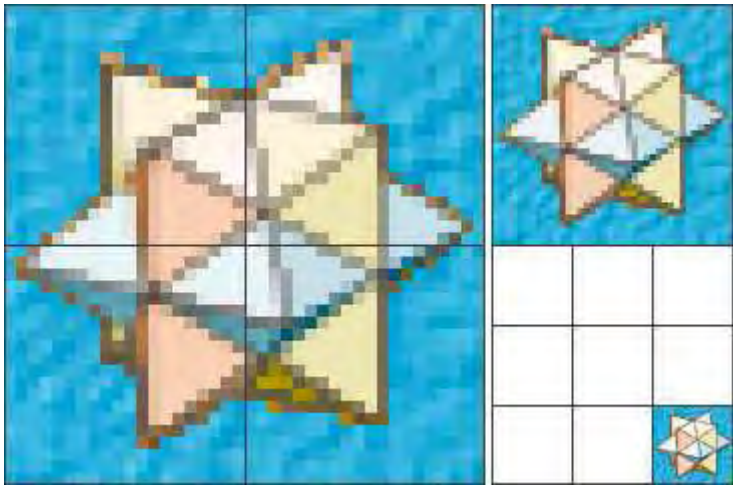


Figura E.5: La resolución es inversa al tamaño superficial. La misma imagen de 40 x 40 píxeles que ocupa 4 pulgadas cuadradas cuando se imprime a 20 ppp, a 40 ppp ocupará 1 pulgada cuadrada, y a 120 ppp cubrirá 1/9 parte de pulgada cuadrada.

La resolución así entendida la podríamos decidir en el momento de imprimir. Para la cámara, no obstante, es obligatorio que el número de píxeles por pulgada figure como dato al crear un formato de archivo como JPEG o TIFF. Se asigna una resolución por defecto, habitualmente 72, 180 ó 300 ppp. No tiene importancia, es un dato que podemos modificar sin estropear nada.

E.8 Controlar la resolución al imprimir

Comprender la resolución sirve para predecir el resultado en la impresión. En general, queremos evitar que los píxeles sean tan grandes que resulten evidentes. A una distancia normal de 40 ó 50 cm, y si la resolución es de 150 ppp o menor, distinguimos claramente la frontera entre un píxel y el siguiente. Aumentando la resolución, los píxeles serán más pequeños, pero seguiremos notando la estructura de filas y columnas hasta unos 180 ppp.

Por encima de esta resolución ya no notaremos escalones, aunque seguiremos percibiendo mejoras en la riqueza del color y en la suavidad de los degradados hasta unos 220 ppp. Por encima de este nivel es muy difícil estar seguro de notar ningún cambio, por lo que podemos considerarlo el umbral de seguridad para una impresión fotográfica.

Así pues, la referencia habitual de 300 ppp supone un amplio margen que podemos permitirnos cuando no hay problemas de espacio informático, pero siempre a sabiendas de que con 240 ppp

estamos en un nivel que no desmerece la impresión en papel fotográfico, y que en documentos con papel corriente se cumple dignamente incluso con resoluciones de 200 ppp.

E.9 Conclusiones del apéndice

En este apéndice se analizó de manera exhaustiva el concepto de resolución, y sus diferentes acepciones en procesamiento digital de imágenes.

Apéndice F

Televisión Digital y sus Técnicas de Compresión y Codificación para su Transmisión Terrestre

F.1 Introducción

En este apéndice se desarrollan los conceptos relativos a la TV Digital, su codificación y compresión para su proceso de transmisión por antena y la noción de exploración entrelazada y progresiva, así como toda otra noción en forma complementaria a las desarrolladas en el Capítulo 3. Esto quiere decir, que se analizarán las que en el mismo quedaran pendientes.

F.2 Prolegómenos

La televisión digital permite a los operadores la transmisión de contenidos de televisión con gran calidad de vídeo y audio, unida a una información adicional de datos que puede recibir el usuario final.

Un sistema de televisión digital se lo puede entender como un sistema típico cliente/servidor, en donde el servidor sería el ambiente de una radiodifusora (parte izquierda de la Fig.F.1) como un servidor de contenido, y el cliente o ambiente del usuario telespectador (parte derecha de la Fig. F.1)

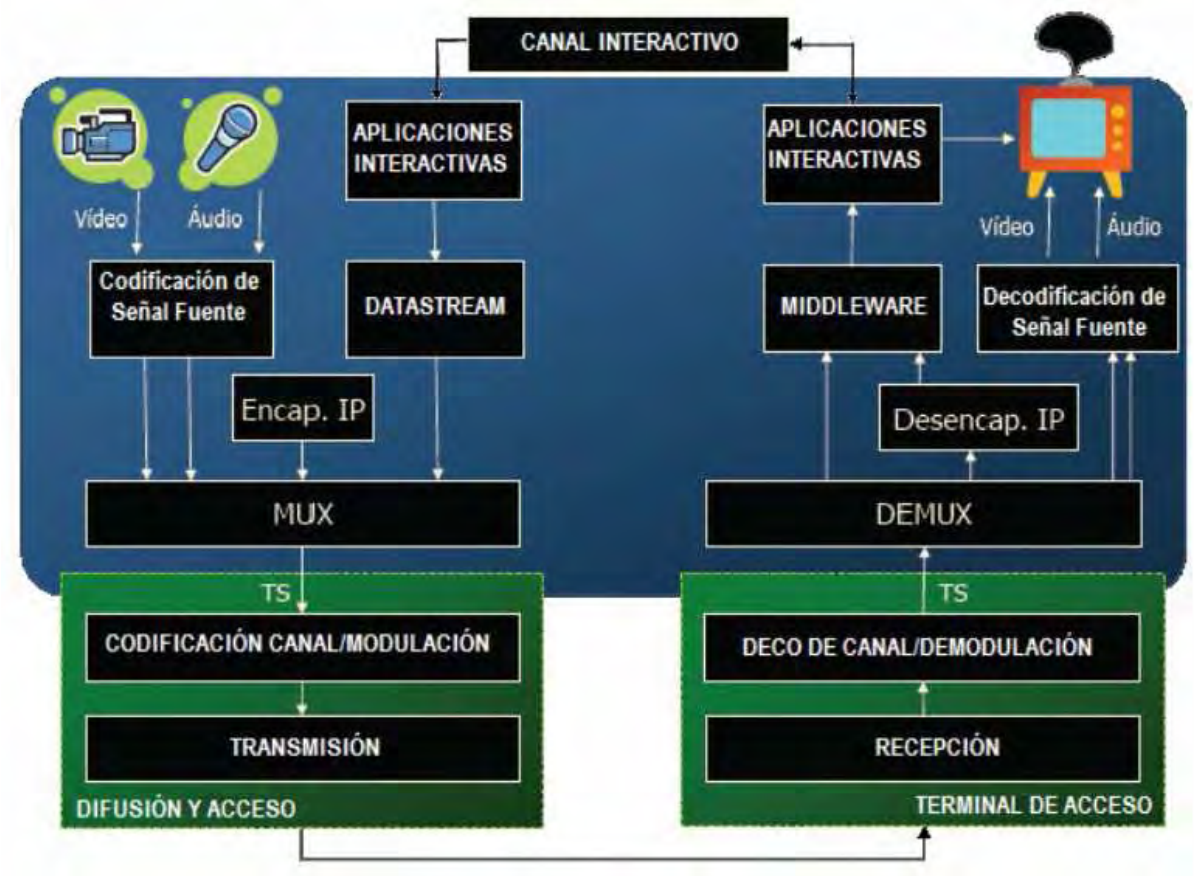


Figura F.1: Visión general de un sistema de TV Digital.

Un programa está compuesto de audio y video principal, capturado en vivo por una cámara o proveniente de un servidor de video, y de datos adicionales, los mismos que pueden estar encapsulados en formato IP (Internet Protocol) o en otro formato.

Del lado de recepción la señal que se recibe es demodulada y entregada a un demultiplexador, que separa los flujos de audio y video principal entregándoles a los decodificadores apropiados. Existen diferentes modelos de televisión digital, dependiendo del modo y el medio de transmisión por el que se propagan, entre los que se encuentran los siguientes:

F.2.1 Televisión Digital por Satélite

Como se puede observar en la Fig.F.2 en el esquema de televisión digital por satélite se realiza una transmisión satelital cuyas señales son propuestas directamente al público, en donde los usuarios reciben estas señales mediante el uso de decodificadores suministrados por quien entrega este servicio o dispositivos disponibles en el mercado.

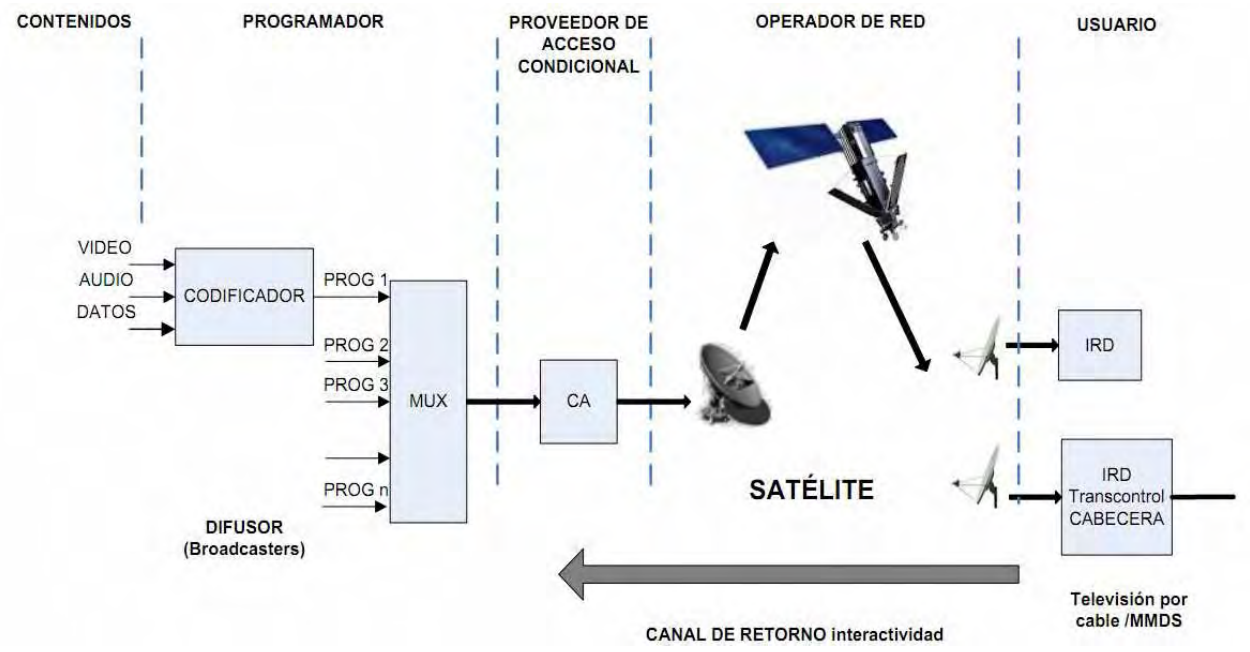


Figura F.2: Esquema de TV Digital por Satélite.

F.2.2 Televisión Digital por Cable

Bajo el sistema mostrado en la Fig.F.3 se puede observar cómo están agrupados los servicios de televisión por suscripción en los cuales se realiza una transmisión por un medio físico, ya sea por cable coaxial o fibra óptica cuyas señales son emitidas a una parte del público general (suscriptores).

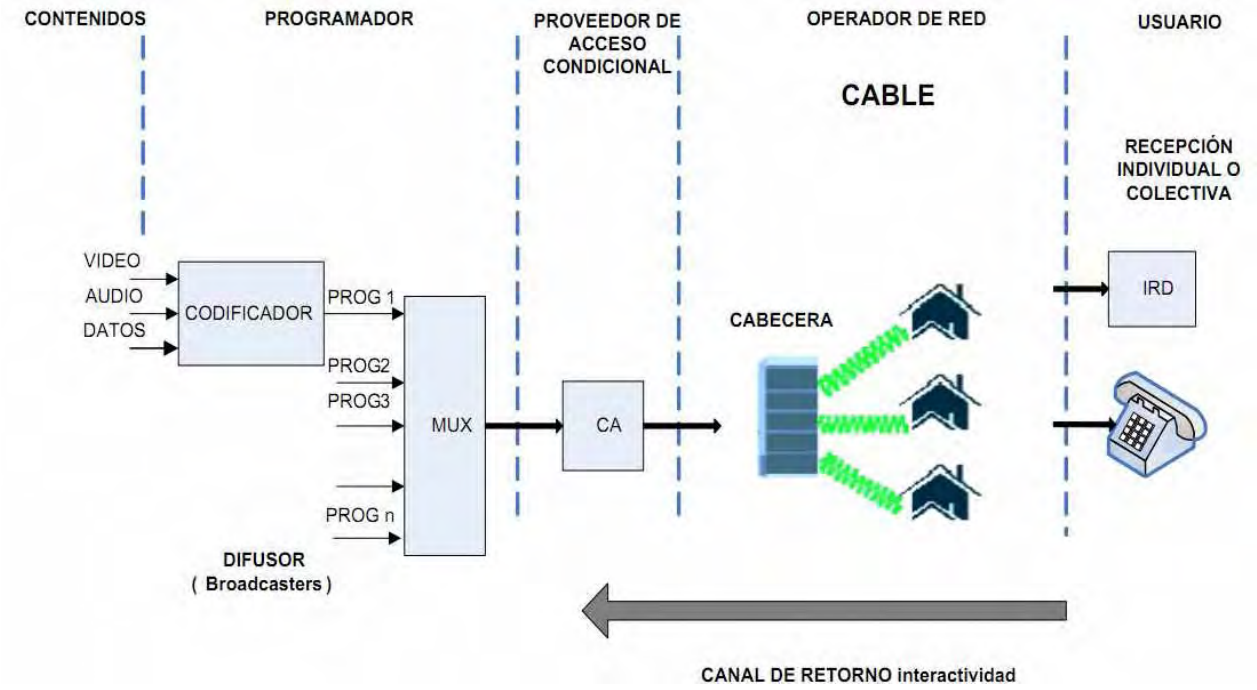


Figura F.3: Esquema de TV Digital por Cable.

F.2.3 Televisión Digital Terrestre (TDT)

La Fig.F.4 muestra un esquema general de cómo están agrupados los servicios de televisión terrestre abierta radiodifundida de operación pública y privada con cobertura nacional, regional y local.

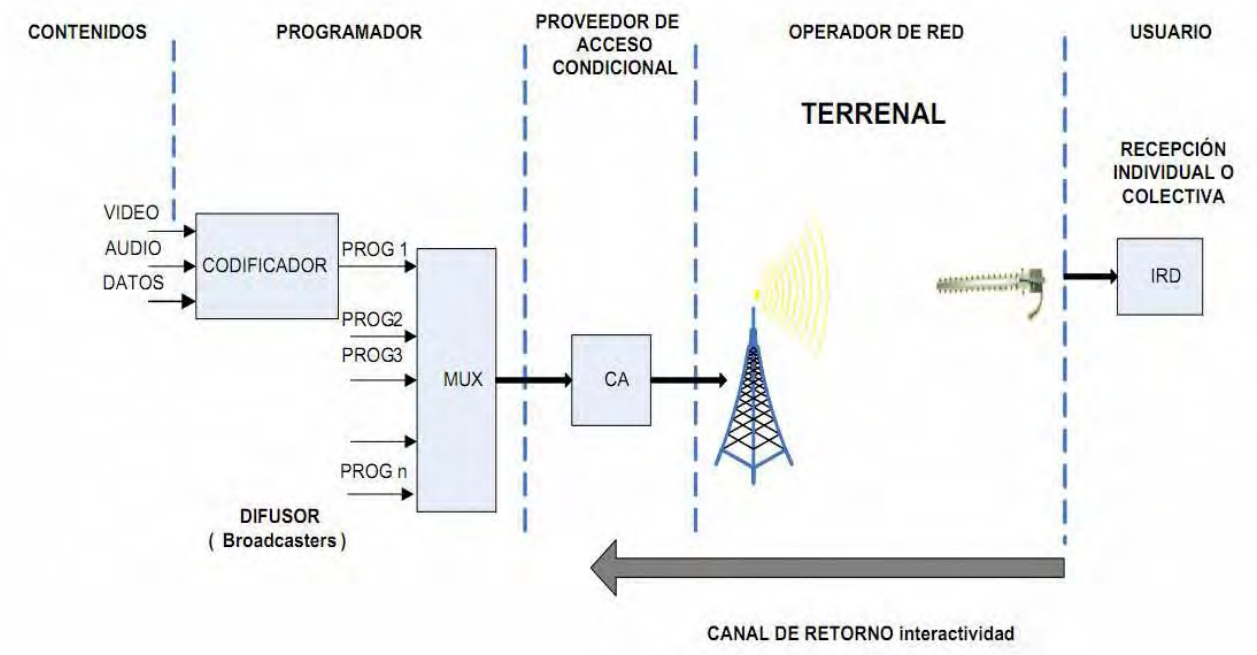


Figura F.4: Esquema de TV Digital Terrestre.

F.3 Técnicas de Compresión y Codificación para TDT

F.3.1 Compresión y Codificación de Video: Moving Picture Expert Group (MPEG)

La compresión de video implica una disminución de calidad, ya que generalmente existe una pérdida de información, pero gracias a los algoritmos de codificación se descarta la información redundante o que no puede ser apreciada por el ojo humano, por este motivo sabemos que la calidad del video es inversamente proporcional al nivel de compresión. La compresión es un arma de doble filo, ya que el video comprimido es más sensible a los errores, un error en video comprimido puede hacer ilegible a la imagen.

MPEG (Moving Picture Experts Group) es el grupo de trabajo del subcomité del ISO/IEC (International Organization for Standarization / International Electrotechnical Commission) encargado del desarrollo de estándares para la compresión, descompresión, procesado y codificación de imágenes animadas, audio o la combinación de ambas.

MPEG define la sintaxis de las señales digitales correspondientes a audio y vídeo, describe su estructura, contenido y regula el funcionamiento de decodificadores estandarizados. El MPEG no define los algoritmos de codificación. Lo que facilita una mejora de los codificadores y su adaptación a aplicaciones específicas dentro de la norma. "Además de la codificación de audio y vídeo, el MPEG también define sistemas para multiplexar la información de audio y vídeo en una única señal digital, describe los métodos para verificar que las señales y los decodificadores se ajustan a la norma y publica informes técnicos con ejemplos de funcionamiento de codificadores y decodificadores" (Norma ISO de codificación de contenidos audiovisuales).

F.3.1.1 Tipos de Imágenes MPEG

MPEG define tres tipos de imágenes como se observa en la siguiente gráfica, las mismas que son el soporte de la codificación diferencial y bidireccional:

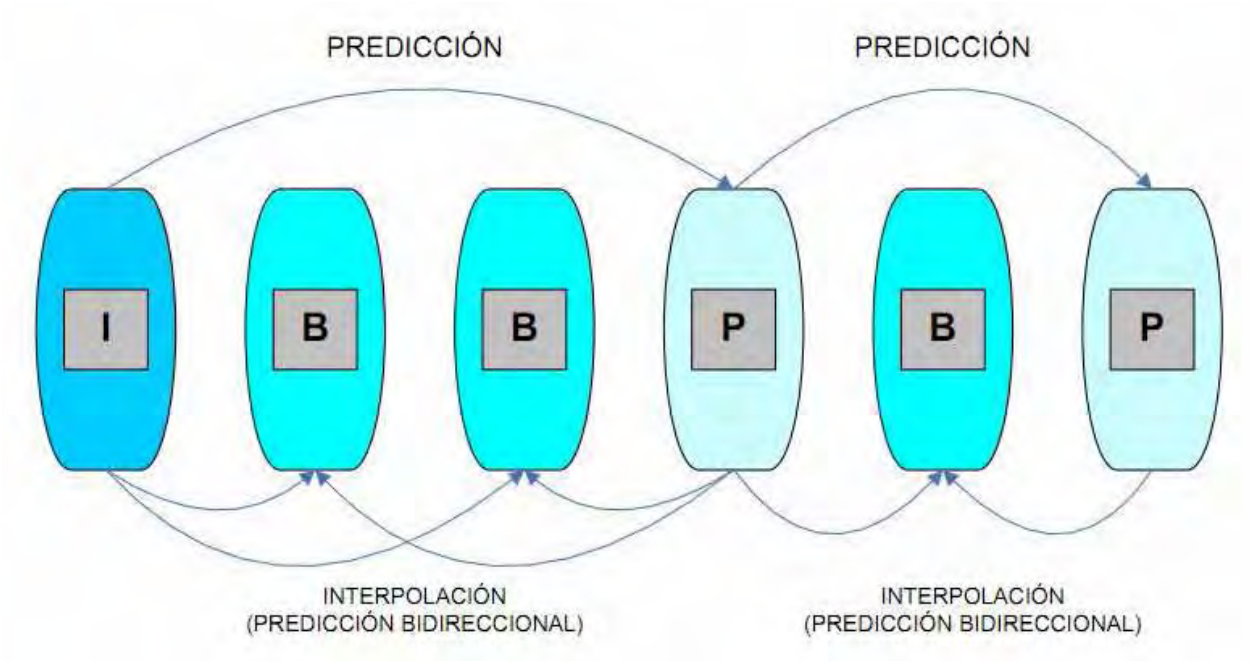


Figura F.5: Tipos de imágenes MPEG (Codificación Bidireccional).

F.3.1.1.1 Imagen I (Intra)

Imagen que no necesita información adicional para su codificación. Son codificadas sin referencia a otras imágenes, contiene todos los elementos básicos para su reconstrucción por el decodificador (inicia el lazo de predicción) y son por ello, el punto de entrada obligatorio para el acceso a una secuencia, las imágenes I se codifican como una única imagen utilizando solo la información de la imagen, en términos de datos transmitidos es la imagen más grandF. Además son utilizadas para facilitar la captura del canal cuando se apaga el decodificador o se cambia el canal.

F.3.1.1.2 Imagen P (Previstas)

Son imágenes de predicción y se codifican con respecto a las imágenes I o P anteriores, gracias a las técnicas de predicción con compensación de movimiento. Su tasa de compresión es mayor que la de las imágenes I, las imágenes P requieren aproximadamente la mitad de los datos de las imágenes I.

F.3.1.1.3 Imagen B (Bidireccionales)

Son imágenes de predicción bidireccional. Para su formación, toman información tanto de una imagen futura (P) como de una imagen previa (I), se codifican por interpolación. Este tipo de imagen es el que ofrece el factor de compresión más alto, que generalmente es de una cuarta parte de los datos de las imágenes I, como no se utiliza para describir otras imágenes, las imágenes B no propagan los posibles errores de codificación.

Dependiendo de la complejidad del codificador utilizado, se podrán codificar las imágenes I, las imágenes I y P o las imágenes I, P y B, cada una de estas codificaciones obtendrá resultados diferentes a nivel del factor de compresión y en cuanto a la posibilidad de acceso aleatorio, así como del tiempo de codificación y de la calidad.

Los parámetros M y N definen la manera en que las imágenes I, P y B se encadenan:
- M es la distancia en número de imágenes entre dos imágenes P sucesivas.
- N es la distancia entre dos imágenes I sucesivas.

Para alcanzar una velocidad de transmisión o flujo de video de 1.15 Mbps con calidad satisfactoria, al tiempo que se mantiene una resolución de acceso aleatorio aceptable (< 0.5 s), se utiliza regularmente los parámetros M=3 y N=12 como se muestra en la Figura 6

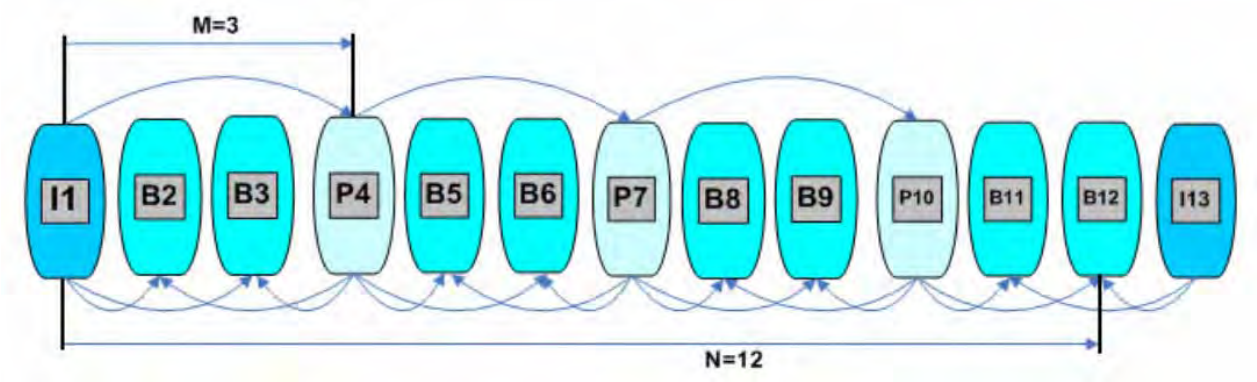


Figura F.6: Encadenamiento de las imágenes I, P y B.

En este caso, una secuencia de video (secuencia de imágenes) se compone de 1/12 de imágenes I (8.33%), 1/4 de imágenes P (25%) y de 2/3 de imágenes B (66.66%). El factor de compresión total se beneficia por el hecho de que las imágenes B que son las imágenes de mayor factor de compresión son las imágenes más frecuentes.

F.3.1.2 Estándar MPEG-1

"MPEG-1 se considera como un vídeo solamente progresivo (no entrelazado), que alcanza una tasa de 1.5 Mbps. La entrada de vídeo es usualmente convertida primero al formato estándar de entrada MPEG SIF (Standard Input Format). El espacio de color adoptado es Y- Cr- Cb. En el MPEG-1 SIF el canal de luminancia es de **352 pixeles x 240 líneas y 30 cuadros/segundo**" (Norma ISO de codificación de contenidos audiovisuales).

El algoritmo para la compresión de video en MPEG utiliza dos técnicas fundamentales: Reducción de la Redundancia Temporal y Reducción de la Redundancia Espacial. La señal restante es codificada utilizando las técnicas basadas en transformaciones. En la Fig.F.7 se muestran las etapas y técnicas utilizadas por MPEG para la compresión de video.

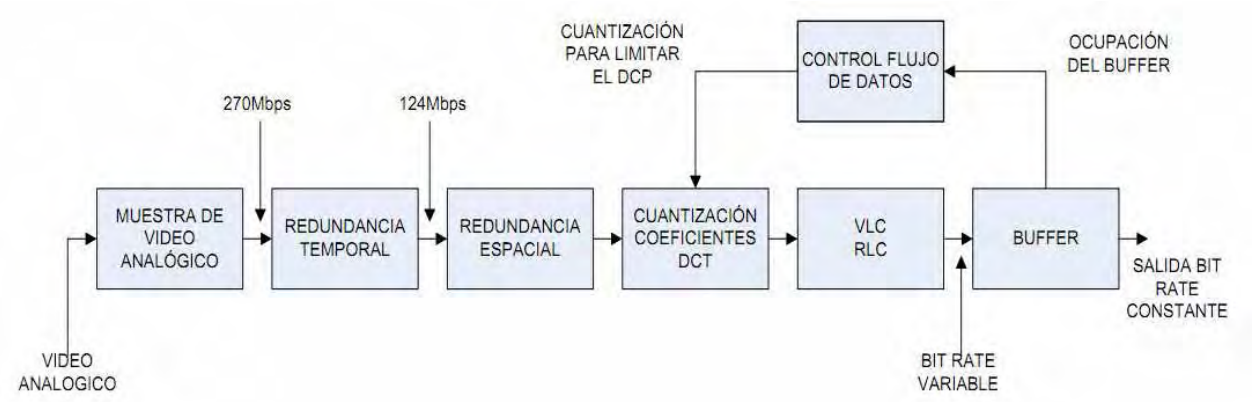


Figura F.7: Técnicas usadas por MPEG para la compresión de video.

F.3.1.2.1 Reducción de la Redundancia Temporal

También conocida como técnica de Predicción con Compensación de Movimiento, consiste en la deducción de la mayoría de imágenes de una secuencia de video, comparando un determinado cuadro con su antecesor adicionando un mínimo de información, se aplica en ambas direcciones hacia adelante o causal (forward) y hacia atrás o no causal (backward), para soportar el acceso aleatorio al video almacenado se defines tres tipos de imágenes o cuadros: codificados internamente (I), predictivos (P) e interpolados bidireccionalmente (B).

La redundancia temporal consiste en el aprovechamiento de las similitudes existentes entre cuadros sucesivos que forman una imagen dinámica con el objetivo de reducir considerablemente la cantidad de información necesaria para su transmisión o almacenamiento, este proceso se lo puede distinguir claramente en la Fig.F.8.

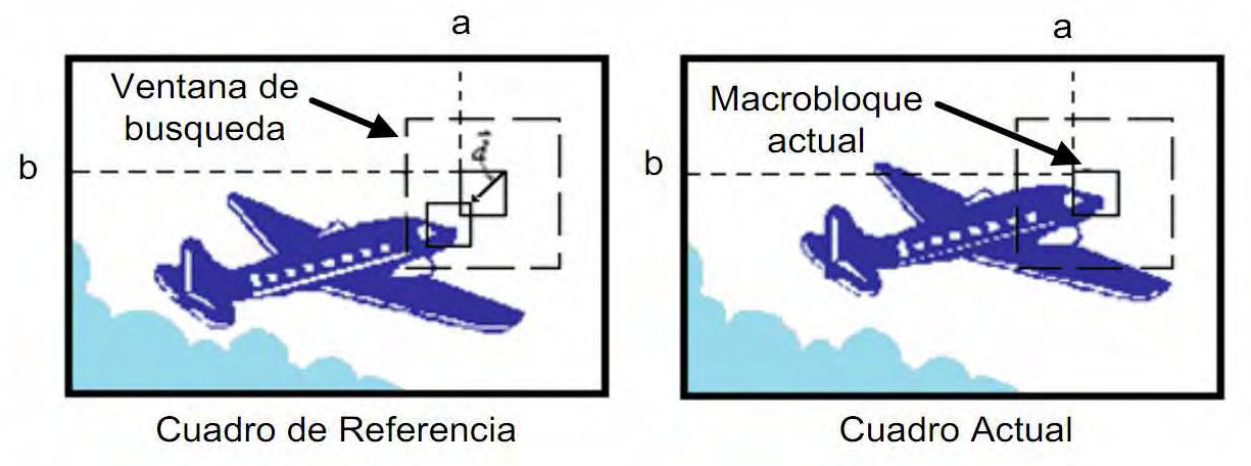


Figura F.8: Redundancia temporal.

F.3.1.2.2 Reducción de la Redundancia Espacial

Consiste básicamente en la semejanza de dos pixeles adyacentes de una misma imagen, como se muestra la Fig.F.9 se puede ver que la variación entre los pixeles es muy poca.

Figura F.9: Redundancia espacial.

En cada frame (imagen) I o en la predicción de errores en frames P o B, MPEG utiliza técnicas de codificación basadas en DCT (Discrete Cosine Transform).

"La DCT incluye la Transformada Rápida de Fourier FFT (Fast Fourier Transformate) y su operación básica es transformar una serie de puntos del dominio espacial a una representación idéntica en el dominio de la frecuencia. La DCT se aplica sobre una matriz generalmente de 8x8 cuya salida es otra matriz de iguales dimensiones que contiene los coeficientes DCT que están ordenados de forma que los que contienen información útil están en la esquina superior izquierda. El coeficiente DC es la posición (0,0) y representa la media de los 63 valores" (Norma ISO de codificación de contenidos audiovisuales).

F.3.1.2.3 Codificación

La codificación de las imágenes en MPEG se realiza a través de los siguientes pasos:

1. Codificación del elemento DC con un valor relativo respecto a valores anteriores, ya que este valor tiene altos grados de correlación

2. Reordenación de los valores DCT en zig-zag ya que hay tantos de estos elementos cuyo valor es cero que deben ser codificados de forma diferente que los que no son cero. Se usa RLE (run-length encoding) que cuenta el número de ceros en la imagen.

F.3.1.2.4 Run Lenght Code (RLC) y Variable Lenght Code (VLC)

Run Lenght Code: Su funcionamiento es sencillo, en lugar de transmitir una palabra entera ejemplo ARAQUARA se transmite el número de letras que conforman la palabra: 5A, 3R, 1Q y 1U. Cuanto mayor número de elementos repetidos mayor eficiencia de codificación.

Variable Lenght Code: También denominado algoritmo de Huffman, cuyo principio básico es atribuir símbolos de menor compresión a las informaciones más frecuentes, ejemplo ARAQUARA, Símbolo "0" letra A, Símbolo "01" letra R, Símbolo "111" letra Q y Símbolo "0000" letra U.

F.3.1.2.5 Secuencia de Video

Es la combinación de un código inicial, un encabezamiento y un código final, la secuencia de soporte especifica el tamaño horizontal y vertical de la imagen, la tasa de imágenes, norma de barrido, si se usa barrido progresivo o entrelazado, velocidad de transferencia de bits, el perfil, el nivel, y cuales matrices de cuantificación se utilizan para codificar imágenes espaciales y temporales.

La secuencia de soporte de datos permite al decodificador entender el flujo de bits y así empezar la tarea de decodificación correcta, esto sucede generalmente cuando un televidente cambia de canal de un lugar a otro.

F.3.1.2.6 Grupo de imágenes

El GOP (Group of Pictures) es la unidad fundamental de codificación temporal, un GOP está conformado por imágenes I, B y P que se encadena según se muestra en la Fig.5. La utilización de estos tres tipos de imágenes, aporta alta compresión, buen acceso aleatorio, y capacidades de adelanto y retroceso rápidamente, son el soporte de la codificación diferencial y bidireccional, reduciendo así la transmisión de errores.

Desafortunadamente si se utiliza una secuencia ilimitada de imágenes previstas (P) existe la posibilidad de transmitir errores, por lo cual se utiliza una cantidad limitada de estas imágenes y así garantizar una mejor transmisión, por lo que cada determinado tiempo se envía una imagen la cual no ha sido tratada y que es idéntica a la imagen original, de esta forma se refresca y actualiza los datos en la secuencia de transmisión.

