Universidad Zaragoza
1542

Proyecto Fin de Carrera

Desarrollo de software de compresión de vídeo en MPEG-2 para uso en el entorno académico

Autor:

Daniel Prol Martín-Ambrosio

Director:

Santiago Cruz Llanas

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2016

RESUMEN

Desarrollo de software de compresión de vídeo en MPEG-2 para uso en el entorno académico

En asignaturas del ámbito universitario que cubren los conceptos de la compresión de vídeo la docencia puede mejorar notablemente mediante formación práctica. Existen herramientas software en el mercado, como analizadores gráficos de vídeo comprimido, que resultan muy atractivas para este fin. Estas, en cambio, presentan el inconveniente de tener un coste elevado, lo que provoca que puedan quedar inaccesibles para su uso en la docencia.

Este proyecto surge entonces con el objetivo de desarrollar una herramienta para estudiar la tarea de compresión de vídeo en el entorno académico. La implementación se realiza en Matlab debido al enfoque didáctico, un entorno con el que el alumnado de ingenierías del ámbito de las telecomunicaciones suele estar familiarizado. Así se podrá aprovechar también la optimización que ofrece Matlab para el cálculo de algunas operaciones matemáticas que intervienen en el proceso de compresión de vídeo.

El desarrollo del proyecto incluye un estudio previo de las técnicas empleadas en la compresión de vídeo digital. Este análisis se centra en las particularidades del estándar MPEG-2 vídeo, una norma ampliamente extendida y compatible con la mayoría de reproductores multimedia. Se toma este estándar como referencia para la realización de la herramienta objeto de este proyecto.

La implementación de la herramienta divide el proceso de compresión de vídeo en dos partes. En la primera de ellas se almacena la señal en una estructura, acorde a los procesos de transformada, cuantificación y predicción temporal que exige la norma MPEG-2. En la segunda, toda esta información es codificada en binario conformando un flujo elemental de vídeo MPEG-2.

En aras de maximizar el valor didáctico de la herramienta se incluye también una interfaz gráfica, que ofrece al alumno un uso sencillo sin necesidad de profundizar en el código. Además se proponen ejemplos de utilización en prácticas, donde el estudiante podrá conocer todo el proceso de compresión de vídeo digital y sus implicaciones en el resultado, comprobando también de primera mano el compromiso que existe entre la compresión de la secuencia de vídeo resultado y su calidad.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE FÓRMULAS

1 - INTRODUCCIÓN

1.1 – MOTIVACIÓN Y CONTEXTO

En mayor o menor medida los avances en las tecnologías de la información y la comunicación hacen que la mayoría de los ciudadanos en las sociedades desarrolladas sean consumidores de contenidos multimedia. En concreto el vídeo digital es un contenido fundamental presente en muchas aplicaciones, mostrándose las que quizá son más populares en la Tabla 1. Además se caracteriza por ser uno de los tipos de datos que más recursos consume cuando se transmite o se almacena (ancho de banda, memoria requerida, etc.), por lo que, en la mayoría de los casos y aplicaciones, la señal nativa se comprime o codifica previamente. Este es el ámbito principal en el que se incide en este trabajo.

Aplicación	Tasa de bits
Televisión Digital Terrestre	SD: 1.5 - 6 Mbps HD: 5 - 20 Mbps
DVD-Vídeo	5 - 20 Mbps
Disco Blu-ray	Hasta 40 Mbps
Transmisión de vídeo por internet	100 - 2000 Kbps
Vídeoconferencia	20 - 2000 Kbps
Vídeo sobre red móvil	3G: 100 - 500 Kbps 4G: 300 - 1500 Kbps

Tabla 1 – Ejemplo de aplicaciones de vídeo digital comprimido.

A lo largo de los años se han ido desarrollando técnicas de compresión que han propiciado la aparición de estándares de codificación de vídeo. El tándem técnicas de compresión / estándares de codificación ha permitido hacer frente a los retos que iban planteando las nuevas aplicaciones que surgían como las de la Tabla 1. Para alumnos que se forman en el ámbito de las tecnologías de comunicación no es suficiente con saber que es importante o necesario comprimir vídeo. También deberían conocer los fundamentos teóricos que posibilitan esta compresión y las familias de normas y estándares más populares que se destinan a este uso. Sería una cualidad positiva que esta parte de su formación se proporcionara de una forma práctica. Contribuir a esta tarea es la principal motivación de este proyecto, que ha surgido dentro de la Universidad de Zaragoza con la idea de dar cobertura a las asignaturas en las titulaciones del ámbito de la telecomunicación donde más se trabaja este aspecto.

Existen herramientas disponibles en el mercado que aportan un valor práctico en este ámbito, pero suelen derivar en un gasto extraordinariamente alto, debido al elevado coste de las licencias necesarias del software propietario. Esto ocurre por

ejemplo en los analizadores de flujos elementales de vídeo[1], donde cada licencia de usuario tiene un precio de al menos cuatro dígitos. Por ello, la creación de una herramienta, diseñada con fines didácticos y enfocada al entorno académico, solventará este inconveniente y facilitará la labor del profesorado a la hora de transmitir todos los conceptos relacionados con la compresión y codificación de las señales de vídeo digital.

1.2 – OBJETIVOS Y ALCANCE DEL PFC

Conociendo la necesidad de la creación de una herramienta de compresión de vídeo con fines educativos, los objetivos se enfocan en maximizar los valores didácticos que esta pueda aportar al entorno académico. Se pretende así distanciarse de las características del software profesional: entornos de trabajo opacos con demasiada complejidad para el ámbito académico.

Se ha decidido desarrollar la herramienta en Matlab, ya que se trata de un lenguaje de programación y entorno con el que los alumnos suelen estar familiarizados. Esto les permitirá seguir el proceso de compresión de vídeo sobre el mismo código que lo implementa. El uso de Matlab para esta aplicación concreta de procesado de la señal de vídeo es además adecuado por otras razones. Matlab trabaja de forma nativa con vectores, matrices e *hyperarrays*, y también integra herramientas útiles eficientes para labores de procesado de señal, como pueden ser los algoritmos optimizados y las transformadas rápidas, entre otros. Adicionalmente, incluye de forma sencilla capacidades de representación gráfica y reproducción multimedia, lo cual lo hace especialmente atractivo para la aplicación de compresión de vídeo.

Se ha escogido MPEG-2 vídeo [1] como el estándar de referencia que la herramienta debe seguir para la compresión de vídeo digital. Su uso en el mundo digital es ampliamente conocido y su proceso de compresión presenta además menor complejidad y, por tanto, resulta más didáctico que otros que han aparecido en los últimos años. Así, la implementación deberá ser capaz de generar flujos elementales de vídeo acordes a la norma MPEG-2. Se encuentra que el hecho de que la herramienta siga un estándar de uso en el mundo real como MPEG-2 puede constituir un elemento incentivador para el alumno, que puede comprobar cómo las secuencias de vídeo comprimido generadas se pueden visualizar en reproductores multimedia estándar.

Para un aprovechamiento óptimo de la herramienta los alumnos podrían analizar, depurar o modificar de forma guiada el código en el que se basa, siguiendo y comprendiendo así el proceso de compresión. No obstante, pensando en que puedan existir asignaturas en las que esto suponga demasiado tiempo, se ha pensado además en el interés de incluir en el desarrollo una interfaz gráfica que permita usar la herramienta y analizar resultados sin necesidad de ahondar en los entresijos de la programación. El uso de la herramienta gráfica podría ser suficiente, por ejemplo para

[1] Algunos de los analizadores de flujos elementales de vídeo disponibles en el mercado son CodecVisa, Elecard StreamEye Studio e Intel Vídeo Pro Analyzer.

ilustrar el compromiso entre la tasa de compresión y la calidad final obtenida al descomprimir una secuencia de vídeo.

Para complementar esa ayuda al usuario, se pretende incorporar también un guión de prácticas sobre la compresión de vídeo usando la herramienta, que aporte al alumno el valor que tienen algunas partes del proceso implementado.

En resumen, el objetivo principal de este proyecto es la creación de una herramienta didáctica que genere un flujo elemental de vídeo comprimido MPEG-2. Esta debe además aportar un análisis adicional sobre el proceso de compresión y facilitar al usuario el acceso al mismo. La implementación deberá también aprovechar las características del entorno que ofrece Matlab en la medida de lo posible. El cumplimiento de estos objetivos requerirá también de un análisis previo del estándar, con el que se realice la toma de decisiones sobre qué opciones elegir a lo largo del desarrollo que maximicen el valor didáctico de la herramienta.

