ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Implementación de un codificador de video para videoconferencia basado en el estándar H.261

(Implementation of a video codec for videoconference based on the H.261 standard)

Para acceder al Título de

Graduado en

Ingeniería de Tecnologías de Telecomunicación

Autor: Sergio Sainz Ruiz

Octubre - 2017

E.T.S. DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACION

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Sergio Sainz Ruiz
Director del TFG: Pablo Pedro Sánchez Espeso
Título: "Implementación de un codificador de video para videoconferencia basado en el estándar H.261"
Title: **"**Implementation of a video códec for videoconference based on H.261 standard**"**

Presentado a examen el día: 27 de Octubre de 2017

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): Pablo Pedro Sánchez Espeso
Secretario (Apellidos, Nombre): Begoña Sánchez Madariaga
Vocal (Apellidos, Nombre): Iñigo Ugarte Olano

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del TFG
(sólo si es distinto del Secretario)

Vº Bº del Subdirector

Trabajo Fin de Grado Nº
(a asignar por Secretaría)

Agradecimientos

A mi familia, por darme la oportunidad de recibir una gran educación y por depositar en mi la fe necesaria para lograr este objetivo.

A mis amigos, gran pilar donde poder apoyarse cuando las cosas no van bien.

A mi tutor, Pablo, por sus enseñanzas y su entusiasmo contagioso que han hecho que no me arrepienta de mi decisión de estudiar electrónica. Sin él, este trabajo no hubiera salido adelante.

Al resto de compañeros y profesores del departamento que te echan una mano cuando lo necesitas.

RESUMEN

En este trabajo se va a realizar el estudio de un codificador de video basado en el estándar H.261. Se llevará a cabo un análisis de cada uno de los módulos necesarios para su implementación. Se realizará un código software en C/C++ de los algoritmos y se obtendrá un resultado a evaluar.

ABSTRACT

For this work, a video encoder based on the H.261 standard will be studied. To do this, an analysis of each modules is necessary for its implementation. A software code will be made in C / C ++ of the algorithms and a result to be evaluated will be obtained.

Palabras Clave

Codificación, Compresión, Huffman, Frame, DCT, Cuantificación, Macrobloque, Estándar, H.261, Redundancia, Estándar, GOB.

Key Words

Coding, Compression, Huffman, Frame, DCT, Quantization, Block, Standard, H.261, Redundancy, Standard, GOB.

Índice

Tabla de Figuras

Tabla de Ecuaciones

Capítulo **1**

1.1 Introducción

Una de las principales características del ser humano es la necesidad de comunicarse. Somos seres sociales que necesitamos interactuar con otras personas.

Esta forma de comunicarse ha ido cambiando con el paso de los años, dando un gran giro en los últimos años.

Actualmente, vivimos en un planeta globalizado en el que los distintos países, empresas o personas necesitan comunicarse continuamente y en este proceso de globalización, las tecnologías digitales han tenido gran importancia.

La revolución digital, que comenzó aproximadamente a finales de los años 50, supuso un gran avance en la evolución de las comunicaciones, que consistió en el paso de la tecnología analógica a la tecnología digital, y se basa principalmente en la generación, transmisión o procesado de señales digitales.

Las señales analógicas poseían una calidad y una resolución inferior respecto a las señales digitales, así como una menor robustez frente al ruido. Por tanto, esta digitalización supuso un gran avance tecnológico. Sin embargo, y a pesar de estas mejoras significativas, la cantidad de información que se comenzó a generar superó notablemente los sistemas de procesado de los que se disponían.

Hacia mediados de los años 70, un grupo de investigadores japoneses comenzaron a estudiar la posibilidad de realizar un sistema digital de grabación de imagen con el fin de no perder calidad tras las grabaciones, pero no fue hasta una década después cuando Sony presentó su primer videograbador basado en esta tecnología.

Años más tarde, esta tecnología se fue introduciendo en los estudios de TV, hasta finalmente llegar al usuario en forma de videograbador portátil. Sin embargo, estos avances, generaban grandes archivos de información cuyo almacenamiento se hacía insostenible y por ello, surgió la idea de la compresión.

A principios de los años 90, un grupo de expertos de la ITU (unión internacional de telecomunicaciones) desarrolló un estándar de codificación enfocado a las videoconferencias denominado H.261. Por otro lado, el grupo experto de imágenes en movimiento (MPEG) desarrolló un estándar de compresión con el fin de almacenar videos en discos compactos.

Posteriormente, los estandares definidos fueron mejorando con nuevos algoritmos y técnicas de compresión, surgiendo así estándares como el H.263, el MPEG-2 o el H.264/MPEG-4 parte 10.

Estos estándares se basan en los conceptos de redundancias con el fin de comprimir los datos sin generar grandes pérdidas de calidad.

1.2 Objetivos

En este proyecto se va a realizar un estudio del estándar de codificación de video H.261. Para lograr este objetivo, se llevará a cabo un estudio de los diferentes algoritmos de compresión, con y sin perdidas, así como las técnicas que se pueden emplear, con los que sentar las bases del codificador.

Se realizará una implementación software del estándar para:

- Asentar bases de conocimiento sobre los codificadores de vídeo
- Servir como fuente para nuevas implementaciones
- Generar archivos de video comprimidos
- Ser implementado en aplicaciones de videoconferencia

1.3 Estructura

El trabajo se va a estructurar en los siguientes capítulos:

- En el capítulo 2 se presentará el estado del arte en el que se realizará una breve explicación del espacio de color, así como una introducción de las técnicas y los algoritmos de compresión que existen y de los diferentes estandares de codificación de imagen y video.

- En el capítulo 3 se explicará de forma detallada el estándar de codificación seleccionado y la implementación realizada.

- En el cuarto capítulo, se realizará un análisis de los resultados obtenidos.

- Por último, en el capítulo 5 se realiza la conclusión del trabajo.

Capítulo 2

Como se ha comentado anteriormente, el objetivo principal del proyecto consiste en realizar un codificador de video H261. Pero antes de presentar su desarrollo en profundidad, se ha de conocer que es un codificador, los tipos de codificadores que existen, tanto de imagen como de video, y los algoritmos de compresión que se pueden utilizar.

2.1 Códec de vídeo

Códec es el acrónimo de codificador-decodificador y se trata de un programa capaz de codificar o decodificar una señal de datos digitales. Este sistema de compresión se encargará de reducir los datos en un número más pequeño de bytes.

Estos sistemas de compresión se componen de una serie de algoritmos en función de los datos que se van a comprimir. Una vez obtenidos, se requiere de un descompresor para poder recuperar la señal original, valiéndose de algoritmos opuestos al caso de la compresión.

2.1.1 Técnicas de Compresión

Las técnicas de compresión usan la redundancia para reducir la cantidad de información a enviar.

La redundancia es una propiedad que indica que existen partes que pueden ser predichas en un mensaje, que no aportan información adicional o simplemente la repiten. Por tanto, a partir de la eliminación de está redundancia, se pueden comprimir los datos más importantes con los algoritmos que se describirán a continuación.

Existen tres tipos de redundancias, la espacial, la temporal y la estadística.

- Redundancia Espacial

  Este tipo de redundancia tiene en cuenta el valor de pixeles próximos de una imagen y que, en general, mantienen una gran similitud. Se asocia generalmente a objetos o paisajes que cuentan con superficies uniformes y, por tanto, no tendrán grandes cambios entre pixeles adyacentes.

  Esta característica, va a permitir enviar únicamente un valor codificado, en vez de mandar todo el conjunto de valores iguales.

  Para poder eliminar está redundancia se pueden emplear algoritmos como la transformada discreta del coseno.

- Redundancia Estadística

La redundancia estadística tiene en cuenta la repetición de ciertos valores en una trama, pudiéndose asignar un código en función del número de repeticiones.

Para eliminar la redundancia estadística, se pueden utilizar algoritmos como Run Length Code (RLC) o variable Length Code (VLC).

- Redundancia Temporal

La redundancia temporal, al contrario que los otros dos casos, se basa en la relación de pixeles entre diferentes imágenes de una secuencia de video. Esto se debe a que, en general, las variaciones entre fotogramas son pequeñas, ya que el video no se encuentra continuamente cambiando de plano, y por tanto, se puede suponer que un fotograma será similar al anterior y al siguiente.

De esta forma, las secuencias de video se suelen descomponer en "Key frame" y en "Delta frames".

Los "Key frame" son los fotogramas clave y en los cuales sólo se realizarán los algoritmos correspondientes a la eliminación de la redundancia espacial, mientras que en el resto de fotogramas se tratarán de suprimir todas las redundancias.

Para evitar el crecimiento del error en la codificación de los "Delta frames", de vez en cuando se ha de enviar un "Key frame" codificado.

La técnica típicamente empleada para la eliminación de la Redundancia Temporal se conoce como compensación de movimiento.

La compensación de movimiento es una técnica capaz de analizar el movimiento que realizan los objetos en la imagen. Para ello, se ha de buscar un bloque dentro de la imagen actual y se debe comparar con alguno o todos los bloques de la imagen de referencia, ya que los vectores de movimiento se pueden relacionar con la imagen completa o con partes específicas de ella. En esta técnica se elige un predictor y se resta al bloque actual, el de referencia, de forma que se obtiene un bloque residual que se codificará y se enviará.

  - Algoritmo Block Matching

    El algoritmo de Block Matching trata de localizar los macrobloques que coinciden en una secuencia de frames. Para ello, se ha de dividir el frame actual en macrobloques y compararlos con un bloque correspondiente y un marco cercano en la imagen de referencia.

    La estimación del movimiento determina los vectores de movimiento que muestran la transformación de las imágenes dentro de unos marcos cercanos. La aplicación de estos vectores para predecir el movimiento se denomina compensación de movimiento y se trata de una de las claves de la compresión

de video. Esta compresión permite enviar diferencias de imágenes codificadas, pero computacionalmente es muy costoso.

Dentro de este algoritmo, se pueden encontrar varios tipos:

- Búsqueda Exhaustiva

  Este algoritmo realiza la comparación de todos los macrobloques del frame actual con todos los macrobloques del frame de referencia. La imagen compensada es la mejor de todos los algoritmos de compensación, sin embargo, al ser una ventana de búsqueda tan extensa, computacionalmente es el más costoso de todos los mecanismos de búsqueda.

- Búsqueda Tres Pasos

  Se trata de un algoritmo rápido de búsqueda de coincidencia de bloques. Se comienza la búsqueda desde el centro y se buscan 8 ubicaciones en torno a esa posición inicial con paso S=4. De entre esas 9 ubicaciones resultantes, se escoge aquella que más se parezca al macrobloque seleccionado y se define un nuevo origen de búsqueda. Se repite la operación hasta que S=1 y esa será la ubicación de mayor semejanza.

2.1.2 Algoritmos de Compresión

Se conoce como algoritmo de compresión toda técnica de codificación cuya finalidad sea, como se ha mencionado con anterioridad, representar unos datos con el menor número de bits posible.

Estos algoritmos se clasifican en dos grandes grupos. Por un lado, se encuentran los algoritmos de compresión con pérdidas (lossy) y, por otro lado, los algoritmos de compresión sin perdidas (lossless).

El primer caso incluye aquellos procedimientos, que, una vez realizada la compresión, es imposible recuperar la señal original exacta. En el segundo caso, la señal reconstruida concuerda exactamente con la señal original [1].

Los algoritmos de compresión con pérdidas son los más comunes en la codificación, tanto de audio como de video, y esto se debe a que el ratio de compresión obtenido es mayor. Las pérdidas se pueden aceptar siempre y cuando la calidad de las imágenes decodificadas se encuentre en un rango aceptable.

Si se quiere conocer cuánto de cerca se encuentra la aproximación de la señal original se pueden emplear módulos de distorsión, siendo esta distorsión la diferencia numérica entre los datos originales y los reconstruidos [2].

La distorsión se puede medir con el error cuadrático medio (ECM)[1]. Esta medida de distorsión consiste básicamente en medir el cuadrado del promedio de los errores, siendo estos, la diferencia entre la señal original y la reconstruida.

$$\sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - y_n)^2$$

Ecuación 1: Error Cuadrático Medio

Sin embargo, en ocasiones se mide el error relativo de la señal, empleándose la relación señal a ruido (SNR), que relaciona el valor cuadrado medio de la señal original y el ECM.

$$SNR = 10\log_{10}\frac{\sigma_x^2}{\sigma_d^2}$$

Ecuación 2: Relación Señal a Ruido

Ambas son medidas de distorsión muy comunes en la compresión de imagen y evalúan el impacto de la cuantificación, proceso que introduce las pérdidas en la codificación.

2.1.2.1 Cuantificación

La cuantificación es un procedimiento que lleva implícito la perdida de información y cuyo objetivo es de reducir el rango de valores a codificar para aumentar el ratio de compresión. Existen varios tipos de cuantificación:

- Cuantificación no adaptativa:

  Se trata del cuantificador más sencillo de implementar. Sin embargo, los resultados que se obtienen no son los mejores y esto se debe a que las distancias entre los niveles de reconstrucción son siempre iguales, ya que no tiene en cuenta los datos a cuantificar.

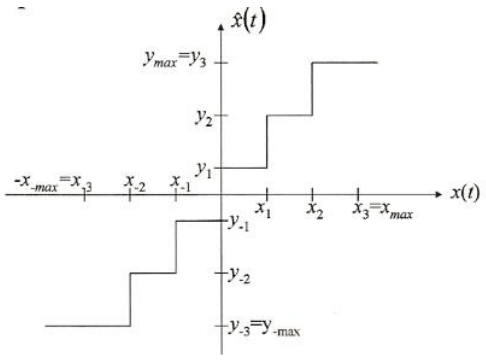


Figura 1: Cuantificador Uniforme [3]

[1] En inglés MSE (mean square error). Se empleará en el codificador seleccionado en este trabajo.

- Cuantificador adaptativo:

Para el caso en el que las entradas no se encuentren distribuidas de forma uniforme, se ha de emplear un cuantificador adaptativo, en el que los niveles se van modificando en función de los datos que llegan. Para obtener estos niveles, se puede utilizar el algoritmo de Max Lloyd, que define la función de densidad de probabilidad (fdp).

$$\sigma_q^2 = \sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} (x - y_i)^2 f_X(x) d_x$$

Ecuación 3: Error de cuantificación

Suponiendo el error de cuantificación de una fdp (ecuación 3), e igualando la derivada a 0, se obtiene la siguiente ecuación.

$$y_i = \frac{\int_{b_{j-1}}^{b_j} x f_x(x) d_x}{\int_{b_{j-1}}^{b_j} f_x(x) d_x}$$

Ecuación 4:Ecuación obtenida a partir de la derivada

A partir de cual, se puede obtener que el valor óptimo (b_j) para la reconstrucción es el centro ponderado del intervalo (ecuación 5).

$$b_j = \frac{y_{j+1} + y_j}{2}$$

Ecuación 5: Valor Óptimo Reconstruido

Existe además otra forma de obtener los niveles de un cuantificador no uniforme y consiste en la utilización de un compresor que transforme los datos de forma que parezca una distribución uniforme, para poder aplicar un cuantificador como el mencionado previamente y después emplear un expansor para deshacer el cambio.