Una de las estructuras de GOP más usuales y eficaces es de 12 cuadros de largo, como se muestra en la Fig.F.6 y se forma de la siguiente manera: IBBPBBPBBPBB.

El mismo que funciona de la siguiente forma, el codificador envía la información de la diferencia existente entre la imagen previa y la actual, el codificador necesita de una imagen previamente almacenada para ser comparada entre imágenes sucesivas y de la misma forma el decodificador se basa en la imagen almacenada para generar las imágenes siguientes.

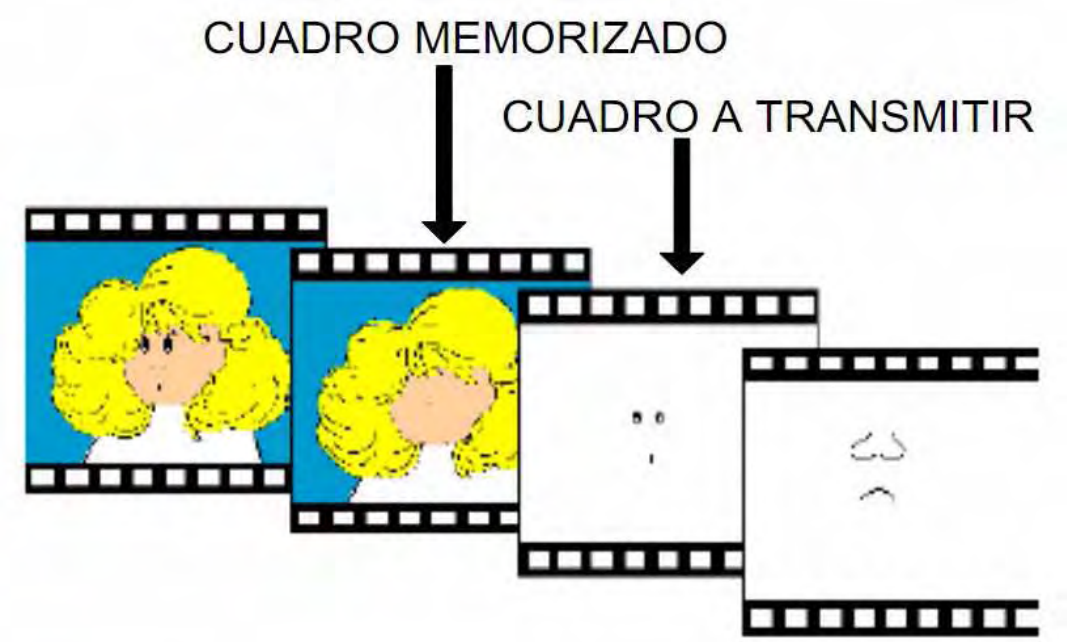


Figura F.10: Compresión de cuadros.

F.3.1.2.7 Bloque

Es la ***unidad fundamental de la información de la imagen***, se representa por una matriz de coeficientes DCT de 8x8 pixeles, los mismos que representan datos Y, Cr o Cb.

F.3.1.2.8 Macrobloque

"Es la unidad fundamental de la imagen que además está compensada en movimiento. Cada macrobloque es un vector de desplazamiento en dos dimensiones situado en la parte superior de la secuencia. En una imagen B, el vector puede ser hacia adelante o hacia atrás" (Norma ISO de codificación de contenidos audiovisuales).

En un formato de codificación 4:2:0, cada macrobloque tendrá 4 bloques Y, y dos bloques de color diferente, para poder identificar cada bloque y sus componentes, estos se envían en un orden especifico. Cada macrobloque tiene un tamaño de 16x16 pixeles.

F.3.1.2.9 Franja (Slice)

Es un conjunto de macrobloques que siempre deben representar una fila horizontal la cual se ordena de izquierda a derecha.

Las Franjas son la ***unidad fundamental de sincronización*** para la codificación de la longitud variable y diferencial, los vectores iniciales en una Franja son enviados completamente, mientras que los demás vectores son transmitidos diferencialmente.

F.3.1.2.10 Imágenes de Tipo I, P o B

Es la parte activa de un cuadro y está formada por un número de Franjas, la imagen de soporte inicial define que imágenes I, P o B codifica e incluye una referencia temporal para que la imagen pueda ser representada en el momento apropiado.

F.3.1.3 Estándar MPEG-2

MPEG-2 posee una calidad superior a MPEG-1 por lo que inicialmente se convirtió en el estándar de facto en el mundo de la televisión digital ya que mejora muchos de los problemas propios de MPEG-1, como es la escalabilidad, resolución y manejo de vídeo entrelazado. MPEG-2 permite imágenes de mejor calidad (hasta niveles de HDTV) y permite que muchos canales de diferentes velocidades de transmisión se multiplexen dentro de un mismo flujo de datos.

En MPEG-2 se definen dos sistemas de capas, el flujo de programa y el flujo de transporte. En donde se usa un solo flujo pero no los dos a la vez. El flujo de programa funcionalmente es similar al sistema MPEG-1.

El encapsulamiento y multiplexación de la capa de compresión produce paquetes grandes y de varios tamaños.

"Los paquetes grandes producen errores aislados e incrementan los requerimientos de buffering en el receptor/decodificador para demultiplexar los flujos de bits. En contraposición el flujo de transporte consiste en ***paquetes fijos de 188 bytes*** lo que decrementa el nivel de errores ocultos y los requerimientos del buffering receptor" (Norma ISO de codificación de contenidos audio-visuales).

El proceso de codificación de las imágenes animadas es similar al descrito para MPEG-1, especialmente desde el bloque hasta la secuencia.

F.3.1.3.1 Codificación MPEG-2

Al igual que MPEG-1, MPEG-2 no define explícitamente el método de codificación, sino únicamente la sintaxis que controla el tren binario a la salida del codificador.

Con la imagen digitalizada en formato 4:2:0, el codificador elige para cada imagen su tipo (I, P o B) y si esta imagen debe ser codificada en modo imagen o campo, el siguiente paso del codificador es estimar los vectores de movimiento para cada macrobloque de 16x16 pixeles. El número de vectores depende del tipo de imagen y modo de codificación escogido.

La unidad básica de codificación es el macrobloque, 4:2:0 (4 bloques de luminancia 8x8 pixeles y 2 bloques de crominancia Cr y Cb 8x8 pixeles) que abarcan la misma zona de la imagen. Todos los macrobloques se codifican de arriba abajo y de izquierda a derecha, eligiéndose un modo de codificación independiente para cada uno.

"En una imagen con estructura frame, el codificador deberá elegir entre realizar la DTC en modo frame o field, esto depende de la amplitud del movimiento entre los campos de la imagen" (Norma ISO de codificación de contenidos audiovisuales).

F.3.1.3.2 Decodificación MPEG-2

En la decodificación no se realiza ninguna estimación de movimiento, "la memoria intermedia (buffer) de entrada recibe los datos del canal de transmisión y el decodificador lee el tren binario hasta encontrar el principio de una imagen, su tipo (I, P o B) y su estructura (frame o field).

Empieza la decodificación con la primera imagen I, almacenándola en su memoria, así como la imagen P siguiente, para servir de referencia a las imágenes P o B que dependen de ella" (Norma ISO de codificación de contenidos audiovisuales).

Cuando todos los macrobloques han sido tratados se efectúa la reconstrucción de la imagen.

La necesidad de memoria para el codificador es de tres imágenes (dos imágenes de referencia y una imagen en vía de reconstrucción), siendo para una imagen 4:2:0 de aproximadamente 16 Mbps.

F.3.1.3.3 Niveles y Perfiles de MPEG-2

Un **perfil** es básicamente el grado de complejidad esperada en la codificación, mientras que un nivel describe el tamaño de la imagen, la resolución o la velocidad de transferencia de bit usada en ese perfil.

• **The SP@ML profile (Simple Profile @ Main Level)** El perfil simple no soporta una codificación bidireccional, por lo tanto solo genera imágenes I y P. Esto disminuye la tasa de compresión simplificando de esta manera tanto el codificador como el decodificador, el perfil simple tiene una tasa máxima de 15 Mbps.

• **The MP@ML profile (Main Profile @ Main Level)** Es el formato digital más comparable al NTSC 4:2:0 y su velocidad de transmisión de datos es de 15 Mbps, el video codifica con una proporción de 4 elementos de los datos para la luminancia a 2 para la crominancia (4 muestras Y por cada muestra Cb y Cr), esta proporción de la codificación iguala la característica de percepción visual, y un desempeño óptimo para la transmisión a bajas velocidades, el perfil principal soporta imágenes I, P y B, por lo que tanto el codificador como el decodificador son más complejos que los del perfil simple.

Tabla F.1: Perfiles y niveles de MPEG-2

		PERFILES					
		SIMPLE	PRINCIPAL	4.2:2	SNR	ESPACIAL	ALTO
NIVELES	ALTO		4:2:0 1920 X 1152 80 Mbps				4:2:0 o 4:2.2 1920 X 1152 100 Mbps
	ALTO 1440		4:2:0 1440 X 1152 60 Mbps			4:2:0 1440 X 1152 60 Mbps	4:2:0 o 4:2.2 1440 X 1152 80 Mbps
	PRINCIPAL	4:2:0 720 X 576 15 Mbps Sin B	4:2:0 720 X 576 15 Mbps	4:2:2 720 X 608 50 Mbps	4:2:0 720 X 576 15 Mbps		4:2:0 o 4:2.2 720 X 576 20 Mbps
	BAJO		4:2:0 352 X 288 4 Mbps		4:2:0 352 X 288 4 Mbps		

• **The MP@HL profile (Main Profile @ High Level)** Es utilizado para HDTV 4:2:2. Su máxima velocidad de transferencia de datos alcanza los 80 Mbps, por lo que no puede ser usado en su plenitud en el ancho de banda de 6 MHz, donde aproximadamente se hace a 19.4 Mbps, el video se codifica con una proporción de 4 elementos de los datos para la luminancia a 4 para la crominancia (4 muestras Y por cada 2 muestras Cb y 2 Cr), la actuación es ligeramente mejor que el perfil 4:2:0, solo cuando la velocidad de transmisión es mayor de 10 Mbps.

Los diferentes perfiles y niveles bajo la norma de compresión MPEG-2, se muestran en la siguiente Tabla F.1.

F.3.1.4 Estándar MPEG-4

MPEG-4 se creó para mejorar la calidad del video codificado a bajas velocidades a través de nuevas técnicas de comprensión, las cuales estuvieron orientadas inicialmente a las videoconferencias e internet. Las emisoras de TDT se benefician por la alta tasa de compresión de audio (AAC, parte 3 del estándar) y video (H.264 parte 10 del estándar), las nuevas características de este estándar son:

- Las escenas se descomponen en dos componentes básicas: audio y video los cuales son codificados de forma independiente.
- Los objetos pueden ser tanto video natural como imágenes sintéticas.
- Ofrece soporte para manipulación de las imágenes sintéticas.
- Permite interacción de los usuarios sobre la escena que se está renderizando.
- Tiene un mejor algoritmo que incrementa la robustez para el trato de errores.

MPEG-4 posee muchas características de sus antecesores MPEG-1 y MPEG-2, añadiendo otras tales como:

VRML (Virtual Reality Modeling Language) que trabaja con objetos 3-D, a demás de soportar varios tipos de interactividad, la principal diferencia de MPEG-4 con referencia a otros estándares audiovisuales, es que representa un modelo audio visual basado en objetos.

F.3.1.4.1 MPEG-4 AVC (H.264)

Es también conocido como MPEG-4 parte 10 o MPEG-4 AVC (Advanced Video Coding), busca ofrecer una alta tasa de compresión sin pérdida de la calidad de video, obteniendo entre un 40 a 70% más compresión comparado con MPEG-2.

F.3.1.4.2 Estructura de Capas MPEG-4

La Fig.F.11 muestra las capas del sistema H.264, primero se tiene un codificador de video (**VCL**-Video Coding Layer), que reduce la cantidad de información a ser transmitida, una capa red (NAL – Network Abstraction Layer) es responsable por formatear el stream a la salida de VLC, segmentando e incrementando la información de cabecera, esta capa también configura el pro-tocolo para el transporte de información.

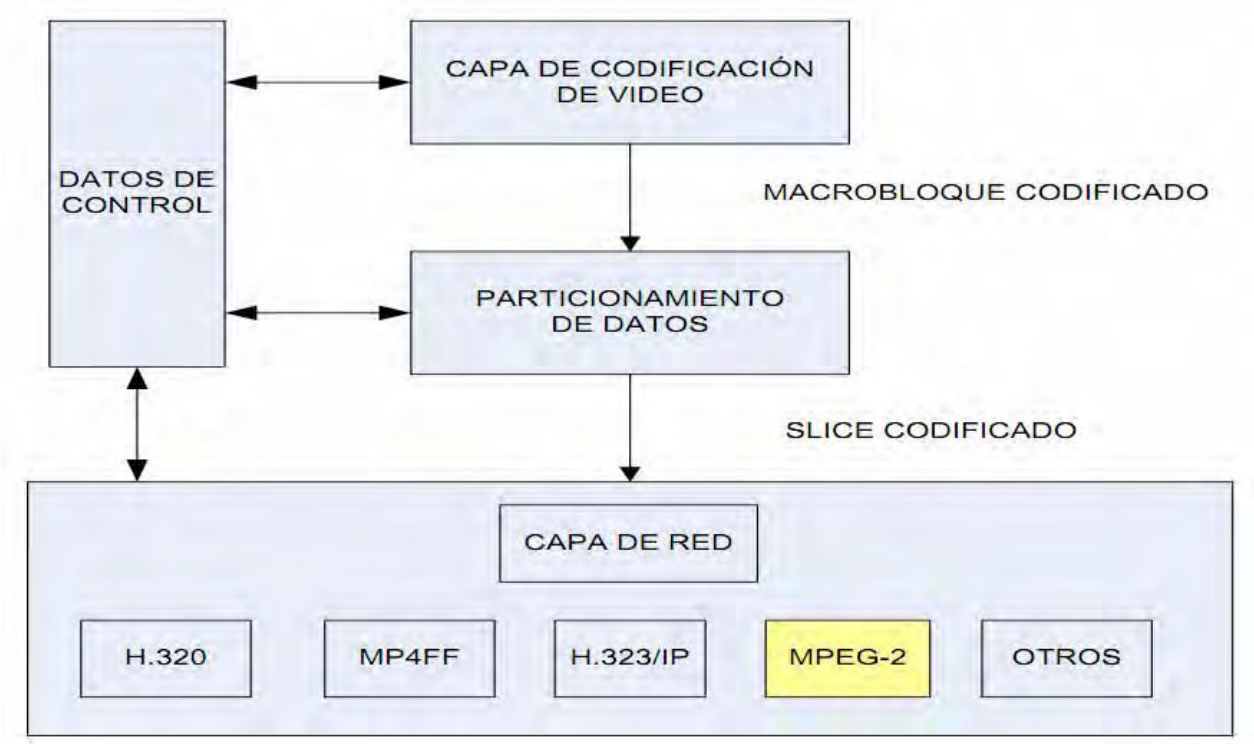


Figura F.11: Estructura de capas del estándar H.264.

F.3.1.4.3 Tipos de Imagen

Existen cinco tipos básicos de imágenes en MPEG-4, a más de las imágenes descritas anteriormente en el literal 1.2.1.1 I, P y B, se tienen las imágenes SP y SI.

F.3.1.4.3.1 Imagen SP (Switching P)

Una imagen tipo SP se codifica de forma que permite la transición entre dos streams que representan una misma secuencia, con calidad diferente, o también permite avanzar o retroceder una imagen de un mismo stream, transmitiendo así menos información que una imagen I.

F.3.1.4.3.2 Imagen SI (Switching I)

Una imagen tipo SI representa un punto de sincronismo para una transición entre dos streams, diferenciándose de las imágenes tipo SP por presentar todos los macrobloques codificados utilizando predicciones intra.

F.3.1.4.4 Tipos de Predicción

En MPEG-4 se utiliza los mismos tipos de predicción descritos en los puntos F.3.1.2.2 Redundancia Espacial (predicción espacial) y F.3.1.2.1 Redundancia Temporal (predicción temporal), en H.264 se soporta segmentaciones de los macrobloques en dimensiones de 4x4 pixeles, presentando una estimación y compresión de movimiento con una resolución de 1/4 para luminancia y 1/8 de pixel para crominancia.

F.3.1.4.5 Niveles y Perfiles MPEG-4

MPEG-4 H.264 tiene una flexibilidad de aplicaciones debido a una jerarquía de niveles y perfiles que define un stream de bits codificados, los perfiles soportados por H.264 son los siguientes.

Tabla F.2: Perfiles y Niveles de MPEG-4 H.264

Nivel #	Max macroblocks por segundo	Max frame size macroblocks	Max video bit rate VCL for Baseline, Extended and Main Profile	Max video bit rate VCL for High Profile	Max video bit rate VCL for High 10 Profile	Max video bit rate VCL for High 4.2:2 and High 4:4:4 Profile	Examples for high resolution @ frame rate (max stored frames) in level
1	1485	99	64 Kbps	80 Kbps	192 Kbps	256 Kbps	128x96@30.9 (8) 176x144@15.0 (4)
1b	1485	99	128 Kbps	160 Kbps	384 Kbps	512 Kbps	128x96@30.9 (8) 176x144@15.0 (4)
1.1	3000	396	192 Kbps	240 Kbps	576 Kbps	760 Kbps	176x144@30.3 (8) 320x240@10.0 (3) 352x288@7.5 (2)
1.2	6000	396	384 Kbps	480 Kbps	1152 Kbps	1536 Kbps	320x240@20.0 (7) 352x288@15.2 (6)
1.3	11880	396	768 Kbps	960 Kbps	2304 Kbps	3072 Kbps	320x240@36.0 (7) 352x288@30.0 (6)
2	11880	396	2 Mbps	2.5 Mbps	6 Mbps	8 Mbps	320x240@36.0 (7) 352x288@30.0 (6)
2.1	19800	792	4 Mbps	5 Mbps	12 Mbps	16 Mbps	352x480@30.0 (7) 352x576@25.0 (6)
2.2	20250	1620	4 Mbps	5 Mbps	12 Mbps	16 Mbps	352x480@30.7 (10) 352x576@25.6 (7) 720x480@15.0 (6) 720x576@12.5 (5)
3	40500	1620	10 Mbps	12.5 Mbps	30 Mbps	40 Mbps	352x480@61.4 (12) 352x576@51.1 (10) 720x480@30.0 (6) 720x576@25.0 (5)
3.1	108000	3600	14 Mbps	17.5 Mbps	42 Mbps	56 Mbps	720x480@80.0 (13) 720x576@66.7 (11) 1280x720@30.0 (5)
3.2	216000	5120	20 Mbps	25 Mbps	60 Mbps	80 Mbps	1280x720@60.0 (5) 1280x1024@42.2 (4)
4	245760	8192	20 Mbps	25 Mbps	60 Mbps	80 Mbps	1280x720@68.3 (9) 1920x1088@30.1 (4) 2048x1024@30.0 (4)
4.1	245760	8192	50 Mbps	62.5 Mbps	150 Mbps	200 Mbps	1280x720@68.3 (9) 1920x1088@30.1 (4) 2048x1024@30.0 (4)
4.2	522240	8704	50 Mbps	62.5 Mbps	150 Mbps	200 Mbps	1920x1088@64.0 (4) 2048x1088@60.0 (4)
5	589824	22080	135 Mbps	168.75 Mbps	405 Mbps	540 Mbps	1920x1088@72.3 (13) 2048x1024@72.0 (13) 2048x1088@67.8 (12) 2560x1920@30.7 (5) 3680x1536@26.7 (5)
5.1	983040	36864	240 Mbps	300 Mbps	720 Mbps	960 Mbps	1920x1088@120.5 (16) 4096x2048@30.0 (5) 4096x2304@26.7 (5)

• **Base Line**: Soporta imágenes I y P, código de compresión variable basado en (CAVLC Context-Adaptive Variable-Lenght Coding: Basado en códigos de Huffman) y orden flexible de macrobloques (FMO), es utilizado en aplicaciones conversacionales, como video-conferencias y video en estaciones móviles (celulares).

• **Main**: Soporta imágenes I, P y B, CAVLC, código aritmético binario adaptivo (CABAC) y codificación de video entrelazado usando codificación cuadro/campo adaptiva por imagen (PAFF) o codificación cuadro/campo adaptiva por macrobloque (MBAFF), y utilizado en aplicaciones de radiodifusión de televisión digital.

• **Extended**: Soporta herramientas del perfil Base Line, imágenes B, codificación de video entrelazado (PAFF o MBAFF), imágenes SI e imágenes SP, se utiliza en streamings de video.

• **High**: Soporta herramientas del perfil Main, formato 4:2:0 con 8 bits por muestra, utiliza transformadas 8x8 o 4x4, matrices de escalamiento para cuantización, presenta variaciones: High 10, High 4.2:2, High 4:4:4, es utilizado en aplicaciones de radiodifusión de televisión digital.

En la Tabla F.2 se presentan los niveles y perfiles soportados por el estándar MPEG-4 (H.264).

F.4 Exploración

La recolección de datos procedentes de la cámara consta de dos exploraciones, la vertical de arriba hacia abajo y la exploración horizontal de izquierda a derecha, la cual se subdivide a su vez en dos tipos, ver Fig.F.12:

• Exploración horizontal progresiva o ***p***, todas las líneas de pixeles se recolectan en una sola pasada.

• Exploración horizontal entrelazada o ***i***, donde primero se recolectan todas las líneas impares de pixeles y luego (en otra pasada) todas las pares las pares.

En una exploración 1080p se envían 2 cuadros idénticos de 1920x1080 pixeles c/u, mientras que en 1080i solo se envían 2 cuadros de 1920x540 pixeles c/u, como resultado de la exploración primero impar y luego par, es decir, la exploración progresiva maneja el doble de volumen de información que la entrelazada.

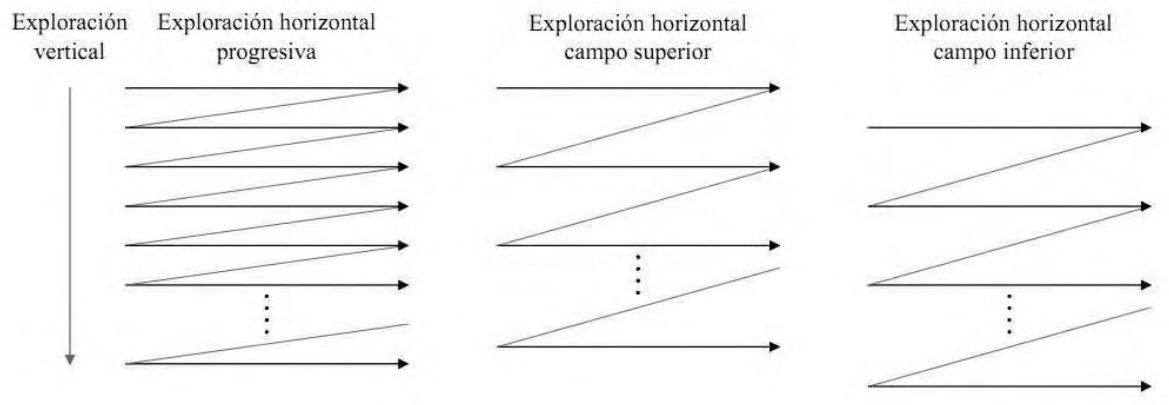


Fig.F.12: Exhibición progresiva y entrelazada (campos superior e inferior).

F.5 Conclusiones del apéndice

En este apéndice se analizaron todos los elementos complementariamente y no desarrollados en el Capítulo 3 en función de los esquemas de compresión/descompresión y codificación, además de todas las formas posibles de transmisión de TV Digital por aire, cable y satélite, así como las tan mencionadas exploraciones progresivas y entrelazadas.

Apéndice G

Filtrado anti-aliasing

G.1 Introducción

En este apéndice se desarrollan los conceptos relativos a las técnicas de filtrado anti-aliasing mencionadas en el Capítulo 3 como parte del esquema del codificador. Existen dos grandes clasificaciones de las técnicas anti-aliasing: Por la ubicación del algoritmo y por el tipo de filtro pasa-bajos

G.2 Por la ubicación del algoritmo

Los métodos de suavizado de bordes y líneas (anti-aliasing) se han desarrollado para combatir los efectos de aliasing. Hay tres clases principales de algoritmos de anti-aliasing.

- Cuando el problema de aliasing se debe a la baja resolución, una solución fácil es aumentar la resolución, haciendo que los puntos de muestreo tenga una mayor frecuencia frecuencia. Esto aumenta el costo computacional y de almacenamiento de las imágenes.

- La imagen se crea en alta resolución y luego se filtra digitalmente. Este método se llama super-muestreo o post-filtrado y elimina las altas frecuencias que son la fuente de alias.

- La imagen se puede calcular teniendo en cuenta las intensidades de más de una región en particular. Esto se conoce como pre-filtrado.

G.2.1 Pre-filtrado

En estos métodos de tratamiento de pre-filtrado se trata a un píxel como un área, y se calcula el color del píxel sobre la base de la superposición de los objetos de la escena sobre el área del píxel. Estas técnicas calculan los tonos de gris sobre la base del tamaño del área del píxel cubierto por el objeto, ver Fig.G.1.

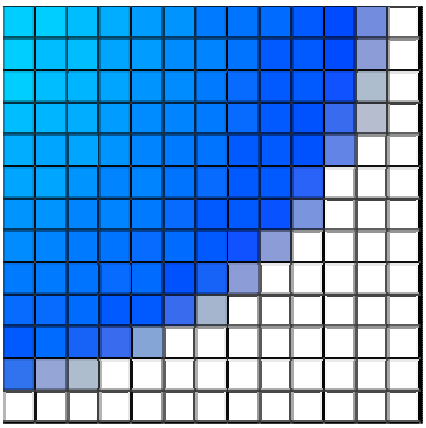


Figura G.1: Área del pixel.

Por ejemplo, en una modificación al algoritmo de Bresenham [2] desarrollado por Pitteway y Watkinson [2], a cada píxel se le da una intensidad dependiendo de la zona de superposición de los píxeles y la línea. Por lo tanto, debido al efecto de desenfoque a lo largo de los bordes de la mencionada línea, el efecto de suavizado de líneas no es muy importante, a pesar de que todavía existe.

El pre-filtrado equivale, pues, a muestrear la forma del objeto muy densamente en una región de píxeles. Para formas no poligonales, esto puede ser muy computacionalmente muy costoso.

G.2.2 Imagen original vs pre-filtrada

Por ejemplo, en la Fig.G.2 sin anti-aliasing, los dientes de sierra son claramente evidentes. En cambio, en la Fig.G.3 a lo largo de la frontera de la ***W***, los colores son una mezcla de los colores de la misma letra y los del fondo.

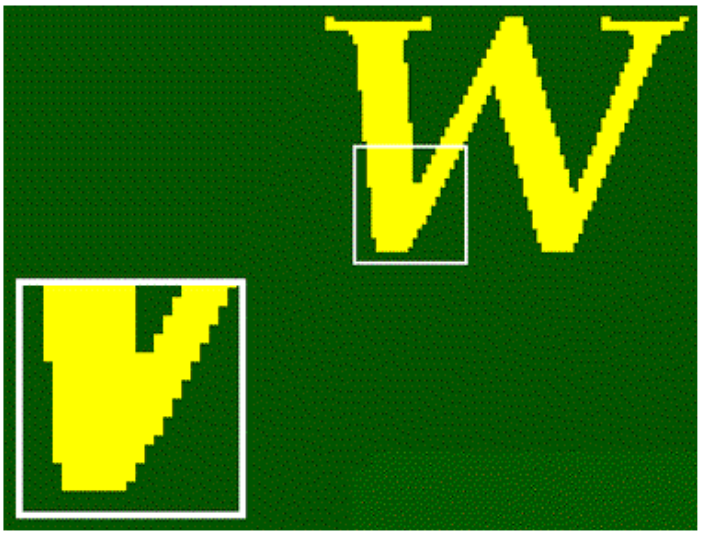


Figura G.2: Sin anti-aliasing.

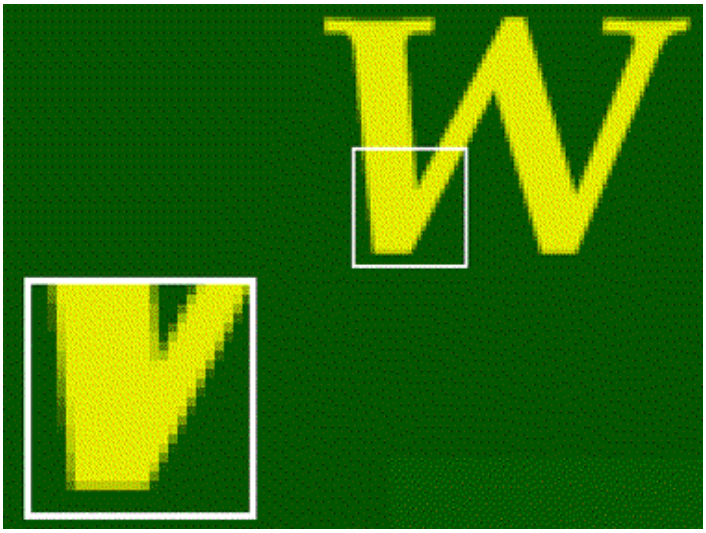


Figura G.3: Pre-filtrada.

G.2.3 Pre-filtrado o super-muestreo

En cambio, el post-filtrado o super-muestreo es el proceso por el cual los efectos aliasing en los gráficos se reducen al aumentar la frecuencia de la grilla de muestreo y luego promediando los resultados bajan. Este proceso implica el cálculo de una imagen virtual con una resolución espacial más alta que la resolución del cuadro almacenado y de este modo el promedio desciende a la resolución final. Se llama post-filtrado pues el filtrado se lleva a cabo después del muestreo.

Hay dos inconvenientes con este método

- El inconveniente es que hay un límite técnico y económico para aumentar la resolución de la imagen virtual.

- Dado que la frecuencia de las imágenes se puede extender hasta el infinito, sólo se reduce el aliasing al aumentar el efecto del desplazamiento-límite de Nyquist en el espectro de frecuencias.

El super-muestreo es básicamente un proceso de tres etapas.

- Una imagen continua $\boldsymbol{I}(x,y)$ se muestra a $\boldsymbol{n}$ veces la resolución final. La imagen se calcula en $\boldsymbol{n}$ veces la resolución del cuadro. Esta es una imagen virtual.

- La imagen virtual es entonces filtrada con un filtro paso-bajos

- La imagen filtrada se vuelve a muestrear en la resolución del cuadro final.

G.2.4 Algoritmos para super-muestreo

- Para generar la imagen original, tenemos que considerar una región en la imagen virtual. El alcance de esta región determina las regiones involucradas en la operación de filtrado paso-bajo. Este proceso se denomina convolución.

- Después de obtener la imagen virtual, la cual se encuentra en una resolución más alta, los píxeles de la imagen final se encuentran sobre los super-pixeles en la imagen virtual. Para calcular el valor de la imagen final en (S_i, S_j), se coloca el filtro sobre la super-imagen y se calcula la suma de los pesos del filtro y los píxeles circundantes. Un píxel adyacente de la imagen final es calculado al mover el filtro de super-pixeles $\boldsymbol{S}$ a la derecha. Así, el tamaño de paso es el mismo que el factor de escala entre la imagen real y virtual.

- Los filtros combinan las muestras para calcular el color de un píxel. El filtro ponderado toma una muestra de una combinación de nueve pixeles considerando el pixel del centro y los ocho que lo rodean. Cada muestra se multiplica por su peso correspondiente y los productos se suman para producir un promedio ponderado, que se utiliza como el color del píxel. En este filtro, la muestra del centro tiene la mayor influencia. El otro tipo de filtro es el filtro no ponderado. En un filtro no ponderado, cada muestra tiene la misma influencia en la determinación del color del píxel. En otras palabras, un filtro no ponderado calcula un promedio no ponderado.

- La extensión espacial del filtro determina la frecuencia de corte. Cuanto mayor sea el

filtro, menor es la frecuencia de corte y más borrosa es la imagen.

Las opciones disponibles en super-muestreo son:

- El valor del factor de escala ***S*** entre la imagen virtual y las reales.
- La elección de las extensiones y los pesos del filtro

En lo que respecta al primer factor se refiere, cuanto más alto sea el valor, mejor será el resultado obtenido. El compromiso que se asumió es el alto costo de almacenamiento.

La principal desventaja consiste en que no se trata de una técnica sensible al contexto y con ello da lugar a una gran cantidad de cálculos inútiles.

G.3 Por el tipo de filtro pasa-bajo

Los filtros de suavizado de la imagen en el dominio de la frecuencia son:

- Filtro pasa-bajo ideal: filtrado brusco
- Filtro pasa-bajo Gaussiano: filtrado suave
- Filtro pasa-bajo de Butterworth: intermedio

En televisión digital el filtro anti-aliasing más usado es el filtro Gaussiano, por lo tanto, es el único que desarrollaremos aquí, junto con sus propiedades.

Esta clase de filtros simulan una distribución Gaussiana bivariante. El valor máximo aparece en el pixel central y disminuye hacia los extremos tanto más rápido cuanto menor sea el parámetro de desviación típica ***s***. El resultado será un conjunto de valores entre 0 y 1. Para transformar la matriz a una matriz de números enteros se divide toda la matriz por el menor de los valores obtenidos. La ecuación para calcularla es:

$$g(x,y) = e^{-\frac{x^2+y^2}{2s^2}}$$

(G.1)

$$G(x,y) = \frac{g(x,y)}{\min_{x,y}(g(x,y))}$$

(G.2)

Son una manera de obtener filtros de tipo genérico. Pueden ser útiles, por ejemplo, cuando se asume que la respuesta espectral de un pixel es función de la reflectividad de los pixeles vecinos atenuada en función de la distancia. El alcance de esta atenuación (***r***) viene marcado por el tamaño de la ventana de filtrado (***w*** = 2***r*** +1) que debe especificarse previamente. Por esto, es necesario seleccionar el tamaño de una máscara adecuada que se aplica a la imagen y calcular el valor del elemento (pixel) central de la máscara como la suma de los productos de entre los valores contenidos en la vecindad de la máscara y los de la imagen original [2].