1.3 – ORGANIZACIÓN DE LA MEMORIA

Este documento se encuentra dividido en cuatro capítulos. Este primero sirve como preámbulo o introducción para conocer los factores por los que se tomó la decisión de llevar a cabo este proyecto.

En el segundo capítulo se presenta el estado del arte en la compresión de vídeo y una introducción a los fundamentos teóricos que permiten dicha compresión. Se pretende presentar en él conceptos generales de cualquier estándar de vídeo digital comprimido, pero se hace inevitable particularizar en algunas partes que requieren entrar en detalle. Para estos casos se elige MPEG-2 como estándar de referencia al ser uno de los objetivos contemplados en el proyecto.

En el tercer capítulo se detalla el enfoque al problema que presenta el proyecto y la solución que se ha adoptado para un mejor desarrollo del mismo. Se describe así el proceso de diseño, desarrollo y validación de la herramienta creada para cumplir los objetivos.

El cuarto y último capítulo está dedicado a las conclusiones del proyecto. Se presentan también una síntesis de las dificultades encontradas durante el desarrollo, su influencia sobre el mismo y cómo se han abordado. Para cerrar el capítulo se plantean también se plantean las futuras líneas de desarrollo que pueda tener la herramienta.

Para complementar toda la información presentada en los capítulos anteriores, se añaden diferentes anexos con los que se intenta aportar más detalles o plasmar casos prácticos que puedan describir la materia de una forma distinta. Aparecen así los detalles sobre la jerarquía de capas de MPEG-2, un ejemplo de la codificación implementada sobre un bloque de la imagen, la estructura de datos definida para almacenar la información del vídeo comprimido y un ejemplo de un guión de prácticas.

2 – COMPRESIÓN DE VÍDEO

En este capítulo veremos la necesidad de comprimir la señal de vídeo digital y revisaremos brevemente los estándares internacionales más populares de codificación. Posteriormente se introducirán los fundamentos teóricos de la compresión de vídeo mediante la presentación de los elementos que componen el diagrama de bloques de un codificador híbrido. Teniendo en cuenta que se ha impuesto como requisito que la herramienta desarrollada en este proyecto sea compatible con la norma MPEG-2 vídeo, se introducen finalmente algunos aspectos sobre esta norma. Concretamente, la jerarquía de datos que utiliza y la sintaxis sobre el flujo elemental de vídeo comprimido que impone. Si se desea ampliar estos aspectos, las referencias que más se han consultado para la realización de referencia sobre la norma MPEG-2 vídeo son [1], [2], [3] y [4].

2.1 – NECESIDAD DE COMPRESIÓN EN EL VÍDEO DIGITAL

Al ritmo en el que la resolución de las pantallas digitales aumenta, el tamaño de los datos de la secuencia de vídeo para cubrir esa calidad en la imagen lo hace cuadráticamente (Figura 1). Así como tanto los medios de almacenamiento como los anchos de banda de los canales de transmisión no siguen este ritmo, se ve necesario introducir un proceso previo a la transmisión de los datos. Este proceso consiste en la reducción de la cantidad de datos a enviar adaptando así el flujo de datos de la secuencia de vídeo a las características del medio y su aplicación: una vídeoconferencia en un *smartphone* no dispone de los mismos recursos que un estudio de producción podría disponer para sus proyectos.

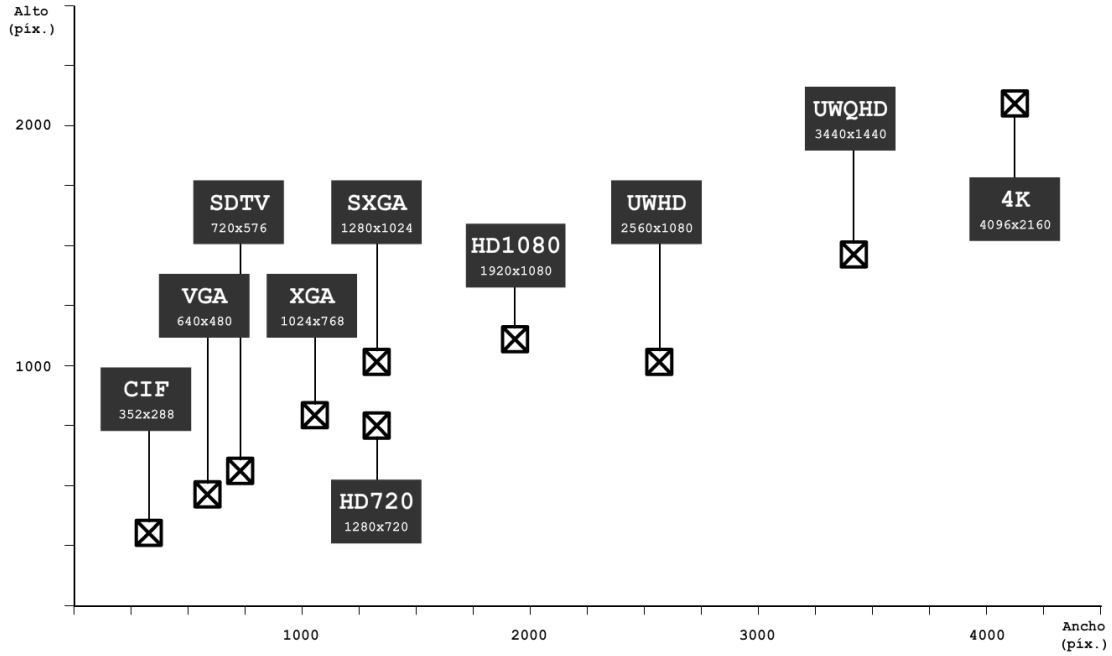

Figura 1 – Diferentes resoluciones de pantallas.

Como ejemplo, en la Televisión Digital Terrestre (TDT) se especifican múltiplex de 19,9 Mbps para la transmisión de varios canales de televisión, en los que las empresas

propietarias distribuyen la tasa de bits según el número de canales y la calidad de éstos. Un sólo canal HD sin compresión supondría algo más de 1,3 Gbps[2], por lo que, salvo se comprima previamente, no podría ser emitido. Esta compresión, en el mejor de los casos, significaría reducir el flujo de datos a menos del 2% del original.

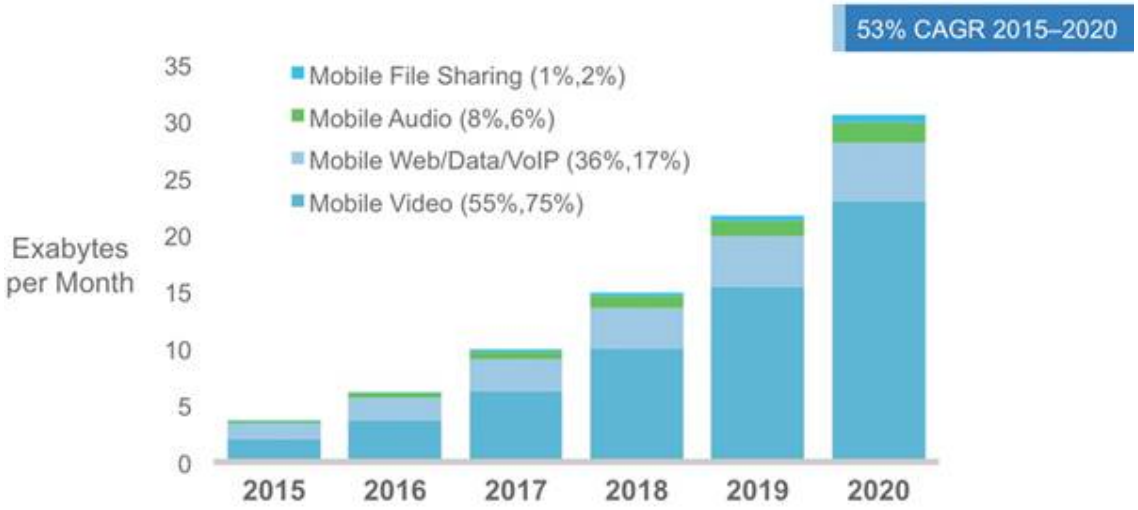

Figura 2 – Distribución del tráfico de datos móvil. [5]

Además, gracias a la evolución de los dispositivos móviles y a la mejora de su rendimiento, hoy en día tenemos la posibilidad de visualizar contenido multimedia en las pantallas de alta resolución que ofrecen. El tráfico de datos de vídeo en dispositivos móviles corresponde al 55% sobre el total, y la estimación sobre su evolución indica que en pocos años esta cifra puede convertirse en el 75% (Figura 2). Aun así debemos tener en cuenta que se trata de dispositivos de bajo consumo con recursos menores a los de los equipos fijos y que además trabajan sobre un canal inalámbrico con un ancho de banda menor a la disponible sobre la línea fija. Es por ello por lo que los compresores de vídeo digital no deben perseguir solamente la reducción de la cantidad de datos o la calidad de la imagen final, sino también optimizar los recursos disponibles y mejorar el rendimiento de los equipos necesarios para lograrlo.