Figura 2:Proceso Cuantificación no Lineal [4]

- Cuantificador Vectorial:

Este tipo de cuantificador se diferencia de los dos casos anteriores en que no considera las muestras como entradas independientes, sino que los datos se han de cuantificar en bloques de N muestras, empleando un proceso similar a los descritos previamente.

2.1.2.2 Transformada Discreta del Coseno (DCT)

La transformada discreta del coseno es una de las herramientas base en los estándares de compresión de imagen y video. Se emplea para tratar de eliminar la redundancia espacial de las imágenes. La DCT es una variante de la transformada discreta de Fourier, que permite descomponer las imágenes como sumas solamente de cosenos.

Esta técnica trata de concentrar gran parte de la información en unos pocos coeficientes. Esto supone una gran ventaja ya que a partir de la codificación de estos coeficientes se puede obtener la imagen y para ello se emplearía la transformada inversa.

Existen diferentes variantes básicas de la DCT unidimensionales, siendo la más utilizada la DCT-II.

$$DCT\ I: \quad y_j = \frac{1}{2}\left[x_0 + (-1)^j X_{N-1}\right] + \sum_{K=1}^{N-2} X_K \cos\left(\frac{\pi}{N-1} j\right)$$

$$DCT - II: \quad y_j = \sum_{K=0}^{N-1} X_K \cos\left(\frac{\pi}{N} j\left(k + \frac{1}{2}\right)\right)$$

$$DCT - III\ (IDCT): \quad x_k = \frac{1}{2} y_0 + \sum_{j=1}^{N-1} y_K \cos\left(\frac{\pi}{N}\left(k + \frac{1}{2} j\right)\right)$$

$$DCT - IV: \quad y_j = \sum_{K=0}^{N-1} X_K \cos\left(\frac{\pi}{N}\left(j + \frac{1}{2}\right)\left(k + \frac{1}{2}\right)\right)$$

Ecuación 6: Variantes DCT

En la ecuación de la DCT-II, se puede observar que y_j es la salida, x_k es la entrada, K es el índice de los coeficientes de salida calculados, desde 0 hasta N-1, N es el número de elementos a transformar. [6]

Componiendo dos o más funciones básicas de la DCT, se pueden formar transformadas de dos o más dimensiones.

- 1D DCT/IDCT:

Al contrario que con la DCT 2D, esta transformada únicamente trabaja con vectores de 8 elementos, de forma que requiere operaciones menos costosas

$$F(u) = \frac{C(u)}{2} \sum_{i=0}^{7} \cos \frac{(2i+1)u\pi}{16} f(i),$$

Ecuación 7: 1D DCT

$$\tilde{f}(i) = \sum_{u=0}^{7} \frac{C(u)}{2} \cos \frac{(2i+1)u\pi}{16} F(u),$$

Ecuación 8: 1D IDCT

computacionalmente. Sin embargo, la finalidad es la misma que en el caso anterior para descomponer la señal en sus componentes de Dc y Ac correspondientes.

La DCT convierte los valores f(i) en el dominio del tiempo a valores F(u) en el dominio de la frecuencia (ecuación 9), mientras que como se puede observar, la IDCT realiza la función contraria (ecuación 10). En ambos casos se emplean los mismos coeficientes descritos en la 2D DCT.

Existen casos particulares de la 1D DCT que mejoran el rendimiento, computacionalmente hablando, como son el algoritmo de Chen o el de Loeffler.

- 2D DCT/IDCT:

La 2D DCT trabaja con matrices de 8x8 elementos y trata de descomponer la imagen en una suma de frecuencias espaciales (ecuación 6), siendo los coeficientes los mostrados a continuación.

$$C_u = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } u = 0 \\ 1 & \text{else} \end{cases}$$
$$C_v = (\text{similar to the above})$$
$$F_{vu} = \frac{1}{4} C_v C_u \sum_{y=0}^{N-1} \sum_{x=0}^{N-1} S_{yx} \cos\left(v\pi \frac{2y+1}{2N}\right) \cos\left(u\pi \frac{2x+1}{2N}\right)$$

Ecuación 9: 2D DCT y coeficientes [8]

Una vez realizada la transformada, el resultado queda ordenado con un valor de DC (componente de continua) en la primera posición del array, siendo este, el elemento con menor frecuencia, seguido por el resto de coeficientes denominados AC (componente de alterna).

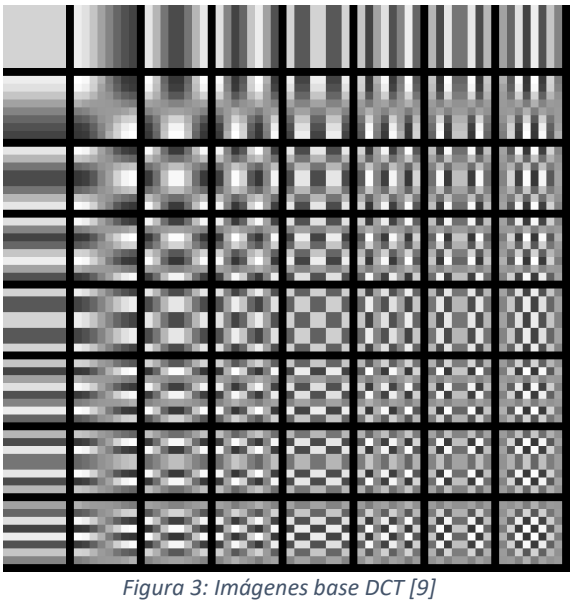


Figura 3: Imágenes base DCT [9]

Para las funciones de la DCT bidimensional, se emplean las imágenes base (Figura 3), siendo el blanco el correspondiente a los valores positivos y el negro para los negativos.

Como se observa en la imagen, tanto la primera columna como la primera fila, solo tienen variaciones de intensidad en una única dimensión, mientras que el resto de bloques posee cambios en ambas dimensiones. Los 64 productos que se obtienen producen una imagen de frecuencia espacial [2].

En el caso de la IDCT (Transformada inversa del coseno), la función es la misma, cambiando f(i,j) y F(u,v) e incluyendo los coeficientes (mismos valores que en el caso anterior) dentro de los sumatorios.

$$f(i,j) = \sum_{u=0}^{7}\sum_{V=0}^{7}\frac{C(u)}{C(v)}\cos\left(\frac{(2i+1)u\pi}{16}\right)\cos(\frac{(2j+1)u\pi}{16})F(u,v)$$

Ecuación 10: 2D IDCT

2.1.2.3 Codificación basada en Wavelet

Otra técnica que se puede utilizar como transformada de imagen en codificación de video es la transformada wavelet, la cual pretende representar señales con gran resolución tanto en el dominio del tiempo como en el de la frecuencia, a partir de unas funciones base.

El propósito de esta codificación radica en la descomposición de la señal de entrada en componentes más sencillos con los que trabajar y con los que ser capaces posteriormente de reconstruir la imagen original.

- Wavelet 1D:

Uno de los tipos de codificación wavelet es la transformada simple o 1D que consiste en sustituir una secuencia de entrada $X_{n,i}$ por el valor medio, $X_{n-1,i}$ (ecuación 10), y la diferencia, $d_{n-1,i}$, de cada par de valores consecutivos (ecuación 11), de forma que el resultado de ambas secuencias calculadas tenga el mismo número de elementos que la secuencia original [2].

$$X_{n-1,i} = \frac{X_{n,2i} + X_{n,2i+1}}{2}$$

Ecuación 11: Valor Medio Wavelet

$$d_{n-1,i} = \frac{X_{n,2i} - X_{n,2i+1}}{2}$$

Ecuación 12: Valor diferencias Wavelet

Esta nueva secuencia contará con los valores medios en la primera mitad y las diferencias en la segunda, de forma que se puede recuperar la secuencia original a partir de las siguientes ecuaciones.

$$X_{n,2i} = X_{n-1,i} + d_{n-1,i}$$

$$X_{n,2i+1} = X_{n-1,i} - d_{n-1,i}$$

Ecuación 13: Inversa Wavelet

Esta transformada recibe el nombre de transformada de Haar.

- Wavelet 2D

A wavelet 2D se emplea en procesamiento de imágenes como herramienta capaz de resolver análisis de imágenes, eliminación de ruido, segmentación y otros.

En este caso, en vez de considerar únicamente una secuencia de N elementos, se tiene una matriz de MxN elementos. Para poder realizar la codificación wavelet 2D, en primer lugar, se han de aplicar las mismas operaciones mencionadas anteriormente (ecuaciones 10 y 11) a cada par de columnas de la matriz de entrada, de forma que la matriz resultante sigue siendo una matriz MxN formada en su primera mitad por valores medios (L1) y en la segunda por las diferencias(H1).

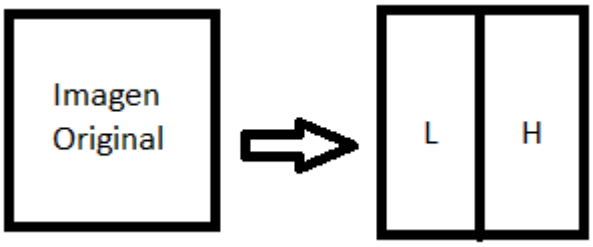


Figura 4: Paso 1 Wavelet 2D

Una vez obtenida esta nueva matriz, se vuelven a aplicar las mismas ecuaciones, pero esta vez por filas, de forma que se obtendrá una matriz como la que se observa a continuación.

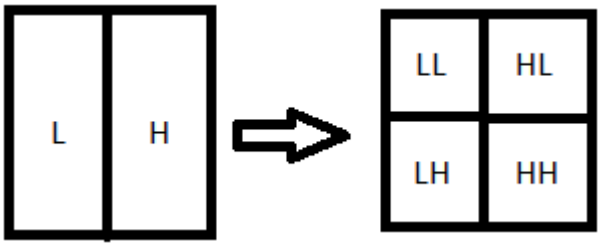


Figura 5: Paso 2 Wavelet 2D

Cada una de estas partes posee un cuarto del tamaño de la imagen original, siendo la matriz LL los coeficientes de aproximación, la LH horizontal, HL vertical y los

coeficientes de detalle corresponden a la parte HH. A continuación, se observa un ejemplo de esta codificación.

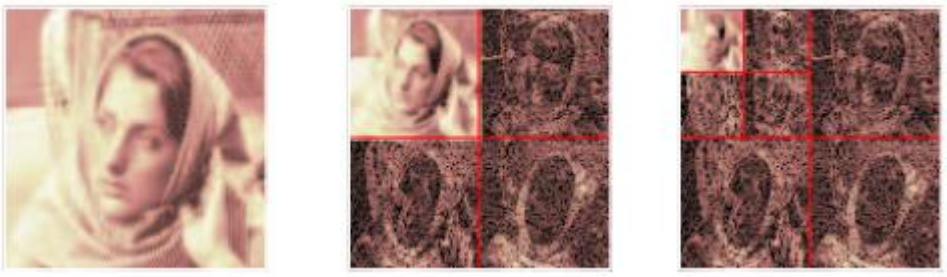


Figura 6: Ejemplo imagen comprimida en 3 escalas [11]

2.1.3 Algoritmos de codificación de Símbolos

Como se ha mencionado anteriormente, los algoritmos de compresión sin pérdidas o lossless son aquellos procedimientos que permiten codificar cierta información ocupando el menor espacio posible y a partir de los cuales se puede obtener una imagen reconstruida exactamente igual que la original.

- Run Length Coding (RLC)

Se trata de uno de los algoritmos de codificación más sencillos. Su función es comprimir con un único valor las secuencias de datos con el mismo valor, seguido del número de veces que se repite.

Si se dispone de una cadena de caracteres, AAACCEEEAAAAAANN, y se aplica el run length coding, se obtendría, 3A2C3E6A2N. Si se observan ambas cadenas, se puede ver como la cadena original poseía 16 caracteres y una vez aplicado el RLC, la cadena se queda en 10 caracteres.

Se trata de un algoritmo que funciona muy bien con imágenes en blanco y negro, pero no así en imágenes donde el color varí constantemente.

- Variable Length Coding (VLC)

El empleo del algoritmo VLC proporciona mayor eficiencia de codificación, ya que asigna códigos cortos a caracteres que aparecen con mayor frecuencia y códigos largos a aquellos caracteres que rara vez aparecen. Existen diversos tipos de algoritmos de este tipo, siendo los más comunes Shannon-Fano, Huffman, diccionario o el aritmético.

  - Codificación Shannon-Fano

Este algoritmo valora la probabilidad de cada carácter y le asigna un código con una longitud determinada. Se emplea mayoritariamente en sistemas

sencillos de recursos mínimos con un gran rendimiento, sin embargo, en comparación con otros algoritmos no tiene gran importancia, ya que su eficiencia es menor que empleando Huffman.

Para poder obtener el árbol de codificación Shannon-Fano, se ordenarán los caracteres de mayor a menor probabilidad, se harán dos grupos de una probabilidad similar y a uno de ellos se le codificará como "1" y al otro como "0". Este proceso se repetirá con cada grupo hasta que sólo quede un dato por grupo.

Si la palabra que se va a codificar es "HELLO", se han de ordenar los caracteres de mayor a menor número de apariciones, en este caso, L-2, H-1, E-1, O-1 y a continuación se realizarán los pasos mencionados en el párrafo anterior como se muestra en la siguiente imagen.

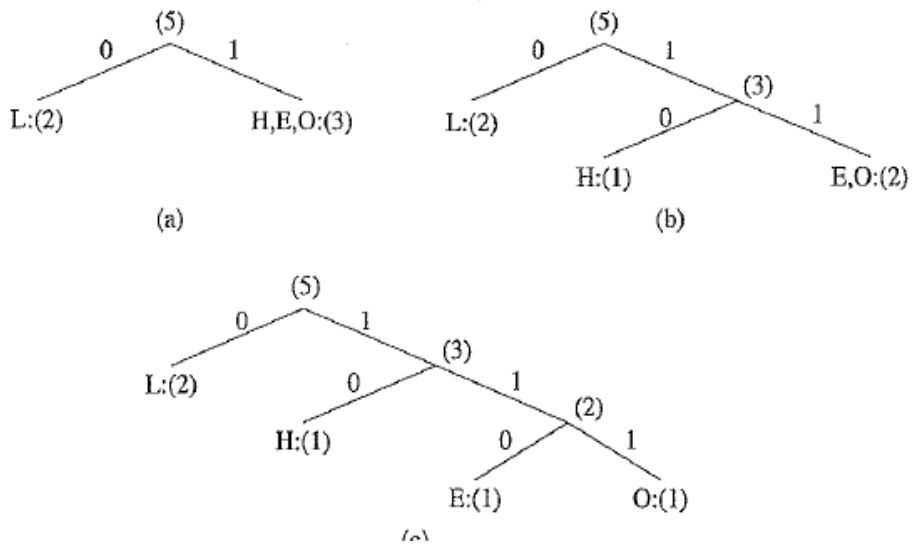


Figura 7: Formación Árbol Shannon-Fano [2]

Una vez terminada la codificación, se observan los siguientes resultados:

Symbol	Count	$\log_2 \frac{1}{p_i}$	Code	Number of bits used
L	2	1.32	0	2
H	1	2.32	10	2
E	1	2.32	110	3
O	1	2.32	111	3
TOTAL number of bits:				10

Figura 8: Resultados ejemplo Shannon-Fano [2]

Como se puede ver los datos de compresión que la codificación Shannon-Fano proporciona son satisfactorios, pero como se ha comentado con anterioridad, esta codificación se ha visto superada por Huffman.