Por ejemplo, la máscara para un filtro Gaussiano *g(x,y)* asociado a una matriz de 5x5 obtenida mediante la Ecuación G.1 para ***s*** = 1, es:

Tabla G.1: Máscara *g(x,y)* del filtro Gaussiano con ***s*** = 1 y ***r*** = 2.

1	4	7	4	1
4	20	33	20	4
7	33	55	33	7
4	20	33	20	4
1	4	7	4	1

Mientras que su correspondiente *G(x,y)* resulta:

Tabla G.2: Máscara *G(x,y)* del filtro Gaussiano con ***s*** = 1 y ***r*** = 2.

0.02	0.08	0.14	0.08	0.02
0.08	0.37	0.61	0.37	0.08
0.14	0.61	1.0	0.61	0.14
0.08	0.37	0.61	0.37	0.08
0.02	0.08	0.14	0.08	0.02

Esta misma máscara sirve no solo como filtro anti-aliasing sino como filtro para ruido y para detección de contornos [2-4].

La principal ventaja de este tipo de filtro en términos de la complejidad computacional empleada y su uso en placas del tipo GPGPU es su separabilidad. De la Ecuación E.1 tenemos:

$$g(x,y) = e^{-\frac{x^2}{2s^2}}\ e^{-\frac{y^2}{2s^2}}$$

(G.3)

Lo que para una máscara de 3x3 se expresa como indica la Fig.G.4.

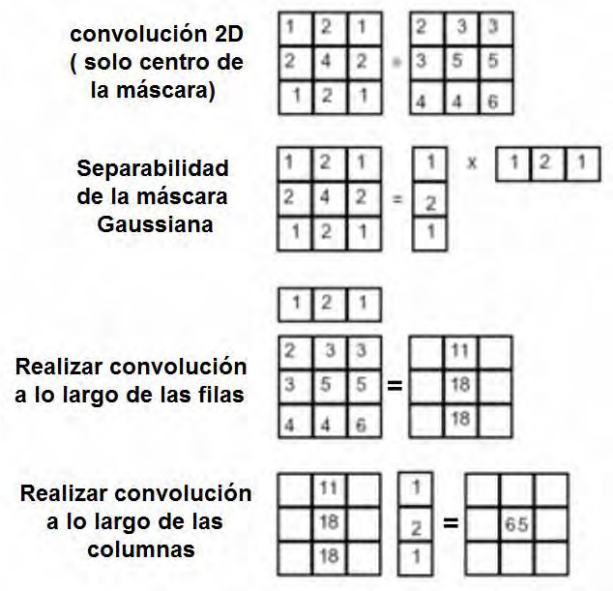


Fig.G.4: Ejemplo de separabilidad para una máscara de 3x3.

G.4 Conclusiones del apéndice

En este apéndice se analizaron las principales técnicas de filtrado anti-aliasing, con especial énfasis en el filtrado por máscara Gaussiana, el cual resulta ser en la práctica el más empleado en TV Digital.

Apéndice H

Nuevos encoders y normas para TDT y Argentina Conectada

H.1 Descripción Operativa de los Algoritmos de Supercompresión Desarrollados: Concepto de Supercompresión

Llamamos Supercompresión al procedimiento mediante el cual se aplica un catalizador algorítmico antes del encoder del algoritmo de compresión elegido, y luego del decoder de dicho algoritmo, a los efectos de aumentar en forma dramática la tasa de compresión de este, sin introducir pérdidas adicionales, ni latencia. El precio que se paga es un aumento de la complejidad computacional. No obstante, los catalizadores desarrollados basados fundamentalmente en dos técnicas: wavelets y compressed sensing, son altamente atomizables (paralelizables o distribuídos), por lo cual, su implementación se realiza sobre placas del tipo General Purpose Graphics Processing Units (GPGPU), que hacen a un formidable desempeño transparente al usuario final.

Estos catalizadores, no introducen pérdidas (es decir, son del tipo lossless), por lo cual, si el algoritmo de compresión elegido también lo es, se alcanzará una mayor tasa de compresión, haciendo trabajar al conjunto al límite de los principios de Nyquist y Shannon. En otras palabras, el catalizador no comprime, sino que convierte al grupo de frames entrantes en uno nuevo, con un mayor compromiso morfológico inter-cuadro respecto del original. Esto mejora notablemente el trabajo inter-cuadro del códec empleado (el cual, para ISDB-Tb se trata del AVC/MPEG4-10/H.264).

Hemos desarrollado esta tecnología con objeto de ser empleada en dos pruebas a saber:

1. Codificación y transmisión de 1080p-3D en 6 segmentos ISDB-Tb. Por primera vez en la historia se podrán enviar señales de semejante tasa de bits, 1080p-3D en el equivalente de bit-rate de 1080i o 720p, es decir, 6 segmentos ISDB-Tb por la modalidad terrestrial. Esta prueba se realiza con catalizador y encoder en las instalaciones de Canal 7 y el descatalizador y decoder se ubica en el mismo canal con objeto de evaluar:
   a) Calidad visual de la recuperación,
   b) Reducción del bit-rate, la cual es de 4:1 adicionales al ya generado por H.264 y se condice con una reducción equivalente del ancho de banda,
   c) Latencia adicional, y
   d) No incidencia del catalizador/descatalizador sobre el middleware de interactividad (Ginga)
   Dicha prueba se detalla más adelante.

2. Con objeto de la realización de la Copa América en nuestro país a inaugurarse el 1 de Julio del presente año, se transmitirán imágenes captadas en estudio con una cámara 4Kp-3D (3840x2160p stereo) como mínimo a 120 FPS y máximo 600 FPS, generadas en las instalaciones de Canal 7 y transmitidas por fibra óptica al Palacio del Bicentenario (ex Palacio de Correos) donde estará el descatalizador y un TV 4Kp-3D. Esta prueba permitirá probar compresor y catalizador, ambos lossless sobre una red óptica transmitiendo imágenes de TV Digital/Cine Digital de ultra-alta resolución como preámbulo de los posibles servicios que brindará la futura red óptica "Argentina Conectada".

Para ambos tipos de servicios, el descatalizador será encapsulado en un simple ARM (chip, circuito integrado) diseñado en el país, permitiendo incorporarlo en futuros set-top-boxes, conexiones punto a punto, enlaces satelitales, etc.

H.2 PRUEBA #1: "Codificación y transmisión de 1080p 3D en 6 segmentos ISDB-Tb"

Transmisión de una señal estereoscópica (3D) de televisión digital en alta definición de escaneo progresivo desde las instalaciones de la TV Pública a través del Canal 23, a ser recibidas y exhibidas en el mismo lugar.

La señal será generada mediante la televisación de escenas en vivo a través de 2 cámaras 1080p en sincro conectadas a un "sistema de catalización algorítmica" de manera previa a su ingreso al encoder regular del Canal de marca NEC, ver Fig.H.1. Del lado de la recepción, obrará una antena y un Set-Top-Box regulares, mediando un "sistema descatalizador" entre la salida de este y la entrada HDMI del TV. La visualización se producirá a través de anteojos de sincronización, ver Fig.H.2.

Figura H.1: Diagrama de captura, procesamiento y transmisión.

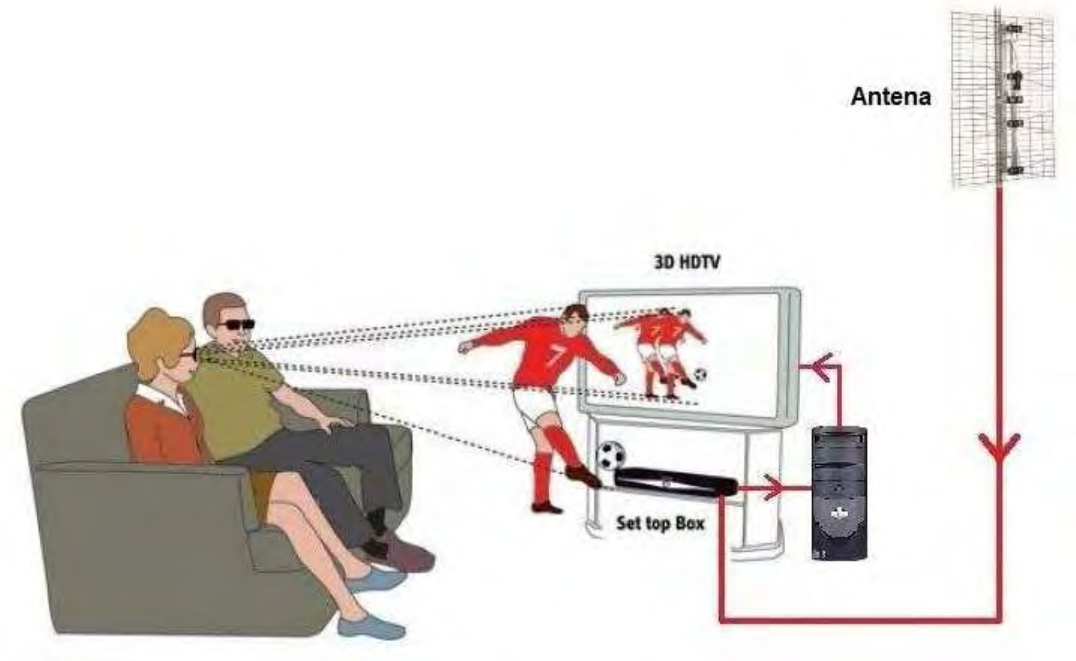


Figura H.2: Recepción en el hogar. En el futuro cercano, la CPU del hogar es reemplazada por un chip en el set-top-box.

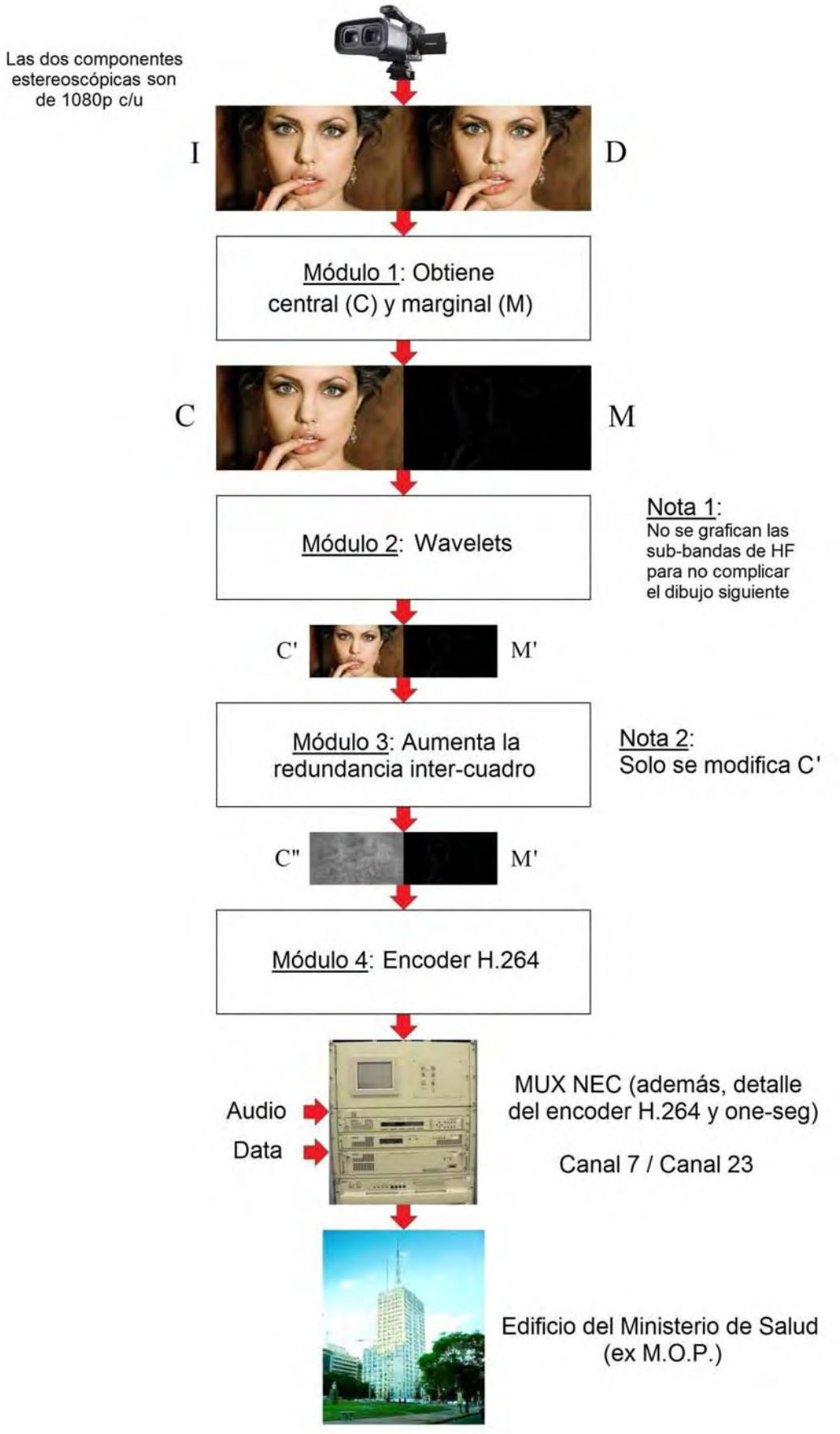


Figura H.3: Detalle de la Fig.H.1.

La Fig.H.3 muestra el detalle del procedimiento empleado en la Fig.H.1, donde cada cámara genera imágenes 1080p, las cuales luego de pasar por el Módulo 1, son convertidas en dos nuevos cuadros, uno central (como si la estereoscopía no existiera, y que hace al sistema compatible para todos aquellos que no tengan TV 3D) y una marginal (necesario para recuperar la estereoscopía). El Módulo 2 consiste en la aplicación de una base de wavelets, de la cual solo se grafica la sub-banda de baja frecuencia, las tres componentes de alta frecuencia no se grafican para no complicar el esquema. El Módulo 3 aumenta la redundancia inter-cuadro solo en la componente central (C), mientras la marginal queda intacta. Finalmente, el Módulo 4 encodea en H.264 para luego en el MUX incorporar audio y datos, para finalmente ser transmitidos los tres componentes. Un aspecto importante se grafica en la Fig.4, donde se puede ver en el esquema completo de encoder y decoder que la super-compresión deja intacto al datastream, las aplicaciones interactivas y su eventual canal de retorno.

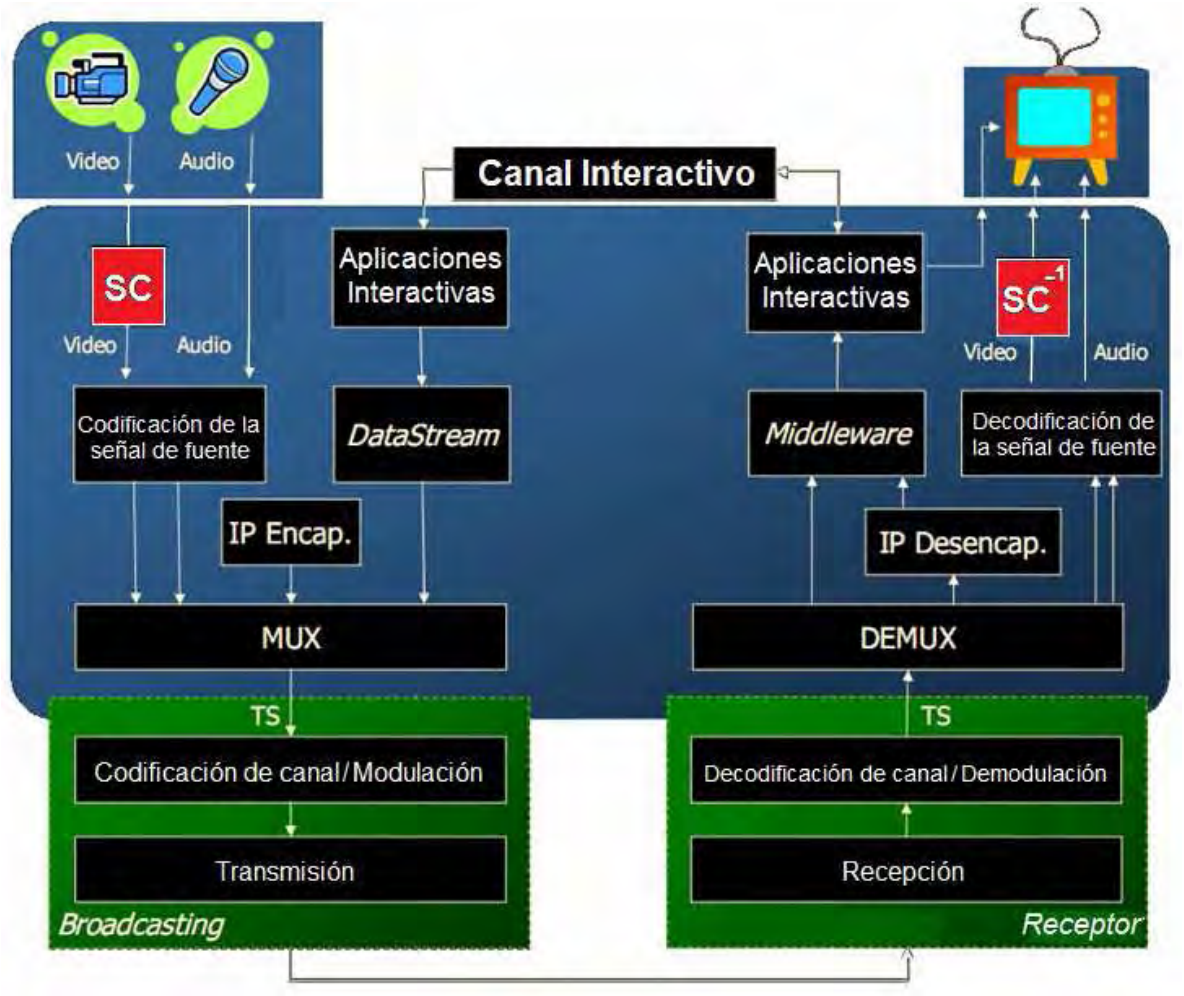


Figura H.4: Independencia de la super-compresión del datastream de interactividad, las aplicaciones interactivas y su eventual canal de retorno.

La Fig.H.5 nos permite apreciar la ubicuidad de las aplicaciones interactivas, con el middleware de interactividad (Ginga), así como la codificación del audio y video, la capa de transporte (MPEG2) y finalmente y al más bajo nivel la modulación BST-OFDM. A la derecha de la estructura en capas, puede observarse el set-top-box.

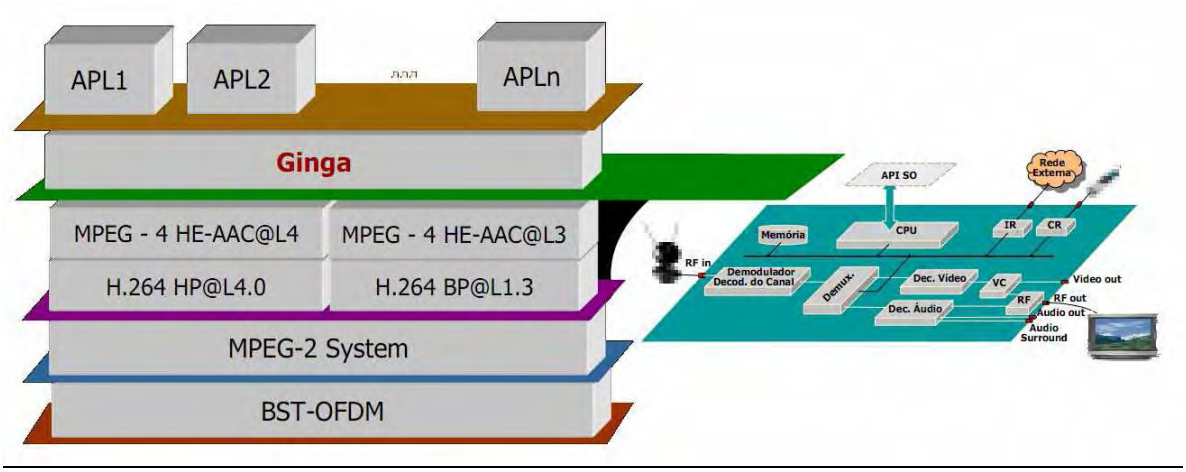


Figura H.5: Detalle del set-top-box y las capas desde las aplicaciones interactivas hasta la codificación y modulación.

Hoy en día se dispone de cámaras integradas 3D, como muestra la Fig.H.6. No obstante, las imágenes generadas por un sistema de TV 3D pueden ser de tres tipos a saber:

1. Anaglíficas, como muestran las Figuras H.7 y H.8, las cuales hacen al sistema económicamente competitivo, pero deben lidiar con el problema de los desagradables colores al quitarse las gafas.

Figura H.6: Cámara integrada 3D.

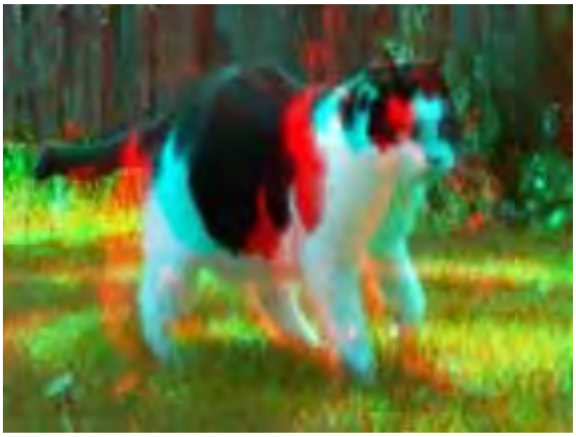


Figura H.7: Ejemplo anaglífico 1.

Figura H.8: Ejemplo Anaglífico 2

2. Las polarizadas, de mejor calidad que las anteriores, pero más caras y con serios problemas de balance de luminancia en ambos canales. Ver Fig.H.9.

Figura H.9: Ejemplo de polarizada.

3. Finalmente, las de obturación, las cuales son de una calidad superior, con lentes más caros y con la necesidad de emplear un dispositivo sincronizado con los anteojos que alterne la obturación de los lentes (transparente al usuario) y que completa la sensación 3D. Ver Figuras H.10 y H.11.

Los tres sistemas mencionados deben lidiar con el mismo problema, el excesivo ancho de banda que insumen en su transmisión. Hasta la fecha, solo se han podido realizar transmisiones de tipo terrestrial experimentales, como es el caso de la NHK de Japón, pero en 1080i 3D empleando la totalidad de los 6 Mhz del ancho de banda del canal, esto empleando la norma ISDB-T.

Figura H.10: Ejemplo de obturación 1.

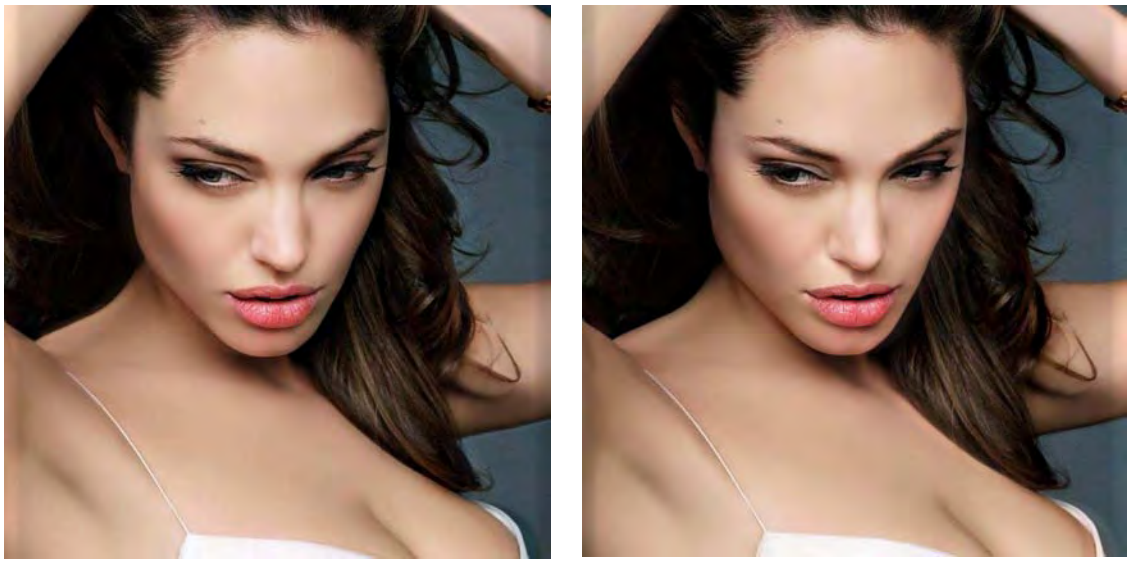


Figura H.11: Ejemplo de obturación 2.

Las diferencias más notorias entre la versión anaglífica y obturación son las siguientes:

1. el costo relativo entre ambas, donde tanto el TV como las gafas son más baratas en la versión anaglífica, pero
2. al sacarnos las gafas la versión anaglífica se ve como en las Figuras H.7 y H.8, mientras que en la versión de obturación se ve tan nítido como las Figuras H.10 y H.11

Por lo tanto, y especialmente en Japón, EE.UU. y Europa se emplean las opciones 1080i 3D vía satélite, mediante satélites de comunicaciones geoestacionarios y gracias a la opción conocida como Televisión Directa al Hogar (TDH), ver Fig.H.12. Demás está decir, que dicho sistema implica un costo de servicio para el usuario final fuera del alcance del gran público latinoamericano en general y argentino en particular. También en los países mencionados se emplea la opción digital para el cine de alta definición 3D.

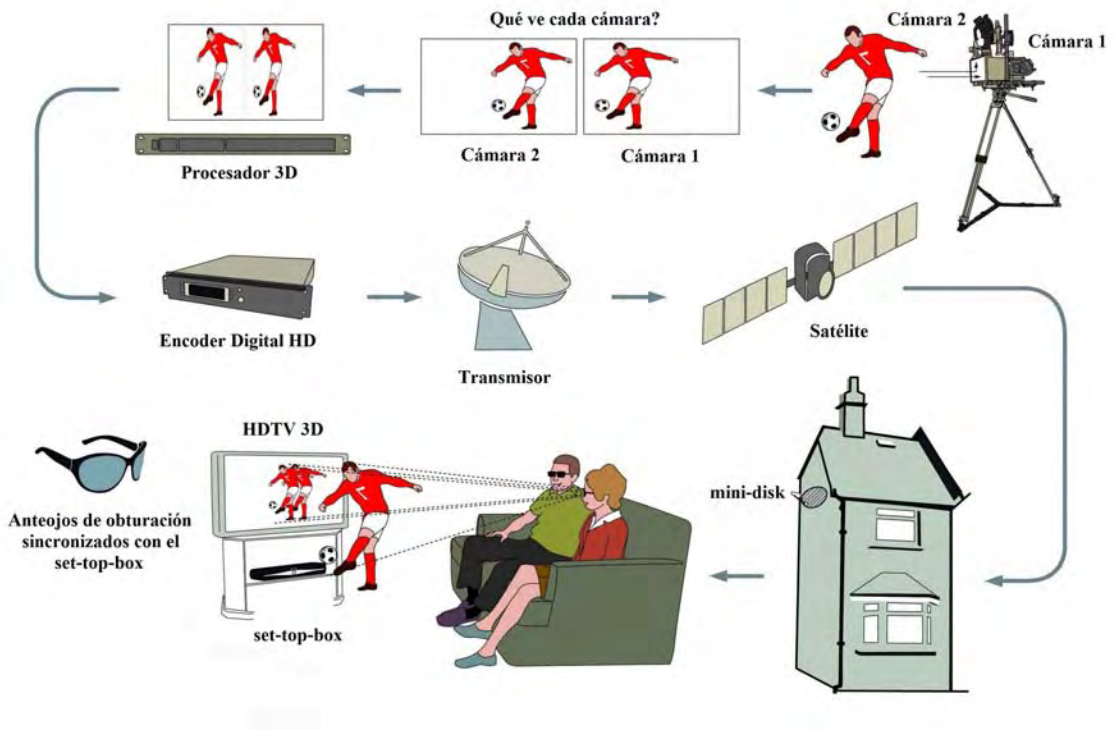


Figura H.12: Transmisiones 1080i 3D satelitales.

La masterización de esta opción la podemos ver en la Fig.H.13, en la cual ambas componentes estereoscópicas se alternan una detrás de la otra, en amarillo para el ojo derecho y en celeste para el izquierdo.

Fig.H.13: Masterización para cine digital 3D de alta definición.

Por otra parte, las Figuras H.14 y H.15 nos muestran las proporciones/resoluciones comparativas de las normas de Cine Digital vs Full-HD TV Digital.

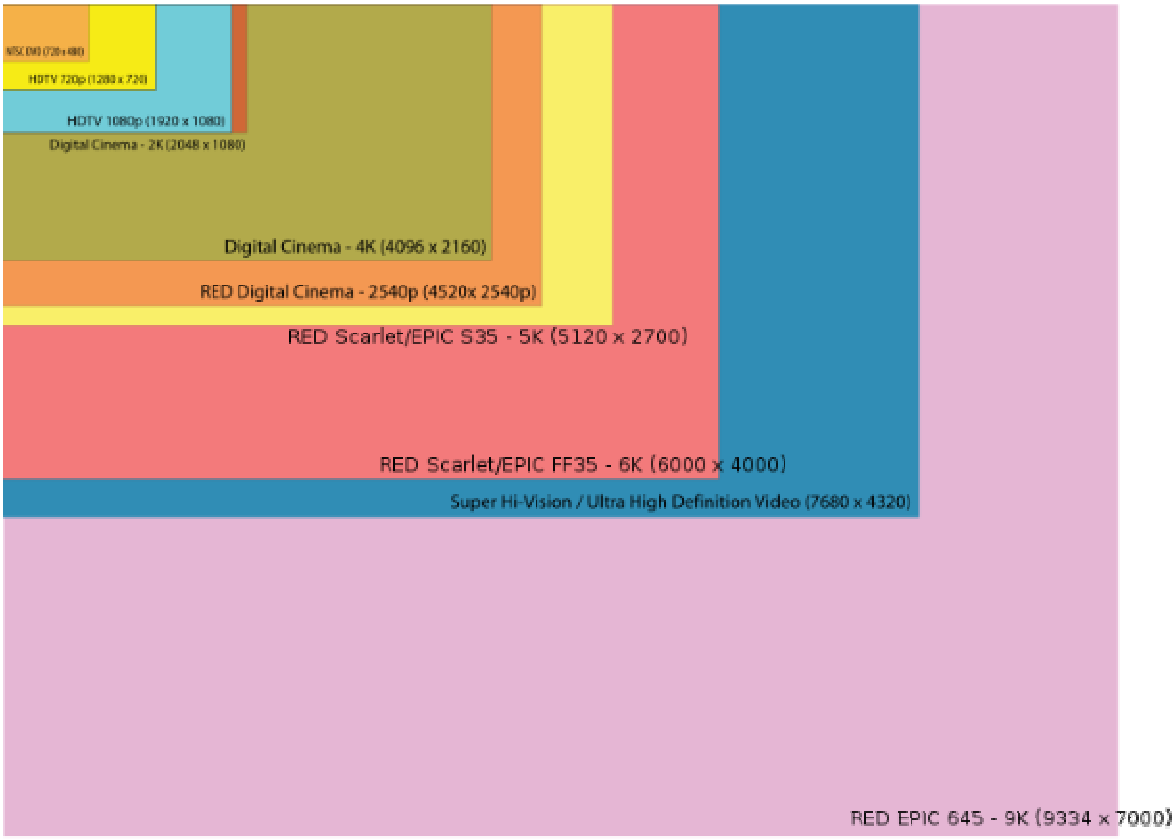


Figura H.14: Detalle de la norma de Cine Digital RED EPIC 645 – 9K (9334x7000 pixeles) en violeta.
Full HD para TV Digital está en celeste en el margen superior izquierdo.

Figura H.15: Detalle de la norma de Cine Digital RED EPIC 617 – 28K (28000x9334 pixeles) en rojo.
RED EPIC 645 quedó en carmín, mientras que también aquí
Full HD para TV Digital está en celeste en el margen superior izquierdo.

Estas tecnologías de ultra alta resolución/definición son altamente reusables en infinidad de otras aplicaciones de fundamental importancia para la sociedad constituyendo un verdadero derrame a todo tipo de incumbencias, a saber:

Super-resolución para:
• Imágenes Médicas (DICOM 3.0)
• Imágenes Satelitales
• Procesamiento Forense de Imágenes
• Procesamiento de Imágenes Multimediales
• Inteligencia de Imágenes
• Esteganografía y Esteganálisis
• Microscopía Electrónica
• Procesamiento de Microarrays en Bioinformática
• Vigilancia y Seguridad

Super-compresión para:
• IPTV
• WebTV
• TV Geométrica
• Documentología Digital
• Video como Youtube
• Diarios y Revistas Electrónicas
• e-Learning
• Galerías y Museos para Arte Virtual
• Diseño Digital
• GIS
• Hospital Electrónico y Telemedicina
• i-pod y Libros Electrónicos
• MP5
• Realidad Virtual
• Teletrabajo

Consecuente con lo dicho, no debemos dejar de apreciar la actual tendencia de incorporación de tridimensionalidad a los dispositivos móviles, ver Figuras H.16 y H.17.

Figura H.16: Celulares 3D.

Figura H.17: iPAD y tablets 3D.

Con respecto a la utilización del canal (y los respectivos segmentos que lo componen), así como los distintos servicios que soportan las distintas versiones de la norma ISDB-T, tenemos:

1. ISDB-T (ver Fig.H.18) actualmente en uso en Japón, la cual emplea como compresor al Algoritmo MPEG2, y que consiste en 14 segmentos, de los cuales 13 son útiles y cuyo orden se detalla en la Fig.H.19. esta norma permite varios servicios: 1 servicio HD de 1080i o 720p que emplea 12 segmentos, 1 servicio MD de 1440 x 1125i que emplea 8 segmentos y el cual puede complementarse con 1 servicio SD de 576i el cual emplea 4 segmentos, o bien, 3 servicios SD. El segmento cental o S0 se reserva para el one-seg de 320x240p. La Tabla H.1 nos muestra los caprichosos formatos de esta norma, algunos de los cuales solo son conocidos en el Japón.