2.2 – NORMAS DE COMPRESIÓN DE VÍDEO DIGITAL

Desde la aparición de las primeras herramientas y conversores digitales, los organismos internacionales han tratado de unificar y adecuar estándares de uso en el ámbito del vídeo digital. Estos han establecido unas reglas para definir la sintaxis del flujo de datos de vídeo digital comprimido, junto con la descripción del proceso de descodificación, pero dejando a elección del fabricante el diseño del codificador y sus algoritmos. Esto se hizo intencionadamente para promover la investigación en el ámbito de la compresión de vídeo entre los fabricantes y fomentar su competitividad en el mercado.

De entre todas las recomendaciones publicadas por el organismo ITU-T para la compresión de vídeo digital, la H.262 probablemente sea la que más presencia ha

[2] $720\,\frac{líneas}{fotograma} \cdot 1280\,\frac{píxeles}{línea} \cdot 3\,\frac{colores}{píxel} \cdot 8\,\frac{bits}{color} \cdot 60\,\frac{fotogramas}{segundo} = 1{,}327104\ Gbps$

tenido en el mundo digital. Publicada en 1995, la recomendación H.262 aportó nuevas características que su estándar antecesor, el H.261, no tenía: permitía trabajar con calidades y tamaños de imagen comparables a los de la televisión analógica del momento, flexibilizaba más el proceso de cuantificación y además incluía como novedad el soporte de vídeo entrelazado [3] en su flujo de datos. Todas estas características abrían la posibilidad de transmitir vídeo digital comprimido en aplicaciones de difusión de televisión, planteándose así alternativas a los sistemas de transmisión analógica tradicionales como PAL o NTSC. Así, el organismo ISO-IEC adoptó el H.262 para la parte de vídeo del estándar ISO-IEC 13818, más conocido como MPEG-2, utilizado en la difusión de vídeo digital terrestre DVB-T (TDT) y que también está presente en almacenamiento de vídeo sobre formatos físicos como es el caso del DVD.

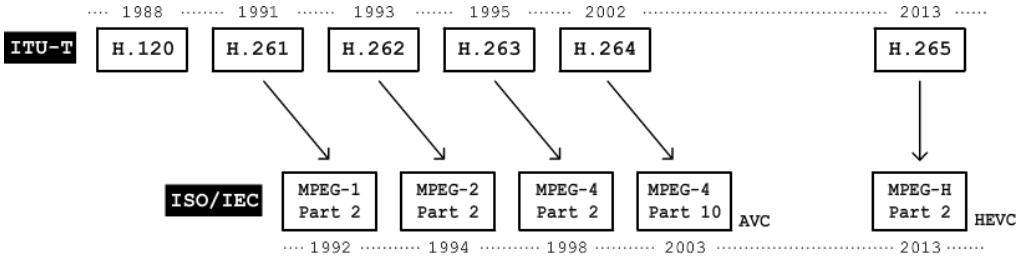

Figura 3 – Los estándares de vídeo digital y su aparición a lo largo del tiempo.

Aunque este trabajo se centra en MPEG-2 vídeo, es importante subrayar las dos causas fundamentales de la evolución en compresión de vídeo y del surgimiento de nuevos estándares antes y después de MPEG-2. Por un lado han surgido estándares cuando aparecían nuevas aplicaciones de transmisión o almacenamiento de vídeo digital: vídeoconferencia, vídeo en Internet (*streaming*, descargas de servidores o *peer-to-peer*), vídeo sobre redes inalámbricas, etc. Pero también ha habido una necesidad constante de incrementar las tasas de compresión alcanzables por otras causas, como la adaptación a nuevos dispositivos reproductores que ofrecen cada vez más resolución y calidad. Con estos propósitos apareció H.263 (adoptado en MPEG-4 parte 2), aunque H.264 (MPEG-4 parte 10, también llamado AVC - *Advanced Vídeo Coding*) pronto le robaría protagonismo al conseguir tasas de compresión del 50% sobre MPEG-2. H.264 es el estándar de facto en los discos Blu-Ray y en la televisión digital de alta definición[4].

La constante evolución de las pantallas de televisión, su resolución y el vídeo en tres dimensiones, han provocado una vez más la creación de un estándar optimizado para estas aplicaciones. H.265, definido en MPEG-H parte 2 (llamado HEVC - *High Efficiency Vídeo Coding*) en 2013, mejora de nuevo un 50% la compresión de su antecesor y añade soporte específico para resoluciones de ultra alta definición como 4K.

[3] El vídeo entrelazado trata cada cuadro o imagen en dos semicuadros denominados campos, de forma que uno de ellos esté compuesto por las líneas pares y el otro por las impares, correspondiendo cada uno de ellos a instantes temporales consecutivos. Lo opuesto a este término sería el vídeo progresivo, en el que no existe división en campos y cada cuadro se trata de forma completa.

[4] La televisión en definición estándar (SDTV) considera una variedad de tamaños de imagen en torno a 720x576 píxeles en Europa. En alta definición (HDTV) se consideran tamaños como 1280x720 o 1920x1080 (Full HD). La ultra alta definición o los sistemas 4K duplican los tamaños de imagen de la alta definición proporcionando calidades comparables a las del cine.

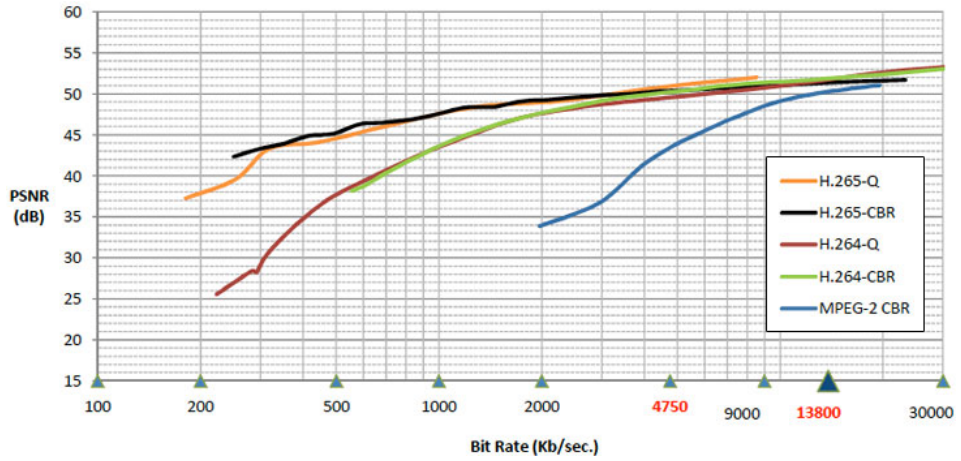

Figura 4 – Calidad de imagen vs. *bitrate* en diferentes estándares. [6]

2.3 – CODIFICADOR HÍBRIDO DE VÍDEO

La mayoría de codificadores de vídeo digital presentan una estructura similar, si bien cada vez los estándares añaden más elementos o incrementan la complejidad de alguno de ellos con la finalidad de conseguir un mejor resultado final, debido a la evolución en compresión de vídeo comentada. Todos ellos emplean un esquema de **codificador híbrido** enfocado a reducir o eliminar de forma simultánea la redundancia espacial, temporal y estadística, y que trabaja fotograma a fotograma sobre la secuencia de vídeo digital.

A lo largo de este capítulo se utilizará como referencia la definición en MPEG-2 vídeo, cuyo estudio es el propósito de este proyecto, para cualquier detalle que no sea común entre los estándares. Así, en la Figura 5, se presenta un esquema de codificador híbrido acorde a dicho estándar.