- Codificación Huffman

El algoritmo de codificación Huffman fue propuesto a principios de los años 50 con la intención de codificar de forma simple los caracteres de entrada con un código de longitud óptima. Al igual que la codificación Shannon-Fano, Huffman es una codificación que genera un árbol binario, pero su formación se realiza de forma diferente, de la parte baja a la parte alta del árbol.

Para ello, se ordenan los caracteres de menor a mayor probabilidad de aparición, se cogen los dos datos menos probables, se les asigna un "0" y se forma un nuevo dato cuya probabilidad es la suma de las dos. Y de esta forma, se va repitiendo el proceso hasta codificar todos los caracteres y obtener un árbol como el mostrado a continuación.

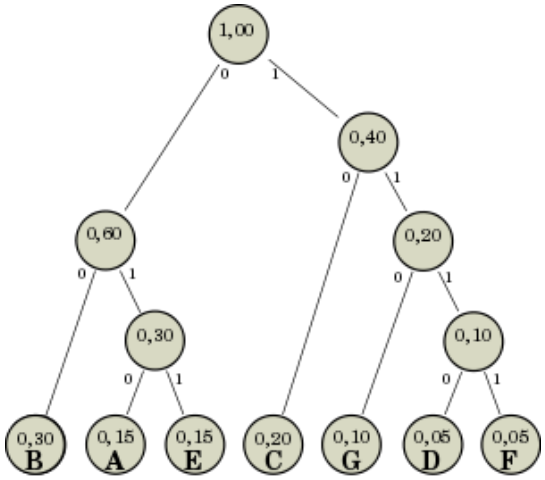


Figura 9: Ejemplo Árbol Huffman [13]

Sin embargo, para emplear el algoritmo de Huffman, se necesita conocer previamente las probabilidades de aparición de los caracteres, como ocurre en multimedia que se conocen los datos futuros antes de su llegada. Sin embargo, a pesar de conocer estas probabilidades, la tabla de caracteres codificados puede ser demasiado grande. Para solucionarlo, se emplea un método adaptativo de la codificación Huffman.

Este método adaptativo consiste en ir creando el árbol a medida que se van leyendo los datos de entrada, de esta forma se realiza una compresión más rápida, aunque sacrificando parte de la razón de compresión.

- Codificación basada en diccionario (LZW)

Se trata de un algoritmo desarrollado por Welch para mejorar el de Lempel y Ziv. Este algoritmo emplea códigos de longitud fija para representar los símbolos que suelen aparecer juntos. De la misma forma que el Huffman adaptativo, el algoritmo LZW va introduciendo de forma iterativa los datos en

el diccionario mientras que el transmisor y el receptor generan el diccionario de forma dinámica.

Para generar la compresión se emplea el siguiente pseudocódigo:

```
BEGIN
   s = next input character;
   while not EOF
       { c = next input character;

         if s + c exists in the dictionary
            s = s + c;
         else
           { output the code for s;
             add string s + c to the dictionary with a new code;
             s = c;
           }
        }
   output the code for s;
END
```

Figura 10: Pseudocódigo Compresión LZW

Y el pseudocódigo correspondiente a la descompresión es:

```
BEGIN
   s = NIL;
   while not EOF
       {
         k = next input code;
         entry = dictionary entry for k;
         output entry;
         if (s != NIL)
            add string s + entry[0] to dictionary with a new code;
         s = entry;
       }
END
```

Figura 11: Pseudocódigo Descompresión LZW

- Codificación Aritmética

Este tipo de codificación es considerada como el mejor método para codificar símbolos según su probabilidad de aparición. La longitud del código corresponde con el mínimo posible y considera todo el mensaje como una única unidad. Cada unidad se representa mediante un intervalo semiabierto entre 0 y 1. A medida que el mensaje crece, este intervalo se va reduciendo, pero el número de bits necesarios para codificarlo aumenta.

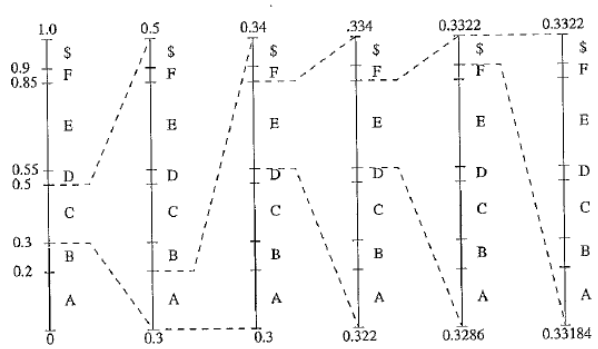


Figura 12:Intervalos Codificación Aritmética

A medida que este rango se va reduciendo, se necesitan números de gran precisión para hacer la codificación y esto supone que su implementación es muy compleja.

A diferencia de la codificación Huffman, la codificación aritmética ofrece una tasa de compresión más alta, sin embargo, su uso se encuentra restringido por patentes [14].

2.2 Espacio de Color

Antes de comenzar con la explicación de los diferentes Estándares de Compresión de imagen y video, se ha de tener unos conocimientos base del espacio y el modelado del color.

La luz es una onda electromagnética y su espectro contiene todas las longitudes de onda. Este espectro es muy grande y el ser humano no es capaz de visualizarlo todo, solamente puede percibir la franja denominada Espectro visible (figura 13).

Por tanto, cuando se habla de color, se está hablando ni más ni menos de una longitud de onda perteneciente al espectro visible. El ser humano es capaz de percibir el color focalizando la luz en la retina. En ella, se encuentran dos tipos de fotorreceptores, los conos y los bastones.

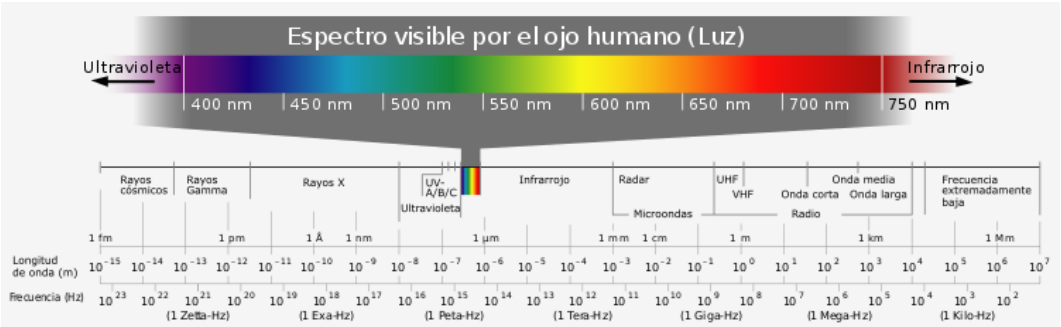


Figura 13: Espectro Radioeléctrico [13]

Los bastones son los encargados de generar imágenes en escala de grises y son activos cuando los niveles de luz son reducidos.

Los conos, por el contrario, son activos con altos niveles de luz y se pueden dividir en tres clases en función de la longitud de onda a la que son más sensibles, rojo (R), verde (G) y azul (B).

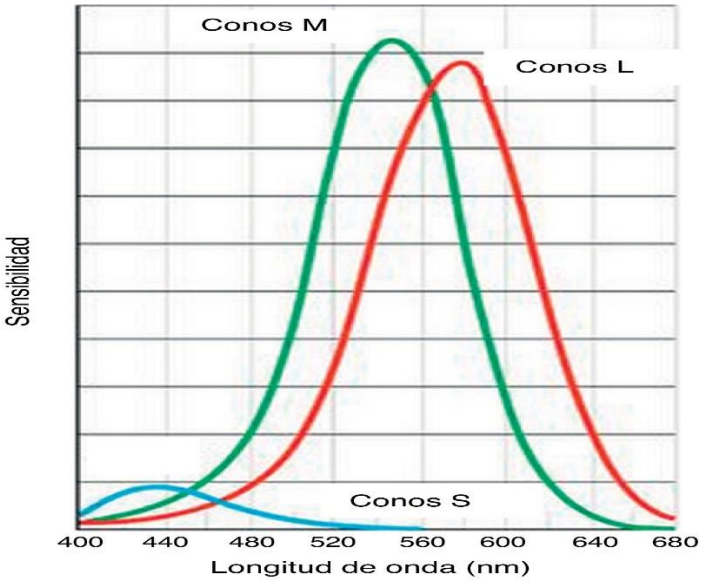


Figura 14: Sensibilidad Ojo

El cerebro humano es capaz de emplear las diferencias de estas componentes para generar otros colores, así como combinarlos todos para generar un canal acromático[2].

Como se puede observar en la imagen 14, el ojo humano es menos sensible al color azul y más sensible a las frecuencias centrales del espectro visible (Verde y Rojo).

2.2.1 Modelo RGB

Se trata de espacio de color más común en sistemas de imagen, se forma a partir de la composición de los tres colores primarios (rojo, verde y azul). Es un espacio aditivo, es decir, se forman nuevos colores a partir de la adición de los tres.

A continuación, se muestra representado en un cubo el conjunto de todos los colores. En la formación de una imagen, a cada pixel se le asocian 3 componentes del color (R, G, B) con 8 bits por color, de forma que el rango de intensidad se encontrará entre 0 y 255.

[2] Que no tiene color.

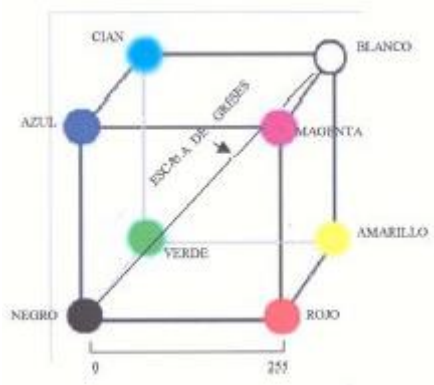


Figura 15: Conjunto Colores RGB

Cuando se habla de la formación de una imagen, se ha de tener en cuenta que las superficies claras reflejan mayor cantidad de luz que las superficies oscuras. Cuando una luz E(λ), se impregna sobre una superficie S(λ), esta es reflejada con una determinada señal de color C(λ) y es filtrada por los conos.

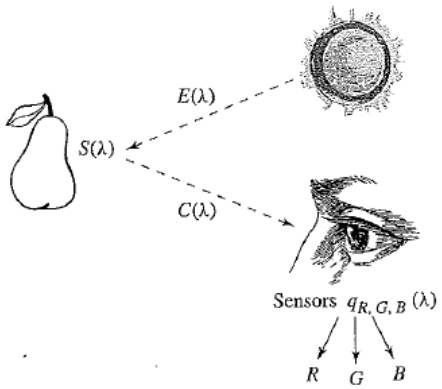


Figura 16: Esquema Formación Imagen [2]

2.2.2 Modelo CIE[3] XYZ

Se trata de uno de los primeros espacios de color definidos de forma matemática. Se emplea para definir los colores y otros espacios de color que el ojo humano puede percibir.

En este modelo, las componentes XYZ se pueden generar a partir de unas ecuaciones que las relacionan con las componentes de color del espacio RGB.

$$X = 0.490R + 0.310G + 0.200B$$

$$Y = 0.177R + 0.812G + 0.011B$$

$$Z = 0.010G + 0.990B$$

Ecuación 14: Transformación RGB a XYZ

[3] CIE: Comission Internationale de l'Eclairage

La Y corresponde con la luminosidad, la X se asocia con la curva de sensibilidad del rojo al verde y la Z se trata del estímulo del color azul, es decir, los parámetros XZ corresponden con la cromaticidad del color.

A partir de esto, se puede definir la cromaticidad como:

$$x = \frac{X}{X + Y + Z}$$

$$y = \frac{Y}{X + Y + Z}$$

$$z = 1 - x - y$$

Ecuación 15: Cromaticidad XYZ

Por tanto, se puede representar el color conociendo únicamente x e y, a partir del diagrama de cromaticidad (Figura 17).

En este diagrama, se puede observar un triángulo que delimita los colores que quedan fuera de rango y que se aproximan por el color que corresponde al punto donde se cruzan la línea que une el punto blanco (D65) y el color con la recta del triángulo.

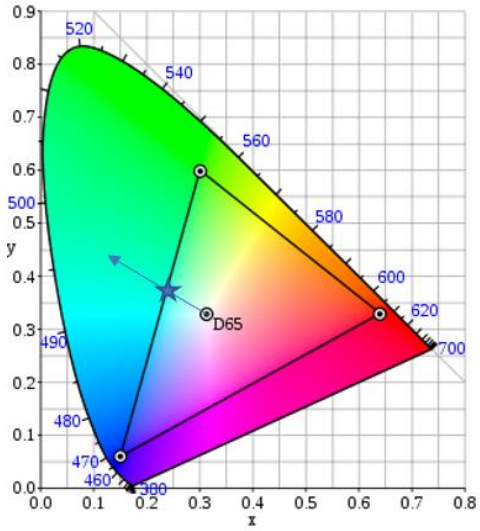


Figura 17: Diagrama de Cromaticidad

2.2.3 Modelo CMY

Al contrario que el modelo RGB, este modelo se basa en una síntesis sustractiva del color. Sus siglas se forman a partir de los colores Cyan, Magenta y Yellow (amarillo) y son las cantidades que se restan al color blanco.

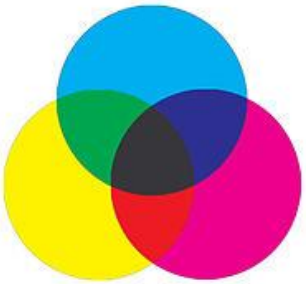


Figura 18: Modelo CMY

Su modelado se realiza de forma simple a partir de las siguientes ecuaciones:

$$\begin{matrix} C \\ M \\ Y \end{matrix} = \begin{matrix} 1 \\ 1 \\ 1 \end{matrix} - \begin{matrix} R \\ G \\ B \end{matrix}$$

Ecuación 16: Transformación RGB a CMY

Un modelo derivado del CMY es el conocido CMYK, que además de los colores ya mencionados, incluye el negro. Este modelo se emplea en las impresoras, ya que de esta forma ahorran tinta al añadir un cartucho de color negro y no tener que escribirlo sumando los otros tres colores.

2.2.4 Modelo YUV

Este modelo define el espacio de color que se emplea generalmente en video analógico PAL[4]. La luminancia viene definida por Y, mientras que la crominancia viene dada por los valores U y V. Se relaciona con el modelo RGB a partir de las siguientes ecuaciones:

$$\begin{matrix} Y' \\ U \\ V \end{matrix} = \begin{matrix} 0.299 & 0.587 & 0.144 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{matrix} * \begin{matrix} R \\ G \\ B \end{matrix}$$

Ecuación 17: Transformación RGB a YUV

En el caso en que la crominancia U=V=0, la Y' representa una imagen en blanco y negro.

2.2.5 Modelo YIQ

Este modelo es un espacio de color empleado por el estándar americano de TV analógica NTSC[5].