2. ISDB-Tb (ver Fig.H.20) actualmente en uso en Argentina y Brasil, la cual emplea como Algoritmo de compresión al Advanced Video Coding (AVC), o más conocido por su nombre en ISO, MPEG4-10, o mejor aún las siglas impuestas por la Unión Internacional de Telecomunicaciones (ITU), es decir, H.264. Esta norma aunque organizada también en 13 segmentos útiles (dado que en ambas el segmento 14 es fraccionado en dos mitades con objeto de usarlo al comienzo y al final de salvaguarda, a los efectos de evitar el solapamiento con otros broadcasters) hace un mejor aprovechamiento del espectro por lo cual permite los siguientes servicios: 2 HD de 1080i o 720p, 4 SD de 576i y el one-seg de 240p. Es importante mencionar que esta norma no prevé ninguna de las siguientes resoluciones: 1080p mono y 1080p stereo (3D) para su empleo en la modalidad terrestrial, por lo cual, su ventaja frente a ISDB-T aunque importante, es escasa.

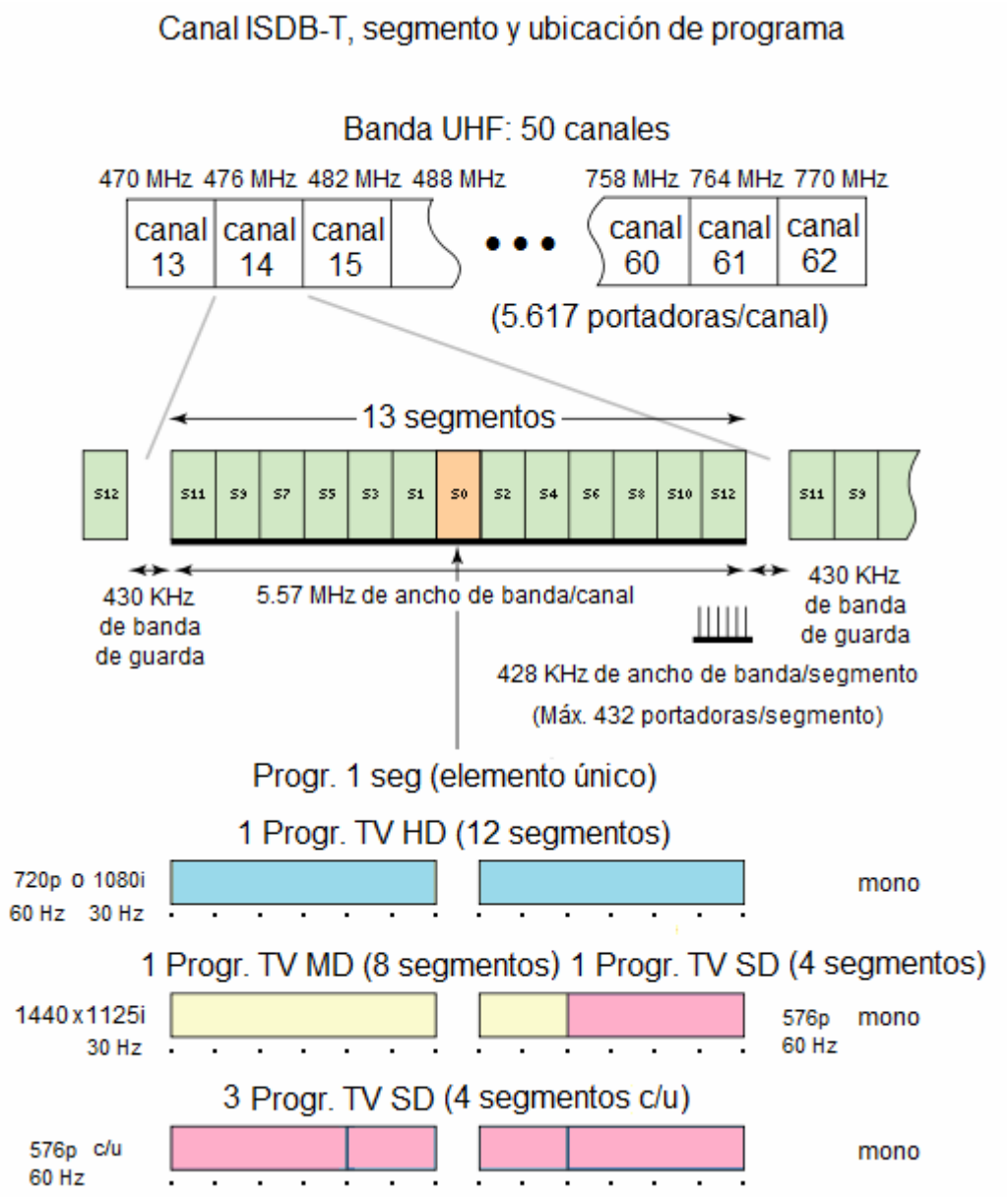


Figura H.18: Actual organización de servicios en ISDB-T (Japón).

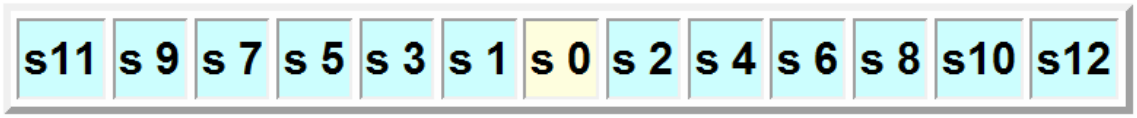


Figura H.19: Detalle de ordenamiento de los 13 segmentos en ISDB-T.

3. ISDB-Ta (ver Fig.H.21) es la norma que proponemos para nuestro país, la cual utilizaría la super-compresión, la cual permite en cambio y como muestra la Fig.H.21, 8 servicios 720p o bien 1080i, 12 servicios 576p, y por último 2 servicios 1080p-3D, sin afectar el original *one-seg*, en ninguno de los casos mencionados. Esto conlleva a un salto cuantitativo y cualitativo respecto a la norma ISDB-Tb.

Tabla H.1: Formatos de resoluciones en ISDB-T.

ISDB-T formats						
	format	horizontal pixels	vertical scan lines	aspect ratio	scan mode	frame rate
HDTV	1125i	1920	1080 *	16:9	interlaced	29.97 Hz
	1125i	1440	1080 *	16:9	interlaced	29.97 Hz
	750p	1280	720	16:9	progressive	59.94 Hz
SDTV	525p	720	480	16:9	progressive	59.94 Hz
	525i	720	480	16:9	interlaced	29.97 Hz
	525i	544	480	16:9	interlaced	29.97 Hz
	525i	480	480	16:9	interlaced	29.97 Hz
	525i	720	480	4:3	interlaced	29.97 Hz
	525i	544	480	4:3	interlaced	29.97 Hz
	525i	480	480	4:3	interlaced	29.97 Hz

Cómo podemos observar en la Fig.H.20 para los 6 Mhz de la norma ISDB-Tb cada servicio de 576i emplea 3 segmentos c/u, es decir, 4.5 MBPS. Mientras que en la ISDB-Ta solo utiliza 1 segmento, pero además es de escaneo progresivo en lugar de interlaceado.

De esta manera pasamos de ISDB-Tb a ISDB-Ta (donde la “a” es de advanced). No obstante, este tipo de tecnología es absolutamente independiente del algoritmo de compresión empleado y de la norma utilizada. En otras palabras, se lo puede utilizar indistintamente en MPEG2, H.264, VP8, etc. tanto para las normas DVB, ISDB-T, ISDB-Tb y ATSC. Incluso, es extensivo su uso no solo a TDT sino también a IP-TV, WebTV/OTT, etc, como así también cualquier otro medio de transporte dedicado punto a punto, o satelital.

Otro avance importante de la norma ISDB-Ta es que tanto la super-resolución, como la super-compresión pueden implementarse en un chip de tipo ARM, el cual iría en el set-top-box, haciendo a estos, más baratos, de menor consumo y más pequeños, como se mencionó en la Fig.H.2. A este respecto la UNTreF aportaría tanto el Algoritmo de super-resolución como el de super-compresión, mientras que la UNS aportaría el diseño del chip, la Corporación MOSIS de EE.UU. prototiparía el chip, y seguramente la faundry china Celestial manufacturaría el chip en escala comercial. En otras palabras, estaríamos en presencia del más importante proyecto tecnológico de la Argentina.

La Fig.H.22 muestra en detalle los atributos de la super-resolución desarrollada (la cual es un engranaje fundamental de la super-compresión), la cual elimina efectos espúreos tales como impainting y aliasing.

La Fig.H.23 muestra en detalle el interior del chip y sus bloques constitutivos fundamentales, mientras que la Fig.H.24 nos muestra también en detalle los pasos en la creación del mencionado chip. Como se ha mencionado, el diseño de este dispositivo queda absolutamente en manos de los profesionales de la Universidad Nacional del Sur.

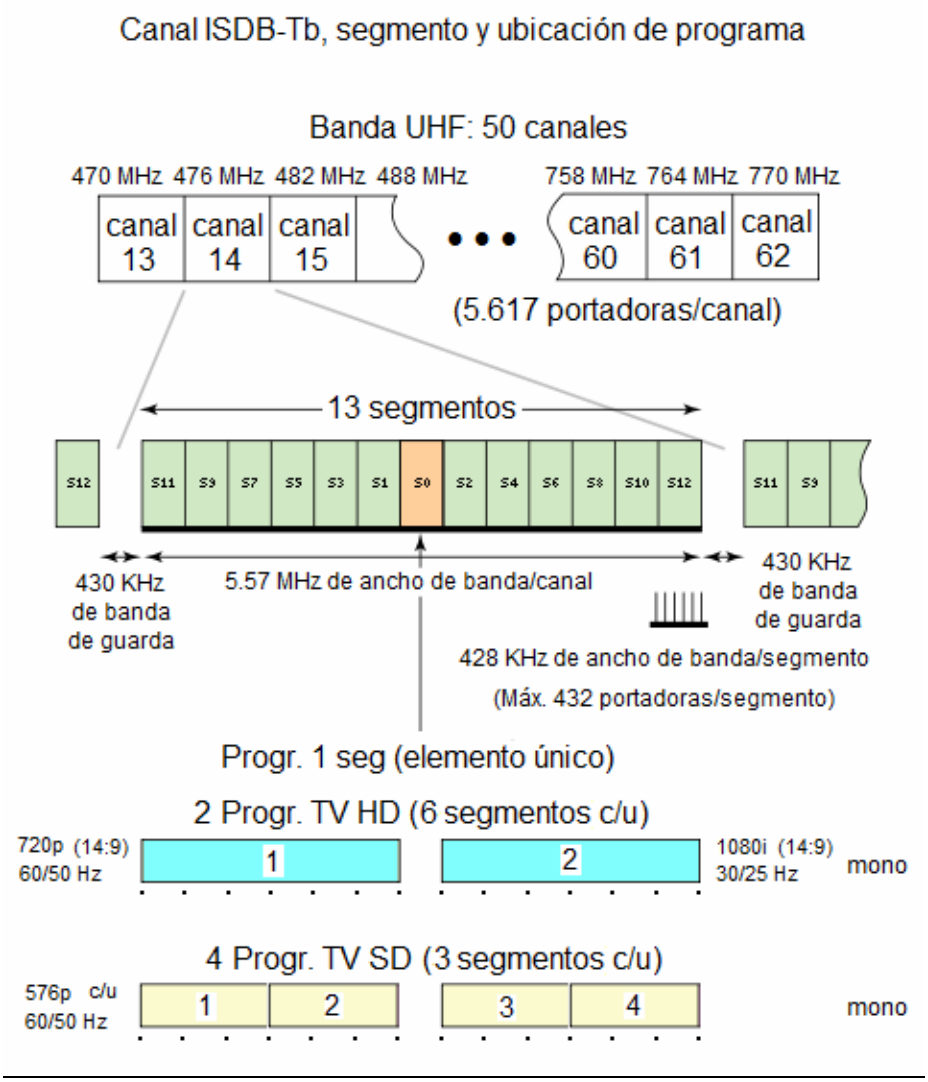


Figura H.20: Actual organización de servicios en ISDB-Tb (Brasil y Argentina).

La Fig.H.25 nos muestra la cobertura de la Televisión Digital Terrestre (TDT) en la República Argentina, como es lógico, con un marcado sesgo urbano.

Finalmente, la Tabla H.2 nos muestra la taxonomía Tecnologías vs Sistemas para EE.UU., Europa, Japón, Brasil y la Argentina. En dicha tabla, el texto en verde indica la innova-ción de la norma ISDB-Tb frente a la ISDB-T, mientras que a su vez, el texto en celeste nos indica las ventajas de la norma ISDB-Ta frente a la ISDB-Tb.

Es importante aclarar en este punto, que la norma ISDB-Tb es en realidad un experimento japonés de la NHK, el cual utilizando a Latinoamérica como laboratorio experimentó H.264 según todas las métricas conspicuas a considerar:

a) Latencia
b) Calidad de recuperación de la imagen
c) Baja del bit-rate

Una prueba de esto lo constituye el hecho de que la NHK de Japón ha decidido realizar un upgrade de su norma (conocida como ISDB-T2) reemplazando MPEG2 por H.264.

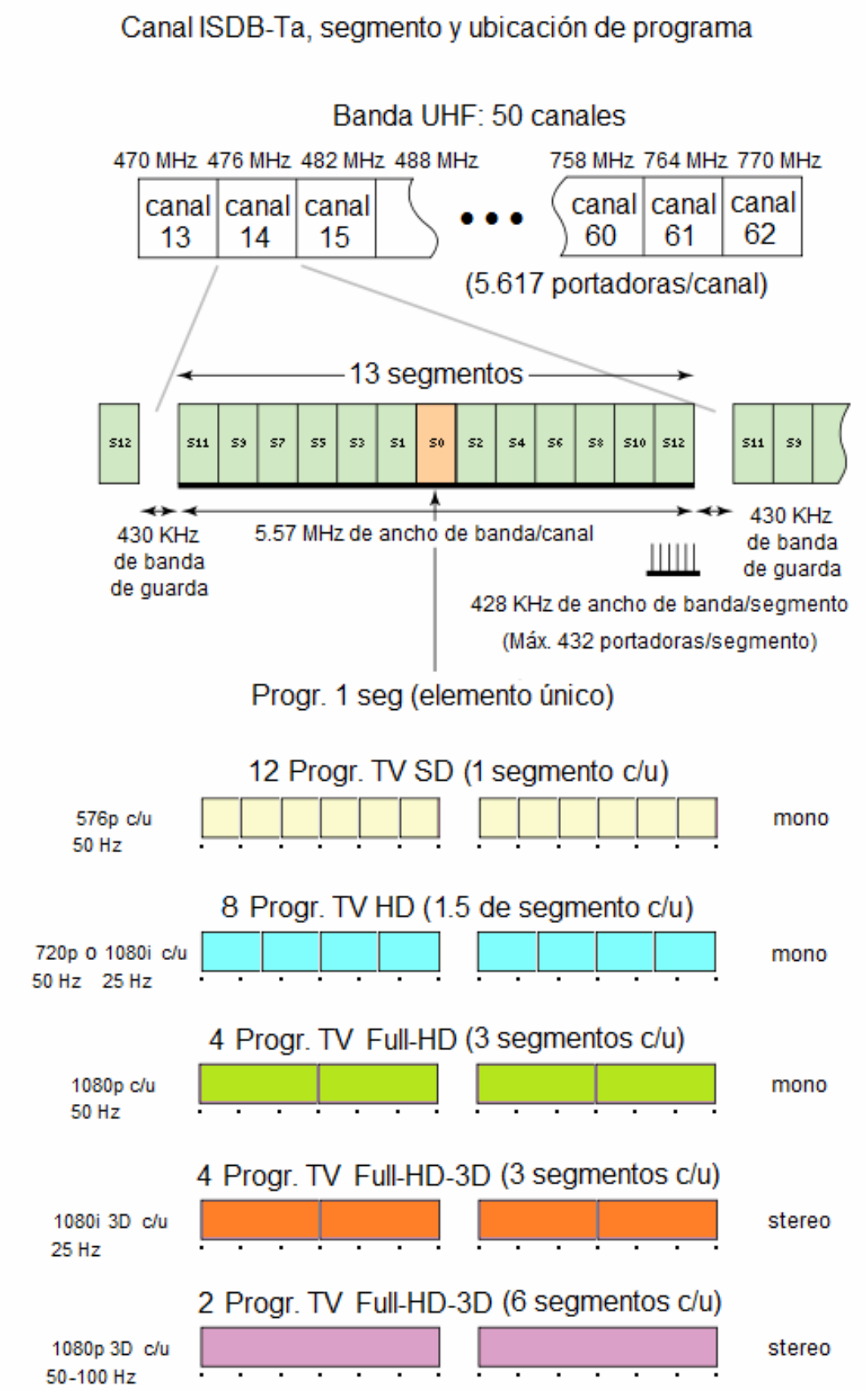


Figura H.21: Empleando seis segmentos para cada transmisión 1080p stereo y a la vez ahorrando una enorme porción del espectro radioeléctrico (ver parte baja de la figura en violeta), así como pasando de 4 576p a 12 576p y de 2 a 8 1080i/720p.

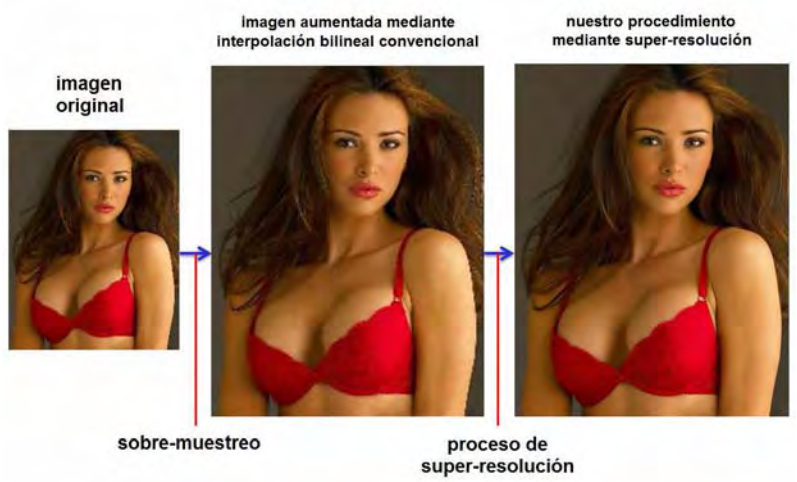


Figura H.22: Detalle de la super-resolución.

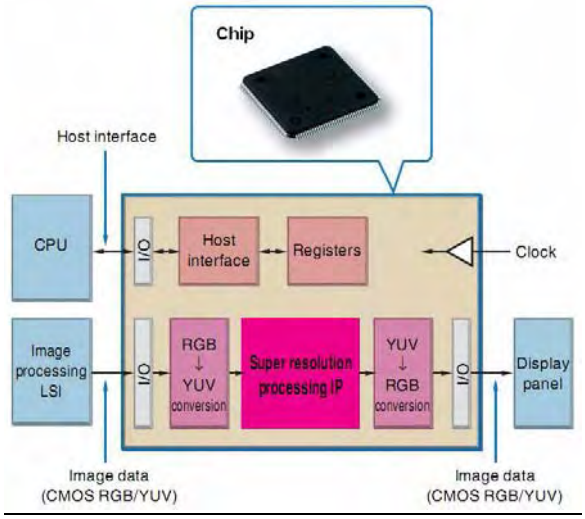


Figura H.23: Detalle del chip.

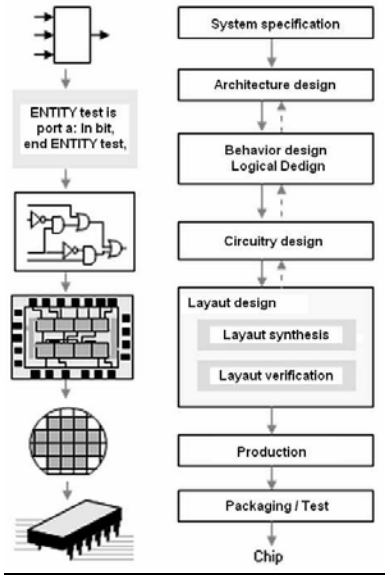


Figura H.24: Pasos en la gestación del chip.

Figura H.25: Cobertura de la TDT en la República Argentina.

Tabla H.2: Tecnologías vs Sistemas.

Tecnologías	Sistemas				
	ATSC	DVB	ISDB-T	ISDB-Tb	ISDB-Ta
	USA	Europa	Japón	Brasil	Argentina
Aplicativos	Interactivo	Interactivo	Interactivo	Interactivo	Interactivo
Middleware	DASE	MHP	ARIB	Ginga	Ginga
Compresión de Audio	Dolby AC3	MPEG-1 L-II	MPEG-II AAC	MPEG-II AAC	MPEG-II AAC (SC+GPGPU)
Compresión de Video	MPEG-2	MPEG-2	MPEG-2	H.264 (no es brasilero)	H.264 (SC+GPGPU+chip)
Transporte	MPEG-2	MPEG-2	MPEG-2	MPEG-2	MPEG-2
Transmisión y Modulación	8-VSB	COFDM	BST-OFDM	BST-OFDM	BST-OFDM (GPGPU y eOFDM)
Escaneo	576i, 720p y 1080i	576i, 720p y 1080i	576i, 720p y 1080i	576i, 576p, 720p y 1080i	576i, 576p, 720p, 1080i y 1080p
3D en terrestrial	No	No	Si a 1080i y 12 seg. (6 MHz)	No	Si a 576p, 720p, y 1080p a 6 seg.
Gestión de Contenidos	No tiene	No tiene	No tiene	No tiene	Tiene

H.3 PRUEBA #2: "Codificación y transmisión de 4Kp-3D y 8Kp-3D sobre red óptica"

Transmisión de una señal estereoscópica (3D) de televisión digital en ultra-alta definición 4Kp-3D (3840 x 2160p x 2) a 120 a 600 frames-per-second (FPS) sobre una red óptica. La excesiva cantidad de FPS se debe a requerimientos relativos a la supresión de blur y una mayor sensación de movimiento. Estas resoluciones permiten realizar panorámicos más amplios, asi como observar detalles en aproximación. En estas resoluciones no se utilizan encoders lossy (salvo la versión japonesa), y tampoco se emplean encoders con tratamientos inter-cuadro, solo intra-cuadro, dada la excesiva latencia que un análisis inter-cuadro generaría en relación a la compensación de movimiento, la cual insume el 80 % del tiempo de codificación.

Aclaraciones:

1. Las pruebas a realizar en Canal 7 son tendientes a demostrar que los procesos de compresión/descompresión del encoder propuesto no generan latencia en comparación a los que tienen lugar con H.264, VP8 y todos los demás: Argentina y Brasil 4.5 segundos, Europa 5.5 segundos (MPEG2/DVB), etc. Es decir, el encoder propuesto es el primero verdaderamente low-latency de la historia.

2. La latencia de un sistema de TV Digital frente a su versión analógica responde a dos factores a saber: a) El sistema de codificación, modulación y transmisión y b) El tratamiento inter-cuadro del códec empleado por la norma. Este tratamiento está organizado en tres partes, a saber: I. Detección de escena, II. Detección de componentes fijas y móviles, donde las fijas solo se codifican la primera vez que aparecen, mientras que se sigue a las móviles con la creación de un vector de movimiento. III. Soporte a Regions on Interest (ROI) para una compresión más eficiente de las mismas.

Por otra parte, los sistemas experimentales actualmente en prueba en el mundo, como el caso japonés de la NHK (ver Fig.H.26) el cual fue desarrollado para resoluciones de 4K, 4K-3D, 8K (7680 x 4320 pixeles) y 8K-3D, y que tiene pérdidas (lossy) infligidas por el uso de H.264 en cada tile de 1920x1080 en el que es dividida la imagen, a través de un encoder Fujitsu IP9500 (ver Fig.H.27) para c/tile (cuyo costo individual en Japón es de 35.500 U$S). Si consideramos que se necesitan 16 de estos dispositivos para 8K, y 32 para 8K-3D solo del lado del encoder, más una cantidad similar del lado del decoder, nos deja entrever que esta propuesta solo sirve para conexiones ópticas punto-a-punto.

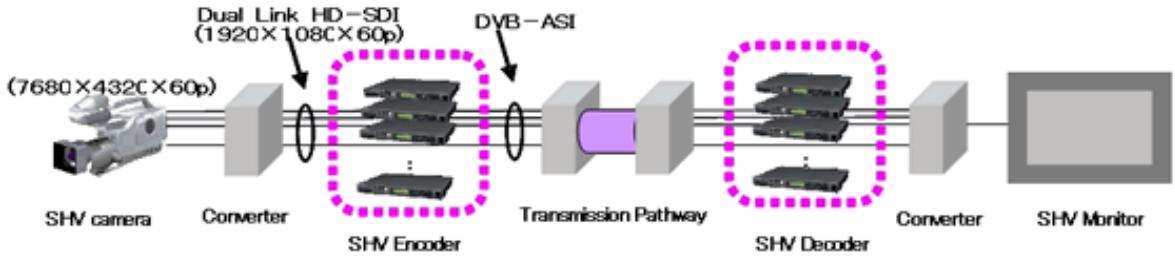


Figura H.26: Propuesta japonesa (NHK) para 4K, 4K-3D, 8K y 8K-3D.

Figura H.27: Encoder IP9500 Fujitsu.

Dado que, para una adecuada proyección de las resoluciones 4K y 8K se necesitan monitores de 152” y 304” respectivamente, se hace evidente que estas resoluciones solo tienen una adecuada justificación en el terreno de las salas digitales, para cine digital o shared exposure.

La propuesta japonesa establece que la resolución 8K permite una aproximación de 0.75 de la altura del monitor, frente a Full-HD, en la cual esta aproximación no debía ser menor a 3 veces la altura correspondiente en ese caso. Por otra parte, una segunda justificación de esta resolución tiene que ver con el aumento panorámico para una misma escena, ver Fig.H.28. Estas elucubraciones de la NHK no contemplan el hecho no trivial del escaso espacio disponible en Japón en general y Tokio en particular. Al solo efecto de dar una idea comparativa, un departamento de 4 ambientes en Tokio tiene la misma superficie en metros cuadrados que uno de 2 ambientes en Buenos Aires. Por lo cual surge una pregunta automáticamente: en cuántos departamentos de Tokio (y Japón) podemos ubicar estos televisores, los cuales de poder ser instalados, solo podrían ser recibidos al solo momento de la construcción misma de la vivienda anfitriona.

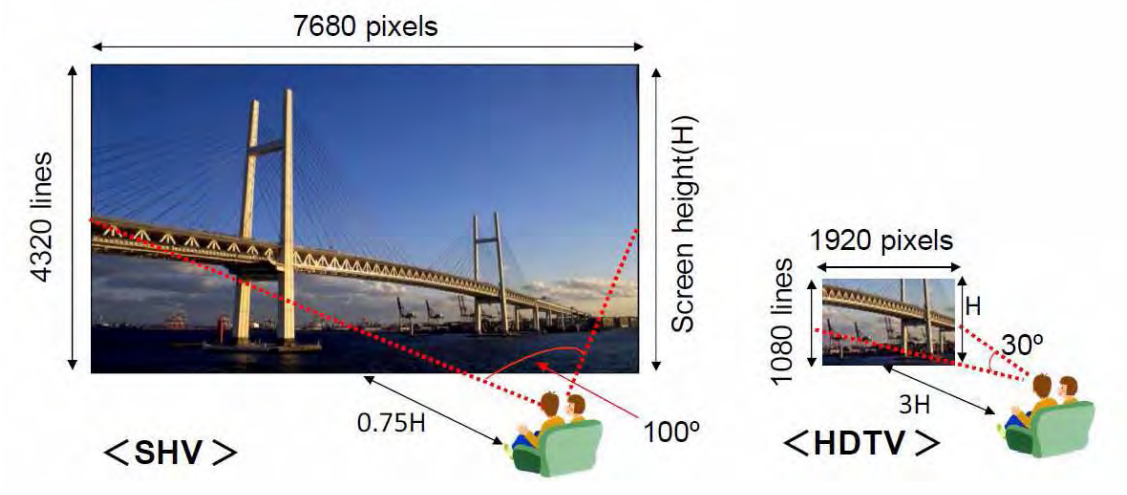


Figura H.28: Justificación japonesa para 8K.

En otro orden de cosas, y con objeto de su mundial de futbol en el 2014 y las olimpíadas 2016, Brasil está trabajando en su propio sistema de TV 4K-3D. El proyecto se llama 2014K.org (ver Fig.H.29), y en relación al cual ya se están haciendo transmisiones experimentales sobre redes ópticas, con objeto que llegado el día del mundial se transmitan imágenes del mismo en resoluciones:

1. 4K-3D a USA, Japón y Europa mediante fibra óptica sub y trans-oceánica
2. 2K-3D (1920x1080px2x120-600 FPS) al interior del Brasil, para lo cual ya se están construyendo 40 salas
3. 1080i (y si es posible 1080i-3D, como el caso actual de la NHK de Japón) en modalidad terrestrial al gran público

Figura H.29: Proyecto brasilero 2014k.org para el Mundial 2014 y las Olimpíadas 2016, ambos en 4K-3D.

La versión hacia el interior de Brasil en 2K-3D la están proyectando considerando el uso de la futura red óptica brasilera, conocida como Brasil Conectado. Similar proyecto está llevando a cabo la República Argentina, el cual se llama Argentina Conectada. De hecho, en el año 2014 ambas redes se conectarán entre sí a la altura de la provincia de Misiones, ver Fig.H.30.

Un problema muy serio que se presenta en ambos países es que con objeto de la creación de ambas mega-redes se podría intentar enviar imágenes 4Kp-3Dx120-600 FPS lo cual ocuparía completamente el ancho de banda de ambas, inhibiendo el aprovechamiento de las mismas para otros usos de fundamental importancia, como lo constituyen, por dar solo un ejemplo, la Telemedicina, es decir, la transmisión de imágenes médicas según el protocolo DICOM 3.0.

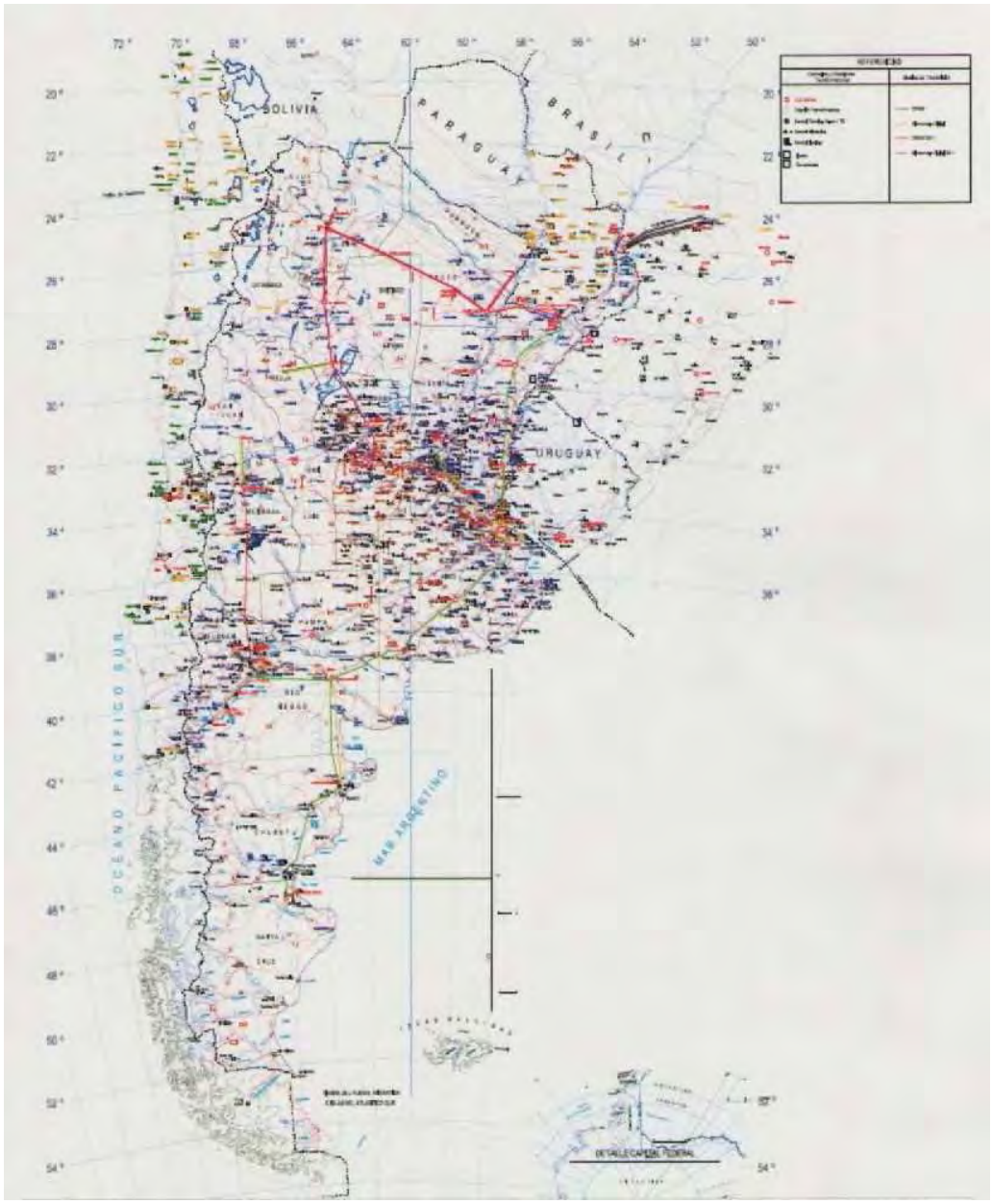


Figura H.30: Mapa de Argentina Conectada.