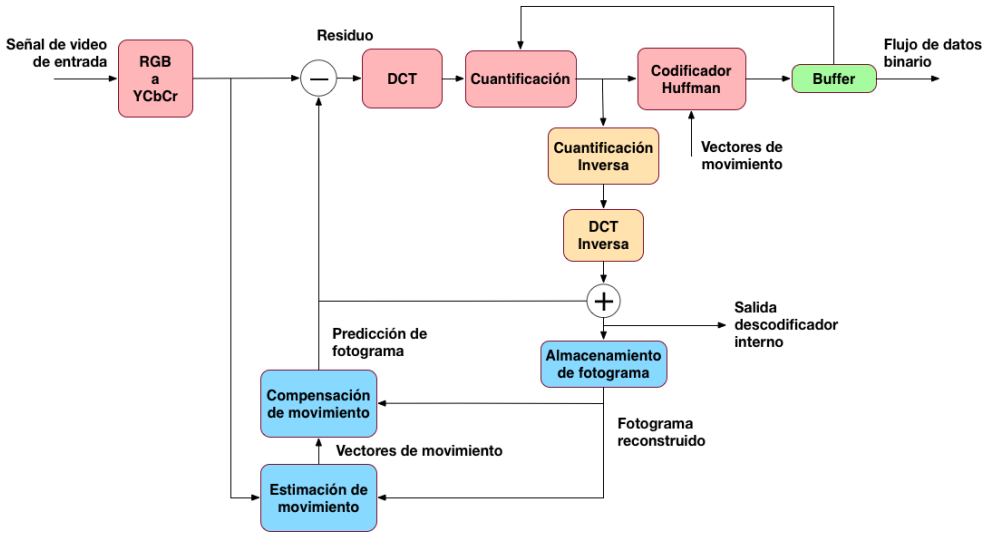

Figura 5 – Esquema de codificador híbrido utilizado en MPEG-2.

En vídeo digital una imagen en color la componen dos dimensiones de muestras (horizontal y vertical) para tres componentes diferentes. Para almacenamiento y transmisión de vídeo estas componentes no suelen ser los niveles habituales de rojo,

verde y azul. En lugar de esto, los codificadores digitales de vídeo trabajan con una componente de luminancia y dos de crominancia [7] [8]. La luminancia (representada como *Y*) determina el nivel de gris de la imagen, mientras que la crominancia representa la variación del color respecto al color azul (C_B) y el color rojo (C_R).

Por lo expuesto en el párrafo anterior, el volumen de datos a comprimir para una imagen en color sería 3 veces superior al de la misma imagen representada en niveles de gris (formato conocido como 4:4:4). Sin embargo se explota una propiedad conocida del sistema de visión humano consistente en que este es más sensible a la variación de los niveles de gris que a la variación en el color. Esto permite reducir las dimensiones de los planos de crominancia a la mitad (formato 4:2:0) sin que el usuario final perciba degradaciones a causa de esto. De esta manera, el hecho de usar color supone solo un incremento de 1,5 en el volumen de datos respecto al caso en que se trabajara solamente en niveles de gris.

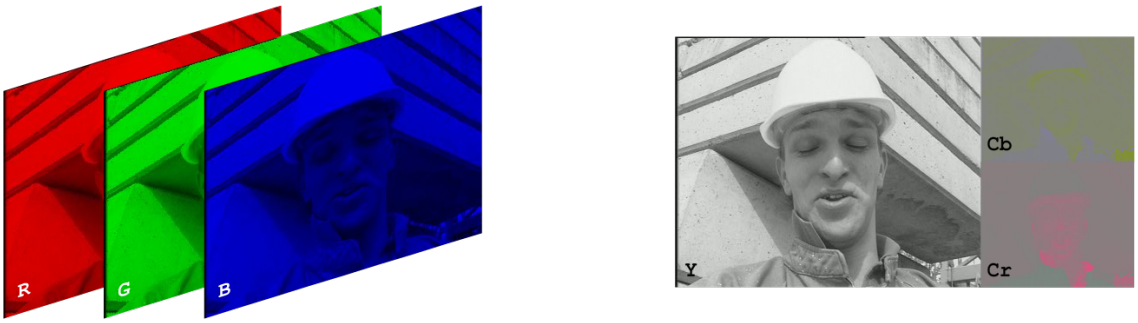

Figura 6 – Representación 4:4:4 RGB y 4:2:0 YC_BC_R de una imagen.

TRANSFORMADA Y CUANTIFICADOR

Una peculiaridad importante de los codificadores híbridos de vídeo es que no introducen directamente en el flujo binario los valores de los píxeles de la imagen. En su lugar, introducen unos **coeficientes** obtenidos a través de una transformación matemática. En el caso de MPEG-2 vídeo los coeficientes están asociados a una representación frecuencial de bloques de pequeño tamaño de la imagen (8x8 píxeles). Por lo general los coeficientes asociados a frecuencias altas son menos relevantes tanto estadísticamente (es más frecuente que sus magnitudes sean bajas) como perceptualmente (el ojo humano ya no es capaz de resolver o discernir frecuencias espaciales altas o detalles demasiado finos en la imagen). Por esto se podrán eliminar o cuantificar de forma más grosera los coeficientes correspondientes a las altas frecuencias. Se consigue así reducir la cantidad de datos introducida en el flujo binario.

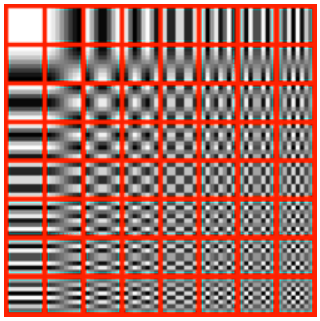

Figura 7 – Frecuencias espaciales de un bloque transformado de tamaño 8x8.

La representación frecuencial concreta que emplea MPEG-2 es la Transformada Discreta del Coseno (DCT – *Discrete Cosine Transform)* de forma bidimensional. Se escoge esta frente a otras ya que posee buenas propiedades teóricas (los valores de los coeficientes son reales, compacta la energía en pocos coeficientes) y prácticas (existencia de algoritmos que permiten su cálculo eficientemente) [9].

$$F(u,v) = \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) cos \frac{(2x+1)u\pi}{2N} cos \frac{(2y+1)v\pi}{2N}$$

$$C(u), C(v) = \begin{cases} 1/\sqrt{2} & si\ u,v = 0 \\ 1 & si\ u,v \neq 0 \end{cases}$$

Fórmula 1 – Transformada Discreta del Coseno Bidimensional NxN. N=8 en MPEG-2.

Gracias al uso de esta representación frecuencial los bloques correspondientes a partes de la imagen que contienen colores sólidos o pocos detalles únicamente tendrán coeficientes relativos a las bajas frecuencias, lo que conlleva a transmitir menos información.

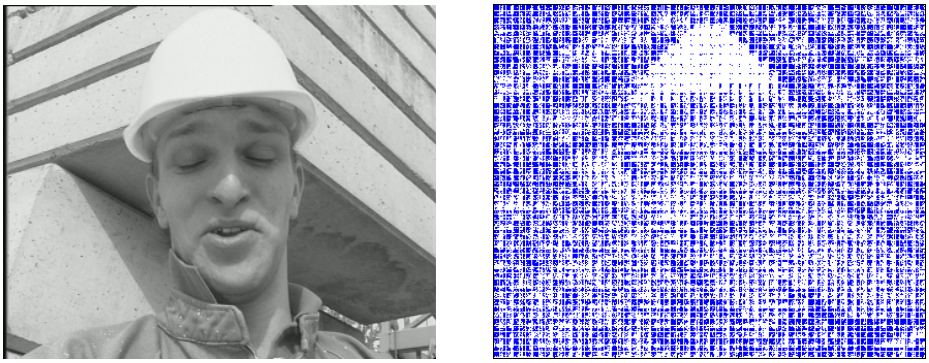

Figura 8 – Imagen original (izquierda) y coeficientes DCT retenidos para su reconstrucción (derecha).

Como se ha adelantado, una vez calculados los coeficientes a continuación se procede a realizar un procesado sobre los mismos que sirva a fines de compresión. Los bloques 8x8 se dividen elemento a elemento por matrices de cuantificación, como la que se presenta en la Figura 9, y el resultado se cuantifica mediante redondeo. Se escogen matrices de cuantificación como la del ejemplo ya que premian las bajas frecuencias espaciales, situadas en la esquina superior izquierda. Esta matriz debe ser conocida por el receptor para poder reconstruir los niveles originales o aproximados de la señal original.