La luminancia del modelo YIQ es la misma que en el caso anterior, mientras que los valores I y Q son una rotación de 33º respecto U y V y se calculan mediante:

[4] Línea de fase alternada. Codificación empleada en TV analógica en Europa.
[5] Comité Nacional de Sistema de Televisión

$$\begin{matrix} Y' \\ I \\ Q \end{matrix} = \begin{matrix} 0.299 & 0.587 & 0.144 \\ -0.599 & -0.2773 & -0.3217 \\ 0.213 & -0.5251 & 0.3121 \end{matrix} * \begin{matrix} R \\ G \\ B \end{matrix}$$

Ecuación 18: Transformación RGB a YIQ

2.2.6 Modelo YCbCr

Se trata de un estándar de video, que no depende del tipo de codificación de TV como en los dos casos anteriores y está formado por una luma (Y) y dos cromas, la croma azul (Cb) y la croma roja (Cr).

Las componentes de croma de este modelo tienen menos detalle que en el resto de modelos y es por eso, por lo que se suele emplear en la compresión de video.

El ser humano ve el color con menor resolución que la luminosidad y es por esto, que se necesita reducir el número de muestras de croma, lo que se conoce como Submuestreo.

- Submuestreo 4:4:4 → No se realiza Submuestreo de la croma, por lo que habrá 4 muestras de luma (Y), 4 de croma azul (Cb) y 4 de croma roja (Cr).

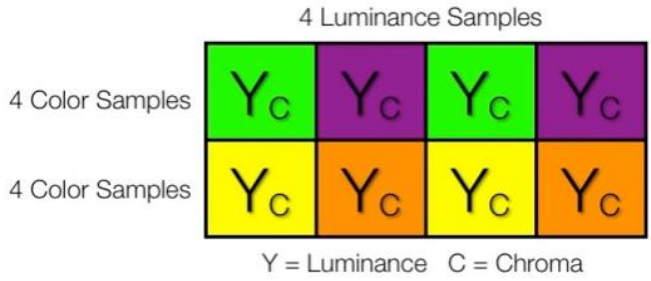


Figura 19: Submuestreo 4:4:4 [16]

- Submuestreo 4:2:2 → Se trata de un Submuestreo horizontal de factor 2, por lo que habrá 4 muestras de luma (Y), pero la croma reduce la resolución de color a la mitad. A pesar de ello, la calidad no se ve degradada notablemente, incluso se puede adaptar mejor a la sensibilidad de ojo humano.

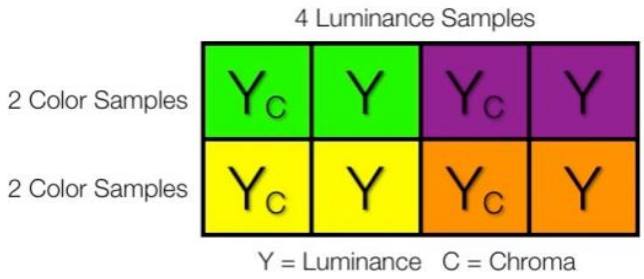


Figura 20: Submuestreo 4:2:2

- Submuestreo 4:1:1→ Se realiza un Submuestreo horizontal de factor 4. Seguirá habiendo 4 lumas, pero la información de croma se ve reducida a una por línea.

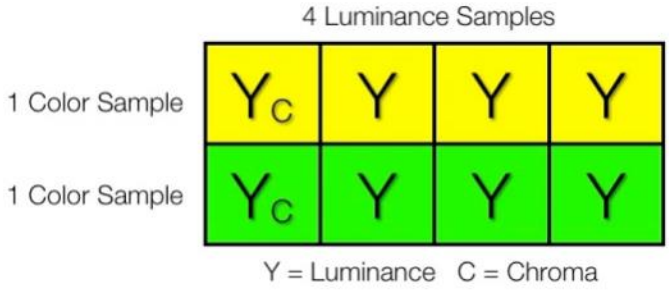


Figura 21: Submuestreo 4:1:1

- Submuestreo 4:2:0 → Se realiza un Submuestreo de factor dos en la horizontal y en la vertical.

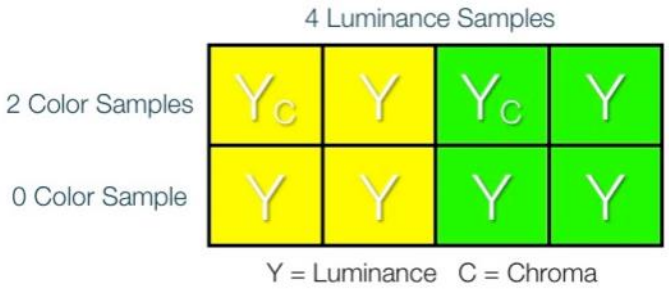


Figura 22: Submuestreo 4:2:0

2.3 Estándares de Compresión de Imagen

Las imágenes digitales han ido creciendo con el paso de los años y esto se debe en gran medida a que el número de dispositivos digitales también ha aumentado considerablemente. Esto ha supuesto, la necesidad imperativa de procesar y almacenar un gran número de imágenes digitales. Debido a esto, surgieron diferentes estándares de compresión de imagen en función de las necesidades del usuario.

2.3.1 Estándar JPEG

JPEG es un estándar de compresión y codificación de imagen cuyas siglas significan Joint Photographic Experts Group.

Este formato emplea algoritmos de compresión con pérdidas, como los mencionados anteriormente, y, por tanto, no se podrá obtener la imagen original exacta en el momento de su descompresión.

JPEG permite ajustar el grado de compresión deseado, de forma que, si se quiere obtener un archivo muy pequeño, el grado de compresión será elevado, sacrificando en el proceso gran parte de la calidad.

A continuación, se muestra una imagen del diagrama de bloques del codificador JPEG:

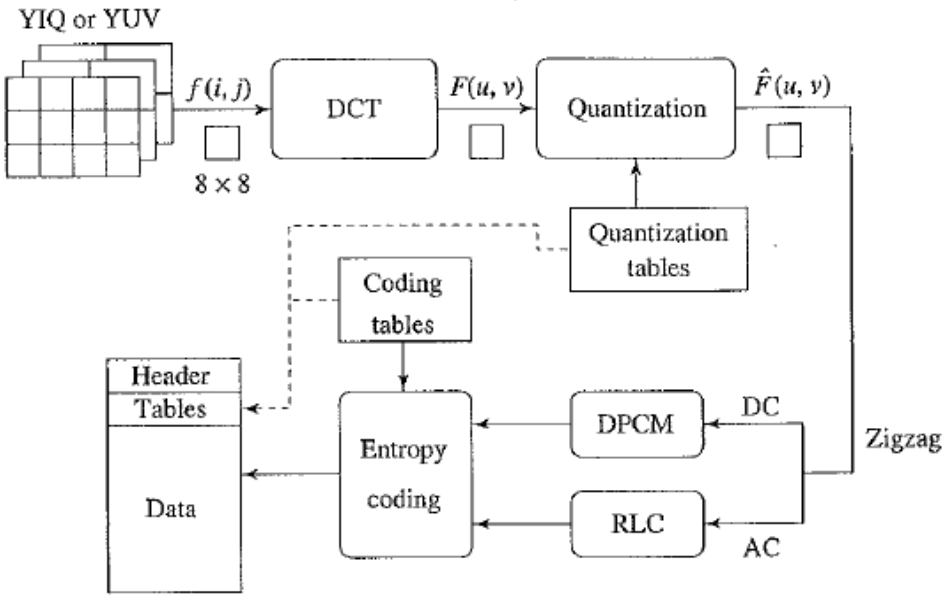


Figura 23: Diagrama Códec JPEG

Inicialmente se dispone de una imagen RGB que se ha de transformar al modelo YIQ o YUV como se mencionó anteriormente. A continuación, se puede realizar un Submuestreo de las cromas, teniendo en cuenta la pérdida de calidad que eso conlleva. A continuación, se divide la imagen en macrobloques de 8x8 elementos, de forma que la computación será más rápida.

Cada uno de estos macrobloques pasará por los diferentes bloques del diagrama (figura 25). En primer lugar, se aplicará la DCT con el fin de reducir los elementos de alta frecuencia, empleando alguna de las técnicas ya comentadas, y quedarse así concentrar la información en los elementos de baja frecuencia que son los que más importan.

El paso siguiente consiste en llevar a cabo la cuantificación al macrobloque resultante de la DCT, con el fin de reducir el número total de bits para la imagen comprimida. Para ello se emplean dos tablas de cuantificación, una para la luminancia y otra para la crominancia.

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

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Tabla 1: Tablas cuantificación JPEG Luminancia y crominancia [2]

Tras la cuantificación, se obtiene un macrobloque de 8x8 en el que el primer término será el coeficiente de DC y el resto los coeficientes AC. Solo habrá un coeficiente de DC por cada macrobloque. Esta matriz resultante se ordenará mediante el método de ordenación en zigzag que obtendrá secuencias largas de ceros.

Para codificar los valores AC, se empleará el método conocido de run length coding (RLC). En el caso de los coeficientes DC, se utilizará la codificación DPCM (Differential Pulse Code Modulation) mediante las ecuaciones:

$$d_i = DC_{i+1} - DC_i$$

$$D_0 = DC_0$$

Ecuación 19: Codificación DPCM

Finalmente se realiza la codificación de entropía para todos los coeficientes, con el fin de aumentar la compresión, utilizando el método de Huffman.

Cada coeficiente DC codificado se representa mediante un par de símbolos (tamaño, amplitud), siendo el tamaño el número de bits con los que se ha de representar y la amplitud, el valor del número codificado [2].

En el caso de los coeficientes AC, se envían dos símbolos de 4 bits. El primer símbolo indica el número de ceros seguidos en la trama y el segundo símbolo, el valor del primer elemento no nulo.

Dentro del estándar de codificación de imagen JPEG se pueden encontrar modificaciones que tratan de mejorar la codificación. A continuación, se mencionan dichas modificaciones:

- Estándar JPEG sin pérdidas:

  Se trata de un añadido al estándar original para permitir la compresión sin perdidas. Para ello, se utiliza un modelo de predicción sin perdidas a partir de la modulación de código de pulso diferencial (DPCM).

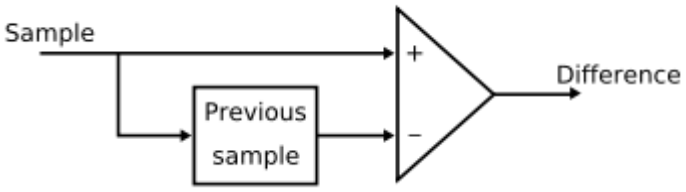


Figura 24: JPEG sin pérdidas

  Se basa en la predicción de valores de las muestras cercanas que han sido previamente codificadas. De esta forma, se genera una imagen residual que se codificará y se enviará.

- Estándar JPEG 2000:

Se trata de la versión modificada en el año 2000 del estándar JPEG y actualmente es uno de los mejores estándares para compresión de imagen. Esto se debe en gran parte a que los ratios de compresión son mayores sin generar imágenes tan borrosas como pasaba con JPEG.

Este estándar se caracteriza por emplear la transformada wavelet en lugar de utilizar la transformada discreta del coseno que se usaba en el estándar original.

2.3.4 Estándar JBIG

JBIG es un estándar del Joint Bi-Level Image Processing Group para imágenes binarias. Se trata de un estándar sin perdidas cuya aplicación principal es la codificación de imágenes escaneadas y de fax. Se basa en la codificación aritmética.

2.4 Estándares de Compresión de Video

Al igual que ocurría con las imágenes, el ser humano también se ha visto con la necesidad de comprimir y transmitir videos. Pero como ya hemos visto anteriormente, estos videos introducen redundancias que no aportan información y que se deben eliminar. Por ello, se han desarrollado diferentes estándares a lo largo de los años con el fin de cumplir esta función.

Estos estándares, proporcionan una gran compresión de video a partir de diferentes algoritmos que minimizan la redundancia espacial y la redundancia temporal.

Un video, se trata simplemente de una secuencia ordenada de imágenes y, por tanto, se podría aplicar una codificación predictiva a partir de imágenes anteriores.

A continuación, se explicarán de forma abreviada algunos de los estándares de codificación más sencillos que se pueden implementar.

2.4.1 MJPEG

El Motion JPEG (JPEG en movimiento) es una amplificación del estándar JPEG de imágenes aplicada al tratamiento de video. Consiste, básicamente, en enviar imágenes codificadas en formato JPEG de forma consecutiva, sin embargo, este estándar únicamente trata la redundancia espacial de cada imagen, pero no minimiza la redundancia temporal debido a que los frames son codificados de forma independiente.

2.4.2 H.120

Se trata del primer estándar de compresión de video desarrollado en los años 80 por el CCITT[6]. Este estándar consistía en una cuantificación escalar y una posterior codificación de longitud variable (VLC) y a pesar de proporcionar muy buena resolución espacial, la calidad temporal era nefasta y por ello, no tuvo gran éxito. Sin embargo, sentó las bases para su sucesor, el codificador de video h.261.

2.4.3 H.261

H.261 es considerado realmente el primer estándar de compresión de video. Introdujo el empleo de compresión mediante compensación de movimiento que posteriormente, se emplearía en el resto de codificadores.

El h.261 fue desarrollado originalmente para aplicaciones de videoconferencia debido a que emplea un algoritmo que optimiza el ancho de banda pudiendo realizar videoconferencias bidireccionales en tiempo real si el codificador tiene un retraso mínimo. A diferencia de los estandares de imagen, el H.261 no solo permite reducir la redundancia espacial, sino que también reduce la redundancia temporal a partir de la predicción de frames utilizando estimación de movimiento.

Por tanto, este estándar permitirá codificar dos tipos de frames, los intra frames (I-frame) que son imágenes completas a las que solo se le aplica la redundancia espacial y los inter frames (P frames), en los que se reducirá también la redundancia temporal.

2.4.4 H.263

H.263 es un estándar de codificación de video mejorado del estándar H.261. Emplea, tanto en los I frames como en los P frames, la codificación de transformada con el fin de reducir la redundancia espacial y la codificación predictiva en los p frames para la reducción de la redundancia temporal.

A diferencia del H.261 que solamente soportaba dos formatos (Capítulo 3), el H.263 es capaz de soportar hasta 5 formatos.

Formato de imagen	Número de píxels de luminancia (dx)	Número de líneas de luminancia (dy)	Número de píxels de crominancia (dx/2)	Número de líneas de crominancia (dy/2)
sub-cuarto de CIF	128	96	64	48
cuarto de CIF	176	144	88	72
CIF	352	288	176	144
4 veces CIF	704	576	352	288
16 veces CIF	1408	1152	704	576

Tabla 2: Formatos Entrada H.263 [18]

[6] Comité Consultivo Internacional Telegráfico y Telefónico

Este estándar se diferencia del H.261 en que los GOB no poseen un tamaño fijo y empiezan y terminan a izquierda y derecha de los bordes de la imagen [2].

El H.263 emplea la precisión de medio pixel en la compensación de movimiento, a diferencia del estándar anterior que usaba la precisión de pixel completo.