Por lo tanto, entre la propuesta japonesa de transmisión con pérdidas (lossy) y la brasilera sin pérdidas (lossless) pero a costa de un restricción total en el uso de la red, surge la propuesta objeto de esta prueba, y la cual consiste en el desarrollo de un encoder nacional especialmente desarrollado para Argentina Conectada, que sin tener pérdidas permita aprovechar en forma muy superior la red óptica, permitiendo la simultaneidad de servicios sobre la misma. Este encoder trabaja de una forma absolutamente inédita, es decir, constituye en si mis-mo un nuevo paradigma tecnológico en lo que a encoders se refiere, generando en forma artificial un aumento de la redundancia tanto inter-cuadro como intra-cuadro de las distintos frames de la imagen para una posterior supresión controlada de la misma. Por ejemplo, si nos basamos en la Fig.H.31, luego de aplicar un procedimiento que separa en cuadrantes las sub-bandas espectrales de la imagen pero siendo dichas sub-bandas marcadamente similares desde el punto de vista espectral, aparecerán los cuatro cuadrantes

Figura H.31: Angelina.

mencionados, los cuales se pueden apreciar en la Fig.H.32. Este procedimiento es el único que permite sub-bandas espectrales todas de baja frecuencia (es decir, de aproximación) con un notable compromiso morfológico entre las misma. De hecho, los cuatro cuadrantes de la Fig.H.32 resultan idénticos al ojo cuando en realidad son marcadamente diferentes. Si practicamos sustracciones inter-cuadrantes surge la Fig.H.33, en la cual la codificación de tres de sus cuadrantes insumirá muchos menos bpp que los de la Fig.H.32.

Figura H.32: Angelina con Deeplets_MG.

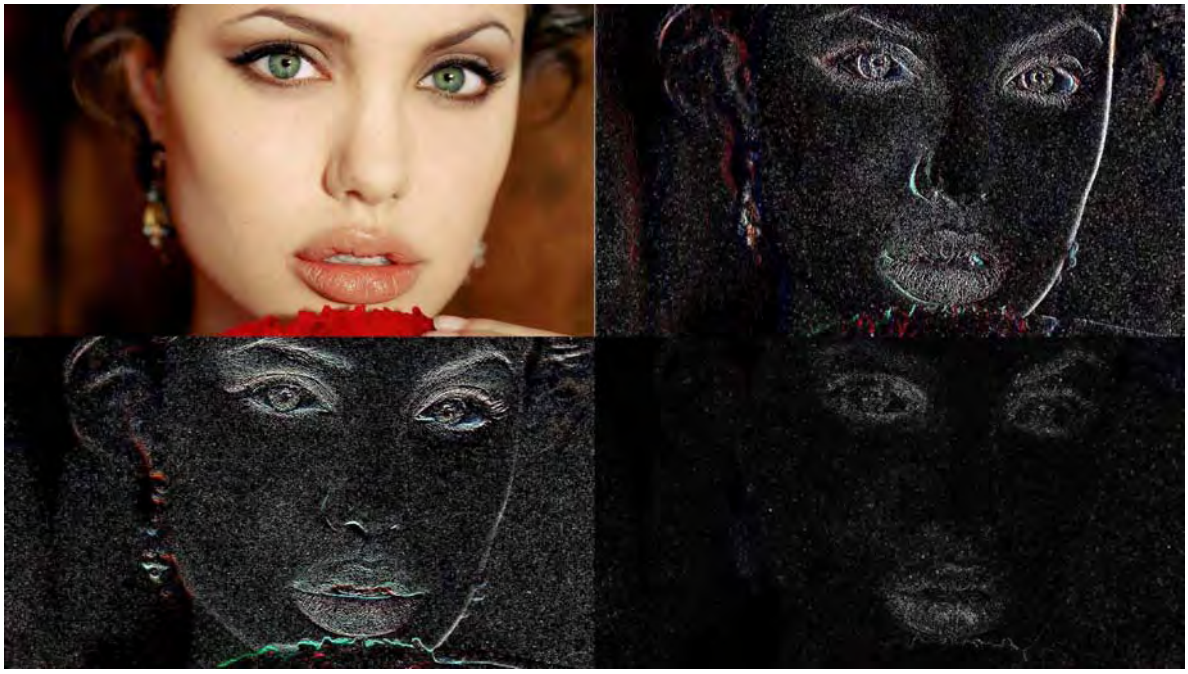


Figura H.33: Angelina con Deeplets_MG luego de la sustracción interbandas.

Figura H.34: Secuencia de tomografías computadas en corte oxiaxial.

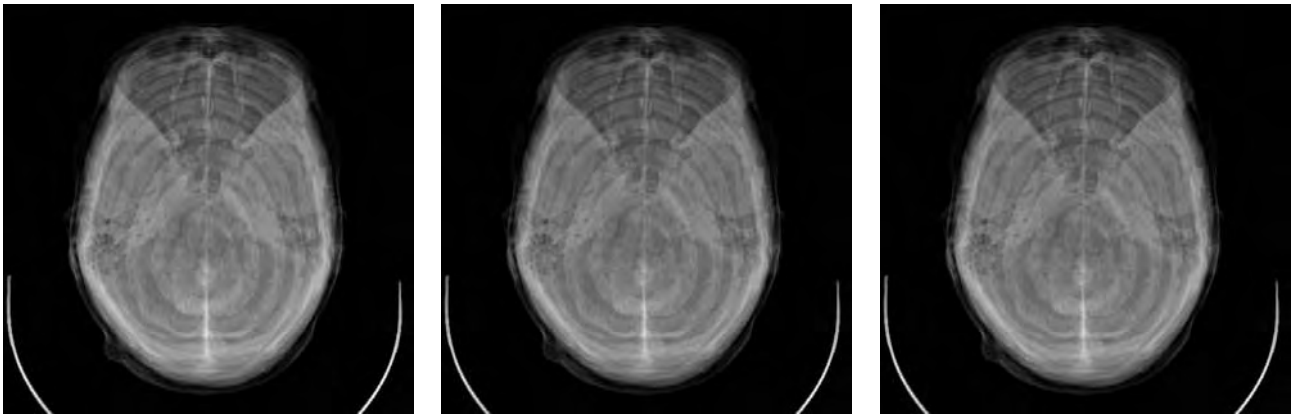


Figura H.35: Secuencia de tomografía de la Fig.H.34 luego de aplicarles Deeplets_MG.

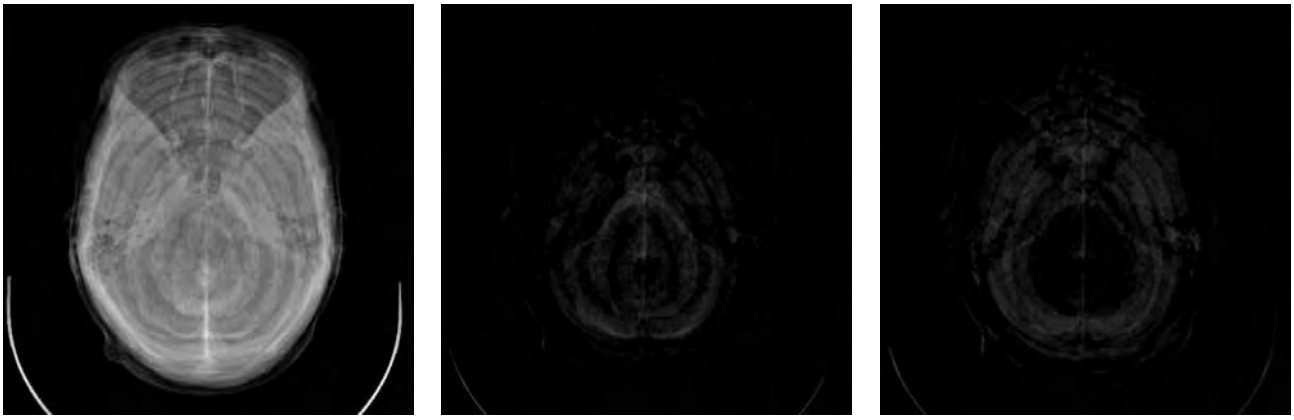


Figura H.36: Secuencia de tomografías de la Fig.H.35 luego de la sustracción intercuadros.

Esta herramientas reciben el nombre de Deep Wavelets of Minor Gap, o simplemente, Deeplets_MG para el caso inter-cuadro y Haar's Wavelets of Minor Gap, o simplemente, Haar_MG para el caso intra-cuadro, dando lugar a posibilidades antes nunca alcanzadas en ningún encoder conocido. Por ejemplo, si para el caso de la serie de imágenes de la Fig.H.34 le aplicamos Deeplets_MG obtenemos la secuencia de imágenes de la Fig.H.35 claramente similares morfológicamente (aunque en realidad no idénticas) que al practicarles la sustracción inter-imagen surge la nueva secuencia mostrada en la Fig.H.36, donde el nuevo trio es codificable en muchos menos bpp que el de la Fig.H.34. En estas nuevas herramientas se basa el encoder propuesto, el cual también tiene su versión para audio, por lo cual la solución propuesta se completa de la siguiente manera:

• **Encoder para video**:

Honomástico Científico:
Lossless Wavelets Coding – Four Dimensions (LWC-4)

• **Encoder para audio**:

Honomástico Científico:
Lossless Wavelets Coding – One Dimension (LWC-1)

• **Plataforma de interactividad**:

Honomástico Científico:
National Interactive Middleware (NIM)

• **Norma**:

Honomástico Científico:
Lossless Services on Optical Networks (LSON)

Versiones:
Cable (LSON-C)
Terrestrial (LSON-T)
Satelital (LSON-S)

Tabla H.3: Tecnologías vs Sistemas

Tecnologías	Sistemas				
	USA	Europa	Japón	Brasil	Argentina
Norma	No definida	No definida	No definida	No definida	LSON
Aplicativos	No definido	No definido	No definido	No definido	Interactivos
Middleware	No definido	No definido	No definido	No definido	NIM
Compresión de Audio	No definido	No definido	No definido	No definido	LWC-1
Compresión de Video	Solo intracuadro JPEG2000	Solo intracuadro JPEG2000	Splitting y H.264	No usa	LWC-4
Medio	Fibra óptica	Fibra óptica	Fibra óptica	Brasil Conectado	Argentina Conectada
Gestión de Contenidos	No tiene	No tiene	No tiene	No tiene	Tiene

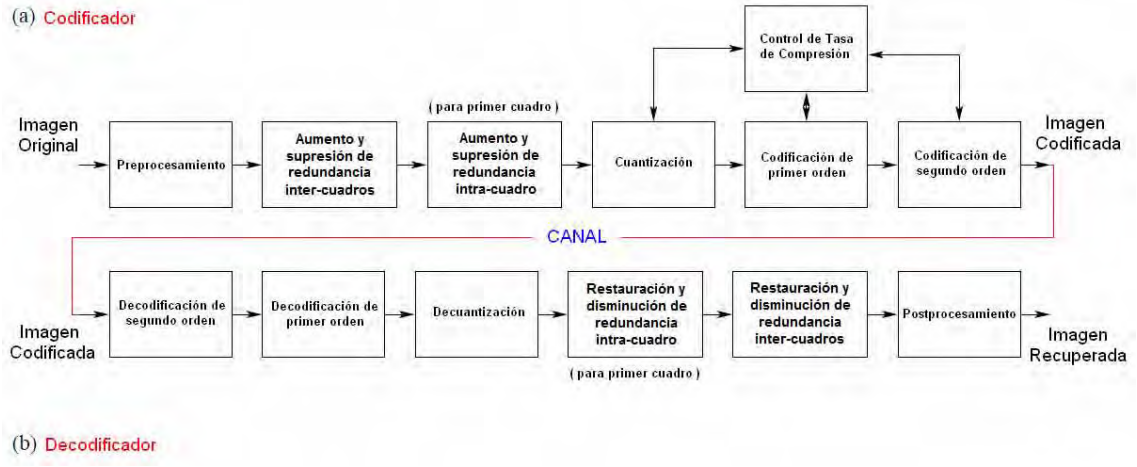


Figura H.37: Broadcasting de 4Kp-3D sobre la red óptica Argentina Conectada.
Detalle del encoder/decoder LWC.

La Tabla H.3 nos muestra el panorama internacional de Tecnologías vs Sistemas, donde se pueden apreciar los aportes e innovaciones de la versión Argentina. La Fig.H.37 muestra el diagrama en bloques o pipeline del encoder/decoder propuesto con particular énfasis en la funcionalidad de cada bloque. Finalmente, la Fig.H.38 nos muestra el detalle del encoder LWC con los módulos intervinientes y la organización de los bloques y los y los tiles en que se divide el primer cuadro de cada trama.

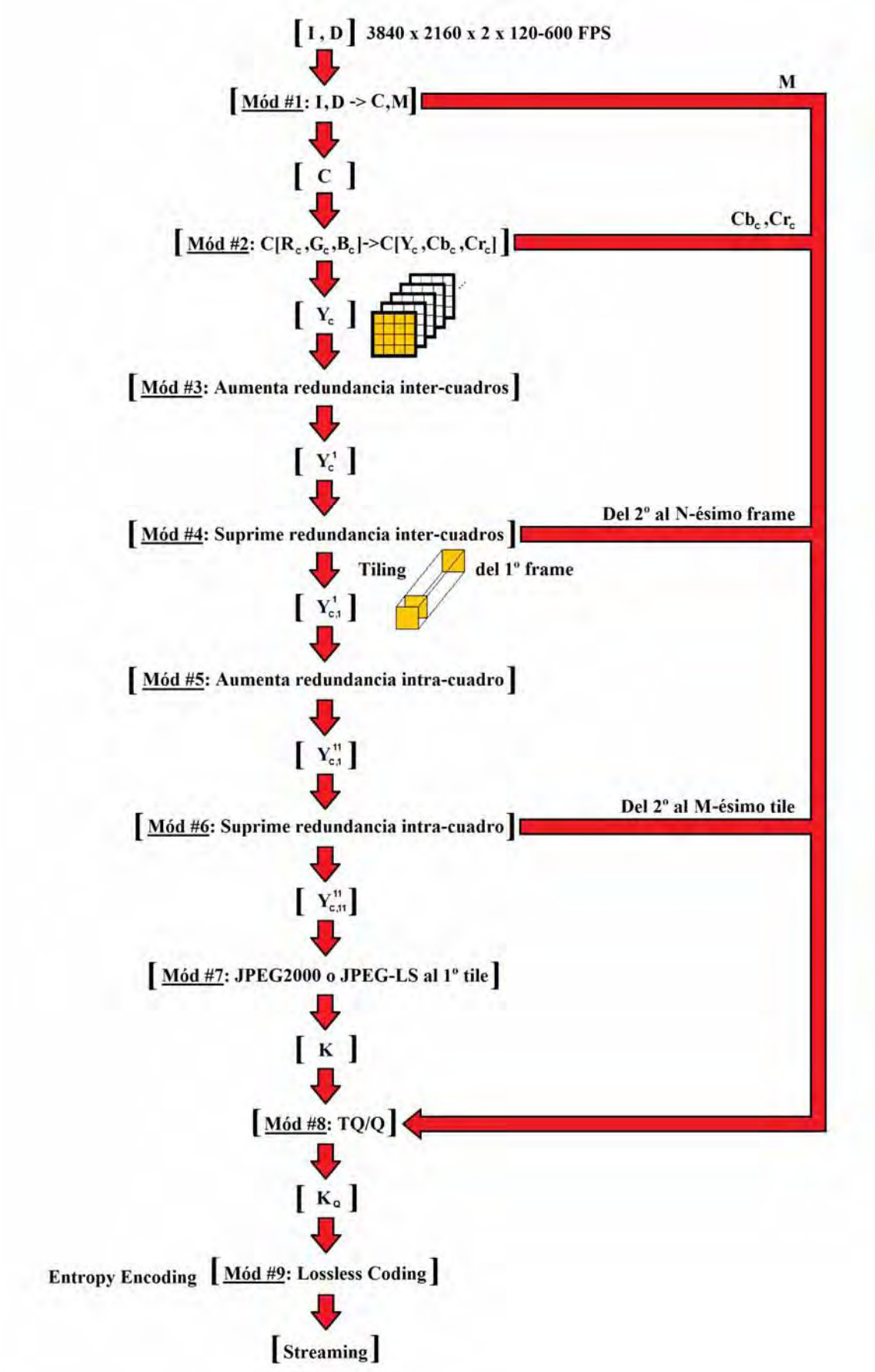


Figura H.38: Encoder LWC-4 para la norma LSON.

H.4 Conclusiones del apéndice

En este apéndice se enumeraron todas las pruebas a realizarse de manera oficial empleando los aportes del Capítulo 3, tanto pata TDT como para Argentina Conectada. En ambos casos los aportes permiten realizar pruebas con características nunca antes vistas en el manejo de estas tecnologías.

Apéndice I

Protocolos de las Pruebas para TDT y Argentina Conectada

I.1 Protocolos de las Pruebas para TDT y Argentina Conectada: Prolegómenos a la evolución de las normas empleadas

I.1.1. Pruebas:

I.1.1.1 Transmisión 576p

La Norma ISDB-Tb lo hace empleando 3 segmentos = 4.5 MBits.

Figura I.1: Cuadro 720x576p.

Encoder:

1.e.1. CPU con GPGPU y embebido en ella el encoder del Algoritmo Catalizador

1.e.2. CPU con GPGPU y embebido en ella el decoder del Resolution Control Coding (RCC)

Decoder:

1.d.1. STB + CPU con GPGPU y embebido en ella el decoder del Resolution Control Coding (RCC), el cual permite ocupar solo 1 o 2 segmentos (es decir, 1.5 o 3 Mbit, respectivamente)

1.d.2. STB con Algoritmo Descatalizador Embebido
Permite ocupar solo 2 segmentos (es decir, 3 Mbit, respectivamente)

I.1.1.2. Transmisión 1080p-3D (no existe norma que lo contemple)

NOTA: La NHK de Japón transmite en forma experimental 1080i-3D ocupando los 6 Mhz y con cuestionable calidad, dado que la baja en el bit-rate se genera a partir de forzar al máximo las posibilidades de compresión del encoder MPEG2. En cambio, la presente propuesta permite 1080p-3D en solo 6 (o menos, por la acción del catalizador) segmentos de la norma de referencia, es decir, ISDB-Tb.

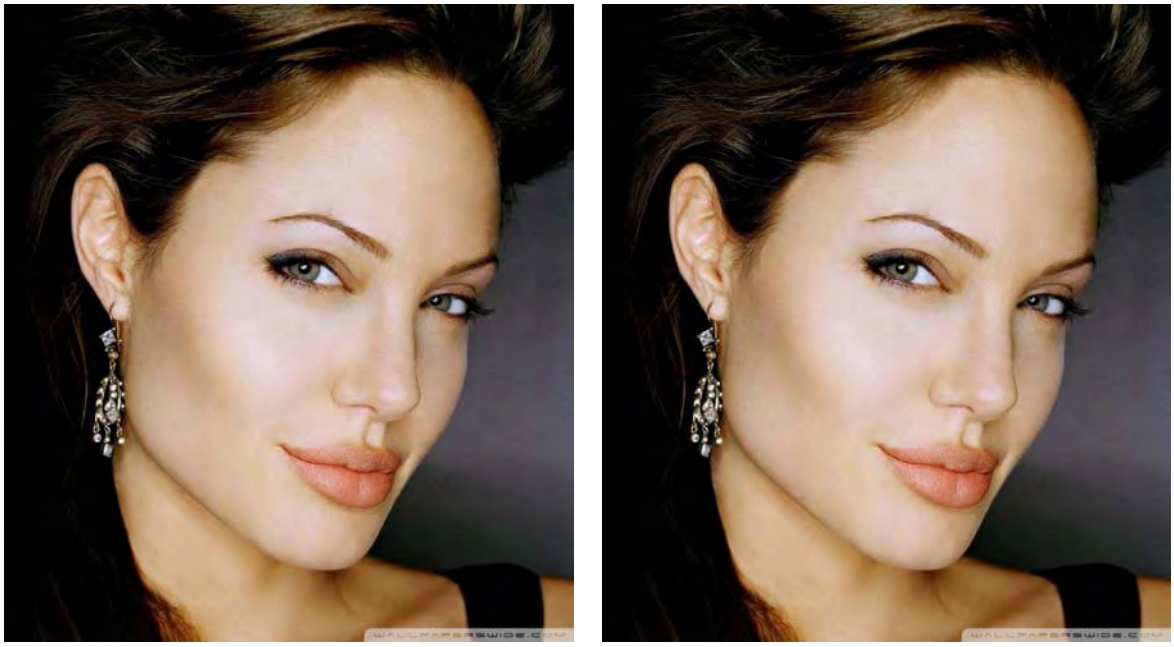


Figura I.2: Cuadros izquierdo y derecho de 1920x1080p c/u.

Encoder:

2.e.1. CPU con GPGPUs y embebido en ellas el encoder RCC

2.e.2. CPU con GPGPUs y embebido en ellas el encoder RCC + Catalizador (el cual aumenta dramáticamente la redundancia morfológica inter-cuadro de los frames originales, dando lugar a un nuevo grupos de frames con un considerable aumento de información mutua entre ellos, lo cual, hace mucho más eficiente a AVC/MPEG4-10/H.264 a la hora de comprimir).

Decoder:

2.d.1. STB + CPU con GPGPUs y embebido en ellas el decoder de RCC

2.d.2. STB + CPU con GPGPUs y embebido en ellas el decoder de RCC + Descatalizador (inversa del catalizador mencionado en el ítem anterior)

I.1.1.3. 4K-3D terrestrial (no existe norma que lo contemple)

NOTA: La diferencia con el caso anterior reside fundamentalmente en: a) la necesidad de más y/o más potentes GPGUs, tanto en el encoder como en el decoder, y b) la eventual necesidad de un splitter, a los efectos de fraccionar (tiling) cada imagen 4K en mosaicos (o tiles) de 1920x1080p c/u. También en este caso se utilizarán solo 6 segmentos de la norma de referencia ISDB-Tb, aunque se prevé jugar con los 12 segmentos, dejando libre el servicio de one-seg.

Figura I.3: Cuadros izquierdo y derecho de 3840x2160p c/u.

Encoder:

3.e.1. CPU con GPGPUs y embebido en ellas el encoder RCC

3.e.2. CPU con GPGPUs y embebido en ellas el encoder RCC + Catalizador

Decoder:

3.d.1. STB + CPU con GPGPUs y embebido en ellas el decoder de RCC

3.d.2. STB + CPU con GPGPUs y embebido en ellas el decoder de RCC + Descatalizador

I.1.1.4. 4K-3D y 8K-3D Argentina Conectada (no existe ni norma ni encoder oficialmente reconocido en ninguna parte del mundo)

Figura I.4: Tendidos de fibra óptica en Argentina Conectada.

Encoder:

4.e.1. Sistema Japonés con H.264 + Catalizador
Esto implica el uso del splitter mencionado en la prueba anterior, donde cada mosaico de 1920x1080p es codificado en H.264. El aporte novedoso reside en bajar notablemente el bit-rate a transmitir, al utilizar el catalizador mencionado en las pruebas anteriores.
Estas prueba (4K y 8K-3D) pensadas para Argentina Conectada implican el uso de encoders propietarios implementados en CPU+GPGPUs, así como el uso de iluminadores de fibra oscura consecuentes con lo dicho aquí.

4.e.2. Empleo del encoder Lossless Wavelets Coding 4 (LWC-4) el cual es implementado mediante CPU + GPGPUs en la Norma Lossless Services on Optical Networks (LSON). También se contempla el uso del datastream de Interactividad National Interactive Middleware (NIM).

Decoder:

4.d.1. Sistema Japonés con H.264 inverso + descatalizador

4.d.2. Empleo del decoder LWC-4 inverso, obviamente dentro de La Norma LSON y usando el middleware NIM.

I.2 Notas Finales:

1. Existe versión de audio del encoder LWC-4, cuya denominación es LWC-1.

2. En general, en 4K, 4K-3D, 8K y 8K-3D de aplicarse compresión, no se emplean compresiones con pérdidas (lossy), sino sin pérdidas (lossless).

3. Los encoders tradicionales están constituidos por varios módulos, a saber: Detección de escenas, compensación de movimiento y soporte a Regions of Interest (ROI). Estos son los responsables de la demora de la transmisión digital frente a la versión analógica y conocida como latencia, la cual en Europa (con DVB/MPEG2) es de 5.5 segundos, mientras que en Brasil/Argentna (ISDB-Tb/H.264) es de 4.5 segundos.

4. Los encoders utilizan el 80 % del tiempo de encodeo en la detección de movimiento.

5. En 4K, 4K-3D, 8K y 8K-3D no se emplea el tratamiento inter-cuadro con los módulos descriptos en el Punto 3, puesto que la latencia sería de minutos, lo cual es inaceptable para el broadcasting. Solo se emplea tratamiento intra-cuadro, generalmente se comprime con JPEG2000.

6. Las Pruebas 3 y 4 implican URGENTE la firma de un acuerdo entre la RTA y la NHK de Japón, por el aprovisionamiento de las cámaras 4K, el splitter y el LCD 4K-3D.

7. Estamos trabajando en modificar y mejorar H.264. Estas modificaciones y mejoras consisten en eliminar sus módulos de Detección de Escena, Compensación de Movimiento y Soporte ROI, para reemplazarlos por un par de procedimientos nóveles en el tratamiento intra-cuadro e inter-cuadro, convirtiendo a la versión mejorada o eH.264 en un encoder low-latency.

8. Estamos trabajando en un encoder/MUX para canales de televisión con la norma ISDB-Tb y encoder H.264 construido exclusivamente con GPGPUs.

9. Estamos trabajando en un procedimiento del tipo OFDM (y sus variantes, es decir, COFDM, BST-OFDM, etc.) prescindiendo del uso de la FFT^{-1} y reemplazándola con wavelets, bajando de esta manera de forma dramática la complejidad computacional. El OFDM mejorado o eOFDM también será implementado con GPGUs.

I.3 Conclusiones del apéndice

En este capítulo se enlistaron el total de las pruebas concernientes a la modalidad TDT, es decir, 576p, 1080p-3D y 4K-3D en este último caso para Argentina Conectada. Por otra parte, se describen las herramientas y metodologías necesarias para llevar a cabo las mencionas pruebas.

Apéndice J

Catalizadores de compresión para más eficientes enlaces, broadcasts y multicasts satelitales en DVB-S2

J.1 Descripción Operativa de las Funcionalidades de los Algoritmos. Catalizadores de Compresión Desarrollados. Concepto de Catalizador

Llamamos *Catalizador de Compresión* al procedimiento mediante el cual se puede aumentar adicionalmente la tasa de compresión de un algoritmo de compresión (o encoder) sin detrimento de la calidad final de recuperación de la entidad a ser comprimida, es decir, audio, imagen o video. No obstante, el precio que se paga – por lo general – es un desmedido aumento de la complejidad computacional, fundamentalmente del lado del encoder, la cual, en nuestro caso (el cual es el único, dado que el concepto de catalizador de compresión nos pertenece), es compensada empleando placas para procesamiento paralelo multi-core del tipo General-Purpose Computing on Graphics Processing Units (GPGPU). A este respecto, se debe desarrollar una Ingeniería Algorítmica tal que permita implementar en forma distribuida tanto el catalizador como el códec a ser catalizado, ya sea, AVC (H.264), DVB-S2, etc. Incluso en aquellos casos donde el códec no disponía de una implementación distribuida conocida.

Por otra parte, los catalizadores de compresión que hemos desarrollado pueden ser del tipo:

1. Con pérdidas (lossy)

2. Sin pérdidas (lossless)

La Fig.J.1 muestra cómo aplicar los mismos en ambos casos.

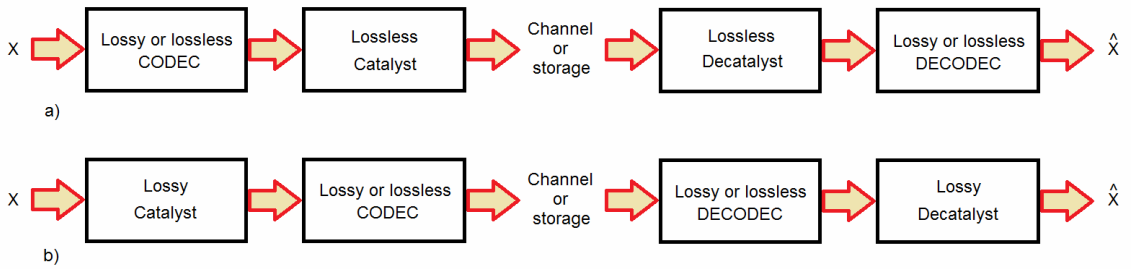


Figura J.1: Uso de los catalizadores según las pérdidas. Si el catalizador es:
a) Lossless (sin pérdidas) debe ir en el interior,
b) Lossy (con pérdidas) debe ir en los extremos.

En la Fig.J.1, X representa a cualquier señal, audio, imagen o video, mientras que $\hat{X}$ es el recuperado de la misma, el cual se espera que con las menores pérdidas adicionales a las ya generadas por el codec empleado por la norma.

En la Fig.J.1(a) podemos observar que si el catalizador en sin pérdidas debe ir luego del encoder en el transmisor y antes del decoder en el receptor, sino la distorsión sería inmanejable, mientras que en la Fig.J.1(b) podemos observar que si el catalizador es con pérdidas debe ir antes del encoder en el transmisor y luego del decoder en el receptor. En el caso a), el catalizador actúa como catalizador del proceso, mientras que en el caso b), lo hace como catalizador del encoder en sí.

Otro concepto que cobra fundamental protagonismo en el contexto de esta tecnología es el de Televisión Directo al Hogar (TDH), el cual implica la intervención satelital, y el cual es ideal para el empleo de catalizadores de compresión que permitan un mejor aprovechamiento del ancho de banda, permitiendo la transmisión de paquetes con más señales, aumentando así la oferta de canales en el hogar.

Por último, abordaremos el concepto de enlace satelital punto a punto mediante la norma europea DVB-S2.

J.2 Empleo de los enlaces satelitales: "Posibilidades reales"

La Fig.J.2 muestra las posibilidades de los enlaces satelitales, a saber:

1. Enlace de radiodifusión
2. Enlace de distribución
3. Enlace de contribución.

Figura J.2: Enlaces satelitales.

De acuerdo al tipo de enlace, se requieren diferentes recursos de tierra, a los efectos de darles viabilidad técnica. La Fig.J.3 muestra un típico rack de dispositivos de encoder y subida típica a satélite.

Figura J.3: Rack con equipos para encoder y MUX previa a la subida satelital.

Dichos enlaces son sensiblemente afectados por diferentes factores, los cuales se hacen particularmente evidentes en DVB-S, ver Fig.4. Ellos son:

1. La no-linealidad de los amplificadores de potencia
2. Los distintos tipos de reuido y la temperatura de antena
3. Los errores IQ

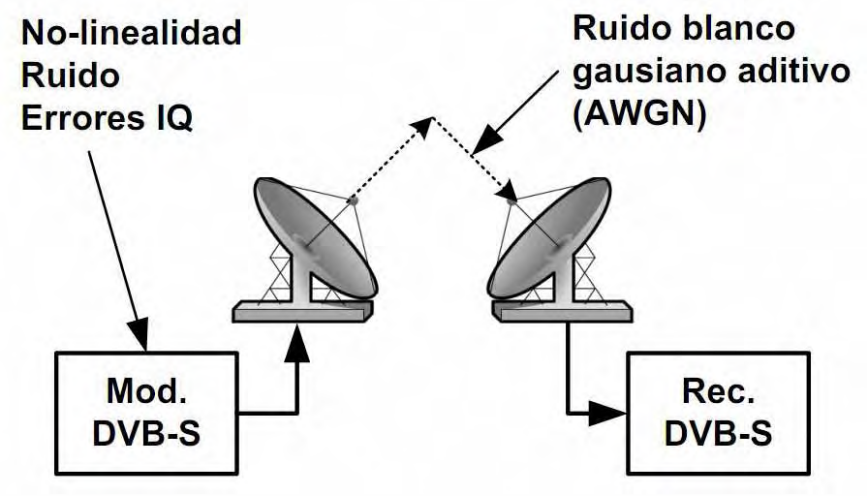


Figura J.4: Influencias que afectan la transmisión por satélite.

El no control de dichas influencias inutiliza todas las posibilidades de las conexiones punto a punto, multipunto y todos sus servicios asociados. La Fig.J.5 muestra una conexión típica punto a punto digital, mientras que la Fig.J.6 hace lo propio con una conexión típica multipunto digital. En este tipo de transmisiones digitales se ponen en evidencia como en ningunas otras los errores que se generan en la recuperación de los datos por las influencias nocibas fuera de control.

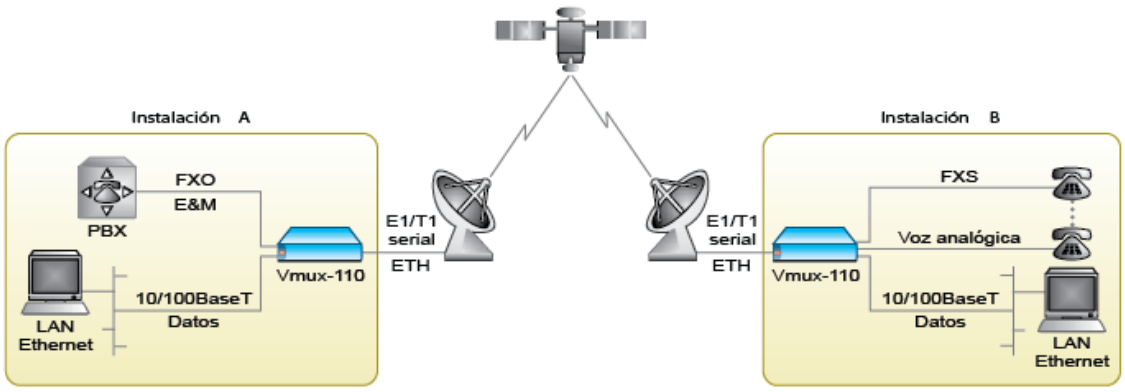


Figura J.5: Conexión punto a punto típica satelital.

Estas influencias nocibas castigan la calidad de transmisión y recuperación, tanto en banda baja como en banda alta (ver Fig.J.7), por lo cual debe hacerse particular incapié en las zonas de emplazamiento de las antenas. Estos aspectos se potencian dependiendo de ciertas características de la norma empleada en la transmisión, o sea, DVB-S o DVB-S2.