Es en este proceso de cuantificación es donde el codificador híbrido introduce pérdidas, puesto que, en general, todos los coeficientes DCT pierden resolución y muchos de ellos se harán nulos, lo que hará que no puedan ser reconstruidos convenientemente en el receptor. Las pérdidas se traducirán en que la señal final reconstruida no será exactamente igual a la original (Figura 10). Pero el hecho de que muchos coeficientes DCT se anulen es precisamente lo que posibilita tasas de compresión altas. Los métodos de compresión sin pérdidas (*lossless coding*) dan lugar

a tasas de compresión demasiado modestas y únicamente se emplean en aplicaciones específicas como la transmisión de vídeo por cable HDMI (*High-Definition Multimedia Interface*).

$$\begin{bmatrix} 8 & 16 & 19 & 22 & 26 & 27 & 29 & 34 \\ 16 & 16 & 22 & 24 & 27 & 29 & 34 & 37 \\ 19 & 22 & 26 & 27 & 29 & 34 & 34 & 38 \\ 22 & 22 & 26 & 27 & 29 & 34 & 37 & 40 \\ 22 & 26 & 27 & 29 & 32 & 35 & 40 & 48 \\ 26 & 27 & 29 & 32 & 35 & 40 & 48 & 58 \\ 26 & 27 & 29 & 34 & 38 & 46 & 56 & 69 \\ 27 & 29 & 35 & 38 & 46 & 56 & 69 & 83 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 5 & 6 & 14 & 15 & 27 & 28 \\ 2 & 4 & 7 & 13 & 16 & 26 & 29 & 42 \\ 3 & 8 & 12 & 17 & 25 & 30 & 41 & 43 \\ 9 & 11 & 18 & 24 & 31 & 40 & 44 & 53 \\ 10 & 19 & 23 & 32 & 39 & 45 & 52 & 54 \\ 20 & 22 & 33 & 38 & 46 & 51 & 55 & 60 \\ 21 & 34 & 37 & 47 & 50 & 56 & 59 & 61 \\ 35 & 36 & 48 & 49 & 57 & 58 & 62 & 63 \end{bmatrix}$$

Figura 9 – Ejemplo de matriz de cuantificación y matriz de ordenación de MPEG-2.

Una vez obtenidos los coeficientes cuantificados se ordenan desde la menor frecuencia espacial hasta la mayor, utilizando una matriz de ordenación que dictará la disposición en la que aparecerán en el flujo de datos, también conocida como matriz de zig-zag (Figura 9).

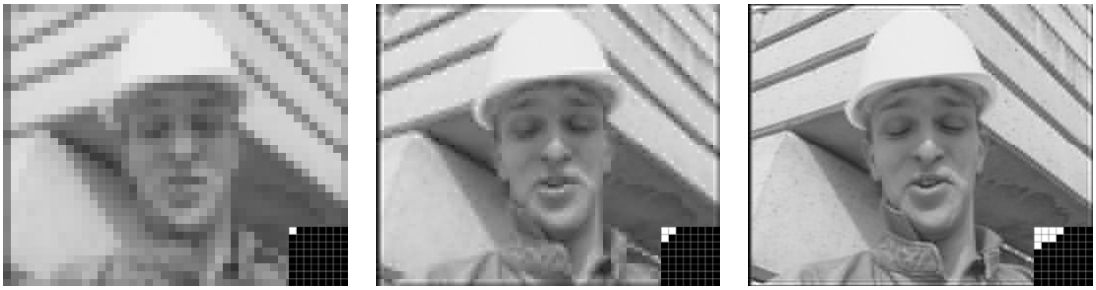

Figura 10 – Calidad del resultado según el número de coeficientes retenidos, indicado en la esquina inferior derecha.

PREDICCIÓN TEMPORAL

Una de las técnicas que utiliza el codificador híbrido para minimizar la redundancia temporal es la estimación de movimiento. Las secuencias de vídeo, por lo general, están compuestas por imágenes de un mismo plano que varía a lo largo del tiempo. Esto provoca que aparezcan partes que apenas cambien en la escena o que si lo hacen lo hagan en bloque. Habrá algunas que permanecerán estáticas, como pueden ser las relativas al fondo de la secuencia, y otras que se desplazarán a lo largo del cuadro, como podrá ocurrir con elementos tanto del fondo como del primer plano.

La estimación y compensación de movimiento aparece para optimizar todos estos casos al minimizar la redundancia temporal. En vez de transmitir todos los coeficientes correspondientes a un bloque de la imagen, la idea es que el descodificador pueda reconstruir ese mismo bloque a partir de uno previamente transmitido. Por ello se busca ese mismo bloque en un fotograma anterior de referencia dentro de un área acotada relativa a su posición actual. El vector diferencia entre la posición del macrobloque en la imagen de referencia y su posición en la imagen actual se denomina **vector de movimiento**. Una componente positiva significa que el bloque de referencia se encuentra a la derecha o abajo del bloque actual, en función la componente de la que se trate.

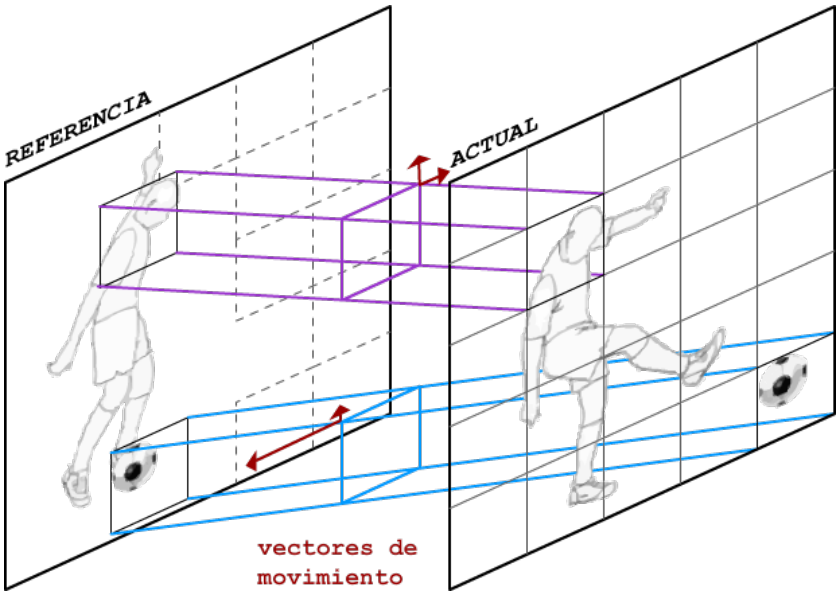

Figura 11 – Relación entre dos fotogramas mediante los vectores de movimiento.

Dentro de un bloque, además de haberse movido o cambiar su posición de una imagen a otra, también pueden cambiar los valores de sus píxeles. Si se estima que ese cambio es lo suficientemente importante para que el descodificador pueda reconstruir ese bloque con cierta calidad no basta con enviarle el vector de movimiento. Necesitará también el **residuo de predicción**, que es la diferencia entre los valores de píxel actuales y los predichos, y que también será transformado y cuantificado. En el caso de los bloques con predicción temporal MPEG-2 permite añadir estos dos parámetros (vectores de movimiento y coeficientes DCT del residuo) al flujo de datos.

Figura 12 – Vectores de movimiento, residuo y residuo transformado. Nótese que el residuo transformado genera menos coeficientes no nulos que sin predicción temporal (Figura 8).

Las imágenes de referencia que el codificador emplea en la predicción temporal no se corresponden con las existentes en la secuencia original de vídeo. Internamente el codificador realiza el proceso inverso a la transformada y la cuantificación de la señal para calcular las imágenes que obtendrá el decodificador. Se utilizarán estas para realizar la estimación de movimiento y posteriormente su compensación. Con otras palabras, el codificador alberga internamente el decodificador para simular la compensación que realizará éste.

Hasta ahora, cuando se ha hablado de imagen de referencia, se ha pensado implícitamente en una sola imagen que tuvo lugar en un instante anterior para que

pueda disponer de ella también el descodificador. Esto no siempre es así. Se puede usar como referencia también una imagen que tendrá lugar en un instante temporal posterior, pero en este caso el orden en que se codifican las imágenes y el orden en el que se reproducen han de ser distintos. De hecho, para cierto tipo de imágenes, MPEG-2 permite que la imagen de referencia pueda ser pasada, futura o, incluso, usar ambas opciones y promediar el resultado.