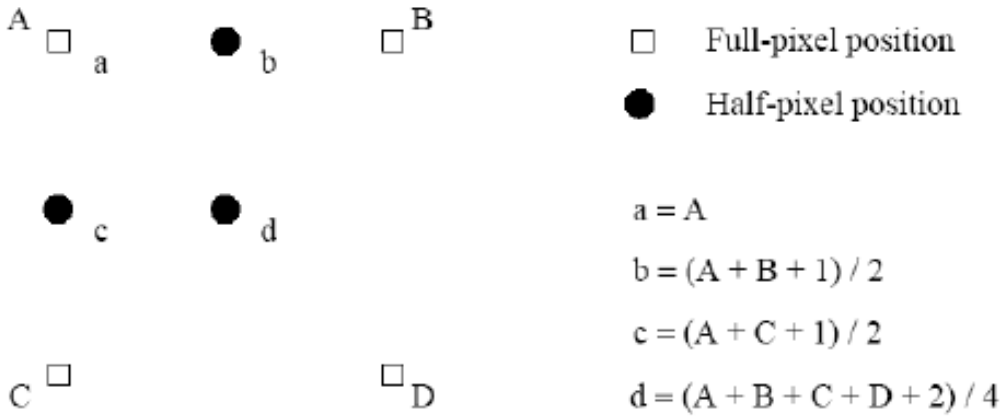


Figura 25: Precisión Medio Pixel

Esta aproximación se realiza con el fin de minimizar el error en la predicción.

2.4.5 MPEG

MPEG son las siglas de Motion Picture Experts Group y se trata de un estándar establecido a finales de los años 80 para la transmisión de señales de video y tv digital con imágenes a altas velocidades. A diferencia de los estándares H.261 y H.263, este estándar no se diseñó para realizar videoconferencias, sino que se emplea en videos ya grabados previamente.

A diferencia del H.261, emplea estimación de movimiento bidireccional comparando los fotogramas actuales tanto con los anteriores como con los siguientes, de forma que se consigue un mayor ratio de compresión.

Dentro de MPEG se pueden encontrar varios estándares:

- MPEG-1:

  Se trata del primer estándar del grupo MPEG de principios de los 90 y estaba orientado al empleo de videoCD.

  Este estándar incluye un tercer tipo de frame en la estimación de movimiento, denominado B frame. A continuación, se muestra un ejemplo del esquema básico por el que se rigen los nuevos frames introducidos.

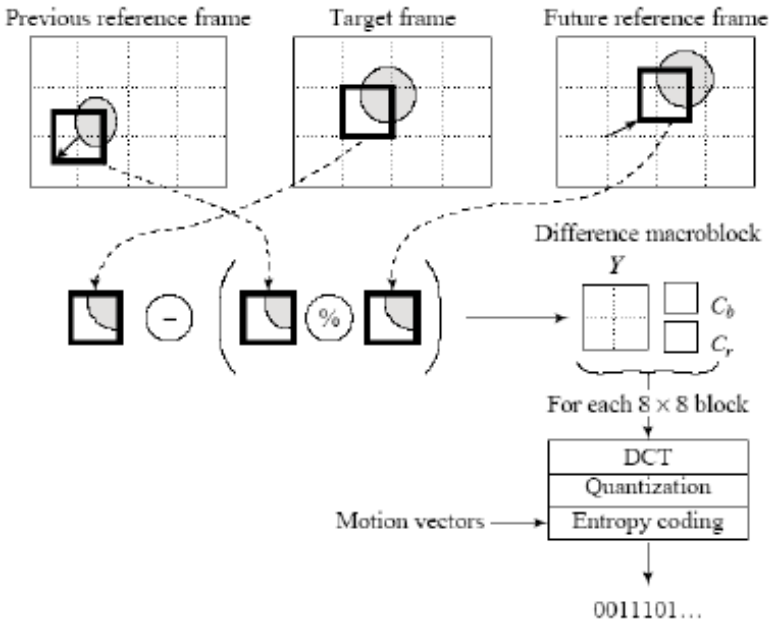


Figura 26: Formación B frames [2]

Como se puede observar en la imagen, cada B frame proporciona dos vectores de movimiento, uno relacionado con el frame anterior y otro con el siguiente, que serán promediados con el fin de generar el frame diferencia. Una vez obtenido este frame, se realizará la codificación vista anteriormente.

Para realizar la cuantificación, se ha de diferenciar entre los I y los P frames, puesto que las tablas de cuantificación son diferentes.

8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16

Tabla 3: Tablas Cuantificación MPEG Luminancia y Crominancia [2]

Teniendo clara la diferencia, se calculan los coeficientes cuantificados a partir de la siguiente expresión:

$$QDCT[i,j] = round(\frac{8 * DCT[i,j]}{Qx[i,j] * scale})$$

Ecuación 20: Cuantificación MPEG

Donde la Qx referenciará a una de las dos tablas de cuantificación en función del tipo de frame y scale es un entero entre 1 y 31.

El tamaño de los P frames era bastante menor que el de los I frame, sin embargo, el tamaño de los B frames es todavía más pequeño que el de los P.

En el caso del codificador H.261, había que enviar un I frame de forma secuencial para evitar que el error causado por la predicción de los P frame no se hiciera muy grande.

En este caso, se sigue una lógica similar. Se comienza enviando un frame I, seguido de varios frame B o P intercalados y cada X frames se vuelve a enviar un I.

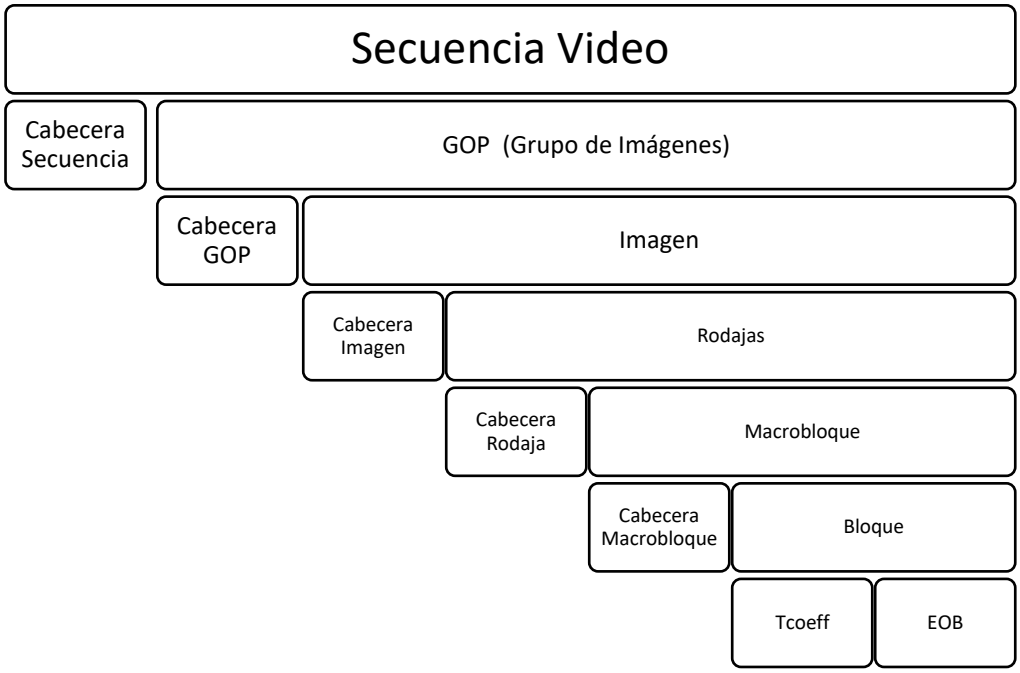


Figura 27: Jerarquía Stream MPEG

La sintaxis del stream de MPEG viene dado por la jerarquía de 6 capas mostrada anteriormente.

En ella se ha de especificar la capa de secuencia que estará formada por uno o más GOPS (Grupo de imágenes). Cada GOP contiene una o más imágenes, las cuales podrán ser de tipo I, P o B. Cada imagen quedará dividida en una o varias porciones donde se encontrarán los macrobloques (4 lumas, 1 Cb, 1Cr) y finalmente, en la última capa se codificarán los coeficientes y un final de bloque.

- MPEG-2

El MPEG-2 se desarrolla 3 años más tarde que su predecesor con el fin de proporcionar mayor calidad y resolución a los videos.

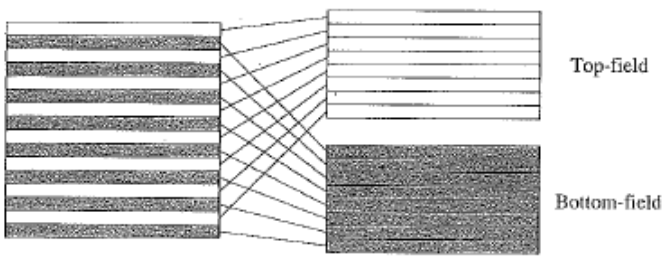


Figura 28: Ejemplo imagen entrelazada

Este estándar permite video entrelazado, de forma que cada frame queda dividido en dos campos (figura 32). En un frame, todas las líneas de escaneado se interrelacionan para formar un solo marco, una vez hecho esto, se dividen en macrobloques de 16x16 y se realiza la codificación por compensación de movimiento [2].

2.4.5 H.264

El estándar H.264 o también conocido como MPEG-4 parte 10 se trata de un codificador avanzado de video. Este estándar se diseñó con la idea de mejorar la eficiencia de los anteriores codificadores a partir de una estimación de movimiento mejorada. Esta mejora consiste en una compensación de movimiento estructurada en árbol con numerosos marcos de referencia y en el empleo principalmente de precisión de un cuarto de pixel.

2.5 VLC Media Player

Se trata de un reproductor multimedia de código abierto capaz de reproducir la mayoría de archivos multimedia, así como la mayoría de los códecs de H.261, H.263, MPEG-2, H.264 [21].

En este trabajo, se proporciona una salida codificada en formato .h261 para su visualización con este reproductor.

2.6 Eclipse

Se trata de una plataforma de código abierto basada en java, que soporta no sólo este lenguaje de programación, sino que incluye soporte para otros lenguajes como C/C++ y que se empleará en la realización de este trabajo [19].

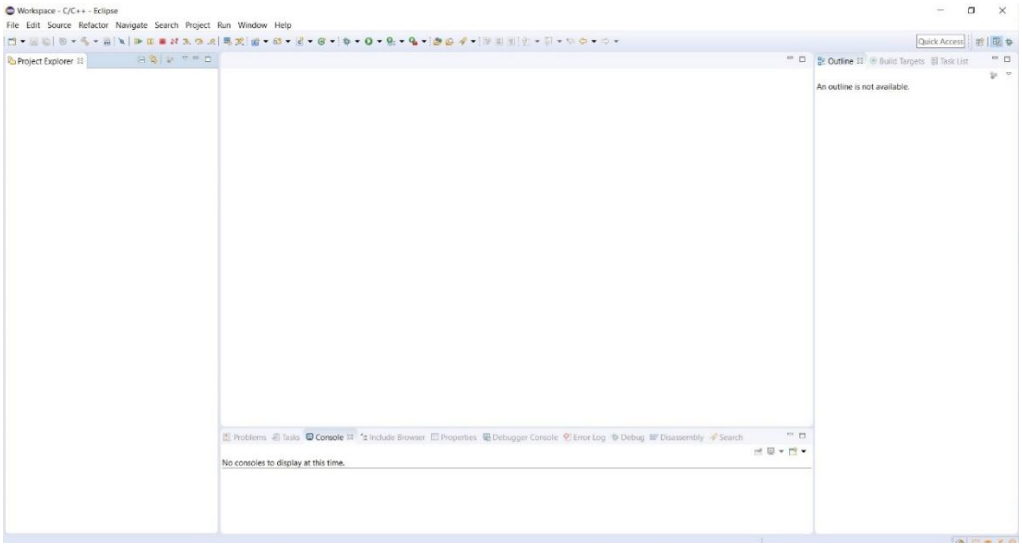


Figura 29: Entorno de trabajo Eclipse

2.7 OpenCV

OpenCV[7] es una librería "Open Source" bajo licencia BSD[8] que permite un uso académico y comercial gratuito. El objetivo principal de OpenCV consiste en proporcionar librerías que mejoren el computo de las aplicaciones en tiempo real [20]. Para el desarrollo de este trabajo se han empleado las siguientes librerías:

- **Core:** Es la librería encargada de definir las funciones básicas que emplean el resto de librerías.
- **Highgui:** Es la encargada de realizar la interfaz de usuario. Permite capturar/escribir imágenes y videos.
- **Imgproc:** Es la librería encargada de procesar las imágenes.
- **Videoio:** Esta librería proporciona una interfaz sencilla para capturar video de la cámara o de archivo y escribir video en un archivo.
- **Video:** Se encarga de realizar el análisis de video y la estimación de movimiento.

[7] Open Source Computer Vision Library
[8] *Berkeley Software Distribution. Licencia de software libre permisiva*

Capítulo 3

3.1 Implementación del Codificador H.261

Como ya se comentó en el capítulo anterior, el codificador H.261 fue de los primeros estándares de codificación de video que se desarrollaron.

Se trata de un codificador que funciona muy bien con secuencias de imágenes pequeñas y con poco movimiento y por eso es ideal para aplicaciones de videoconferencia. Puede soportar bit-rates de px64 KBits, siendo p un valor entre 1 y 30, y el empleo de velocidades binarias hasta 2MB/s [17].

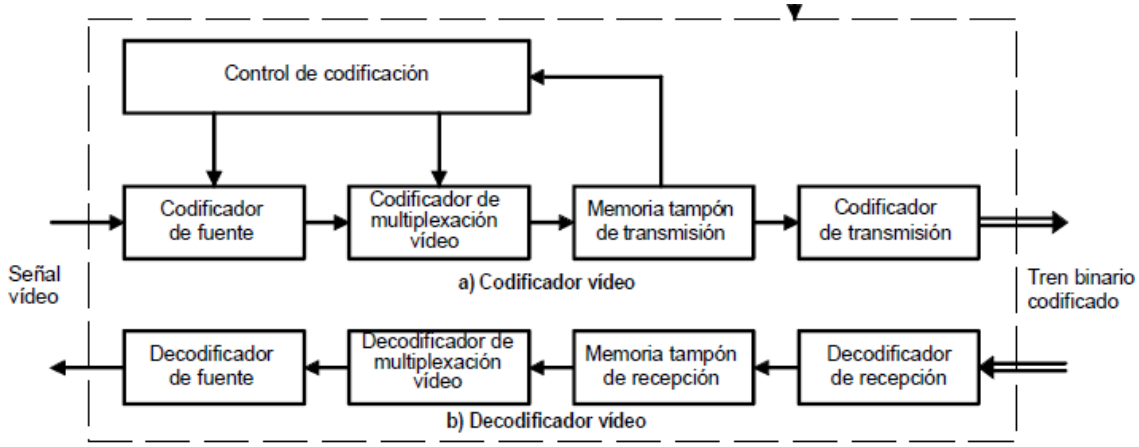


Figura 30: Diagrama de Bloques del Códec H.261

Es un estándar que únicamente soporta dos formatos de imagen, CIF, que posee una resolución de luminancia de 352x288 y una resolución de crominancia de 176x144 (submuestreo 4:2:0) y QCIF, cuya resolución de luminancia es de 176x144 y de crominancia es de 88x72. Para este trabajo las imágenes utilizadas son en formato CIF.

En este estándar se emplean dos tipos de frames, los Intra-frames (I), que son tratados de forma independiente y se codifican de forma similar que los estándares de compresión de imagen, de forma que se elimina la redundancia espacial, y por otro lado están los Inter-frames (P), que se codifican mediante un método de codificación predictiva, con el fin de reducir la redundancia temporal.