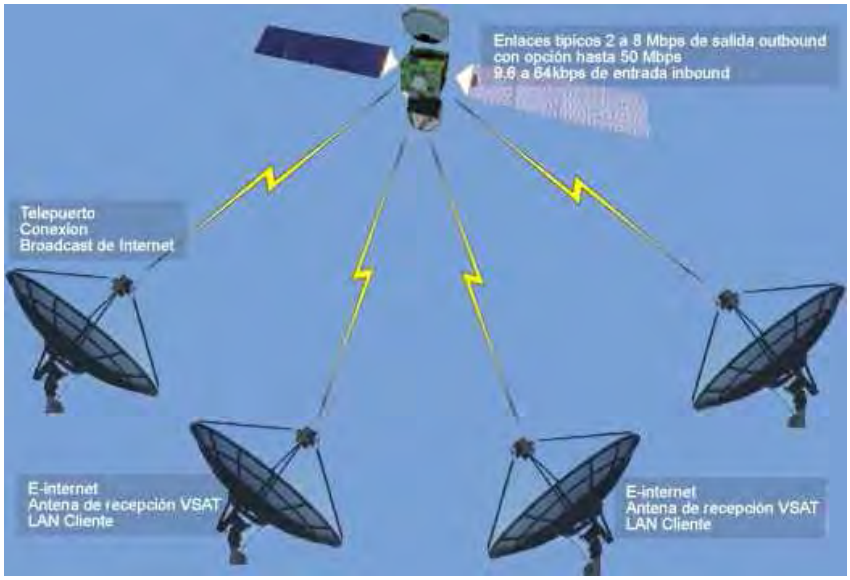


Figura J.6: Conexión multipunto típica satelital.

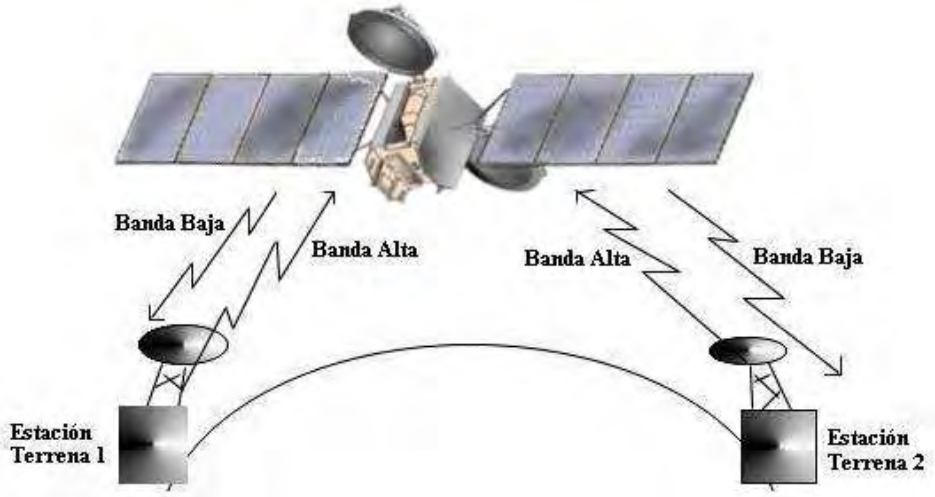


Figura J.7: Banda baja y banda alta de una conexión punto a punto satelital.

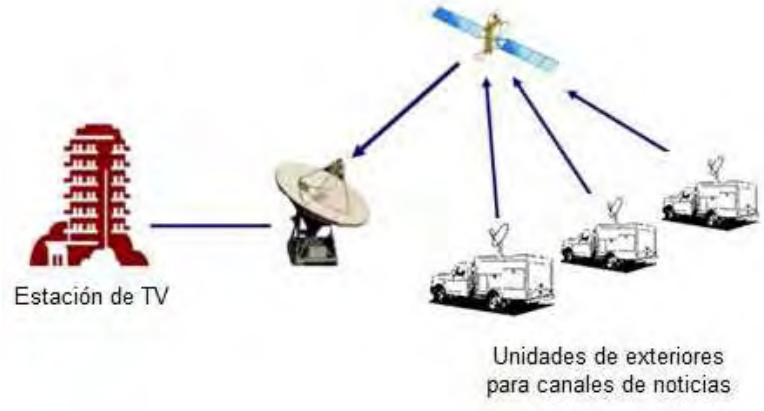


Figura J.8: Enlaces con unidades móviles itinerantes.

La Fig.J.8 es testigo de innumerables problemas de enlace y concurrencia los cuales tienen lugar dada la necesidad de acceso simultáneo a una plataforma satelital que emplea DVB-S. El problema reside en el ancho de banda de los transponders bajo norma DVB-S, los cuales poseen serias limitaciones al respecto en la actual coyuntura de comunicaciones. Por lo tanto, en la actualidad se está realizando a pie firme la migración de DVB-S a DVB-S2, el cual se esta instalando rápidamente en el acervo de la comunidad técnica que emplean los mencionados enlaces, ver Fig.J.9.

Figura J.9: DVB-S2.

Esta relativamente nueva norma, permite como nunca antes un aprovechamiento notable de los transponders satelitales, con un uso racional de los anchos de bandas de los mismos. Y no solo eso, como ninguna otra norma antes es particularmente afin con el empleo de catalizadores de compresión, permitiendo mejorar el aprovechamiento de los transponders, y así lograr multiplicar las señales, y de esta manera meter muchos más canales en el enlace satelital, ver Fig.J.10.

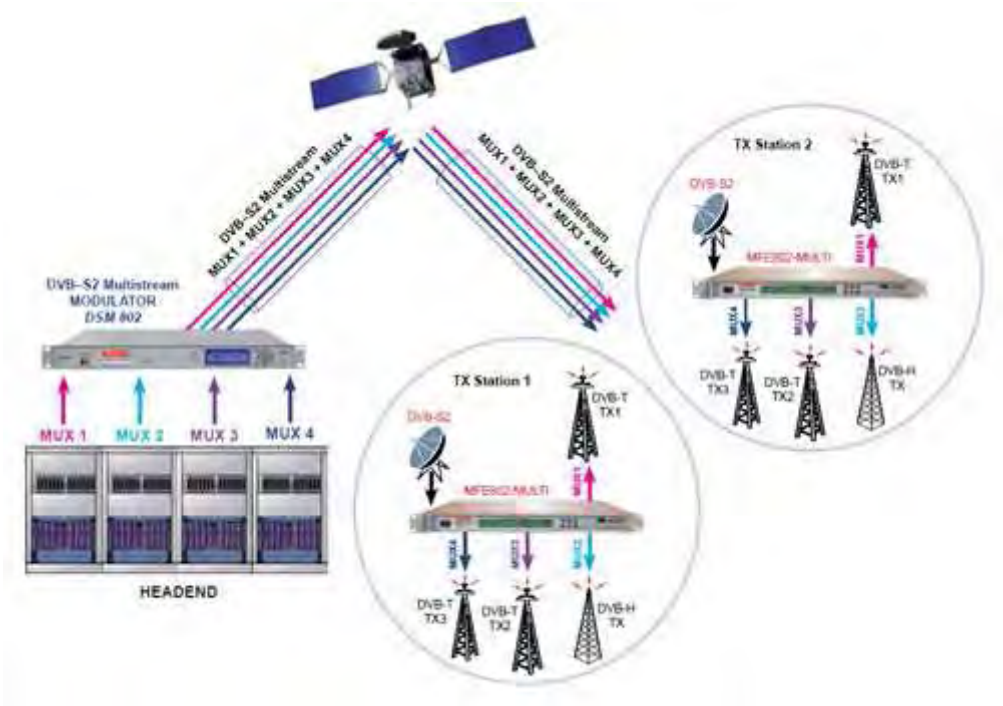


Figura J.10: Posibilidades múltiples y simultáneas con DVB-S2.

DVB-S2 ya nos permite disfrutar día a día de la frivolidad, el horror, la pasión y todo lo que sucede en el mundo en tiempo real y con un altamente superior uso de los transponders satelitales, ver Fig.J.11.

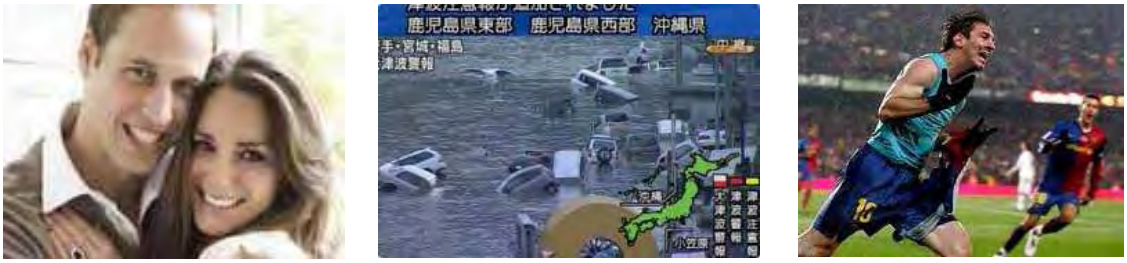


Figura J.11: Noticias y transmisiones en vivo, desde el lugar de los hechos.

El Julio de este año se podrá disfrutar por primera vez de una transmisión 3D en 576p y 720p via satélite (hasta el momento imposible con DVB-S) de la final de damas y caballeros de Wimbledon 2011, en las modalidades cable y TDH, Fig.J.12.

Figura J.12: Final de Wimbledon 2011 en vivo y 3D por cable y transmisión satelital.

Figura J.13: TDH para cubrir las extensas zonas no cubiertas por la TDT en nuestro país.

Esta tecnología permitirá cubrir vastas zonas del país no alcanzadas por las antenas del Sistama Argentini de TV Digital terrestre, ver Fig.J.13.

De esta manera se podrán sumar nuevas señales, así como las tradicionales, y sin dejar de incluir a señales del exterior, ver Fig.J.14.

Figura J.14: Señales nacionales e internacionales.

Por otra parte, permitirá abrir el juego de manera tal de generar más opciones en TDH a lo ya conocido, ver Fig.J.15.

Figura J.15: DirecTV.

Permitiendo actuar a nuevos actores en esta modalidad y fomentando la libre competencia, lo cual abarata los costos en la propuesta que llega al gran público, ver Fig.J.16.

Por otra parte, esta tecnología permite una recuperación automática de ciertos sectores del espectro para aplicaciones diversas, es decir, el tan mentado *efecto derrame* o *articulación horizontal*, por ejemplo para seguimiento y localización de vehículos y personas, en este último caso en un contexto biométrico y de seguridad física, ver Fig.J.17.

En otro orden de cosas, la tecnología DVB-S2 + Catalizador, permitirá romper con la dicotomía cable vs TDH (ver Fig.J.18), configurando más que nunca un espacio de configuración de complementación, de manera tal de usar cada tipo de servicio donde sea más útil y práctico, de manera tal de cubrir todo el territorio nacional con la conectividad necesaria y así no dejar un solo individuo a lo largo y ancho de la Nación sin la cobertura apropiada. La conectividad y acceso a los medios digitales es un derecho inalienable de los pueblos.

Figura J.16: Nuevos consignatarios gracias a DVB-S2 y la catalización de compresión.

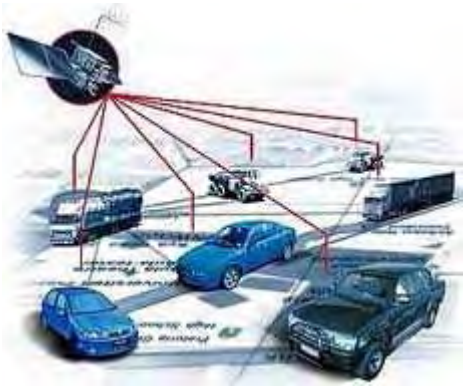


Figura J.17: Seguimiento vehicular.

Figura J.18: Enfrentamiento cable vs. TDH.

En nuestro país, la formulación de los requerimientos y especificaciones de la plataforma satelital de comunicaciones geoestacionaria (ver, Fig.J.19) .la realiza la Empresa Argentina de Soluciones Satelitales S.A. (AR-SAT) Fig.J.20.

Figura J.19: Plataforma satelital de comunicaciones geoestacionaria.

Figura J.20: Empresa Argentina de Soluciones Satelitales S.A. (AR-SAT).

Figura J.21: Telepuerto Benavidez, Pcia. de Bs. As..

Figura J.22: Investigación Aplicada S.E. (INVAP).

AR-SAT S.A. dispone de un moderno telepuerto situado en la localidad bonaerense de Benavidez (Fig.J.21) heredado de nahuelSAt. No obstante, el contratista elegido para la construcción de las futuras plataformas es la empresa del estado rionegrino y conocida como Investigación Aplicada S.E. (INVAP), Fig.J.22. INVAP está construyendo simultáneamente las dos primeras plataformas conocidas como AR-SAT 1 (Fig.J.23) y AR-SAT 2 (Fig.J.24), en sus instalaciones a las afueras de la ciudad de Bariloche.

Figura J.23: AR-SAT 1.

Figura J.24: AR-SAT 2.

Una vez en órbita (Fig. J.25) ambas plataformas e integrando la implementación adecuada de la combinación DVB-S2+Catalizador nos permitirán alcanzar 200 años después (Fig.J.26) la tan ansiada soberanía tecnológica en el espacio, las comunicaciones y los servicios audiovisuales.

Figura J.25: Orbita de las plataformas AR-SAT 1 y 2.

Figura J.26: Bicentenario soberanos tecnológicamente.

J.3 Conclusiones del apéndice

En este capítulo se describieron las potenciales aplicaciones del aporte del Capítulo 3 al caso de TDH y enlaces punto a punto satelitales, potenciando el empleo de la nueva norma europea DVB-S2, a los efectos de mejorar notablemente el uso del canal de comunicaciones satelital.

Glosario: Acrónimos

1080i	1080 líneas en exploración entrelazada (primero impares y luego pares)
1080p	1080 líneas en exploración progresiva (se exploran todas las líneas de una sola vez)
1D	Unidimensional
2D	Bidimensional
3D	Tridimensional
3DTV	3D TV
4k-3D	3840x2160p stereo
720p	720 líneas en exploración progresiva (se exploran todas las líneas de una sola vez)
8k-3D	7680x4320p stereo
AAC	Advanced Audio Coding
AC	Alternative Current
AR	Alta Resolución
AR-SAT	Empresa Argentina de Soluciones Satelitales S.A.
ARIB	Middleware de origen japonés que da lugar a aplicaciones para interactividad
ARM	Advanced RISC Machines
ATSC	Advanced Television System Committee
AVC	Advanced Video Coding
AWGN	Additive White Gaussian Noise
BCS	Block Coding System
BMP	BitMap file format
bpp	Bit-per-pixel
bps	Bit-per-second
BR	Baja Resolución
BST-OFDM	Band-segmented transmission-OFDM
CA	Conditional Access
CABAC	Código Aritmético Binario Adaptativo
CAVLC	Context-Adaptice VLC
CIELAB	Espacio de color
CIELUV	Espacio de color
COFDM	Coded OFDM
CR	Compression Rate
CC	Complejidad Computacional Codificación Cero
CCD	Charge-Coupled Device
CCITT	Comité Consultivo Internacional Telegráfico y Telefónico
CLC	Codificación por Longitud de Corrida
CPU	Central Processing Unit, Unidad Central de Procesamiento
CS	Codificación por Signo
CUDA	Compute Unified Device Architecture
DASE	Middleware de origen americano que da lugar a aplicaciones para interactividad
dB	decibeles
DC	Direct Current

DCT	Discrete Cosine Transform
DEMUX	Demultiplexor
DICOM	Digital Imaging and Communication in Medicine
DIFE	Diferencia
DFT	Discrete Fourier Transform
DM	Delta Modulation
DVB	Digital Video Broadcasting
DVB-S	DVB-by satellite
DVB-S2	DVB-by satellite versión 2
EBCOT	Embedded Block Coding with Optimised Truncation: Codificación por bloque embebido con Truncamiento Optimizado
EDP	Ecuación Diferencial Parcial
EOB	End of Block: Fin de Bloque
EOFDM	Enhanced OFDM
EPIC 617 28k	Norma de cine digital de 28000x9334 pixeles de resolución
EPIC 645 9k	Norma de cine digital de 9334x7000 pixeles de resolución
ETH	Ethernet
ETSI	European Telecommunications Standards Institute
FDCT	Forward DCT
FFT	Fast Fourier Transform
FFT^{-1}	Inverse FFT
FMO	Orden Flexible de Macrobloques
FPS	Frames-per-second
Full-HD	Full-HD: 1920x1080 pixels
FXO	Foreign Exchange Office
FXS	Foreign Exchange Service
GIF	Graphics Interchange Format: Formato de Intercambio de Gráficos
Ginga	Middleware de origen brasilero que da lugar a aplicaciones para interactividad
GOP	Group of Picture
GPGPU	General Purpose Graphics Processing Unit
GPU	Graphics Processing Unit
H.263	Codificador para videoconferencias
H.264	AVC
H.320	**U**mbrella recommendation by the ITU-T for running Multimedia (Audio / Video / Data) over ISDN based networks
HD	High Definition: 1280x720 pixels
HDMI	HD Multimedia Interface
HDTV	HD TV
HINT	Interpolación Jerárquica
HSI	Modelo de color
INVAP	Investigación Aplicada S.E.
IPTV	Internet Protocol TV
IRD	Integrated Reciver Decoder
ISDB-T	Integrated Services Digital Broadcasting-Terrestrial
ISDB-Ta	Integrated Services Digital Broadcasting-Terrestrial Argentine version
ISDB-Tb	Integrated Services Digital Broadcasting-Terrestrial Brazilian version
ISO	International Standarization Organization
ISO/IEC	ISO/Information Security Standard

ITU	International telecommunications Union
JBIG	Joint Bi-level Image Experts Group: Comité de Grupos de Expertos en Imágenes de dos niveles
JPEG	Joint Photographic Experts Group: Grupo de Expertos Enrolados en Fotográfía
JPEG-LS	JPEG - Lossless
JPEG2000	Nueva versión de JPEG basada en la TDO
JVC	Jpint Video Team
KSFM	Kohonen Self-organizing Feature Map
LAN	Local Area Network
LBG	Algoritmo de Linde- Buzo-Gray
LCD	Liquid Display Crystal
LD	Low Definition: 320x240 pixels
LLE	Linear Locally Embedded
LPS	Least Probable Symbol
LSB	Least Significant Bit
LSON	Lossless Services on Optical Networks
LUA	Lenguaje de scripting imperativo de NCL
LWC	Lossless Wavelet Coding
MAE	Mean Absolute Error
MAX	Máximo
MBAFF	Codificación de Video Entrelazado
MBPS	Mega Bits Per Seconds
MCU	Minimum Coded Unit: Unidad Mímima de Codificado
MD	Middle Definition
MHP	Middleware de origen europeo que da lugar a aplicaciones para interactividad
MP	Main Profile
MP4FF	MPEG-4 File Format
MPEG	Moving Picture Expert Group
MPEG2	MPEG2: Códec de video
MPEG4	MPEG4: Códec de video. Parte 4 es H.264 0 AVC
MPS	More Probeble Symbol
MQ	Codificación Aritmética
MSB	Most Significant Bit
MSE	Mean Squared Error: Error Cuadrático Medio
MSR	Módulo de Super-resolución
MUX	Multiplexor
NAL	Network Abstraction Layer
NCL	Nested Context Language
NHK	Nippon Hōsō Kyōkai: Corporación Emisora de Japón o Asociación de Radiodifusión de Japón
NIM	National Interactive Middleware
NL	Nonlocal
NLPS	Number LPS
NMPS	Number MPS
NP	Non Polynomial
OFDM	Orthogonal Frequency Division Multiplexing
One-seg	LD
OpenCL	Open Computing Language

OTT	Over-the-top content
PAFF	Codificación de video entrelazado usando codificación cuadro/campo adaptiva por imagen
PBX	Private Branch Exchange
PCM	Pulse Code Modulation
PNG	Portable Network Graphics: Gráficos Portables de Red
PNN	Algoritmo Pairwise Nearest Neighbour
PSF	Point Spread Function
PSNR	Peak SNR
RCC	Resolution Control Coding
RDP	Reduced Difference Pyramid
RGB	Red Green Blue: Rojo Verde Azul
RISC	Reduced Instruction Set Computer
RLC	Run Length Code
RLE	Run Length Encode
RM	Refinamiento de Magnitud
ROI	Regios of interest
RTA	Radio Televisión Argentina
RX	Receptor
SBTVD	Sistema Brasilero de TV Digital
SC	Supercompresión
SD	Standard Definition: 720x576 pixels
SDI	Serial Digital Interface
SIF	Standard Input Format
SNR	Signal-to-Noise Ratio
SP	Simple Profile
SR	Super-resolución
SRBE	SR basada en ejemplo
SRCMI	SR clásica multi-imagen
STB	Set-top-box
TDC	Transformada Discreta de Coseno
TDCA	Transformada Discreta de Coseno Adaptativa
TDH	Televisión Directa al Hogar
TDKL	Transformada Discreta de Karhunen-Loève
TDO	Transformada Discreta de Onditas
TDT	Televisión Digital Terrestre
TH	Transformada de Haar
TX	Transmisor
TS	Transformada de Slant
TSP	Travelling Salesman Problem: Problema del Viajante de Comercio
TV	Televisión
TVD	TV Digital
TWH	Transformada de Walsh-Hadamard
UHF	Ultra-High Frequency
SC	Super-compresión
SR	Super-resolución
VCEG	Video Coding Experts Group
VLC	Variable-Length Code: Código de Longitud Variable

VCL	Video Coding Layer
VLI	Variable-Length Integer: Entero de Longitud Variable
VLSI	Very Large Scale Integration: Integración de Muy Gran Escala
VP8	Ultimo códec de video de la empresa On2 Tecnologies
VRML	Virtual Reality Modeling Language
VT	Variación Total
VTB	Variación Total Bilateral
WebTV	TV on Web
WHT	Walsh-Hadamard Transform
YIQ	Es el espacio de color definido por la norma de TV color americana y conocida como National Television System Committee (NTSC)
YUV	Mapa de color derivado del RGB
ZRL	Zero Run Length

Bibliografía

[1] **Martín M.,** "Compresión de Imagen", Curso en el Laboratory of Mathematics in Imaging, Harvard University, Brigham and Women's Hospital, 4 de Mayo de 2004. Disponible en: http://lmi.bwh.harvard.edu/papers/pdfs/2003/martin-fernandezCOURSE03f.pdf

[2] **Jain A. K.**, *Fundamentals of Digital Image Processing,* Englewood Cliffs, New Jersey, 1989.

[3] **Gonzalez R. C., Woods R. E.**, Digital Image Processing, 2nd Edition, Prentice- Hall, Jan. 2002, pp.675-683.

[4] **Gonzalez R. C., Woods R. E., y Eddins S. L.,** Digital Image Processing using Matlab. Upper Saddle River, NJ: Pearson Prentice Hall, 2004.

[5] **Schalkoff R. J.**, Digital Image Processing and Computer Vision, Wiley, 1989, New York.

[6] **Pajares G. y de la Cruz J. M.**, Visión por computador, Alfaomega-Ra-Ma, 2002, México D. F.

[7] **Pajares G., de la Cruz J. M., Molina J. M., Cuadrado J. y López A.**, Imágenes Digitales: Procesamiento práctico con Java, Alfaomega-Ra-Ma, 2004, México D. F.

[8] **Faúndez Zanuy M.**, Tratamiento digital de voz e imagen y aplicación a la Multimedia, Marcombo, 2000, Barcelona.

[9] **Teuber J.**, Digital Image Processing, Prentice-Hall, 1993, New York.

[10] **Mastriani M.**, "*Union is Strength in Lossy Image Compression*," International Journal of Signal Processing, Volume 5, Number 2, pp.112-119, 2009. Disponible en: http://www.waset.org/ijsp/v5/v5-2-14.pdf

[11] **M. Mastriani and J. Gambini**, "Uso de la Transformada de Haar para el incremento de la eficiencia y rapidez de la Transformada Discreta de Karhunen-Loève en le compresión de imágenes con pérdidas," 11º Argentine Symposium on Tecnology (AST), 39 Jornadas Argentinas de Informática (39 JAIIO), Volume 1, pp.1563-1574, 30 de Agosto al 3 de Septiembre de 2010, Buenos Aires, Argentina.

[12] **M. Mastriani and J. Gambini**, "Fast Cosine Transform to increase speed-up and efficiency of Karhunen-Loève Transform for lossy image compression," International Journal of Engineering and Mathematical Sciences, Volume 6, Number 2, pp.82-92, 2010.

[13] **Linde, Y., Buzo, A., Gray, R.**, *An Algorithm for Vector Quantizer Design*, IEEE Transactions on Communications, vol. 28, pp. 84–94, 1980.

[14] **W. H. Equitz**, "A new vector quantization clustering algorithm", IEEE Transactions on Acoustics, Speech, and Signal Processing 37 (10), pp. 1568-1575, October 1989.

[15] **Haykin S.**, Neural Networks: A Comprehensive Foundation, Second Edition, Pearson, Delhi, 2005.

[16] -, Handbook of Neural Network Signal Processing, Editado por Yu Hen Hu y Jenq-Neng Hwang, CRC Press, Boca Raton, 2002.

[17] **Wallace G. K.,** The JPEG Still Picture Compression Standard. Communications of the ACM, 34(4):30-40, 1991.

[18] **Wallace G. K.**, The JPEG Still Picture Compression Standard, Submitted in December 1991 for publication in IEEE Transactions on Consumer Electronics. http://www.ijg.org/files/wallace.ps.gz

[19] **Wallace G. K.**, The JPEG Still Picture Compression Standard, *Communications of the ACM*, vol. 34, no. 4, pp. 30-44, April 1991.

[20] **Pennebaker W. B. y Mitchell J. L.**, "JPEG – Still Image Data Compression Standard," New York: International Thomsan Publishing, 1993.

[21] **Pennebaker W. B. y Mitchell J. L.**, "JPEG Tech. Specification," Revision 8. Informal Working paper JPEG-8-R8, Aug. 1990.

[22] **Pennebaker W. B. y Mitchell J. L.**, *JPEG: Still Image Compression Standard*. Kluwer Academic Publishers, 1993.

[23] **Miano J.,** Compressed Image File Formats: JPEG, PNG, GIF, XBM, BPM. Addison Wesley, 1999.

[24] **Léger, A., Omachi, T., y Wallace, G.** The JPEG still picture compression algorithm. In Optical Engineering, vol. 30, no. 7 (July 1991), pp. 947-954.

[25] **Briggs W. L. y Van Emden H.**, The DFT: An Owner's Manual for the Discrete Fourier Transform, SIAM, 1995, Philadelphia.

[26] **Ras Oliva E.**, Análisis de Fourier y Cálculo Operacional Aplicados a la Electro-tecnia, Marcombo, 1979, Barcelona.

[27] **Echebest N. y Liberman E.**, Resolución numérica de ecuaciones diferenciales parciales por aplicación de la transformada rápida de Fourier, con estimación de errores, Métodos Numéricos en la Mecánica del Continuo, Marshall G. ed., pp. 59-69, 1978, EUDEBA, Buenos Aires.

[28] **Hernández Lerma O.**, Métodos de Fourier en la Física y la Ingeniería, Trillas, 1973, México.

[29] **Hsu H. P.**, Fourier Analysis, Simon & Schuster, 1970, New York.

[30] **Tolimieri R., An M. y Lu C.**, Algorithms for Discrete Fourier Transform and convolution, Springer Verlag, 1997, New York.

[31] **Tolimieri R., An M. y Lu C.**, Mathematics of multidimensional Fourier Transform Algorithms, Springer Verlag, 1997, New York.

[32] **Claveau F. y Poirier M.**, "Real time FFT based cross-covariance method for vehicle speed and length measurement using an optical sensor," ICSPAT 96, pp.1831-1835, Boston, MA, Oct.1996.

[33] **Jones H. W., Hein D. N. y Knauer S. C.**, "The Karhunen-Loeve, discrete cosine and related transforms via the Hadamard transform," Proc. Intl. Telemeter. Conf., pp. 87-98, Los Angeles, CA, Nov. 1978.

[34] **Khayam S. A.**, The Discrete Cosine Transform (DCT): Theory and Application Technical Report, WAVES-TR-ECE802.602, 2003. Revised February 2005. Disponible en: http://www.niit.edu.pk/~khayam/pdf/DCT_TR802.pdf

[35] **Strang G.**, "The Discrete Cosine Transform," SIAM Review, Volume 41, Number 1, pp.135-147, 1999.

[36] **Hung A. C. y Meng TH-Y**, "A Comparison of fast DCT algorithms," Multimedia Systems, No. 5 Vol. 2, Dec 1994.

[37] **Aggarwal G. y Gajski D. D.**, "Exploring DCT Implementations," UC Irvine, Technical Report ICS-TR-98-10, March 1998.

[38] **Blinn J. F.**, "What's the Deal with the DCT," IEEE Computer Graphics and Applications, July 1993, pp.78-83.

[39] **Chiu C. T. y Liu K. J. R.**, "Real-Time Parallel and Fully Pipelined 2-D DCT Lattice Structures with Application to HDTV Systems," IEEE Trans. on Circuits and Systems for Video Technology, vol. 2 pp. 25-37, March 1992.

[40] **Haque M. A.**, "A Two-Dimensional Fast Cosine Transform," IEEE Transactions on Acoustics, Speech and Signal Processing, vol. ASSP-33 pp. 1532-1539, December 1985.

[41] **Vetterli M.**, "Fast 2-D Discrete Cosine Transform," ICASSP '85, p. 1538.

[42] **Kamangar F. A. y Rao K. R.**, "Fast Algorithms for the 2-D Discrete Cosine Transform," IEEE Transactions on Computers, v C-31 p. 899.

[43] **Linzer E. N. y Feig E.**, "New Scaled DCT Algorithms for Fused Multiply/Add Architectures," ICASSP '91, p. 2201.

[44] **Loeffler C., Ligtenberg A., y Moschytz G.**, "Practical Fast 1-D DCT Algorithms with 11 Multiplications," ICASSP '89, p. 988.

[45] **Vetterli M., Duhamel P., y Guillemot C.**, "Trade-offs in the Computation of Mono- and Multi-dimensional DCTs," ICASSP '89, p. 999.

[46] **Duhamel P., Guillemot C., y Carlach J. C.**, "A DCT Chip based on a new Structured and Computationally Efficient DCT Algorithm," ICCAS '90, p. 77.

[47] **Cho N. I. y Lee S. U.**, "Fast Algorithm and Implementation of 2-D DCT," IEEE Transactions on Circuits and Systems, vol. 38 p. 297, March 1991.

[48] **Cho N. I., Yun I. D., y Lee S. U.**, "A Fast Algorithm for 2-D DCT," ICASSP '91, p. 2197-2220.

[49] **McMillan L. y Westover L.**, "A Forward-Mapping Realization of the Inverse DCT," DCC '92, p. 219.

[50] **Duhamel P. y Guillemot C.**, "Polynomial Transform Computation of the 2-D DCT," ICASSP '90, p. 1515.

[51] **Saha S.**, "Image Compression - from DCT to Wavalets: A Review," *ACM Cross- roads Student Magazine*, vol. 6.3, Spring 2000. http://www.acm.org/crossroads/ Disponible en: http://www.upatras.gr/ieee/skodras/pubs/ans-c37.pdf

[52] **Tjoa S., Lin W. S. and Ray Liu K. J.**, "Transform coder classification for digital image forensics". Disponible en: http://www.cspl.umd.edu/sig/publications/tjoa_ICIP_200709.pdf

[53] **Preisendorfer R. W.**, Principal Component Analysis in Meteorology and Oceanography, Elsevier, Amsterdam, 1988.

[54] **Jollife I. T.**, Principal Component Analysis, 2/e, Springer-Verlag, New York, 2002.

[55] **Diamantaras K. I. y Kung S. Y.**, Principal Component Neural Networks, Wiley, New York, 1996.

[56] **Rao C. R.**, "The Use and Interpretation of Principal Component Analysis in Applied Research," Sankhya, **26**, 329 (1964).

[57] **Jolicoeur P. y Mosimann J. E.**, "Size and Shape Variation in the Painted Turtle: A Principal Component Analysis," Growth, **24**, 339 (1960).

[58] **Bretherton C. S., Smith C., y Wallace J. M.**, "An Intercomparison of Methods for Finding Coupled Patterns in Climate Data," J. Climate, **5**, 541 (1992).

[59] **Hadi A. S. y Ling R. F.**, "Some Cautionary Notes on the Use of Principal Components Regression," Amer. Statistician, **52**, 15 (1998).

[60] **Naik D. N. y Khattree R.**, "Revisiting Olympic Track Records: Some Practical Considerations in the Principal Component Analysis," Amer. Statistician, **50**, 140 (1996).

[61] **Hotelling H.**, "Analysis of a Complex of Statistical Variables into Principal Components," J. Educ. Psychol., **24**, 417 (1933).

[62] **Hotelling H.**, "The Most Predictable Criterion," J. Educ. Psychol., **26**, 139 (1935).

[63] **Kumar P. S. y Prabhu K. M. M.**, "A special case for the KLT of Markov-1 Process," IEEE Trans. SP (Under review).

[64] **Chan Y. H.**, "On the substitution of the Karhunen-Loeve transform," IEEE Trans. SP (Under review).

[65] **Fleury M., Self R. P., y Downton A. C.**, Development of a Fine-grained Parallel Karhunen-Loève Transform.
Disponible en: http://privatewww.essex.ac.uk/~fleum/jpdsPapv3.pdf

[66] Transformada de Karhunen-Loève (KLT). Disponible en:
http://www.diac.upm.es/acceso_profesores/asignaturas/tdi/tdi/transformadas/pdf/karhunen.pdf

[67] **Dony A. C.**, "Karhunen-Loève Transform", The Transform and Data Compression Handbook, CRC Press, 2001.

[68] **Nakagawa M. y Miyahara M.**, "Generalized Karhunen-Loeve transformation- I: Theoretical Consideration," IEEE Trans. Commun. Vol. C-35, pp. 215-223, Feb. 1987.

[69] **Ziyal N. A. *et al***, "Image representation, recovery and analysis using principal component analysis," ICSPAT 97, San Diego, CA, Sept. 1997.

[70] **Hamilton D. J., Sandham W. A. y Blanco A.**, "Electrocardiogram data compression using non-linear principal component analysis," IEEE SP Letters (in print).

[71] **Kitter J. y Young P. C.**, "A new approach to feature selection based on the Karhunen-Loeve expansion," Pattern Recognition, Vol. 5, pp. 335-352, 1973.

[72] **Chen C. C. T., Chen C. T. y Tsai C. M.**, "Karhunen-Loeve transform for text independent speaker recognition," 1997 Intl. Symp. on Communications, Hsinchu, Taiwan, Dec. 1997.