Como se ha adelantado antes, los estándares de codificación de vídeo digital únicamente describen el proceso de descodificación y la sintaxis del flujo de datos, por lo que recae sobre el diseño del codificador la elección de los algoritmos que generan los vectores de movimiento. La mayor parte de la carga computacional del proceso de compresión recae precisamente sobre la ejecución de estos algoritmos de estimación de movimiento. El proceso de compresión-descompresión es fuertemente asimétrico, puesto que el descompresor simplemente lee los vectores de movimiento del flujo binario sin tener que calcularlos.

CODIFICADOR BINARIO

La última etapa en el proceso de compresión consiste en introducir la representación de la secuencia de vídeo que ha obtenido el codificador híbrido (coeficientes transformados, vectores de movimiento, parámetros de la secuencia) en un flujo binario comprimido o flujo elemental de vídeo. Esta etapa es fuertemente dependiente del estándar de compresión concreto adoptado, puesto que lo que las normas fijan es precisamente el formato de estos flujos.

En el caso concreto de MPEG-2 la codificación de los diversos elementos viene muy marcada por la utilización de tablas que proporciona el estándar que han sido optimizadas para reducir la redundancia estadística. En aras de comprimir más los vectores de movimiento y los coeficientes transformados no se codifican directamente. Por ejemplo:

- Los vectores de movimiento se codifican diferencialmente. Esto es más eficiente cuando, como es frecuente, los vectores de movimiento de macrobloques consecutivos son muy similares.
- La mayoría de coeficientes DCT se codifican mediante RLE (Run-length encoding). Esto quiere decir que sólo se codifican los coeficientes no nulos y las palabras código no representan únicamente el valor de dichos coeficientes sino, además, el número de coeficientes de valor cero que les preceden en el barrido en zig-zag del bloque. Esto se hace así porque estadísticamente ocurre con frecuencia que aparezcan ristras largas de ceros entre coeficientes no nulos en el barrido en zig-zag.

2.4 – ESTRUCTURACIÓN DE LA SECUENCIA DE VÍDEO EN MPEG-2

La descomposición de la secuencia de vídeo en elementos de menor tamaño mejora el proceso de codificación tanto en la eliminación de la redundancia temporal y espacial como para permitir el procesado en paralelo. Además la estructuración de los

datos en una jerarquía o capas ayuda al descodificador a la hora de sincronizar la secuencia de vídeo. Así, en el caso de que aparezca algún error en el flujo de datos, únicamente se verá perjudicado el elemento de menor tamaño afectado.

La secuencia de vídeo en MPEG-2 se divide de la forma en que se muestra en la Figura 13. De esta forma, cada nivel almacenará información correspondiente a distintos aspectos de la secuencia de vídeo.

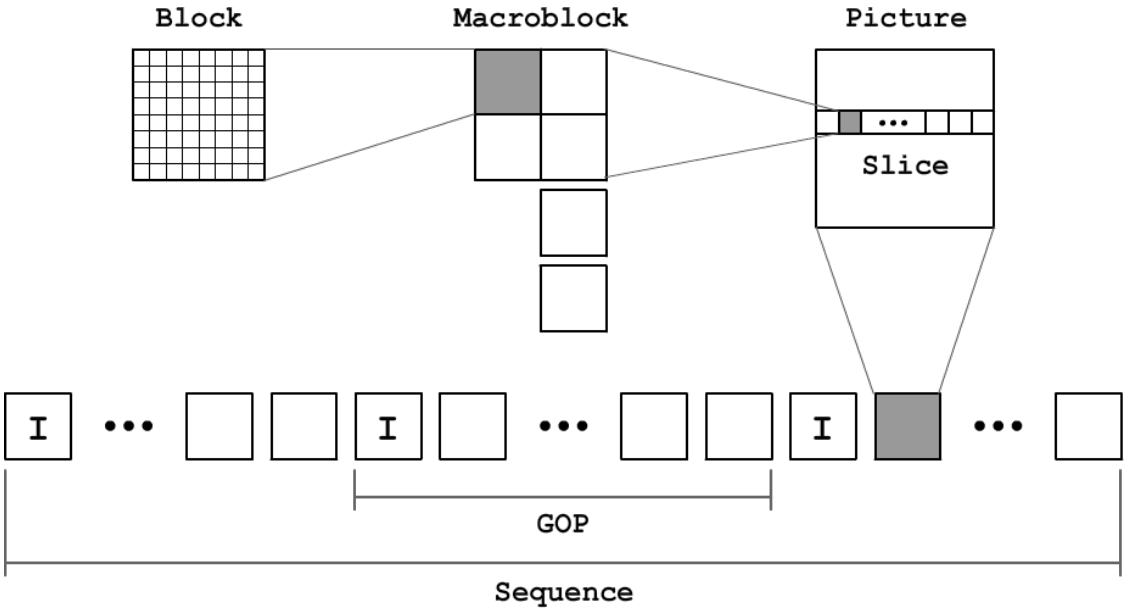

Figura 13 – Elementos que componen una secuencia de vídeo 4:2:0 en MPEG-2.

El elemento básico en MPEG-2 es el bloque (*block*), compuesto por 8x8 píxeles de una componente de una imagen de la secuencia de vídeo. A este elemento se le aplica la transformada DCT y la cuantificación para obtener los coeficientes que serán transmitidos en el flujo de datos.

El elemento por encima de la jerarquía es el macrobloque (*macroblock*). Se trata de una agrupación de seis bloques pertenecientes a un sector de 2x2 bloques de la imagen. Cuatro de ellos corresponderán a la componente de luminancia y existirá uno adicional por cada componente de crominancia, ajustándose al formato 4:2:0 de la señal de vídeo. El macrobloque es la unidad fundamental en la gestión de predicción temporal e información de movimiento, puesto que los vectores de movimiento se obtienen para los macrobloques.

Una rebanada o franja (*slice*) es el siguiente elemento más alto en la jerarquía y se trata del conjunto formado por varios macrobloques consecutivos. Aunque su posición en la imagen y longitud no están determinados, es habitual hacer corresponder un *slice* con cada fila de macrobloques de la imagen. Se añade este nivel para aumentar la resiliencia del descodificador frente a errores y fijar ciertos parámetros, como la resolución de los cuantificadores utilizados para la cuantificación de coeficientes transformados.

El siguiente nivel que aparece es el nivel de *picture*, que por lo general corresponderá a cada imagen de la secuencia de vídeo[5] y así se asumirá aquí. MPEG-2 vídeo contempla tres tipos diferentes de imagen:

- Las imágenes de tipo I (Intra), donde no se explota la predicción temporal. Introducir periódicamente imágenes de este tipo, al suponer un reseteo de los mecanismos de predicción temporal, contribuye a evitar la propagación de errores a través de la compensación de movimiento.
- Las imágenes de tipo P (Predictivas), donde se puede explotar la predicción temporal, pero siempre empleando como referencia un fotograma que ocurrió en un instante anterior.
- Por último, las imágenes de tipo B (Bidireccionales), donde la predicción temporal de cada macrobloque podrá ocurrir tomando como referencia un macrobloque de la imagen anterior, uno de una imagen futura, o el promedio de las dos predicciones anteriores. El usar imágenes futuras como referencia, para que estas ya estén disponibles cuando son necesarias en el descodificador, requerirá que el orden de codificación y de presentación de las imágenes sea distinto. El detalle se explica en el Anexo I.

Las imágenes están agrupadas de forma consecutiva en los llamados Grupos de Imágenes (también llamados GOP – *Group Of Pictures*). Cada GOP contiene al menos una imagen de tipo I y esta se debe encontrar al principio, marcando el punto a partir del cual cualquier receptor puede empezar a descodificar la señal al recibir un flujo de datos MPEG-2 vídeo. En otras palabras, esta división permite un acceso aleatorio de la secuencia de vídeo por parte de cualquier descodificador.

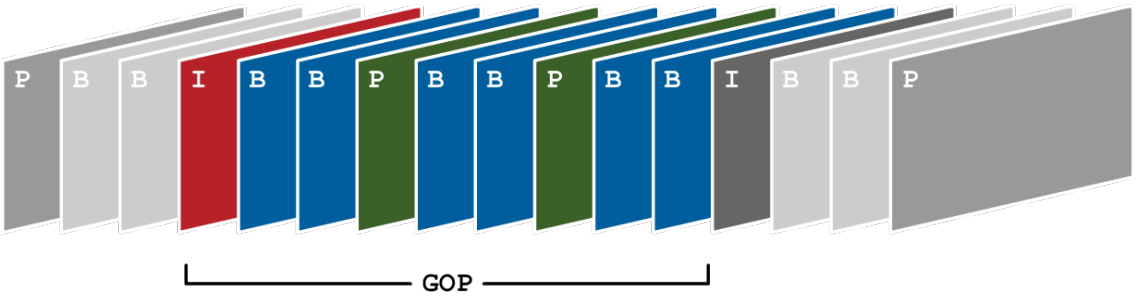

Figura 14 – Ejemplo de GOP dentro de una secuencia de vídeo.