Secuencialmente se ha de codificar un frame I, para que el error introducido en la codificación de los frame P, no se incremente excesivamente.

Como se observa en la figura 29 (diagrama de bloques), el estándar se puede dividir en dos partes, por un lado, se encuentran los bloques asociados a la codificación del video y por el otro, los relacionados con la decodificación. En la siguiente figura, se representa el esquema básico del codificador H.261.

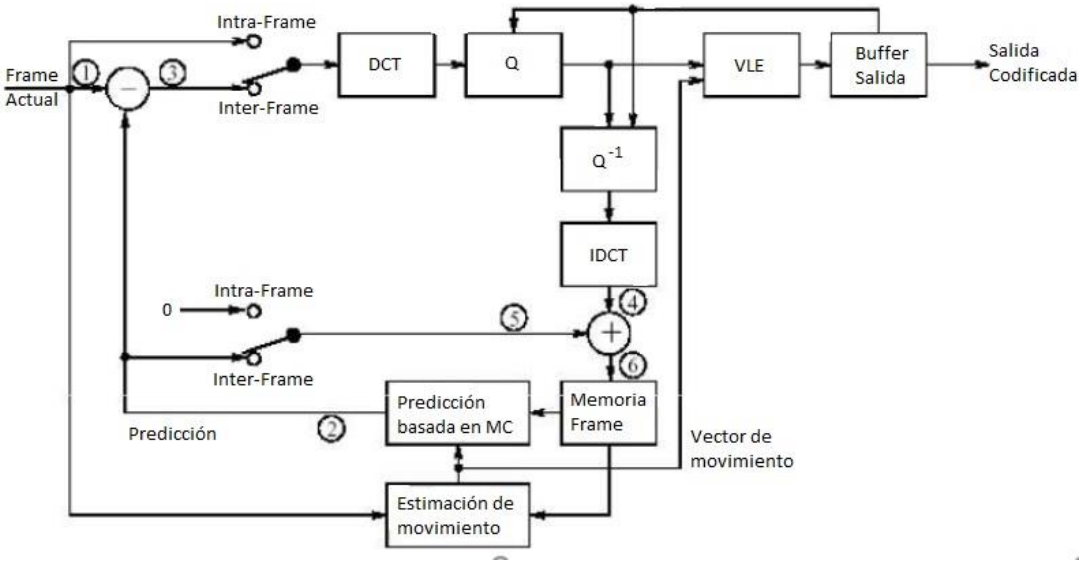


Figura 31: Esquema codificador H.261 [7]

Como ya se ha mencionado, el codificador H.261 puede codificar dos tipos diferentes de frames. Sin embargo, en el presente trabajo sólo se codificarán los frames I.

La ventaja de codificar sólo los intraframes es que el código es computacionalmente menos costoso que en el caso de codificar intraframes e interframes. Esto se debe a que la estimación de movimiento supone las ¾ partes del cómputo del codificador.

En la figura 30 se visualiza el esquema del codificador H.261 y cada uno de los diferentes módulos de implementación que se han de realizar. En el presente trabajo, los módulos de mayor relevancia son el de la DCT y la codificación de la entropía (VLE). Por lo tanto, la parte más importante del esquema será la superior.

3.1.1 Transformada de la Imagen

Las imágenes que se codificarán se capturan desde la webcam y se emplea un filtro de Bayer, filtro más común en el espacio de color RGB. Posteriormente, las imágenes capturadas en RGB se transformarán al espacio de color YCbCr.

$$Y = 16 + (65.481R + 128.553G + 24.996B)$$

$$Cb = 128 + (-37.797R\ - 74.203G\ +\ 112B)$$

$$Cr = 128 + (112R\ -\ 93.786G\ - 18.214B)$$

Ecuación 21: Transformación RGB a YCbCr

A continuación, se realiza un submuestreo 4:2:0 como se indica en el estándar.

3.1.2 Transformada del Coseno

Para poder comprimir y codificar las imágenes, habrá que dividirlas en pequeños bloques de 8*8 elementos. Para cada uno de estos bloques se realizará la DCT, la cuantificación, la ordenación en ZigZag y la codificación de entropía antes de ser enviados.

El primer módulo que se implementa es el de la transformada discreta del coseno para comprimir la mayor parte de la energía de la señal en pocos coeficientes.

La DCT procesa cada bloque de 8*8 elementos descomponiéndolos en 64 señales de la base ortogonal vista en el capítulo anterior. El primer elemento de frecuencia cero se le conoce como DC (corriente continua) y los otros 63 coeficientes son los AC (corriente alterna).

En este trabajo inicialmente se realizó la implementación de la DCT 2D. Pero esta implementación realiza 8*8 multiplicaciones por cada bloque, así que se optó por desarrollar la implementación de la DCT con el algoritmo de Loeffler. Es este un algoritmo rápido de cómputo de la 1 D DCT que se puede implementar únicamente con 11 multiplicaciones, con lo que el computo por frame se ve reducido considerablemente. Se ha de ejecutar de forma secuencial ya que existen dependencias entre los datos, sin embargo, dentro de cada etapa se pueden ejecutar en paralelo.

Esta implementación, en vez de tratar con bloques de 8*8 elementos, solo trabaja con vectores de 8 elementos. Primero se realiza la DCT correspondiente a las 8 filas de 8 elementos del bloque y después, se realiza el mismo proceso con las 8 columnas resultantes. Finalmente, se obtiene un bloque de 8*8 elementos ya cuantificados.

Antes de explicar en qué consiste este algoritmo, se ha de tener un conocimiento previo de que son las unidades de mariposa (Butterfly) y las unidades de rotación, puesto que emplearán a lo largo del algoritmo.

Las mariposas son las unidades básicas de las transformadas y su estructura en el caso de la DCT se observa en la figura 44, así como sus correspondientes operaciones [10].

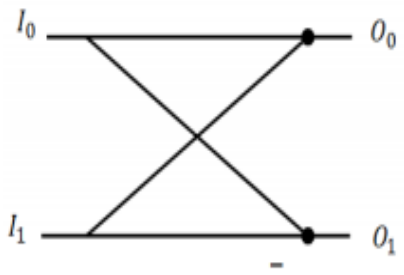


Figura 32: Estructura Mariposa DCT Loeffler

$$O_0 = I_0 + I_1$$

$$O_1 = I_0 - I_1$$

Ecuación 22: Mariposa DCT Loeffler

Por otro lado, se encuentran las unidades de rotación, que transforman parejas de entradas en salidas a partir de cuatro multiplicaciones y dos sumas. En la siguiente imagen se observa la unidad de rotación seleccionada y las formulas asociadas [10].

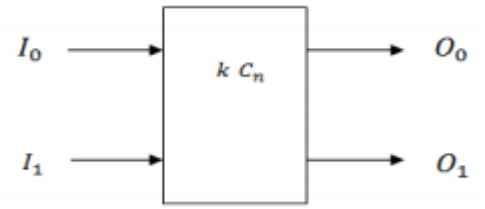


Figura 33: Unidad de rotación DCT Loeffler

$$O_0 = I_0 k cos\left[\frac{n\pi}{16}\right] + I_1 k sin\left[\frac{n\pi}{16}\right]$$

$$O_1 = I_1 k cos\left[\frac{n\pi}{16}\right] - I_0 k sin\left[\frac{n\pi}{16}\right]$$

Ecuación 23: Unidad de rotación DCT Loeffler

Una vez comprendidos estos términos, se puede proceder con la explicación del algoritmo.

El algoritmo de Loeffler se compone de 4 etapas (figura 48). En la primera etapa se cogen los datos de entrada y se procesan con la estructura de las mariposas. En esta etapa se realizan las operaciones correspondientes a 4 mariposas. Una vez realizadas, se pasa a la segunda etapa, en la cual se separan los coeficientes impares y se realizan dos operaciones de rotación y dos operaciones de mariposa. En la tercera etapa se vuelven a separar los coeficientes pares e impares y se realizan 3 operaciones mariposa y una rotación. Finalmente, en la cuarta etapa, se ajustan los coeficientes y se dan las salidas [10].

En este proceso se realizan 11 multiplicaciones y 39 sumas. Computacionalmente es menos costoso que las 8*8 multiplicaciones que se realizan con la 2D DCT

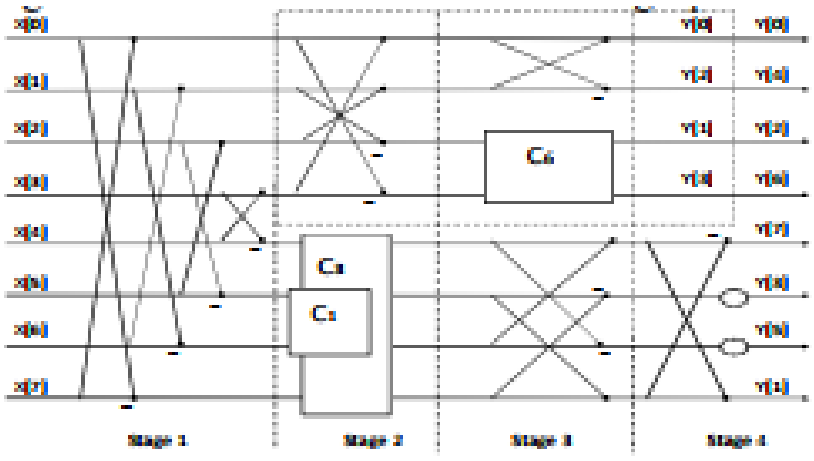


Figura 34: Etapas DCT Loeffler [10]

Una vez realizada la transformada, se obtiene un bloque de 8*8 con un valor de continua (DC) en la primera posición de la matriz y el resto de valores de alterna (AC). Este valor de continua corresponde con el elemento de menor frecuencia espacial y será el que más información aporte del bloque.

A continuación, se muestra un ejemplo de un bloque antes y después de realizar la DCT.

Macrobloque original

200	202	189	188	189	175	175	175
200	203	198	188	189	182	178	175
203	200	200	195	200	187	185	175
200	200	200	200	197	187	187	187
200	205	200	200	195	188	187	175
200	200	200	200	200	190	187	175
205	200	199	200	191	187	187	175
210	200	200	200	188	185	187	186

*Figura 36: Ejemplo Bloque 8*8*

Salida DCT

1537	-16	-12	-8	0	0	2	-1
63	3	6	2	-2	-3	-1	4
-10	1	10	-4	6	0	-4	-1
3	0	0	1	-4	0	3	3
1	-1	2	-1	3	3	2	-2
1	-9	0	-2	0	1	0	1
-6	-1	0	-4	-1	0	-1	-2
4	2	-1	-2	-2	0	2	0

Figura 35: Ejemplo Salida DCT (método 1D DCT)

En el ejemplo se observa como los elementos de las primeras posiciones concentran la mayor parte de la energía. Estos elementos tendrán menor frecuencia espacial.

3.1.3 Módulo de Cuantificación

A continuación, se realizará la cuantificación (Q) de los 64 coeficientes resultantes de la DCT. La cuantificación es el paso donde se producen las pérdidas del codificador. En este caso, el objetivo es obtener el mayor número de ceros en los coeficientes AC para aprovechar al máximo la compresión.

Para el coeficiente de DC, la cuantificación es constante y se emplea la ecuación:

$$QDCT = round(\frac{DCT}{8})$$

Ecuación 24: Cuantificación Valores DC

Sin embargo, para los coeficientes AC se va ajustando el cuantificador:

$$QDCT = \frac{DCT}{2 * scale^{9}}$$

Ecuación 25: Cuantificación Valores AC

El valor de scale será un número entero que variará entre 1 y 31.

Salida DCT

1537	-16	-12	-8	0	0	2	-1
63	3	6	2	-2	-3	-1	4
-10	1	10	-4	6	0	-4	-1
3	0	0	1	-4	0	3	3
1	-1	2	-1	3	3	2	-2
1	-9	0	-2	0	1	0	1
-6	-1	0	-4	-1	0	-1	-2
4	2	-1	-2	-2	0	2	0

Salida cuantificación

192	-1	-1	0	0	0	0	0
4	0	0	0	0	0	0	0
-1	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figura 37: Ejemplo cuantificación

[9] Siendo scale un valor entero en un rango [1,31].

Como se verá mas adelante, la imagen se tiene que dividir en grupos de bloques, que se dividiran en macrobloques y estos, a su vez, en los bloques 8*8 que se están analizando.

Los macrobloques están compuestos por 4 bloques 8*8 de luminosidad Y y dos bloques de 8*8 de croma, Cb y Cr.

El ajuste de cuantificación es la parte mas inportante de este módulo ya que determinará la calidad de la reconstruccion de la imagen. Si se realiza un ajuste de cuantificación malo, la imagen reconstruida se verá pixelada. Este ajuste se puede hacer de varias formas. Se puede emplear un cuantificador variable para cada coeficiente de un bloque 8*8. Otra opcion es mantener un cuantificador de AC común para todos los coeficientes de un bloque y variariarlo para el siguiente bloque en función del error que se comenta en la reconstrucción.

En esta implementación, el ajuste se ha realizado a nivel de grupo de bloques. Para el ajuste, se obtendrá el error cuadrático medio de los bloques originales y de los bloques reconstruidos y se comparará con un umbral, escogido de forma experimental a valor 8.

Si el error cuadrático está por debajo del umbral, se aumenta el paso del cuantificador AC y si está por encima, se reduce el paso.

Un efecto importante de la cuantificación es que gran parte de los coeficientes serán redondeados a cero. Al realizar un ajuste a nivel de grupo de bloques, la resolución es menor y por tanto, el número de coeficientes nulos aumenta y con ello, la cantidad de información que se puede comprimir.

Para obtener los bloques reconstruidos, se han implementado los modulos inversos de la cuantificación y de la DCT.

En primer lugar, se realiza una cuantificación inversa (Q^{-1}). Para reconstruir los valores de DC y AC se emplea la ecuación mostrada a continuación, siendo QDCT el valor obtenido en la cuantificación.

$$Q^{-1} = round(QDCT * Quant^{10})$$

Ecuación 26: Cuantificación Inversa

A continuación, se realiza la transformada discreta inversa del coseno para obtener el bloque reconstruido. Al igual que en el caso de la DCT, la primera implementación se realizó a partir de la IDCT 2D, pero nuevamente se requerian 8*8 multiplicaciones por bloque. Por ello, se optó por realizar un código software basado en el algoritmo de Loeffler.

[10]Tomará los mismos valores que se emplearon en la cuantificación, 8 en el caso de DC y un valor entre 1 y 31 para los AC, que ha de ser el mismo empleado en la cuantificación.

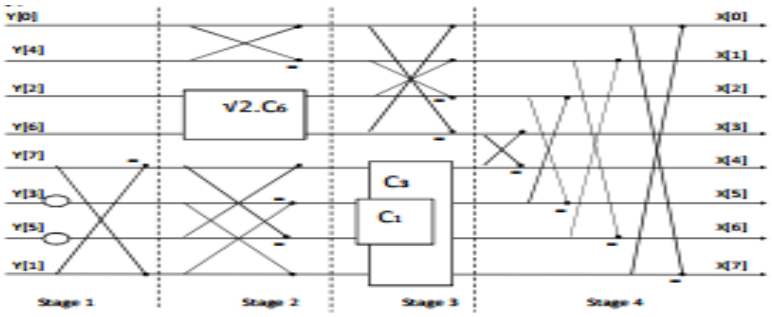


Figura 38: Etapas IDCT Loeffler

Este paso se ha de realizar minuciosamente ya que hay que deshacer los pasos que se realizaron en la DCT. Se seguirá el esquema mostrado en la figura 38.