[73] **Chen C. S. y Huo K. S.**, "Karhunen-Loeve method for data compression and speech synthesis," IEE Proc., Vol. 138, pp. 377-380, Oct. 1991.

[74] **Mudugamuwa D. J. y Bradley A. B.**, "Optimal transform for segmented parametric speech coding," ICASSP 98, pp. 53-56, Seattle, WA, May 1998.

[75] **Chouikha M. F., Gilmore E. T. y Ziyad N.,** "Adaptive principal component extraction (APEX) for image compression," ICSPAT 98, Toronto, Canada, Sept. 1998.

[76] **Tsapatsoulis N., Alexopoulos V. y Kollias S.**, "A vector based approximation of KLT and its application to face recognition," EUSIPCO-98, vol. 3, pp. 1581-1585, Island of Rhodes, Greece, Sept.1998.

[77] **Ziyad N., Gilmore E. T. y Chouikha M. F.**, "Improvements for image compression using adaptive principal component extraction," 32nd Asilomar Conf. on Signals, Systems, and Computers, Pacific Grove, CA, Nov. 1998.

[78] **Swets D. L. y Weng J.**, "Using discriminant eigenfeatures for image retrieval," IEEE Trans. PAMI, vol.18, pp.831-836, Aug. 1996.

[79] **Tang X. y Stewart W. K.**, 'Texture classification using principal component analysis techniques," Proc. of SPIE, vol. 2315, pp. 22-35, 1994.

[80] **Rao K. R.**, Chapter 3 Optimal Decorrelation and the KLT. Disponible en: http://www-ee.uta.edu/Dip/Courses/EE5355/ch3.pdf

[81] **Palacios A., Gunaratne G. H., Gorman M. y Robbins K. A.**, Karhunen-Loève analysis of spatiotemporal flame patterns, Physical Review E Vol 57, Number 5 May 1998, pp.5958-5971.

[82] **Kitajima H. y Shimone T.**, "Some aspects of the fast Karhunen-Loeve transform," IEEE Trans. Commun., Vol. 28, pp. 1773-1776, Sept. 1980.

[83] **Algazi V. R. y Sakrison D. J.**, "On the optimality of Karhunen-Loeve expansion," IEEE Trans. IT, Vol. IT-15, pp.319-321, March 1969.

[84] **Kermani B. G., Schiffman S. S., y Nagle H. T.**, "A Novel Method for Reducing the Dimensionality in a Sensor Array," IEEE Trans. Instr. Meas. **IM-47**, 728 (1998).

[85] **Howard P. G.,** The Design and Analysis of Efficient Lossless Data Compression Systems. PhD thesis, Department of Computer Science, Brown University, 1993.

[86] **Mongkolworaphol S., Rangsanseri Y. y Thitimajshima P.,** Multispectral Image Compression using FCM-Based Vector Quantization. Disponible en: http://www.gisdevelopment.net/aars/acrs/2000/ts9/imgp0011.asp

[87] **Roth P. M. y Winter M,** "Survey of Appearance-Based Methods for Object Recognition", Technical Report ICG-TR-01/08, Graz, January 15, 2008. Disponible en: http://web.mit.edu/~wingated/www/introductions/appearance_based_methods.pdf

[88] **Burl J. B**, "Estimating the basis functions of the Karhunen-Loeve transform," IEEE Trans. ASSP, Vol. 37, pp.99-105, Jan. 1989.

[89] **Musatenko Y. S. y Kurashov V. N.**, "Nonlinear improving of Karhunen-Loeve bases obtained by approximate 2D procedures," IS&T/SPIE's 9th Annual Symp., Electronic Imaging, Vol. 3026, San Jose, CA, Feb. 1997.

[90] **Kirby M. y Sirovich L.**, "Application of the Karhunen-Loève procedure for the characterization of human faces," IEEE Trans. Pattern Anal. Machine Intell, vol. 12, pp. 103-108, 1990.

[91] **Phoong S. M. y Lin Y. P.**, "PLT versus KLT," ISCAS99, pp.15-19, Orlando, FL, May-June 1999.

[92] **Lee J.**, "Optimized quadtree for Karhunen-Loeve transform in multispectral image coding," IEEE Trans. IP, vol. 8, pp. 453-461, April 1999.

[93] **Dony R. D. y Haykin S.**, "Optimally adaptive transform coding" IEEE Trans. IP, Vol. 4, pp. 1358-1370, 1995.

[94] **Epsein B. R. *et al.***, "Multispectral KLT-wavelet data compression for Landsat thematic mapper images, DCC, pp. 200-208, March 1992.

[95] **Kongkchandra R., Tamee K., y Kimpan C.**, "Improving Thai isolated word recognition by using Karhunen-Loeve transformation and learning vector quantization", IEEE ISPACS'99, Pukhet, Thailand, Dec. 1999.

[96] **Kongkchandra R., Tamee K., y Kimpan C.**, "Using Karhunen-Loeve transformation for feature reduction and tones analysis in Thai harmonic frequency speech", ", IEEE ISPACS'99, Pukhet, Thailand, Dec. 1999.

[97] **Lahme B. y Miranda R.**, "Karhunen-Loeve decomposition in the presence of symmetry-part I," IEEE Trans IP, vol. 8, pp. 1183-1190, Sept. 1999.

[98] **Tanaka T. y Yamashita Y.**, "Image coding using vector embedded Karhunen-Loeve transform," IEEE ICIP, Kobe, Japan, Oct. 1999.

[99] **Davila C. E.**, "Blind KLT coding and vector quantization," IEEE 9th DSP Workshop, Hunt, TX, Oct. 2000.

[100] **Dony R. D.**, "Karhunen-Loeve transform," Ch. 1 in the transform and data compression handbook (Eds. K.R. Rao and P.C. Yip), Boca Raton, FL: CRC Press, 2001.

[101] **Kongkchandra R., Tamee K., y Kimpan C.**, "Improving Thai isolated word recognition by using Karhunen-Loeve transformation and learning vector quantization", IEEE ISPACS'99, Pukhet, Thailand, Dec. 1999.

[102] **Kongkchandra R., Tamee K., y Kimpan C.**, "Using Karhunen-Loeve transformation for feature reduction and tones analysis in Thai harmonic frequency speech", IEEE ISPACS'99, Pukhet, Thailand, Dec. 1999.

[103] **Ray W. D. y Driver R. M**, "Further decomposition of the Karhunen-Loeve series representation of a stationary random process", IEEE Trans. IT, vol. IT-16, pp. 663-668, Nov. 1970.

[104] **Jain A. K.**, "A fast Karhunen-Loeve transform for recursive filtering of images corrupted by white and colored noise," IEEE Trans. Comput. Vol. C-26, pp. 560-571, June 1977.

[105] **Jain A. K.**, "A fast Karhunen-Loeve transform for a class of random processes," IEEE Trans. Commun. Vol. COM-24, pp. 1023-1029, Sept. 1976

[106] **Epstein B. R., *et al***, "Multispectral KLT-wavelet data compression for landsat thematic mapper images," In *Data Compression Conference*, pp. 200-208, Snowbird, UT, March 1992.

[107] **Lee J.**, "Optimized quadtree for Karhunen-Loève Transform in multispectral image coding," *IEEE Transactions on Image Processing*, 8(4), pp.453-461, 1999.

[108] **Saghri J. A., *et al***, "Practical Transform coding of multispectral imagery," *IEEE Signal Processing Magazine*, 12, pp.32-43, 1995.

[109] **Kim T-S., *et al***, "Multispectral image data compression using classified prediction and KLT in wavelet transform domain," IEICE Transactions on Fundam Electron Commun Comput Sci, Vol**.** E86-A; No**.**6, pp**.**1492-1497, 2003.

[110] **Saghri J., Tescher A., y Reagan J.**, "Practical Transform Coding of Multispectral Imagery", IEEE Signal Processing Magazine, Vol.12, No.1, January 1995.

[111] **Hunt B. R. y Kubler O.**, "Karhunen-Loeve multispectral image restoration, part 1: Theory," IEEE Trans. ASSP, Vol. ASSP-32, pp. 592-600, June 1984.

[112] **Epstein B. R., *et al***, "Multispectral KLT-wavelet data compression for LANDSAT thematic mapper images," Proc. DCC, Mar. 1992.

[113] **Mastriani M.**, "Denoising and compression in wavelet domain via projection onto approximation coefficients," International Journal of Signal Processing, Volume 5, Number 1, pp.20-30, 2008.

[114] **Mastriani M.**, Nuevas Técnicas Computacionales para el Mejoramiento de Imágenes Médicas, *31 Jornadas Argentinas de Informática e Investigación Operativa*, Santa Fe, Argentina, 9 al 13 de Septiembre de 2002, pp.262-267.

[115] **Mastriani M.**, Imágenes Médicas sobre TCP/IP y UDP: Su problemática de implementación, *INFORMEDICA'2002 - e-Salud Global*, 30 de Octubre al 30 de Noviembre de 2002.

[116] **Mastriani M.**, New wavelet-based superresolution algorithm for speckle reduction in SAR images, International Journal of Computer Science, Volume 1, Number 4, pp.291-298, 2006.

[117] **Mastriani M. y Giraldez A. E.**, Despeckling of SAR images in wavelet domain, GIS Development Magazine, Sept. 2005, Vol. 9, Issue 9, pp.38-40. Disponible en: http://www.gisdevelopment.net/magazine/years/2005/sep/wavelet_1.htm

[118] **Mastriani M. y Giraldez A. E.**, Kalman's shrinkage for wavelet-based despeckling of SAR images, International Journal of Intelligent Technology, Volume 1, Number 3, pp.190-196, 2006. Disponible en: http://www.enformatika.org/jit/v1/v1-3-21.pdf

[119] **Mastriani M. y Giraldez A.**, Enhanced Directional Smoothing Algorithm for Edge-Preserving Smoothing of Synthetic-Aperture Radar Images, *International Journal of*

Measurement Science Review, vol 4, no. 3, pp.1-11, 2004. Disponible en: http://www.measurement.sk/2004/S3/Mastriani.pdf

[120] **Mastriani M. *et al***, Sistema Inteligente de Adquisición de Señales Biomédicas y su Vinculación con una Historia Clínica Electrónica, *33 Jornadas Argentinas de Informática e Investigación Operativa*, Córdoba, Argentina, 22 al 24 de Septiembre de 2004.

[121] **Mastriani M. *et al***, Evaluation of Children's Anthropometric Features Using Wavelet Decomposition, *9th World Congress of Medicine in Internet (MEDNET 2004) of Society for the Internet in Medicine (SIM)*, (8113), Buenos Aires, Argentina, December 5-7, 2004.

[122] **Mastriani M.**, Denoising based on wavelets and deblurring via self-organizing map for Synthetic Aperture Radar images, International Journal of Signal Processing, Volume 2, Number 4, pp.226-235, 2005. Disponible en: http://www.enformatika.org/ijsp/v2/v2-4-33.pdf

[123] **Mastriani M.**, "Denoising based on wavelets and deblurring via self- organizing map for Synthetic Aperture Radar images," ICGST International on Journal of Artificial Intelligence and Machine Learning, Volume 5, 2005.

[124] **Mastriani M., y Giraldez A.**, "Smoothing of coefficients in wavelet domain for speckle reduction in Synthetic Aperture Radar images," ICGST International Journal on Graphics, Vision and Image Processing (GVIP), Volume 6, pp. 1-8, 2005. Disponible en: http://www.icgst.com/gvip/v6/P1150517003.pdf

[125] **Mastriani M.**, "Systholic Boolean Orthonormalizer Network in Wavelet Domain for Microarray Denoising," ICGST International Journal on Bioinformatics and Medical Engineering, Volume 5, 2005.

[126] **Mastriani M.**, "Systholic Boolean Orthonormalizer Network in Wavelet Domain for Microarray Denoising," International Journal of Signal Processing, Volume 2, Number 4, pp.273-284, 2005.

[127] **Mastriani M., y Giraldez A.**, "Microarrays denoising via smoothing of coefficients in wavelet domain," International Journal of Biomedical Sciences, Volume 1, Number 1, pp.7-14, 2006.

[128] **Mastriani M., y Giraldez A.**, "Kalman' Shrinkage for Wavelet-Based Despeckling of SAR Images," International Journal of Intelligent Technology, Volume 1, Number 3, pp.190-196, 2006.

[129] **Mastriani M., y Giraldez A.**, "Neural Shrinkage for Wavelet-Based SAR Despeckling," International Journal of Intelligent Technology, Volume 1, Number 3, pp.211-222, 2006.

[130] **Mastriani M.**, "Fuzzy Thresholding in Wavelet Domain for Speckle Reduction in Synthetic Aperture Radar Images," International Journal of Intelligent Technology, Volume 1, Number 3, pp.252-265, 2006.

[131] **Stollnitz E. J., DeRose T. D., y Salesin D. H.,** Wavelets for computer graphics: A primer, Part I. IEEE Computer Graphics and Applications, 15(3):76–84, May 1995. Disponible en: http://grail.cs.washington.edu/pub/stoll/wavelet1.pdf

[132] **Stollnitz E. J., DeRose T. D., y Salesin D. H.,** Wavelets for computer graphics: A primer, Part II. IEEE Computer Graphics and Applications, 15(4), July 1995. In press. Disponible en: http://grail.cs.washington.edu/projects/wavelets/article/wavelet2.pdf

[133] **Berman D., Bartell J., y Salesin D.,** Multiresolution painting and compositing. In Proceedings of SIGGRAPH 94, pages 85–90. ACM, New York, 1994.

[134] **Beylkin G., Coifman R., y Rokhlin V.,** Fast wavelet transforms and numerical algorithms I. Communications on Pure and Applied Mathematics, 44(2):141–183, March 1991.

[135] **Christensen P. H., Lischinski D., Stollnitz E. J., y Salesin D. H.,** Clustering for glossy global illumination. ACM Transactions on Graphics, Vol. 16, Issue 1, pp. 3–33, January 1997.

[136] **Christensen P. H., Stollnitz E. J., Salesin D. H., y DeRose T. D.,** Wavelet radiance. In G. Sakas, P. Shirley, and S. Müller, editors, Photorealistic Rendering Techniques, pages 295–309. Springer-Verlag, Berlin, 1995.

[137] **Daubechies I.,** Orthonormal bases of compactly supported wavelets. Communications on Pure and Applied Mathematics, 41(7):909–996, October 1988.

[138] **DeVore R., Jawerth B., y Lucier B.,** Image compression through wavelet transform coding. IEEE Transactions on Information Theory, 38(2):719–746, March 1992.

[139] **Finkelstein A. y Salesin D. H.,** Multiresolution curves. In Proceedings of SIGGRAPH 94, pages 261–268. ACM, New York, 1994.

[140] **Gortler S. J. y Cohen M. F.,** Hierarchical and variational geometric modeling with wavelets. In Proceedings of the 1995 Symposium on Interactive 3D Graphics, pages 35–42. ACM, New York, 1995.

[141] **Gortler S. J., Schröder P., Cohen M. F., y Hanrahan P.,** Wavelet radiosity. In Proceedings of SIGGRAPH 93, pages 221–230. ACM, New York, 1993.

[142] **Jacobs C. E., Finkelstein A., y Salesin D. H.,** Fast multiresolution image querying. In Proceedings of SIGGRAPH 95, pages 277–286. ACM, New York, 1995.

[143] **Liu Z., Gortler S. J., y Cohen M. F.,** Hierarchical spacetime control. In Proceedings of SIGGRAPH 94, pages 35–42. ACM, New York, 1994.

[144] **Lounsbery M., DeRose T., y Warren J.**. Multiresolution surfaces of arbitrary topological type. ACM Transactions on Graphics, 16 (3):34-73, 1997.

[145] **Mallat S.,** A theory for multiresolution signal decomposition: The wavelet representation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 11(7):674–693, July 1989.

[146] **Meyers D.,** Multiresolution tiling.Computer Graphics Forum, 13(5):325–340, December 1994.

[147] **Schröder P., Gortler S. J., Cohen M. F., y Hanrahan P.,** Wavelet projections for radiosity. Computer Graphics Forum, 13(2):141–151, June 1994.

[148] **Stollnitz E. J., DeRose T. D., y Salesin D. H.,** Wavelets for Computer Graphics: Theory and Applications. Morgan Kaufmann, San Francisco, 1996.

[149] **Shen J. y Strang G.**, The zeros of the Daubechies polynomials. *Proc. Amer. Math. Soc.,* 1996.

[150] **Chamberlain N. F.**, Introduction to wavelets v1.7, December 2002.
Disponible en: http://www.sdsmt.edu/syseng/ee/courses/ee624/Wavelets1.7.pdf

[151] **Mallat S. G.**, Multiresolution approximations and wavelet orthonormal bases of L2 (R). *Transactions of the American Mathematical Society*, 315(1), pp.69-87, 1989a.

[152] **Grossman A. y Morlet J.**, Decomposition of Hardy Functions into Square Integrable Wavelets of Constant Shape. *SIAM J. App Math*, 15: pp.723-736, 1984.

[153] **Meyer Y.**, Wavelets: Algorithms & applications, SIAM, 1994, Philadelphia.

[154] **Valens C.**, A really friendly guide to wavelets. Disponible en:
http://perso.wanadoo.fr/polyvalens/clemens/wavelets/wavelets.html (Actualizado hasta el 12 de Abril de 2001)

[155] **Burrus C. S., Gopinath R. A. y Guo H.**, Introduction to Wavelets and Wavelet Transforms: A Primer. Prentice Hall, New Jersey, 1998.

[156] **Castro L. R. y Castro S. M.**, Wavelets y sus aplicaciones, *Primer Congreso Argentino de Ciencias de la Computación*, Univ. Nac. Del Sur, 5 al 7 de Octubre de 1995, pp.195-204.

[157] **Daubechies I.**, Different Perspectives on Wavelets. *Proceedings of Symposia in Applied Mathematics*, Vol. 47, American Mathematical Society, United State of America, 1993.

[158] **Walker J. S.**, A Primer on Wavelets and their Scientific Applications. Chapman & Hall/CRC, New York, 1999.

[159] **Daubechies I.**, Ten Lectures on Wavelets, SIAM, Philadelphia, 1992.

[160] **Walker J. S.**, Combined image compressor and denoiser based on tree-adapted wavelet shrinkage, Revision of OE 010337. James S. Walker. Department of Mathematics, University of Wisconsin-Eau Claire.

Disponible en: http://www.uwec.edu/walkerjs/media/CompDen.pdf

[161] **Stollnitz E. J., DeRose T. D., y Salesin D. H.**, Wavelets for Computer Graphics (Theory and Applications). Morgan Kaufmann Publishers, San Francisco, 1996

[162] **Kaiser G. A.**, Friendly Guide To Wavelets, Birkhäuser, Glen Allen, Virginia, 1994.

[163] **-.**, Applied and numerical harmonic analysis, Birkhäuser, C. E. D'Attellis, E. M. Fernández-Berdaguer eds., Boston, 1997.

[164] **Rao R. R. y Bopardikar A. S.**, Wavelet transforms: Introduction to theory and applications, Addison-Wesley Longman, Inc., 1998.

[165] **Hubbard B. B.**, The World According to Wavelets (The Story of a Mathema-tical Technique in the Making), A. K. Peter Wellesley, Massachusetts, 1996.

[166] **Andreopoulos Y., Munteanu A., Van der Auwera G., Cornelis J. y Schelkens P.**, Complete-to-Overcomplete discrete wavelet transforms: Theory and Applications, *IEEE Transactions on Signal Processing*, Vol. 53, No. 4, pp. 1398-1412, April 2004.

[167] Ricoh Innovations, Inc., "Compression with Reversible Embedded Wavelets (CREW) Algorithm." http://www.crc.ricoh.com/CREW/ (as of 01.10.02).

[168] **Misiti M., Misiti Y., Oppenheim G., y Poggi J. M.**. (2005, March). Wavelet Toolbox, for use with MATLAB®, User's guide, version 3. Disponible en: http://www.mathworks.com/access/helpdesk/help/pdf_doc/wavelet/wavelet_ug.pdf

[169] **Strang G.**, Creating and comparing wavelets, *Numerical Analysis:* A. R. Mitchell Birthday Volume, G. A. Watson and D. F. Griffiths, eds., World Scien-tific, 1996.

[170] **Ying L., Ranganath S. y Zhou X.**, Comparison of wavelet and cosine basis for representation of arbitrarily shaped image segments, *Proceedings of Internatio-nal X. Conference on Telecommunications* (ICT 2002), June 2002, Beijing, China.
Disponible en: http://www.itee.uq.edu.au/~zxf/_papers/ICT02.pdf

[171] **Senecal J. G., Duchaineau M. A. y Joy K. I.**, Reversible N-Bit to N-Bit Integer Haar-Like Transforms, *Proceedings of the 7th IASTED International Conference on Computer Graphics and Imaging*, edited by M. H. Hamza. pp 135-140, August 2004. Also available as Lawrence Livermore National Labora-tory technical report UCRL-CONF-202451-REV-1. Disponible en:
http://graphics.cs.ucdavis.edu/~jgseneca/Papers/TLHaar_CGIM2004.pdf

[172] **Mulcahy C.**, Plotting and scheming with wavelets, *AMS Mathematics Magazine,* Vol. 69, No. 5, pp.323-343, December 1996.

[173] **Zhong S. y Cherkassky V.**, Image denoising using wavelet thresholding and model selection, *Proc. IEEE Int. Conf. on Image processing,* Vancouver, BC, Canada, November, 2000.
Disponible en: http://www.ece.utexas.edu/~szhong/papers/denoise.pdf

[174] **Darian Muresan D. y Parks T. W.**, Adaptive principal components and image denoising, *IEEE International Conference of Image Processing*, Barcelona, Spain, pp.101-104, 2003.
Disponible en: http://dsplab.ece.cornell.edu/papers/slides/icip2003.pdf

[175] **Donoho D. L., Johnstone I. M., Kerkyacharian G. y Picard D.**, Wavelet shrin-kage: asymptopia, *Journal of Royal Stat. Soc.*, vol. 57, no.2, pp.301-369, 1995.

[176] **Donoho D. L., Johnstone I. M., Kerkyacharian G. y Picard D.**, Density estimation by wavelet thresholding, *Annals of Stat.*, vol. 24, pp. 508-539, 1996.

[177] **Donoho D. L.**, De-Noising by soft-thresholding, *IEEE Trans. on Inf. Theory*, vol. 41, no. 3, pp. 613-627, 1995.

[178] **Donoho D. L. y Johnstone I. M.**, Adapting to unknown smoothness via wavelet shrinkage, *Journal of the American Statistical Association*, vol. 90, no. 432, pp. 1200-1224, 1995.

[179] **Donoho D. L. y Johnstone I. M**, Ideal spatial adaptation by wavelet shrinkage. *Biometrik*a, 81, 425-455, 1994.

[180] **Coifman R. R. y Donoho D. L.**, Translation-invariant de-noising. Antoniadis, A., & Oppenheim, G. (eds), *Lecture Notes in Statistic*s, vol. 103. Springer-Verlag, pp. 125-150, 1995.

[181] **Gao H. Y. y Bruce A. G.**, WaveShrink with firm shrinkage, *Statistica Sinic*a, 7, 855-874, 1997.

[182] **Adams N., Chen J., Olafsson V. y Yip C.**, Denoising using wavelets: Wavelets & time frequency analysis, December 1991.

[183] **Efromovich S., Lakey J., Pereyra M. C. y Tymes N.**, Data-driven and optimal denoising of a signal and recovery of its derivative using multiwavelets, *IEEE Trans. on Signal Processing*, vol. 52, no. 3, pp.1-8, 2004.

[184] **Chang S. G., Yu B., y Vetterli M.**, Adaptive wavelet thresholding for image denoising and compression, *IEEE Transactions on Image Processing*, vol. 9, no. 9, pp.1532-1546, September 2000.

[185] **Chang S., Yu B., y Vetterli M.**, Spatially adaptive wavelet thresholding with context modelling for image denoising, *IEEE Trans. Image Processing*, vol. 9, pp. 1522-1531, Sept. 2000.

[186] **Choi H. y Baraniuk R.**, Multiscale texture segmentation using wavelet-domain hidden Markov models, in Proc. Int. Conf. Signals, Syst., Comput., vol. 2, 1998, pp. 1692-1697.

[187] **Lang M., Guo H., Odegard J., Burrus C. y Wells R.**, Noise reduction using an undecimated discrete wavelet transform, *IEEE Signal Processing Letters*, vol. 3, no. 1, pp. 10-12., 1996.

[188] **Gopinath R. A., Lang M., Guo H. y Odegard J. E.**, Enhancement of decom-pressed images at low bit rates, *Proceedings of SPIE, Mathematical Imaging: Wavelet Applications is Signal and Image Processing II* (2303-30), San Diego, CA, July 27-29, 1994.
Disponible en: http://cmc.rice.edu/docs/docs/Gop1994Jul5Enhanceme.ps

[189] **Gopinath R. A., Lang M., Guo H. y Odegard J. E.**, Wavelet-based post-processing of low bit rate transform coded images, *Rice University CML Technical Report*, 1994.
Disponible en: http://cmc.rice.edu/docs/docs/Gop1994Non9Wavelet-Ba.ps

[190] **Crouse M. S, Nowak R. D., y Baraniuk R. G**, Wavelet-based statistical signal processing using hidden Markov models. *IEEE Trans. Signal Processing*, vol 46, no.4, pp.886-902, April 1998.

[191] **Nowak R. D. y Baraniuk R. G.**, Wavelet-based transformations for nonlinear signal processing, IEEE Trans. on Signal Processing, Vol. 47, No. 7, pp.1852-1865, July 1999.
Disponible en: http://www.ece.wisc.edu/~nowak/nst.pdf

[192] **Nowak, R. D.**, Wavelet-based Rician noise removal for magnetic resonance imaging, *IEEE Transactions on Image Processing*, Vol. 8, No. 10, pp. 1408-1419, Oct. 1999.

[193] **Nowak R. D. y Baraniuk R. G.**, Wavelet-domain filtering for photon imaging systems, *IEEE Transactions on Image Processing*, Vol. 8, No. 5, pp. 666-678, May 1999.

[194] **Taswell C.**, The what, how, and why of wavelet shrinkage denoising, *Computing in Science and Engineering*, pp.12-19, May/June 2000.

[195] **Abramovich F. y Benjamini Y.**, Adaptive thresholding of wavelet coefficients, *Comput. Statist. Data Anal.*, vol. 22, pp. 351-361, 1996.

[196] **Abramovich F., Sapatinas T. y Silverman B.**, Wavelet thresholding via a Bayesian approach, *J. R. Stat.*, vol. 60, pp. 725-749, 1998.

[197] **Cai Z., Cheng T. H., Lu C. y Subramanian K. R.**, Efficient wavelet based image denoising algorithm, *Electron Lett.*, vol. 37, no. 11, pp. 683-685, May 2001.

[198] **Sylvain D y Nikolova M.**, Restoration of wavelet coefficients by minimizing a specially designed objective function, in *Proc. of IEEE Workshop on Var. and Level Set Meth. in Comp. Vision*, 2003.
Disponible en: http://www.cmla.ens-cachan.fr/Utilisateurs/nikolova/VLSM03.pdf

[199] **Wan S, Raju B. I. y Srinivasan M. A.**, Robust deconvolution of high-frequency ultrasound images using higher-order spectral analysis and wavelets, *IEEE Trans. on Ultrasonics, Ferroelectrics, and Frequency Control*, Vol.50, No.10, pp.1286-1295, October 2003.

[200] **Rangarajan R., Venkataramanan R. y Shah S.**, Image denoising using wavelets: Wavelets & time frequency, December 16, 2002.
Disponible en: http://www-personal.engin.umich.edu/~rvenkata/551_code/Report.pdf

[201] **Pólchlopek W. y Ziólko M.**, Wavelet transform compression and denoising in real-time system, *Proceedings of third international symposium on Communi-cation Systems Networks and Digital Signal Processing*, CSNDSP 2002: 15th-17th July 2002 Staffordshire, UK / ed. R. A. Carrasco. Staffordshire: Sheffield Hallam University Press Learning Centre, pp. 288-291 2002. Disponible en: http://www.scit.wlv.ac.uk/~jphb/cp4040/rolandonotes/CSNDSP2002/Papers/H2/H2.1.pdf

[202] **Calderbank A. R., Daubechies I., Sweldens W. y Yeo B.-L.**, Wavelet thransforms that map integers to integers, August 1996.
Disponible en: http://cm.bell-labs.com/who/wim/papers/integer.pdf

[203] **Thierschmann M., Martin U. y Rösel R.**, New perspectives on image compression, *Photogrammetric Week '97*, Weichmann Verlag, Heidelberg, pp.189-199, 1997.

[204] **Shapiro J. M.**, Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, Vol. 41, No. 12, pp.3445-3462, 1993.

[205] **Creusere C. D.**, A new method of robust image compression based on the embedded zerotree wavelet algorithm, *IEEE Transactions on Image Processing*, Vol. 6, No. 10, pp.1436-1442, 1997.

[206] **Algazi V. R. y Estes R. R.**, Analysis based coding of image transform and subband coefficients, *Proceedings of the SPIE*, Vol. 2564, pp. 11-21, 1995.

[207] **Jia W., He X. y Lin Q.**, Echocardiography sequential images compression based on region of interest, *Proceedings of the 2nd International Conference on Information Technology for Application* (ICITA 2004), pp.232-237, 2004. Disponible en: http://attend.it.uts.edu.au/icita05/CDROM-ICITA04/papers/IC-125.pdf

[208] **Kopp M.**, Lossless Wavelet Based Image Compression with Adaptive 2D Decompo-sition, *Proceedings of the Fourth International Conference in Central Europe on Computer Graphics and Visualization 96* (WSCG96), pages 141-149,edited by Nadia M. Thalmann and Vaclav Skala, Plzen, Czech Republic, February 12-16, 1996.

[209] **Paz Viera J. E.**, Aumento de la tasa de compresión y de la relación señal-ruido en imágenes ruidosas por la combinación de algoritmos de compresión y filtrado, *Informedica 2002: Preparando el Camino para la e-Salud Global, 2do Congreso Virtual Iberoamericano de Informática Médica*, Nov. 4-Nov. 30, 2002. Disponible en: http://www.informedicajournal.org/a1n2/files/papers_informedica/paz.pdf

[210] **Kontoyiannis I., Turek J., Castelli V. y Robinson J.**, Multiresolution lossless image compression. Unpublished manuscript, December 1995.
Disponible en: http://www.dam.brown.edu/people/yiannis/PAPERS/MCIA.pdf

[211] **Popat K.**, Lossy Compression of Grayscale Document Images by Adaptive-Offset Quantization, *Proceedings of IS&T/SPIE Electronic Imaging 2001: Do-cument Recognition and Retrieval VIII*, January 2001.
Disponible en: http://www2.parc.com/istl/members/popat/pdf/Popat2001_offset.pdf

[212] **Brooks S. y Tabbara M.**, Wavelet Image Compression: A Heuristic Approach to Space Frequency Quantisation, *Apple University Consortium Conference*, University of Adelaide, 28 September – 1 October, 2003. Disponible en: http://auc.uow.edu.au/conf/conf03/papers/AUC_DV2003_Tabbara.pdf

[213] **David B. Gerhard y W. Kinsner**, Lossy Compression of Head and Shoulder Images Using Zerotrees of Wavelet Coefficients, *IEEE Canadian Conference on Electrical and Computer Engineering*, Calgary, Alberta. I: 433-437, 1996.
Disponible en: http://www2.cs.uregina.ca/~gerhard/publications/calgary.pdf

[214] **Marcellin M. W., Gormish M. J., Bilgin A. y Boliek M. P.**, An Overview of JPEG-2000, *Proc. Data Compression Conference*, J. A. Storer and M. Cohn, eds., Snowbird, Utah, Mar. 28-Mar. 30, 2000, p. 523-541. Disponible en: http://www.rii.ricoh.com/%7egormish/pdf/dcc2000_jpeg2000_note.pdf

[215] **Van Otterloo S.**, Amazing Wavelet Image Compression, Utrecht, 2000.
Disponible en: http://www.bluering.nl/sieuwert/research/wavcomp.doc

[216] **Vargic R.**, An Approach to 2D Wavelet Transform and its Use for Image Compression, *Radioengineering*, Vol. 7, No. 4, December 1998.
Disponible en: http://www.ktl.elf.stuba.sk/~vargic/papers/radio98/re98_cr.doc

[217] **Amaratunga K.**, A fast wavelet algorithm for the reduction of color information. Disponible en: http://citeseer.ist.psu.edu/rd/21750013%2C479595%2C1%2C0.25%2CDownload/http://citeseer.ist.psu.edu/cache/papers/cs/23491/http:zSzzSzwavelets.mit.eduzSzKevinzSzPaperszSzColorQuant.pdf/a-fast-wavelet-algorithm.pdf

[218] **Almeida R., Martinez J. P., Olmos S., Rocha A. P. y Laguan P.**, Automatic delineation of T and P waves using a wavelet-based multiscale approach, *International Congress on Computational Bioengineering (ICCB'03)*. pp. 243-247. Zaragoza, 2003. Disponible en: http://diec.unizar.es/intranet/articulos/uploads/%20Automatic%20delineation%20of%20T%20and%20P%20waves%20using%20a%20wavelet-ased%20multiscale%20approach.pdf

[219] **Cuesta Frau D., Pérez Cortés J. C., Andréu García G., Eck V. y Sastre Mengual C.**, Reducción del ruido en señales electrocardiográficas mediante la transformada wavelet, *Conferencia Anual de la Sociedad Española de Ingeniería Biomédica (CASEIB-2000)* Cartagena, Spain. September 2000.
Disponible en: http://dcuesta.alc.upv.es/Investigacio/CASEIB2000.pdf

[220] **Cuesta Frau D., Pérez Cortés J. C., Andréu García G. y Novák D.**, Feature extraction methods applied to the clustering of electrocardiographic signals. A comparative study, *International Conference on Pattern Recognition (ICPR-2002)* Quebec, Canada. August 2002.

[221] **Mathias A., Grond F., Guardans R., Seese D., Canela M. y Diebner H. H.**, Algorithms for spectral analysis of irregularly sampled time series, *Journal of Statistical Software*, Vol. 11, No. 2, pp. 1-27, May 2004.