Por último, el elemento que alberga a todos los demás es el de secuencia (*sequence*), que corresponde a la secuencia de vídeo completa y alberga tantos GOP como sean necesarios para representarla. En este nivel es donde se definen parámetros como el tamaño de las imágenes o la velocidad de los fotogramas.

2.5 – SINTAXIS DEL FLUJO DE ELEMENTAL DE VÍDEO MPEG-2

La sintaxis de cada estándar marca la forma en la que la información debe aparecer en el flujo de datos. En el caso de MPEG-2, la estructura de los elementos de la

[5] Para vídeo progresivo cada *picture* en MPEG-2 se corresponde con un fotograma de la imagen. Para vídeo entrelazado existen otras opciones que no se consideran aquí y que son poco usadas.

secuencia de vídeo se corresponde también con una jerarquía de capas o niveles que aparece en la sintaxis de su flujo elemental de vídeo. A cada capa le acompaña una cabecera que contiene parámetros correspondientes a ese nivel que permanecerán activos para todos los elementos contenidos en dicha capa y, salvo que otra cabecera indique lo contrario, en capas inferiores. Estas cabeceras son además las que permiten al descodificador detectar en qué punto del flujo de datos se encuentra para volver a sincronizarse en caso de errores.

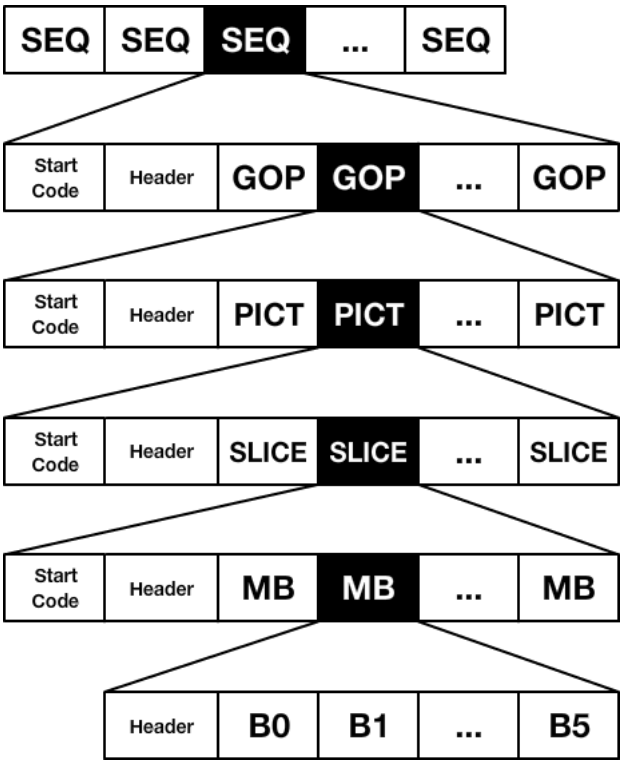

Figura 15 – Estructura o capas del flujo de datos en MPEG-2 vídeo.

En el Anexo I se dispone de más información detallada sobre cada capa de la sintaxis de un flujo elemental de vídeo MPEG-2.

3 - DESARROLLO DE LA HERRAMIENTA

La principal contribución de este proyecto consiste en haber creado una herramienta didáctica en el entorno de Matlab. Dicha herramienta permite que el alumno siga el proceso de compresión de vídeo desde de una fuente de vídeo original sin comprimir en formato 4:2:0 hasta la generación de un flujo elemental comprimido que cumple las especificaciones de la norma MPEG-2. Al comienzo del capítulo se presenta la forma en la que se ha enfocado el problema, separándolo de manera modular en tres partes (compresor híbrido, codificador binario e interfaz gráfica) que se detallan posteriormente. Al final del capítulo se describe cómo se ha validado el desarrollo y se analizan los resultados que proporciona la herramienta.

3.1 – ENFOQUE DEL PROBLEMA

Por lo general los codificadores software de MPEG-2, que no están diseñados para fines didácticos, trabajan sobre la secuencia de vídeo realizando a la vez lo que son propiamente tareas de procesado de vídeo (como la obtención de los coeficientes DCT y la estimación/compensación de movimiento) y tareas de formateado y escritura del flujo binario comprimido.

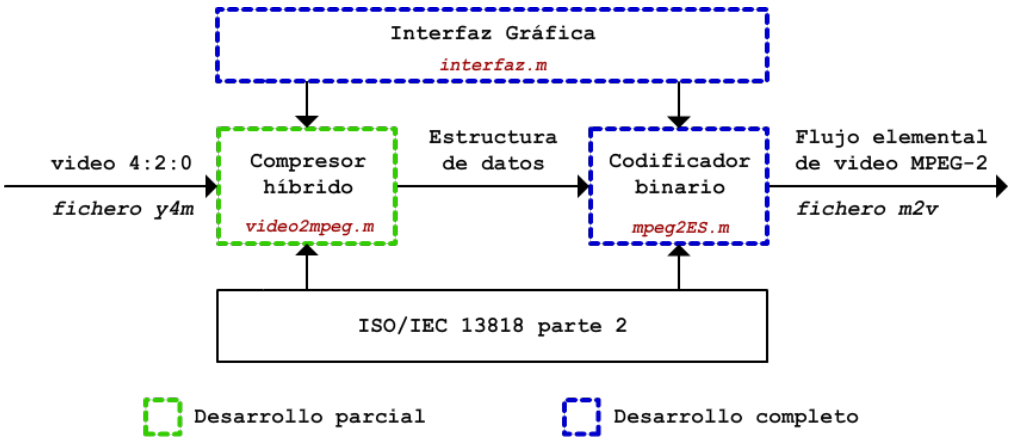

Figura 16 – Esquema del desarrollo del codificador MPEG-2.

En nuestro caso, por tratarse de una herramienta didáctica, se ha decidido separar las partes de procesado y de escritura de flujo binario, que conceptualmente son bastante independientes (Figura 16). Esto permite que el alumno las trabaje por separado. Para la parte de procesado se implementa un codificador híbrido de vídeo. Su entrada es el vídeo sin comprimir. Su salida es una variable de Matlab que aprovecha los tipos de datos de los que dispone este entorno (estructuras, *cell-arrays*) para almacenar de forma organizada todo aquello que debe producir un codificador híbrido: coeficientes DCT cuantificados, vectores de movimiento, etc. La forma exacta de estructurar los datos se introduce en la siguiente sección y se detalla en el Anexo III. Este tipo de estructura tiene por sí misma cierto interés y valor didáctico, pero, además de eso, constituye la entrada al segundo módulo, encargado de la escritura del flujo binario. Este módulo se ocupa menos de aspectos de procesado y más de lo que es el cumplimiento exacto de la norma MPEG-2 vídeo. También se ha creado una

interfaz gráfica que le permite al usuario crear archivos comprimidos y obtener resultados del proceso de compresión sin necesidad de trabajar con código Matlab o de conocer la división en módulos comentada.

Como puede apreciarse en la Figura 16, se considerará que la entrada a la herramienta es vídeo sin comprimir en formato 4:2:0 almacenado en un archivo de tipo extensión *y4m* [10]. Las fuentes de vídeo que se han descargado para trabajar en este proyecto están en este formato [11]. La salida será un flujo elemental de vídeo comprimido que respete la norma MPEG-2. Este flujo se introduce directamente en un archivo de extensión *m2v*, que es la que se usa típicamente para este fin.

3.2 – COMPRESOR HÍBRIDO DE VÍDEO

La razón de que en la Figura 16 el módulo de compresión híbrida aparezca etiquetado como parcialmente desarrollado es porque ya se disponía de un módulo de funcionalidad similar al que se va a presentar aquí desde el inicio del proyecto. No obstante, se realizaron modificaciones importantes sobre él y se añadieron nuevas capacidades sobre esta base de la que se partía. Por ejemplo, hubo que adaptar el proceso de cuantificación para hacerlo compatible a MPEG-2 y hubo que añadir la precisión subpíxel, como se explica más adelante. Con la idea de hacer la exposición que sigue más clara, se describe a continuación este módulo en su totalidad, con independencia de si se trataba de partes que venían dadas o que se modificaron o añadieron en el desarrollo del proyecto.