Para la IDCT de Loeffler, las mariposas toman el valor de:

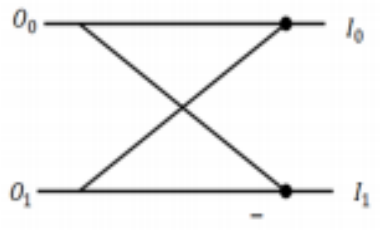


Figura 39: Mariposa IDCT Loeffler

$$I_0 = \frac{O_0 + O_1}{2}$$

$$I_1 = \frac{O_0 + O_1}{2}$$

Ecuación 27: Mariposa IDCT Loeffler

Para el caso de la IDCT, la unidad de rotación y las operaciones asociadas son:

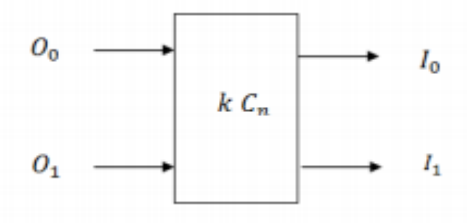


Figura 40: Unidad de rotación IDCT Loeffler

$$I_0 = O_0 k cos\left[\frac{n\pi}{16}\right] - O_1 k sin[\frac{n\pi}{16}]$$
$$I_1 = O_1 k cos\left[\frac{n\pi}{16}\right] + 0_0 k sin[\frac{n\pi}{16}]$$

Ecuación 28: Unidad de rotación IDCT Loeffler

En este caso, el procedimiento es el mismo. Se realiza la IDCT para las 8 filas de 8 elementos del bloque resultante de la cuantificación inversa y se almacena en un array. A continuación, se lleva a cabo la IDCT para las columnas, siguiendo la misma pauta.

Finalmente, se obtiene el bloque reconstruido con el que obtener el error cometido en la reconstrucción y ajustar el cuantificador para los coeficientes AC del siguiente bloque.

Macrobloque original

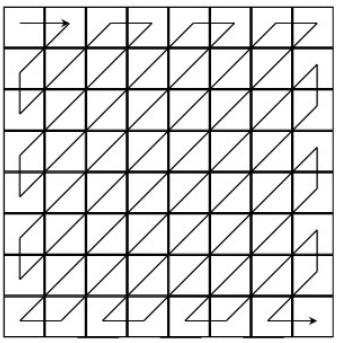
200	202	189	188	189	175	175	175
200	203	198	188	189	182	178	175
203	200	200	195	200	187	185	175
200	200	200	200	197	187	187	187
200	205	200	200	195	188	187	175
200	200	200	200	200	190	187	175
205	200	199	200	191	187	187	175
210	200	200	200	188	185	187	186

Macrobloque reconstruido

197	198	191	185	184	178	173	174
198	199	195	189	188	181	175	175
198	201	199	196	194	186	177	175
199	202	203	202	198	189	180	175
201	202	205	204	199	191	182	176
203	202	203	202	195	189	183	177
205	201	201	198	190	186	184	179
205	200	199	195	187	184	183	179

Figura 41: Ejemplo Bloque Reconstruido

Completada la cuantificación y obtenida la imagen completa cuantificada, se realiza la ordenación en ZigZag de cada uno de los bloques 8*8 que componen la imagen, para agrupar la mayor cantidad de ceros en las últimas posiciones.



Ordenación ZigZag

192	-1	4	-1	0	-1	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	0	-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

Figura 42: Matriz Ordenación ZigZag y ejemplo

A continuación, se llega al punto más importante y complejo del cuantificador, la codificación de entropía. Este paso consiste en una compresión adicional de los coeficientes cuantificados de la DCT basándose en la redundancia estadística.

Para ello, se va a emplear el método de codificación de longitud variable VLC para asignar códigos de palaras de diferentes tamaños en función de la probabilidad de los elementos. Estos elementos codificados se incluirán en un stream de codificación que será la salida del sistema.

3.1.4 Generación de la Trama

En la implementación, se ha realizado las función writev(n,b), donde n es el número de bits con los que codificar y b el valor a codificar. También se ha implementado la función write (b) cuando únicamente se va a codificar un 1 o un 0.

La función writev(n,b) realiza un ciclo de n hasta 0 y compara el valor de b en hexadecimal con la máscara de la figura 43.

```
int bit_set_mask[] =
{0x00000001,0x00000002,0x00000004,0x00000008,
0x00000010,0x00000020,0x00000040,0x00000080,
0x00000100,0x00000200,0x00000400,0x00000800,
0x00001000,0x00002000,0x00004000,0x00008000,
0x00010000,0x00020000,0x00040000,0x00080000,
0x00100000,0x00200000,0x00400000,0x00800000,
0x01000000,0x02000000,0x04000000,0x08000000,
0x10000000,0x20000000,0x40000000,0x80000000};
```

Figura 43: Máscara bits

Para ello, se realiza una operación "and" del valor con las diferentes mascaras en función de la posición n. Tras esta comparación se obtendrá la posición del bit marcado a 1 y a través de la función putc de la librería estándar c, se escribe en el stream.

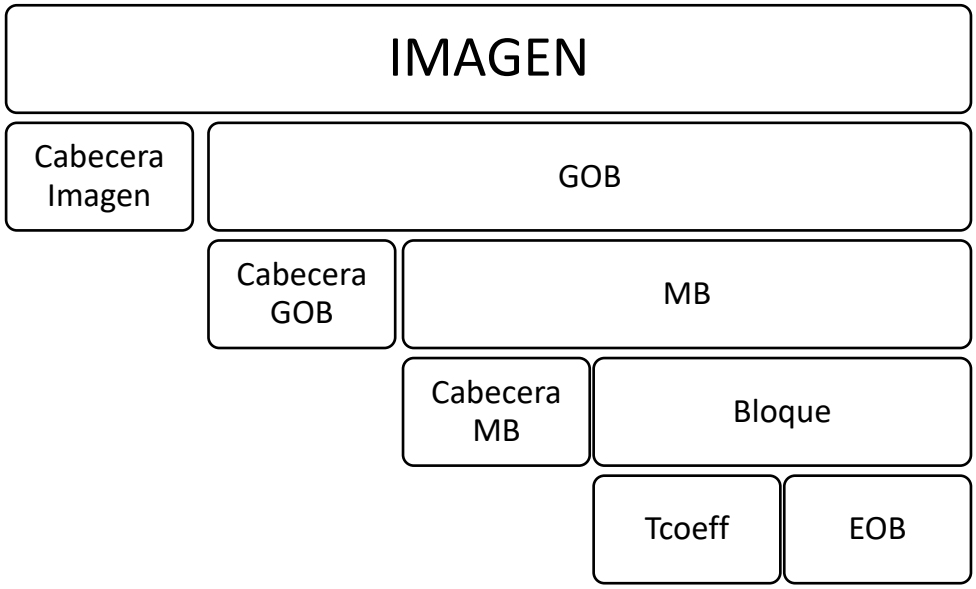


Figura 44: Jerarquía Stream H.261

La estructura del stream es una jerarquía de 4 capas. Estas capas son la capa de imagen, la capa de grupo de bloques (GOB), la capa de macrobloque (MB) y la capa de bloque.

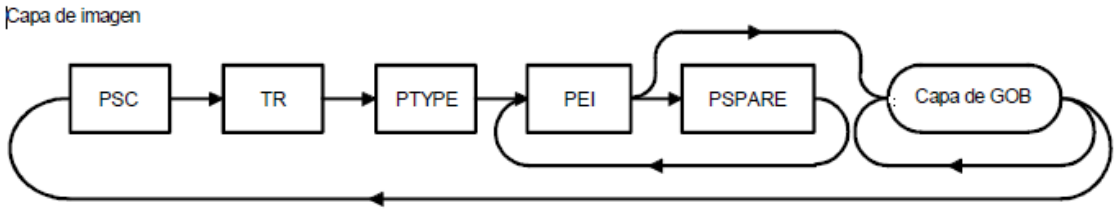


Figura 45: Capa de Imagen

La primera capa de cada imagen contiene un encabezamiento de imagen seguido de los datos relacionados a los GOB. La estructura de esta capa se muestra en la figura 45.

En la implementación se ha realizado una función llamada CabeceraImagen() para escribir los bits de la cabecera en el stream. Estos parametros se codifican en el siguiente orden:

- Se comienza codificando en el stream un código de inicio de imagen (PSC), cuyo valor es 16 y se codifica con 20 bits. writev(20,16)
- A continuación, se codifica el TR (referencia temporal). Es una palabra de 5 bits y en este trabajo su valor se va incrementando en 1 por cada imagen transmitida.
- La siguiente palabra de la cabecera a codificar es el PTYPE que da información acerca de la imagen.
    - Bit 1 Indicador de división de pantalla. «0» no, «1» sí.
    - Bit 2 Indicador de cámara de documentos. «0» no, «1» sí.
    - Bit 3 Liberación de imagen congelada. «0» no, «1» sí.
    - Bit 4 Fuente de formato. «0» QCIF, «1» CIF.
        - Bit 5 Modo opcional de transmisión de imágenes fijas en alta resolución definido en el Anexo D. «0» no, «1» sí.
        - Bit 6 De reserva.

    Para este trabajo toma el valor 0x0F y se codifica con 6 bits.

- El siguiente campo de información (PEI) indica si a continuación viene el campo de información de reserva (PSPARE). Para este trabajo el campo PEI se pone a 0. Por tanto, no habrá campo PSPARE.

A continuación, se envía la información de los GOB (Grupo de bloques). La imagen de entrada cuenta con formato CIF como ya se indicó al inicio y, por tanto, como marca el estándar, se ha de dividir en 12 bloques de tamaño 176*48 para las lumas y 88*48 para las cromas como se indica en la imagen.

1	2
3	4
5	6
7	8
9	10
11	12

Figura 46: Disposición de los bloques en la imagen

Por tanto, la información relativa a los GOB se ha de enviar 12 veces, una por cada bloque.

A continuación, se muestra el diagrama de la capa de GOB.

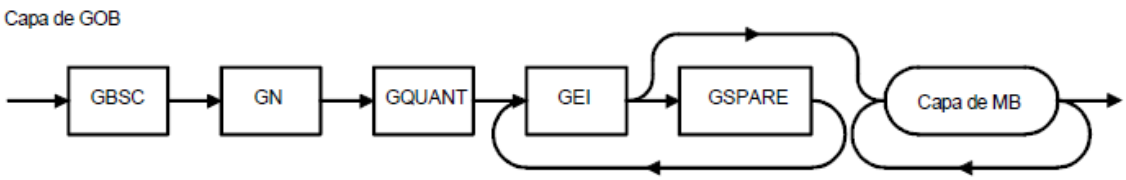


Figura 47: Capa de GOB

En la implementación de esta parte, se ha realizado una función llamada CabeceraGOB() para escribir los bits de la cabecera en el stream. Estos parametros se escribirán a continuación de los datos explicados anteriormente.

Esta cabecera está formada por:

- El primer dato de la cabecera GOB es el código de inicio del grupo de bloques (GBSC) que tiene valor 1 y se codifica con 16 bits. Este valor es común para todos los bloques. Se usa writev(16,1).
- A continuación, se escribe el número de grupo (GN) que indica el GOB que se está codificando. Este valor puede valer entre 1 y 12 (figura 46) y se codifica con 4 bits. Se usa writev(4,GN), siendo GN un entero entre 1 y 12.
- La siguiente palabra que se codifica es el GQUANT que indica el ajuste de cuantificación a nivel de bloque. Como ya se indicó anteriormente, en este trabajo se realiza un ajuste a este nivel. Se empleará el mismo cuantificador para cada pareja de GOB (1-2,3-4,5-6,7-8,9-10,11-12). Se codificará el valor del ajuste de cuantificación con 5 bits. Writev(5, Acquant), siendo Acquant un entero entre 1 y 31.
- Por último, se codifica el campo de información suplementaria (GEI). En este caso se codifica con un bit a valor 0. Esto supone que el siguiente campo (GSPARE) no se codifica.

A continuación, se procede a escribir en el stream la codificación de los macrobloques. Cada GOB mencionado anteriormente, está compuesto de 33 macrobloques. Por tanto, la imagen total está compuesta por 396 macrobloques.

1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22
23	24	25	26	27	28	29	30	31	32	33

Figura 48: Orden de los macrobloques en cada GOB

La disposición de los macrobloques dentro de los 12 GOB se muestra en la figura anterior.

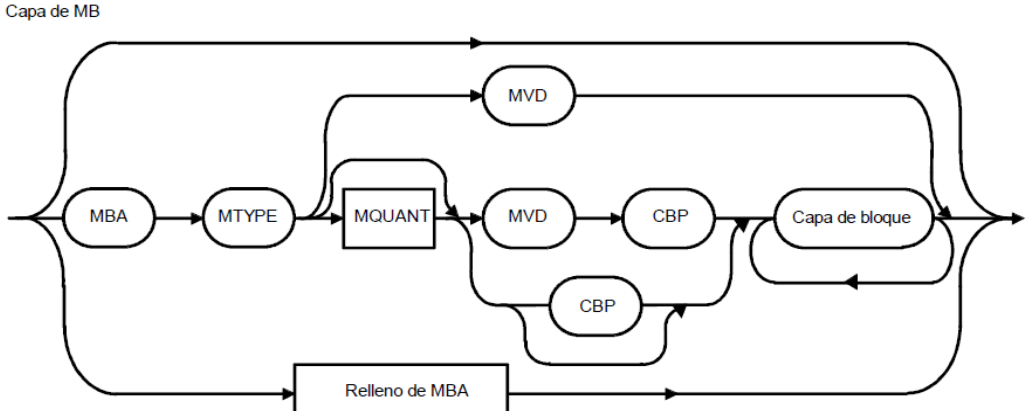


Figura 49: Capa de Macrobloque

La cabecera de esta capa se escribe en el stream a continuación de la cabecera anterior y está formada por:

- La primera palabra que se codifica es el MBA (dirección de macrobloque). Se trata de una palabra de longitud variable que indica la posición relativa del macrobloque que se va a codificar dentro de un GOB. Este valor de MBA varía entre 1 y 33 y la palabra codificada y su longitud están tabuladas en la siguiente tabla:

MBA	Código	MBA	Código
1	1	17	0000 0101 10
2	011	18	0000 0101 01
3	010	19	0000 0101 00
4	0011	20	0000 0100 11
5	0010	21	0000 0100 10
6	0001 1	22	0000 0100 011
7	0001 0	23	0000 0100 010
8	0000 111	24	0000 0100 001
9	0000 110	25	0000 0100 000
10	0000 1011	26	0000 0011 111
11	0000 1010	27	0000 0011 110
12	0000 1001	28	0000 0011 101
13	0000 1000	29	0000 0011 100
14	0000 0111	30	0000 0011 011
15	0000 0110	31	0000 0011 010
16	0000 0101 11	32	0000 0011 001
		33	0000 0011 000
		Relleno de MBA	0000 0001 111
		Código de comienzo	0000 0000 0000 0001

Tabla 4: Tabla codificación MBA

La columna MBA indica la posición del macrobloque que se va a codificar dentro de cada GOB. En la columna de Código se indica el valor y la cantidad de bits con los que se ha de codificar en el stream.