[222] **Paquet A. H., Zahir S. y Ward R. K.**, Wavelet packets-based image retrieval, *IEEE-ICASSP 2002*, Orlando, Florida, May 2002.
Disponible en: http://www.ece.ubc.ca/~alexp/PaperICASSP2002.pdf

[223] **Chandroth G., Sharkey A.J.C. y Sharkey N.E.**, Vibration signatures, wavelets and principal components analysis in diesel engine diagnostics, In: *Marine Technology 99*, Poland, 1999.

[224] **Quian Quiroga R. y García H.**, Single-trial event-related potentials with wavelet denoising, *Clinical Neurophysiology*, 114, pp.376-390, 2003.

[225] **Oweiss K. G. y Anderson D. J.**, Noise reduction in multichannel neural recordings using a new array wavelet denoising algorithm, *Neurocomputing* 38-40, pp.1687-1693, 2001.

[226] **Cárdenas-Barrera J. L., Lorenzo-Ginori J. V. y Rodríguez-Valdivia E.**, A wavelet-packets based algorithm for EEG signal compression, *Medical Informatics*, Vol.29,No.1, pp.15-27, March 2004.

[227] **Mojsilovic A., Popovic M. V., Neskovic A. N. y Popovic A. D.**, Wavelet image extension for analysis and classification of infracted myocardial tissue, *IEEE Trans. on Biomedical Engineering*, Vol.44, No.9, pp.856866, September 1997.

[228] **Donoho D. L., y Johnstone I. M.**, "Adapting to unknown smoothness via wavelet shrinkage," Journal of the American Statistical Assoc., vol. 90, no. 432, pp. 1200-1224., 1995.

[229] **Donoho D. L., y Johnstone I. M.**, "Ideal spatial adaptation by wavelet shrinkage," Biometrika, 81, 425-455, 1994.

[230] **Daubechies I.**, "Different Perspectives on Wavelets," in Proceedings of Symposia in Applied Mathematics, vol. 47, American Mathematical Society, USA, 1993.

[231] **Mallat S. G.**, "A theory for multiresolution signal decomposition: The wavelet representation," IEEE Trans. Pattern Anal. Machine Intell., vol. 11, pp. 674–693, July 1989.

[232] **Mallat S. G.**, "Multiresolution approximations and wavelet orthonormal bases of L2 (R)," Transactions of the American Mathematical Society, 315(1), pp.69-87, 1989a.

[233] **Chang S. G., Yu B., y Vetterli M.**, "Adaptive wavelet thresholding for image denoising and compression," IEEE Trans. Image Processing, vol. 9, pp. 1532–1546, Sept. 2000.

[234] **Misiti M., Misiti Y., Oppenheim G., y Poggi J. M.,** (2001, June). Wavelet Toolbox, for use with MATLAB®, User's guide, version 2.1.

[235] **Burrus C. S., Gopinath R. A., y Guo H.**, Introduction to Wavelets and Wavelet Transforms: A Primer, Prentice Hall, New Jersey, 1998.

[236] **Hubbard B. B.**, The World According to Wavelets: The Story of a Mathematical Technique in the Making, A. K. Peter Wellesley, Massachusetts, 1996.

[237] **Grossman A. y Morlet J.**, "Decomposition of Hardy Functions into Square Integrable Wavelets of Constant Shape," SIAM J. App Math, 15: pp.723-736, 1984.

[238] **Walker J. S.**, A Primer on Wavelets and their Scientific Applications, Chapman & Hall/CRC, New York, 1999.

[239] **Stollnitz E. J., DeRose T. D., y Salesin D. H.**, Wavelets for Computer Graphics: Theory and Applications, Morgan Kaufmann Publishers, San Francisco, 1996.

[240] **Unser M.**, "Wavelets, filterbanks, and the Karhunen-Loeve transform,"EUSIPCO-98, vol. 3, pp. 1737-1741, Island of Rhodes, Greece, Sept.1998.

[241] **Zhang J. y Walter G.**, "A wavelet based KL-like expansion for wide sense stationary random processes," IEEE Trans. SP, Vol. 42, pp. 1737-1745, July 1994.

[242] **Wornell G. W.**, "A Karhunen-Loeve-like expansion for 1/f processes via wavelets," IEEE Trans. IT, Vol. 36, pp. 859-861, July 1990.

[243] **Hall J. y Crowe J.**, "Ambulatory electrocardiogram compression using wavelet packets to approximate Karhunen-Loeve transform," Int. J. Applied SP, Vol. 3, pp. 25-36, 1996.

[244] **Starck J.-L., y Querre P.**, "Multispectral Data Restoration by the Wavelet-Karhunen-Loève Transform," Preprint submitted to Elsevier Preprint, 2000, pp. 1–29.

[245] **M. Mastriani, "***Single frame supercompression of still images, video, HDTV and Digital Cinema*," International Journal of Information and Mathematical Sciences, Volume 6, Number 3, pp.143-159, 2010.

[246] **M. Mastriani, "***Supercompression for Full-HD and 4k-3D (8k) Digital TV Systems*," International Journal of Information and Mathematical Sciences, Volume 6, Number 3, pp.186-199, 2010.

[247] **A. Gilman, D. G. Bailey, S. R. Marsland**, "Interpolation Models for Image Super-resolution," in *Proc. 4th IEEE International Symposium on Electronic Design, Test & Applications*, DELTA 2008, Hong Kong, 2008, pp.55-60.

[248] **D. Glassner, S. Bagon, M. Irani**. Super-Resolution from a Single Image. Available: http://www.wisdom.weizmann.ac.il/~vision/single_image_SR/files/single_image_SR.pdf

[249] **A. Lukin, A. S. Krylov, A. Nasonov**. Image Interpolation by Super-Resolution. Available: http://graphicon.ru/oldgr/en/publications/text/LukinKrylovNasonov.pdf

[250] **Y. Huang**, "Wavelet-based image interpolation using multilayer perceptrons," *Neural Comput. & Applic.*, vol.14, pp.1-10, 2005.

[251] **N. Mueller, Y. Lu, and M. N. Do**. Image interpolation using multiscale geometric representations. Avalilable: http://lcav.epfl.ch/~lu/papers/interp_contourlet.pdf

[252] **T. C. Wittman**, "Variational Approaches to Digital Zooming," Ph.D. dissertation, Dept. Math, Univ. of Minnesota, Saint Paul, MN, 2006.

[253] **M. Mastriani**. Directional smoothing for speckle reduction in synthetic- aperture radar imagery. Available: http://fundesco.eurofull.com/img/paper2.pdf

[254] **M. Kraus, M. Eissele, and M. Strengert**. GPU-Based Edge-Directed Image Interpolation. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.69.5655

[255] **S. C. Park, M. K. Park y M.G. Kang**, "Super-resolution image reconstruction: A technical overview", IEEE Signal Processing Magazine, vol. 20, no. 3, pp. 21-36, 2003.

[256] -., A. K. Katsaggelos, and R. Molina (editors), "*Super resolution (special issue)*", The Computer Journal, 52, 395-396, 2009.

[257] **A. K. Katsaggelos, R. Molina, and J. Mateos**, "*Super resolution of images and video*", Synthesis Lectures on Image, Video, and Multimedia Processing, Morgan & Claypool, 2007.

[258] **M. Elad y A. Feuer**, "Restoration of a Single Superresolution Image from Several Blurred, Noisy, and Undersampled Measured Images", IEEE Transactions on Image Processing, vol. 6, pgs. 1646–1658, Dec. 1997.

[259] **C. A. Segall, R. Molina y A. K. Katsaggelos**, "High-Resolution images from low-resolution compressed video", IEEE Signal Processing Magazine, vol. 20, pgs. 37-48, 2003.

[260] **S. Borman y R. Stevenson**, "Spatial resolution enhancement of low-resolution image sequences. A comprehensive review with directions for future research," Tech. Rep., Laboratory for Image and Signal Analysis, University of Notre Dame, 1998.

[261] **S. Chaudhuri**, ''Super-resolution Imaging'', editor, Kluwer Academic Publishers, 2002.

[262] **M. Irani y S. Peleg**, "Motion analysis for image enhancement: Resolution, occlusion and transparency", Journal of Visual Communications and Image Representation, vol. 4, pgs. 324–335, 1993.

[263] **E. P. Simoncelli, E. H. Adelson y D. J. Heeger**, "Probablity distributions of optical flow", en Proc. IEEE Computer Soc. Conf. Computer Vision and Pattern Recognition, 1991, pgs. 310-315.

[264] **J. Chamorro-Martínez**, "Desarrollo de modelos computacionales de representación de secuencias de imágenes y su aplicación a la estimación de movimiento, Tesis doctoral, Universidad de Granada, 2001.

[265] **O. Nestares y R. Navarro**, "Probablistic estimation of optical flow in multiple band-pass directional channels", Image and Vision Computing, vol. 19, pgs. 339-351, 2001.

[266] **R. Y. Tsai and T. S. Huang**, "Multiframe image restoration and registration", in Advances in Computer Vision and Image Processing, R. Y. Tsai and T. S. Huang, Eds., vol. 1, pp. 317–339. JAI Press Inc., 1984.

[267] **A. M. Tekalp**, Digital Video Processing, chapter 17, Prentice Hall, Upper Saddle River, NJ, 1995.

[268] **A. M. Tekalp, M. K. Ozkan, and M. I. Sezan**, "High-resolution image reconstruction from lower-resolution image sequences and space-varying image restoration", in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, San Francisco, CA, 1992, vol. III, pp. 169–172.

[269] **E. Kaltenbacher and R. C. Hardie**, "High-resolution infrared image reconstruction using multiple low resolution aliased frames", in Proceedings of the IEEE National Aerospace Electronics Conference (NAECON), Dayton, OH, May 1996, vol. 2, pp. 702–709.

[270] **S. P. Kim, N. K. Bose, and H. M. Valenzuela**, "Recursive reconstruction of high resolution image from noisy undersampled multiframes", IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 38, no. 6, pp. 1013–1027, 1990.

[271] **S. P. Kim and W.-Y. Su**, "Recursive high-resolution reconstruction of blurred multiframe images", IEEE Transactions on Image Processing, vol. 2, pp. 534–539, Oct. 1993.

[272] **C. E. Davila**, "Recursive Total Least Squares Algorithms for Adaptive Filtering", in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Toronto, Canada, 1991, pp. 1853–1856.

[273] **C. E. Davila**, "Efficient Recursive Total Least Squares Algorithms for FIR Adaptive Filtering", IEEE Transactions on Signal Processing, vol. 42, no. 2, pp. 268–280, Feb. 1994.

[274] **N. K. Bose, H. C. Kim, and B. Zhou**, "Performance analysis of the TLS algorithm for image reconstruction from a sequence of undersampled noisy and blurred frames", in Proceedings of the IEEE International Conference on Image Processing, Austin, TX, Nov. 1994, vol. III, pp. 571–575.

[275] **S. H. Rhee y M. G. Kang**, "Discrete cosine transform based regularized high-resolution image reconstruction algorithm", Opticla Engineering, vol 38, pgs 1348-1356, 1999.

[276] **H. Ur y D. Gross**, "Improved resolution from sub-pixel shifted pictures", CVGIP: Graphical Models and Image Processing, vol. 54, pgs. 181-186, 1992.

[277] **A. Papoulis**, "Generalized sampling expansion", TEEE Trans. on Circuits and Systems, vol. 24, pgs. 652-654, 1977.

[278] **J. L. Brown**, "Multichannel sampling of low-pass signals", IEEE Trans. on Circuits and Systems, vol. 28, pgs. 101-106, 1981.

[279] **J. J. Clark, M. R. Palmer, and P. D. Laurence**, "A transformation method for the reconstruction of functions from non-uniformly spaced samples", IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 33, no. 4, pp. 1151–1165, Oct. 1985.

[280] **K. Aizawa, T. Komatsu, and T. Saito**, "Acquisition of very high resolution images using stereo cameras", in Visual Communications and Image Processing, 1991, vol. 1605 of Proceedings of the SPIE, pp. 318–328.

[281] **T. Komatsu, K. Aizawa, T. Igarashi, and T. Saito**, "Signal-processing based method for acquiring very high resolution images with multiple cameras and its theoretical analysis", IEE Proceedings I, vol. 140, pp. 19–25, Feb. 1993.

[282] **T. Saito, T. Komatsu, and K. Aizawa**, "An image processing algorithm for a super high definition imaging scheme with multiple different-aperture cameras", in Proceedings of the IEEE International Conference of Acoustics, Speech and Signal Processing, Adelaide, Australia, Apr. 1994, vol. 5, pp. 45–48.

[283] **L. Landweber**, "An iterative formula for Fredholm integral equations of the first kind", American Journal of Mathematics, vol. 73, pp. 615–624, 1951.

[284] **M. S. Alam, J. G. Bognar, R. C. Hardie y B. J. Yasuda**, "Infrared image registration and high-resolution reconstruction using multiple translationally shifted aliased video frames", IEEE Trans. on Instrumentation and measurement, vol. 49, pgs. 915-923, 2000.

[285] **N. R. Shah y A. Zakhor**, "Resolution enhancement of color video sequences", IEEE Trans. on Image Processing, vol. 8, pgs 879-885, 1999.

[286] **B. R. Frieden and H. G. Aumann**, "Image reconstruction from multiple 1-D scans using filtered localized projection", Applied Optics, vol. 26, no. 17, pp. 3615–3621, Sept. 1987.

[287] **D. Keren, S. Peleg, and R. Br**ada, "Image sequence enhancement using subpixel displacements", in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June 1988, pp. 742–746.

[288] **S. Peleg, D. Keren, and L. Schweitzer**, "Improving image resolution by using subpixel motion", Pattern Recognition Letters, vol. 5, no. 3, pp. 223–226, Mar. 1987.

[289] **M. Irani and S. Peleg**, "Improving resolution by image registration", CVGIP: Graphical Models and Image Processing, vol. 53, no. 3, pp. 231–239, May 1991.

[290] **M. Irani, B. Rousso, and S. Peleg**, "Computing occluding and transparent motions", International Journal of Computer Vision, vol. 12, no. 1, pp. 5–16, Feb. 1994.

[291] **C. Kim, K. Choi, K. Hwang, and J. Beom Ra**. Learning-based super-resolution using a multi-resolution wavelet approach. Available: http://www-isl.kaist.ac.kr/Papers/IC/ic123.pdf

[292] **C. S. Boon, O. G. Guleryuz, T. Kawahara and, Y. Suzuki**, "Sparse super-resolution reconstructions of video from mobile devices in digital TV broadcast applications," in *Proc. SPIE Conf. on Applications of Digital Image Processing XXIX, in Algorithms, Architectures, and Devices*, San Diego, CA, Aug. 2006.

[293] **I. E. Richardson,** H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia, Ed. Wiley, N.Y., 2003.

[294] http://www.untref.edu.ar/carreras_de_grado/ing_computacion.htm

[295] **D. E. Goldberg**, Genetic Algorithms in Search, Optimization, and Machine Learning, Ed. Pearson, N.Y., 1989.

[296] **.-**, Kalman Filtering: Theory and Application, Sorenson H. W. ed., IEEE Press, 1985, New York.

[297] **S. H. M. Allon, M. G. Debertrand, and B. T. H. M. Sleutjes**, "Fast Deblurring Algorithms", 2004. Available: http://www.bmi2.bmt.tue.nl/image-analysis/Education/OGO/0504-3.2bDeblur/OGO3.2b_2004_Deblur.pdf

[298] **A. Bennia and S. M. Riad**, Filtering Capabilities and Convergence of the Van-Cittert Deconvolution Technique, IEEE, Trans. Instrum. Meas., Vol. 41, no. 2, pp. 246-250, Apr. 1992.

[299] http://www.forumsbtvd.org.br/

[300] NVIDIA® (NVIDIA Corporation, Santa Clara, CA).

[301] **H. Schwarz, D. Marpe, T. Wieg**, Overview of the scalable H.264/MPEG4-AVC extension, in Proceedings of the IEEE International Conference on Image Processing, ICIP '06. Available: http://iphome.hhi.de/marpe/download/icip06_svc.pdf

[302] **P. S. Mandolesi, P. Julian, A. G. Andreou**, "A scalable and program-mable simplicial CNN digital pixel processor architecture," IEEE Trans. Circuits and Systems – I: Regular Papers, Vol. 51, No. 5, pp. 988- 996, May 2004.

[303] http://elclub.ncl.org.br

[304] http://www-di.inf.puc-rio.br/~colcher/cce/ginga-ncl/main_files/menu/material/transparencias/Lua-ginga-cce.pdf

[305] **T. Ito**, Future Television - Super Hi-Vision and Beyond, IEEE Asian Solid-State Circuits Conference, November 8-10, 2010, Beijing, China, Available: https://web.engr.oregonstate.edu/~venkatha/mwiki/.../Futuretelevision.pdf

[306] ISO/IEC 15444-1:2000. Information technology – JPEG 2000 image coding system – Part 1: Core coding system, 2000.

[307] ISO/IEC 15444-1 and ITU-T Recommendation T.800, "Information technology {JPEG 2000} image coding system," 2002.

[308] **Taubman D., Ordentlich E., Weinberger M., y Seroussi G.,** Embedded block coding in JPEG 2000. In ICIP00, page TA02.02, 2000.

[309] ISO/IEC 15444-1:2000. Information technology – JPEG 2000 image coding system – Part 1: Core coding system, 2000.

[310] **Clark R.**, "An Introduction to JPEG2000 and Watermarking." Elysium Ltd. Disponible en: http://www.jpeg.org/public/jpgintro.pdf (as of 01.10.02).

[311] **Taubman D. S. y Marcellin M. W.**, *JPEG2000: Fundamentals, Standards and Practice.* Kluwer Academic Publishers, Boston, 2002.

[312] **Adams M. D.**, "The JPEG-2000 Still Image Compression Standard," Tech. Rep. N2412, ISO/IEC JTC 1/SC 29/WG 1, September 2001.

[313] **Rabbani M. y Santa Cruz D.**, "The JPEG2000 Still-Image Compression Standard." Course given at the 2001 International Conference in Image Processing (ICIP), October 2001. http://jj2000.epfl.ch/jj_publications/papers/011.pdf (as of 01.10.02).

[314] **Christopoulos C. y Skodras A.**, "JPEG2000 - The Next Generation Still-Image Compression Standard." Tutorial given at the IEEE International Conference on Image Processing (ICIP), October 1999. Disponible en: http://www.etro.vub.ac.be/members/christopoulos.charilaos/jpeg2000_cont%ributions.htm (as of 01.10.02).

[315] **Impoco G.**, JPEG2000 - A Short Tutorial, April 1, 2004 Disponible en: http://www.dmi.unict.it/~impoco/files/tutorial_JPEG2000.pdf

[316] ISO/IEC 11544:1993 and ITU-T Recommendation T.82. Information technology – Coded representation of picture and audio information – progressive bi-level image compression, 1993.

[317] ISO/IEC 15444-1:2000. Information technology – JPEG 2000 image coding system – Part 1: Core coding system, 2000.

[318] **Taubman D., Ordentlich E., Weinberger M., y Seroussi G.,** Embedded block coding in JPEG 2000. In ICIP00, page TA02.02, 2000.

[319] MATLAB® R2010a (Mathworks, Natick, MA).
Disponible en: http://www.mathworks.com/access/helpdesk/help/helpdesk.html

[320] http://dicom.nema.org/

[321] http://www.itu.int/en/pages/default.aspx

[322] **Couch II L. W.**, Sistemas de Comunicación Digitales y Analógicos, Prentice-Hall, 1998, México.

[323] -., E. F. Deprettere, ed., SVD and Signal Processing, North-Holland, New York, 1988.

[324] -., R. J. Vaccaro, ed., SVD and Signal Processing II, Elsevier, New York, 1991.

[325] **Moonen M. y de Moor B.**, SVD and Signal Processing III, Elsevier, New York, 1995.

[326] **_.**, (2005, March). Image Processing Toolbox, for use with MATLAB®, User's guide, version 5. Disponible en: http://www.mathworks.com/access/helpdesk/help/pdf_doc/images/images_tb.pdf

[327] **_.**, (2005, March). Signal Processing Toolbox, for use with MATLAB®, User's guide, version 6. Disponible en: http://www.mathworks.com/access/helpdesk/help/pdf_doc/signal/signal_tb.pdf

[328] **Samet H.**, The Design and Analysis of Spatial Data Structures, Addison-Wesley, 1990.

[329] **Samet H.**, Applications of Spatial Data Structures – Computer Graphics, Image Processing and GIS, Addison-Wesley, 1990.

[330] **Arnold B.**, An Investigation into using Singular Value Decomposition as a method of Image Compression, University of Canterbury Department of Mathematics and Statistics, September 2000. Disponible en: http://math.ucalgary.ca/~laf/teaching/Material/Arnold.pdf

[331] **Tjoa S. et al**, "Transform coder classification for digital image forensics". Disponible en: http://www.cspl.umd.edu/sig/publications/tjoa_ICIP_200709.pdf

[332] **Goyal V**. (2001). Theoretical foundations of transform coding. IEEE Signal Processing Magazine, 18(5), 9–21.

[333] **Gray R. M., y Neuhoff D. L.** (1998). Quantization. IEEE Transactions on Information Theory, 44(6), 2325–2383.

[334] **Huhle B.**, Kernel PCA for Image Compression. Ph.D., Wilhelm-Schickard-Institut für Informatik Eberhard Karls Universität Tübingen 9. April 2006. Disponible en: http://www.gris.uni-tuebingen.de/people/staff/huhle/publications/huhle-2006-thesis.pdf

[335] **Clarke R. J.**, "Transform coding of images," Orlando, FL: Academic Press, 1985.

[336] **Waldemar P. y Ramstad T.**, "Hybrid KLT-SVD image compression," ICASSP 97, Vol. 4, pp. 2713-2716, Munich, Germany, April 1997.

[337] **Gerbrands J. J.**, "On the Relationships Between SVD, KLT, and PCA," Patt. Recogn., **14**, 375 (1981).

[338] **Ruiz V. G.**, Compresión Reversible y Transmisión de Imágenes, Universidad de Almería, Departamento de Arquitectura de Computadores y Electrónica, Tesis Doctoral, Julio 2000. Disponible en: http://www.ace.ual.es/~vruiz/investigacion/tesis.pdf

[339] **Huffman D. A.,** A Method for the Construction of Minimum Redundancy Codes. Proceedings of the Institute of Radio Engineers, 40:1098-1101, 1952.

[340] **Cleary J. G. y Witten I. H.,** Data Compression using Adaptive Coding and Partial String Matching. IEEE Transactions on Communications, 4(32):396-402, 1984.

[341] **Howard P. G. y Vitter J. S.,** Analysis of Arithmetic Coding for Data Compression. Information Processing and Management, 28(6):749-763, 1992.

[342] **Weinberger M. J., Rissanen J., y Arps R. B.,** Applications of the Universal Context Modeling to Lossless Compression of Gray-Scale Images. IEEE Transactions on Image Processing, 5(4):575-586, 1996.

[343] **Knuth D. E.,** Dynamic Huffman Coding. Journal of Algorithms, 6(2):163-180, 1985.

[344] **Nelson M. y Gailly J.,** The Data Compression Book. M&T Books, 1996.

[345] **Ruiz V. G. y García I.,** Compresión de Texto basada en un Modelo Probabilística de Orden 0 y un Codificador de Huffman. Technical Report 1, Depto de Arquitectura de Computadores y Electrónica, Universidad de Almería, 1998.

[346] **Vitter J. S.,** Design and Analysis of Dynamic Huffman Codes. ACM, (4):825-845, 1987.

[347] **Abramson N.,** Information Theory and Coding. New York, McGraw-Hill, 1963.

[348] **Pasco R.,** Source Coding Algorithms for Fast Data Compression. PhD thesis, Stanford University, 1976.

[349] **Langdon G. G. y Rissanen J.,** Compression of Black-White Images with Arithmetic Coding. IEEE Transactions on Communications, 29(6):858-867, 1981.

[350] **Rissanen J. y Langdon G. G.,** Arithmetic Coding. IBM J. Res. Develop., pp.146-162, 1979.

[351] **Guazzo M.,** A General Minimum-Redundancy Source-Coding Algorithm. IEEE Transactions on Information Theory, 26:15-25, 1980.

[352] **Howard P. G. y Vitter J. S.,** Practical Implementations of Arithmetic Coding. In Image and Text Compression, pages 765-779. Kluwer Academic Publishers, 1992.

[353] **Howard P. G. y Vitter J. S.,** Design and Analysis of Fast Text Compression Based on Quasi-Arithmetic Coding. In Data Compression Conference (DCC), pages 89-107, 1993.

[354] **Langdon G. G. y Rissanen J.,** Compression of Black-White Images with Arithmetic Coding. IEEE Transactions on Communications, 29(6):858-867, 1981.

[355] **Mitchell J. L. y Pennebaker W. B.,** Optimal Hardware and Software Arithmetic Coding Procedures for the Q-Coder. IBM J. Res. Develop, 32:727-736, 1988.

[356] **Pennebaker W. B., Mitchell J. L., Langdon G. G., y Arps R. B.,** An Overview of the Basic Principles of the Q-Coder Adaptive Binary Arithmetic Coder. IBM J. Res. Develop, 32:717-726, 1988.

[357] **Rissanen J. y Langdon G. G.,** Arithmetic Coding. IBM J. Res. Develop., pages 146-162, 1979.

[358] **Rissanen J. y Langdon G. G.,** Universal Modeling and Coding. IEEE Transactions on Information Theory, 27:12-23, 1981.

[359] **Rissanen J. J.,** Generalized Kraft Inequality and Arithmetic Coding. IBM J. Res. Develop., 20(198-203), 1976.

[360] **Rissanen J. J.,** A Universal Data Compression System. IEEE Transaction on Information Theory, 29(5):656-664, 1983.

[361] **Rubin F.,** Arithmetic Stream Coding Using Fixed Precision Registers. IEEE Transactions on Information Theory, 25:672-675, 1979.

[362] **Witten I. H., Neal R. M., y Cleary J. G.,** Arithmetic Coding for Data Compression. Communications of the ACM, 30(6):520-540, 1987.

[363] **Arps R. B., Truong T. K., Lu D. J., Pasco R. C., y Friedman T. D.,** A Multi-Purpose VLSI Chip for Adaptive Data Compression of Bilevel Images. IBM J. Res. Develop, 32(6):775-795, 1988.

[364] **Langdon G. G.,** Probabilistic and Q-Coder Algorithms for Binary Source Adaption. In Data Compression Conference (DCC), pages 13-22, 1991.

[365] **Mitchell J. L. y Pennebaker W. B.**, Optimal Hardware and Software Arithmetic Coding Procedures for the Q-Coder. IBM J. Res. Develop, 32:727-736, 1988.

[366] **Pennebaker W. B. y Mitchell J. L.,** Probability Estimation for the Q-Coder. IBM J. Res. Develop, 32:737-752, 1988.

[367] **Pennebaker W. B. y Mitchell J. L.,** Software Implementation of the Q-Coder. IBM J. Res. Develop, 32:753-774, 1988.

[368] **Taubman D. S.,** High performance scalable image compression with EBCOT. In IEEE Trans. Image Proc., volume 9, July 2000.

[369] **Abadir K. M. y Magnus J. R.**, Matrix Algebra, Cambridge University Press, New York, 2005.

[370] **Roman S.**, Advanced Linear Algebra, Third Edition, New York, 2008.

[371] **Meyer C. D.**, Matrix Analysis and Applied Linear Algebra, Siam Press, Philadelfia, 2000.

[372] **Sacchi M. D.**, Statistical and Transform Methods for Geophysical Signal Processing, Chapter 6 KL and Eigen-images, 2001. Disponible en: http://www-geo.phys.ualberta.ca/saig/book/chapter6.pdf

[373] **Freire S. L. M. y Ulrych T. J.**, Application of singular value decomposition to vertical seismic profiling, Geophysics. Vol.53, No.6 (June 1988): pp.778-785

[374] **Cagnoli B., y Ulrych T. J.**, Singular value decomposition and wavy reflections in ground-penetrating radar images of base surge deposits, Journal of Applied Geophysics 48 (2001) 175–182.

[375] **Orfanidis S. J.**, SVD, PCA, KLT, CCA, and All That, Rutgers University, Electrical & Computer Engineering Department, Optimum Signal Processing, 2002–2007. Disponible en: http://www.ece.rutgers.edu/~orfanidi/ece525/svd.pdf

[376] **Winslow P.**, Singular Value Decomposition for Path Reconstruction in High Purity Germanium Detectors for TIGRESS, August 23, 2007. Disponible en: https://gamma.triumf.ca/misc/Student/Peter%20Winslow/SVDReport.pdf/view

[377] **Hoffman K. y Kunze R.**, Álgebra Lineal, Prentice-Hall Hispanoamericana, 1973, México.

[378] **Lang S.**, Introducción al Algebra Lineal, Addison-Wesley Iberoamericana, 1990, Wilmington, Delaware.

[379] **Leon S. J.**, Linear Algebra with Applications, MacMillan, 1990, New York.

[380] **Fraleigh J. B.**, Algebra Abstracta, Addison-Wesley Iberoamericana, 1987, México.

[381] **Noble B. y Daniel J. W.**, Algebra Lineal Aplicada, Prentice-Hall, 1989, México.

[382] **Marcus M. y Minc H.**, Álgebra Lineal, CECSA, 1969, México.

[383] **Birkhoff G. y MacLane S.**, Algebra Moderna, Vicens-Vives, 1980, Barcelona.

[384] **Grossman S. I.**, Algebra Lineal, McGraw-Hill, 2004, México.

[385] **Nicholson W. K.**, Algebra Lineal con aplicaciones, McGraw-Hill, 2004, México.

[386] **Strang G.**, Álgebra Lineal y sus Aplicaciones, Addison-Wesley Iberoamericana, 1990, México.

[387] **Herstein I. N. y Winter D. J.**, A primer on Linear Algebra, MacMillan, 1990, New York.

[388] **Santalo L. A.**, Vectores y Tensores con sus aplicaciones, EUDEBA, 1981, Buenos Aires.

[389] **Máltsev A. I.**, Fundamentos de Álgebra Lineal, MIR, 1978, Moscú.

[390] **Dubreil P. y Dubreil-Jacotin M. L.**, Lecciones de Algebra Moderna, Reverté, 1975, Barcelona.

[391] **Thompson R. C. y Taqub A.**, Introducción al Álgebra Abstracta y Lineal, Uteha, 1976, México.

[392] **Mastriani M.**, An Application for the Symmetric Functions*, Fifth Society Industrial and Applied Mathematics (SIAM) Conference on Applied Linear Algebra*, Snowbird, Utah, June 15-18, 1994.

Sinopsis Curricular

- Ingeniero Electrónico, Especialidad: Control Automático
- Doctor en Ingeniería
- Doctor en Ciencias de la Computación
- Candidato al Doctorado en Ciencia y Tecnología de la UNGS
- Referee de 15 journals internacionales de la IEEE, Springer-Verlag, Taylor & Francis, IET, SPIE, OSA, Elsevier, etc.
- Coordinador de la Carrera de Ingeniería en Computación de la Universidad Nacional de Tres de Febrero (UNTreF)
- Profesor Titular de Procesamiento de Imágenes de la Carrera de Ingeniería en Computación de la UNTreF
- Director del Laboratorio de Imágenes y Señales (LIS) del Sistema UVT-CPA-NeoTVLabs de la UNTreF
- Profesor del curso de Doctorado de la Facultad de Ingeniería de la UBA:
  Tópicos avanzados en procesamiento de señales e imágenes
- Profesor de la Maestría en Optoelectrónica de la Facultad de Ingeniería de la UBA:
  1) Diagnóstico y tratamiento médico.
  2) Aplicaciones Médicas.
- ExDirector del Laboratorio de Investigación y Desarrollo en Nuevas Tecnologías (LIDeNTec) de ANSES
- Coordinador del área de Innovación Técnica de la Gerencia de Informática e Innovación Tecnológica (GIIT) de ANSES
- Evaluador FONCyT en Tecnología Informática, Comunicaciones y Electrónica de la Agencia
- Evaluador Consejo de Investigación Científica (CIC) de la Provincia de Bs. As. en Tecnología Informática, de la Comunicaciones y Electrónica
- Evaluador CONEAU para las carreras de Ingeniería Informática
- Ex responsable del grupo de focalización Synthetic Aperture Radar (SAR) de CONAE
- Contratista de INVAP
- Contratista de la Satellogic
- Representante por UNTreF ante el Programa de Desarrollo de Software para la TV Digital Interactiva del Ministerio de Planificación Federal, Inversión Pública y Servicios
- 49 publicaciones científicas internacionales con proceso de referato severo en journals
- 51 conferencias científicas en el país y el exterior

Links de referencia:
http://www.ucema.edu.ar/u/mmastriani/cv.pdf
http://www.untref.edu.ar/carreras_de_grado/ing_computacion.htm
http://pmcg.minplan.gov.ar/html/eventos/presentacion_universidades/untref.php
http://www.tvpublica.com.ar/tvpublica/articulo?id=3082
http://www.iptv-forum.com.ar/staff.htm
http://softwarelivre.org/portal/comunidade/i-workshop-de-tv-digital-interativa-wtvdi-webmedia-2010
http://www.ufmg.br/swib/?p=135
http://www.sase.com.ar/2010/dr-ing-mario-mastriani
http://www.webmii.es/Result.aspx/Mario/Mastriani
http://www.redusers.com/noticias/desarrollo-100-argentino-un-dispositivo-que-convierte-tu-lcd-en-3d/

http://www.redusers.com/noticias/conozcan-la-tecnologia-argentina-que-transmitira-el-mundial-de-brasil-en-4k-3d/?replytocom=57761
http://www.elargentino.com/nota-139109-Un-invento-argentino-transforma-la-television-3D.html
http://www.prensa.argentina.ar/2011/05/15/19562-experimento-universitario-para-transformar-la-teve-comun-en-3d.php
http://www.redusers.com/noticias/nestor-el-primer-codec-de-video-para-redes-de-fibra-optica-es-argentino/
http://www.redusers.com/noticias/nestor-todo-lo-que-querias-saber-sobre-el-codec-contestado-por-sus-creadores/
http://www.redusers.com/noticias/conozcan-la-supercomputacion-a-base-de-gpu-que-motoriza-la-tv-digital/