- La siguiente palabra que se codifica es MTYPE cuya longitud es variable y determina los siguientes campos que aparecen en la cabecera.
  En la siguiente figura se muestran los diferentes valores que puede tomar este campo en función de los elementos que aparecen a continuación.

Predicción	MQUANT	MVD	CBP	TCOEFF	VLC
Intra				x	0001
Intra	x			x	0000 001
Inter			x	x	1
Inter	x		x	x	0000 1
Inter + MC		x			0000 0000 1
Inter + MC		x	x	x	0000 0001
Inter + MC	x	x	x	x	0000 0000 01
Inter + MC + FIL		x			001
Inter + MC + FIL		x	x	x	01
Inter + MC + FIL	x	x	x	x	0000 01

NOTAS

1 Una «x» significa que el elemento se encuentra presente en el macrobloque.

2 Es posible aplicar el filtro en un macrobloque compensado sin movimiento declarándolo como MC + FIL pero con un vector nulo.

Tabla 5: Tabla codificación MTYPE

En este trabajo solo se van a codificar tramas de tipo I y el ajuste de la cuantificación se realiza en la capa de GOB. Por tanto, esta palabra tendrá valor 1 y se codificará con 4 bits.

Una vez codificada la cabecera del macrobloque, se procede con la codificación de los bloques. Cada macrobloque está compuesto por 4 lumas de tamaño 8*8, un croma azul (Cb) de tamaño 8*8 y una croma roja (Cr) de tamaño 8*8. La disposición de los bloques dentro de cada macrobloque es:

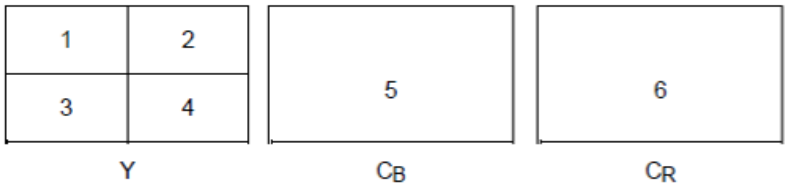


Figura 50: Orden bloques en cada Macrobloque

La imagen completa está compuesta por 2376 bloques de tamaño 8*8. A continuación, se muestra el diagrama de codificación a nivel de bloque.

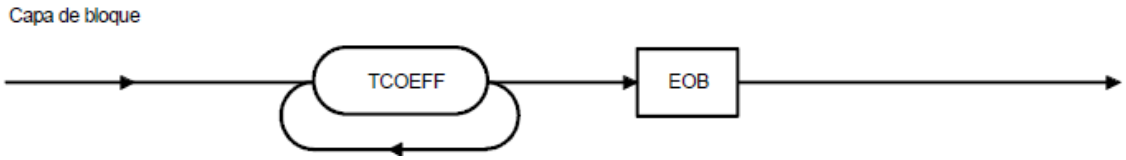


Figura 51: Capa de bloque

Para la codificación de los coeficientes de cada uno de los bloques, se seguirá el orden indicado en la figura 49. Para codificar por un lado las lumas y por otro las cromas, la información se ha almacenado en dos arrays en memoria para recorrer las posiciones de Y y CbCr independientemente. La codificación de estos coeficientes empleará una codificación de longitud variable cuyos valores se encuentran tabulados en una tabla del estándar. Sin embargo, en esta implementación, se utiliza otra forma de codificación permitida en el estándar que se basa en codificar 3 palabras.

- La primera palabra Escape tendrá valor 1 y se codifica con 6 bits. Esta palabra es común para todos los coeficientes de cada bloque.
- A continuación, se escribirá la palabra Run que indica el número de ceros consecutivos hay entre dos elementos no nulos. Se codifica con 6 bits.
- La última palabra que se codifica es Level que contiene el valor del número no nulo.

Tras codificar cada bloque, se incluye en el stream un código que indica el final de bloque (EOB). Esta palabra tiene valor 2 y se codificará con dos bits.

Como se puede observar, la generación de la trama de salida no es tan simple como a priori pudiera parecer. La estructura que presenta se repite continuamente y supone la mayor complejidad del algoritmo.

Una vez codificados todos los elementos de la secuencia de imágenes que componen el video de entrada en el stream, se obtiene la salida final del sistema.

Para visualizar la salida, se ha de decodificar la trama y para ello se tendría que implementar un decodificador. Sin embargo, en este trabajo no se ha realizado dicha tarea, utilizándose un decodificador comercial como VLC.

Capítulo 4

4.1 Análisis de los resultados

Una vez explicados los conceptos de la base teórica y de la implementación realizada, se procede a evaluar el funcionamiento del algoritmo.

Se ha realizado un código software compuesto por aproximadamente 1000 líneas de código, dividido en 13 ficheros para poder manejar la información de forma más sencilla.

El algoritmo se ha implementado siguiendo la jerarquía mostrada en el capítulo anterior. De esta forma, se ha descompuesto la imagen en bloques para analizar las imágenes en partes y que el computo sea más rápido.

Cada módulo del codificador tiene su propio fichero y puede ser implementado en cualquier sistema que utilice los mismos algoritmos. Además, el código desarrollado es libre.

A continuación, se muestran los diferentes tiempos de ejecución de cada uno de los módulos implementados.

```
El módulo de la DCT tarda 3.000000 us
El módulo de la DCT tarda 3.000000 us
El módulo de la DCT tarda 3.000000 us
El módulo de la DCT tarda 3.000000 us
El módulo de la DCT tarda 2.000000 us
El módulo de la DCT tarda 4.000000 us
El módulo de la DCT tarda 4.000000 us
```

*Figura 52: Tiempo Ejecución DCT Bloque 8*8*

El tiempo de ejecución se ha calculado para cada bloque de 8*8 y como se puede observar, oscila entre los 2 y los 4 $\mu segundos$.

```
El módulo de la Cuantificación tarda 1.000000 us
El módulo de la Cuantificación tarda 2.000000 us
El módulo de la Cuantificación tarda 1.000000 us
El módulo de la Cuantificación tarda 1.000000 us
El módulo de la Cuantificación tarda 1.000000 us
```

*Figura 53: Tiempo Ejecución Cuantificación Bloque 8*8*

Para el caso de la cuantificación, el tiempo que emplea por cada bloque está en el rango entre 1 y 2 $\mu segundos$.

```
El módulo de la Cuantificación Inversa tarda 1.000000 us
El módulo de la Cuantificación Inversa tarda 0.000000 us
El módulo de la Cuantificación Inversa tarda 1.000000 us
El módulo de la Cuantificación Inversa tarda 1.000000 us
El módulo de la Cuantificación Inversa tarda 0.000000 us
```

*Figura 54: Tiempo Ejecución Cuantificación Inversa Bloque 8*8*

En el caso de la cuantificación inversa, el tiempo que tarda suele ser 1 $\mu segundo$ o inferior. La precisión de las operaciones no indica los valores inferiores al $\mu segundo$ y, por tanto, aparece representado con un 0.

```
El módulo de la IDCT tarda 4.000000 us
El módulo de la IDCT tarda 4.000000 us
El módulo de la IDCT tarda 5.000000 us
El módulo de la IDCT tarda 4.000000 us
El módulo de la IDCT tarda 5.000000 us
El módulo de la IDCT tarda 5.000000 us
```

*Figura 55:Tiempo Ejecución IDCT Bloque 8*8*

Para el caso de la inversa de la DCT, el tiempo que tarda en ejecutarse el módulo por bloque 8*8 oscila entre 4 y 5 $\mu segundos$.

Como se puede observar, los módulos que más tardan en ejecutarse son los relacionados con la DCT y la IDCT y esto se debe a que el número de operaciones que realizan es superior al resto de módulos.

A continuación, se muestra lo que tarda el sistema en escribir cada frame codificado en el stream. Se observa que los tiempos son inferiores al segundo y se puede determinar que el sistema es rápido.

```
El tiempo que tarda en escribir el stream codificado es: 0.032000 s
El tiempo que tarda en escribir el stream codificado es: 0.015000 s
El tiempo que tarda en escribir el stream codificado es: 0.016000 s
El tiempo que tarda en escribir el stream codificado es: 0.031000 s
El tiempo que tarda en escribir el stream codificado es: 0.031000 s
```

Figura 56: Tiempo Codificación Stream

En la siguiente figura se muestran unos pocos datos de las cabeceras codificadas en el stream para el primer frame.

```
-> Init Picture Layer. Frame=0
        PSC = 00010
        TR = 00
        Ptype = 0f
        pei = 0
--->> GOB Layer
        GBSC = 0001
        GN = 1
        GQUANT = 03
        GEI = 0
        MBA = 1
        Intra  0 0 0 1 - 1
```

Figura 57: Ejemplo Codificación Stream

De esta forma, se puede comprobar que los datos descritos en el capítulo anterior corresponden con los datos escritos.

Además, se ha realizado una prueba con un video de 3 segundos a 29fps y el tiempo de ejecución del sistema completo es:

```
El tiempo que tarda en ejecutar el programa es: 3.141000 s
```

Figura 58: Ejemplo Tiempo Ejecución Programa

Por tanto, el programa realizado es capaz de procesar las imágenes a 27 fps. Esto significa que el código implementado es capaz de capturar y codificar secuencias de video en tiempo real.

En la siguiente figura se muestra un fotograma de un video en tiempo real. La imagen de la izquierda corresponde con la imagen original y la de la derecha con la imagen reconstruida.

Figura 59: Ejemplo Imagen Original e Imagen Reconstruida

Como se puede observar, la imagen reconstruida es prácticamente igual a la original y esto se debe a que el ojo humano no es capaz de percibir las pérdidas que introduce el sistema. Las diferencias se deben al empleo de algoritmos de compresión con pérdidas.

Capítulo 5

5.1 Conclusiones

En este trabajo se ha llevado a cabo la implementación de un codificador de video a partir del estudio de los diferentes módulos del codificador para realizar una implementación óptima.

El algoritmo está basado en el estándar de codificación de video H.261, que fue uno de los primeros estándares de codificación.

El sistema captura una secuencia de imágenes con formato CIF, se transforma la codificación de color de RGB a YCbCr y se realiza un Submuestreo 4:2:0 en la imagen como indica el estándar.

Estas imágenes se han de dividir en pequeños bloques de 8*8 elementos que pasarán por los diferentes módulos del sistema. En el primer bloque se realiza la transformada discreta del coseno para agrupar en pocos coeficientes la mayor parte de la energía de la señal.

A continuación, se realiza la cuantificación, para obtener el mayor número de ceros en coeficientes AC y comprimir más. En este módulo está implícita la pérdida de calidad. Una vez comprimidos, se ordenan con el método ZigZag para agrupar los valores no nulos en las primeras posiciones y se codifican para enviarlos en una trama de salida.

A la vista de los resultados, se trata de un codificador que funciona en tiempo real y que emplea la mayor parte del tiempo en realizar la transformada discreta de los elementos y en la codificación del stream. El resto de módulos se procesan en tiempos inferiores al $\mu segundo$.

Como posibles líneas de trabajo futuras se proponen:

- Realizar un código software que se encargue de la decodificación del stream generado en este trabajo.
- Incluir el módulo de estimación de movimiento en el codificador y comparar los ratios de compresión y el tiempo de ejecución del programa.
- Implementación HW del codificador de video para videoconferencia basado en el estándar H.261.
- Realizar la comparación con estándares de codificación más modernos, como el H.263, MPEG-2 o el H.264.

Capítulo 6

6.1 Referencias

[1]Aguilar Fernández, Elena María. (2008). *Técnicas generales de compresión de la señal de video*. Available: http://bibing.us.es/proyectos/abreproy/11618/fichero/MEMORIA%252F05+-+Cap%C3%ADtulo+2+-+T%C3%A9cnicas+de+compresi%C3%B3n+de+la+se%C3%B1al+de+video.pdf+

[2] LI, Ze-Nian; DREW, Mark S.; LIU, Jiangchuan. *Fundamentals of multimedia*. Upper Saddle River (NJ):: Pearson Prentice Hall, 2004.

[3] (2008). *Formación del Audio Digital: Muestreo, Cuantificación y Codificación.* Available: https://www.lpi.tel.uva.es/~nacho/docencia/ing_ond_1/trabajos_07_08/io5/public_html/Formacion_Audio.htm

[4] Rafael Molina (2012). *Cuantificación Escalar*. Available: https://es.slideshare.net/Rose56/07-cuantificacion-escalar-1

[5] David Aledo Ortega (2013). *Compresión de imágenes optimizada en consumo energético para redes inalámbricas*. Available: http://www.cei.upm.es/media/TFM/Aledo_David_TFM_2013.pdf

[6] Emil Mikulic (2001). *Discrete Cosine Transform*. Available: https://unix4lyfe.org/dct/

[7] Pablo Pedro Sánchez Espeso (2017). *Parte 4.1: Técnicas Básicas de Compresión de Video.* Sistemas Electrónicos Multimedia.

[8] *DCT 2D without FFT.* Available: https://stackoverflow.com/questions/13171329/dct-2d-without-fft

[9] Juan Francisco Rodríguez; Vicente González Ruiz. (2014). *The JPEG standard (ISO/IEC 10918-1) Available:* https://w3.ual.es/~vruiz/Docencia/Apuntes/Coding/Image/03-JPEG/index.html

[10] MAHAJAN, NEHA V.; CHITODE, J. S. DCT/IDCT Implementation with Loeffler Algorithm. *International Journal Of Application or Innovation in Engineering & Management*, 2014, vol. 3, no 5, p. 353-357.

[11] Liesner Acevedo Martínez (2009). *Computación Paralela de la transformada Wavelet.* Available: http://www.dsic.upv.es/docs/bib-dig/tesis/etd-05152009-123504/phd.pdf

[12] Petrová Jana (2011). *Edge detection in medical images using the wavelet transform.* Available: http://www.posterus.sk/?p=10983

[13] *Imágenes Wikipedia. Available:* https://es.wikipedia.org/wiki/Wikipedia:Portada

[14] Example Shannon-Fano Coding. Available: http://www.binaryessence.com/dct/en000046.htm

[15] *¿Por Qué RGB? (2013). Available:*
http://www.pensamientoscomputables.com/entrada/por-que-modelo-color-rgb.html

[16] Dezhine. (2011). *Modelo de color y submuestreo de color. Available:*
https://dezhine.wordpress.com/2011/08/06/modelo-de-color-y-submuestreo-de-color-abelcine/

[17] *Códec Vídeo para servicios audiovisuales a p*64Kbit/s*, Recomendación UIT-T H.261 (03/93).

[18] *Codificación de vídeo para comunicación a baja velocidad binaria*, Recomendación UIT-T H.263 (01/05).

[19] David Gallardo (11/12). *Iniciándose en la plataforma Eclipse.* Available:
https://www.ibm.com/developerworks/ssa/library/os-ecov/index.html

[20] OpenCV Team. *OpenCV.* Available: https://opencv.org/

[21] VideoLAN Organization. *VLC media player.* Available:
https://www.videolan.org/vlc/index.es.html