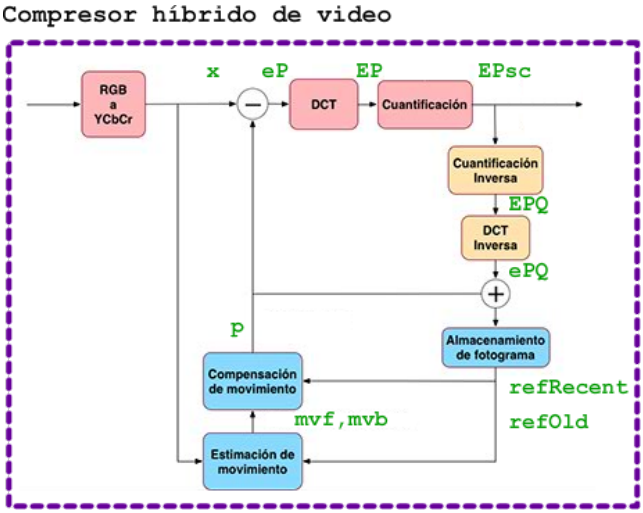

Figura 17 – Bloques del compresor híbrido junto con las variables usadas en el código.

Una sola función de Matlab *vídeo2mpeg.m* (disponible en el Anexo IV) realiza toda la funcionalidad del módulo, implementando el compresor híbrido según el diagrama de bloques que se representa en la Figura 17. Para que el código sea claro y representativo de todo el proceso minimiza el número de llamadas a funciones externas adicionales salvo en casos que complicarían la comprensión, como es el caso de la estimación de movimiento. Para que sea eficiente se hace uso de la característica de indexación de Matlab para realizar cómputos como los de la DCT o los vectores de movimiento sobre toda una imagen simultáneamente. La nomenclatura seguida para nombrar variables sobre el diagrama de bloques de la Figura 17, que se les

proporcionaría a los alumnos como dato, es la misma que para las variables dentro del código. Esto permitirá a los alumnos seguir el diagrama de bloques en un proceso de depuración del código de la función *vídeo2mpeg.m*. El hecho de que haya 2 variables para guardar imágenes de referencia (*refRecent* y *refOld*) y vectores de movimiento (*mvf* y *mvb*) se debe a las peculiaridades del mecanismo de predicción temporal para imágenes de tipo B que se explicó en el capítulo anterior.

Como nivel intermedio entre el nivel de implementación concreta que supone el código *vídeo2mpeg.m* y el de representación abstracta que supone el diagrama de bloques de la Figura 17, se proporciona a continuación pseudocódigo que explicita las tareas principales en el proceso de compresión híbrida:

```
DESDE fotograma = 1 HASTA N ENTONCES

    x = secuencia_de_vídeo[fotograma]

    SI tipo_de_imagen = I ENTONCES
        p = 128

    SI tipo_de_imagen = P ENTONCES
        mvf = vectores_de_movimiento(interpolar(x),refRecent)
        p = predecir_imagen(refRecent,mvf)

    SI tipo_de_imagen = B ENTONCES
        [mvf,mvb] = vectores_de_movimiento(interpolar(x),refRecent,refOld)
        p = predecir_imagen(refRecent,refOld,mvf,mvb)

    eP = x - p
    EP = DCT(eP)
    EPsc = cuantificación(EP)

    guardar_en_estructura(estructura_de_datos,fotograma,EPsc,mvb,mvf)

    EPQ = cuantificación_inversa(EPsc)
    ePQ = iDCT(EPQ)

    SI tipo_de_imagen = I o P ENTONCES
        refOld = refRecent
        refRecent = interpolar(p + ePQ)

DEVUELVE estructura_de_datos
```

Todas las operaciones que se realizan en el pseudocódigo, salvo quizá la de interpolar, han sido explicadas en el capítulo anterior. La necesidad de interpolar en un factor 2 en aquellas imágenes que intervienen en los procesos de predicción temporal surgió para adaptarse a MPEG-2, ya que este estándar establece ½ píxel como unidad de los vectores de movimiento (precisión subpíxel). La interpolación ha de hacerse del modo prescrito en la norma [1], [12]. En este módulo de procesado de vídeo se decidió además permitir factores de interpolación acordes a otras normas (1 en MPEG-1, 4 en H.264 y H.265).

Mediante la separación en módulos se pretendía independizar lo más posible el compresor híbrido del cumplimiento exacto de la norma MPEG-2, pero el desacoplo total entre estas funciones no es posible. Hubo que tomar ciertas decisiones porque MPEG-2 ofrece unas posibilidades de codificación muy amplias y convenía revisar si era conveniente, práctico o didáctico que la herramienta las contemplase todas. Por ejemplo, en MPEG-2 se pueden explotar características especiales de compresión para el caso concreto de vídeo entrelazado a diversos niveles (*picture*, *macroblock*, *block*).

Se decidió no incluir todas estas opciones específicas de vídeo entrelazado porque complican excesivamente el proceso de compresión sin aportar demasiado a la comprensión de los principios fundamentales. Por tanto, la herramienta diseñada no está optimizada para el uso de vídeo entrelazado y sólo incorpora una de las opciones específicas de entrelazado del estándar (el uso de un barrido en zig-zag alternativo en los coeficientes DCT de los bloques).

También hubo que tomar decisiones a este nivel sobre todas aquellas etapas del proceso de codificación que MPEG-2 no fija. Este es el caso, especialmente, de la cuantificación directa de coeficientes (ya que la norma solo define la cuantificación inversa) y de la estimación de movimiento.

Para la cuantificación directa de coeficientes se partió de las expresiones que pueden encontrarse en [13] (páginas 301-303). No obstante, se modificó ligeramente la fórmula de cuantificación para cierto tipo concreto de coeficientes (coeficientes AC en bloques de tipo intra). Esto se hizo así porque con la expresión original de la referencia el cuantificador directo y el inverso no compensaban sus efectos. En el Anexo II se presenta un ejemplo de transformación y cuantificación de coeficientes DCT de bloques.

La estimación de movimiento consiste en buscar dónde aparece una subimagen (macrobloque actual a codificar) dentro de una imagen de mayor tamaño (región de búsqueda en la imagen de referencia). En procesado de imagen esta tarea se conoce como *template matching* [14]. A cada posible posición de la subimagen dentro de la imagen se le asigna una medida de parecido para determinar si esa posición es la mejor a considerar. El parecido se obtiene teniendo en cuenta los valores de píxel de la subimagen (valores t_n que se agrupan en un vector $\bar{t}$) y los de la imagen de mayor tamaño en la posición concreta que se está evaluando (valores x_n que se agrupan en un vector $\bar{x}$). La medida de parecido más típicamente empleada en los codificadores de vídeo es la suma de diferencias absolutas (*SAD – Sum of Absolute Differences*) presentada en la Fórmula 2.

$$SAD = \sum_{n=1}^{N} |x_n - t_n|$$

Fórmula 2 – Suma de diferencias absolutas entre dos vectores.

La medida de parecido *SAD* proporciona resultados similares a la distancia euclídea al cuadrado (*SSD – Sum of Square Differences*) [15]. Como puede apreciarse en la Fórmula 3, esta medida de parecido puede calcularse alternativamente evaluando productos escalares. En el contexto de procesado de señal los productos escalares evaluados para diversas posiciones son correlaciones y estas se pueden calcular de manera eficiente mediante algoritmos FFT. Esta es la razón por la que se usa la medida de parecido *SSD* en nuestro trabajo, ya que Matlab incorpora todas estas herramientas para un cálculo eficiente. Una ventaja adicional de esto es que la búsqueda que realiza la herramienta es exhaustiva (*full search*). Esto quiere decir que se consideran todas las posibles posiciones de la subimagen dentro de la imagen en la

que se busca. Otras opciones, para reducir la cantidad de cálculos, realizan búsquedas de otra naturaleza en las que no se barren todas las posibles posiciones [16]. Esto puede hacer perder precisión en los vectores de movimiento obtenidos, lo que podría aumentar el residuo generado y reducir la compresión, a cambio de ganar velocidad en el cómputo.

$$SSD = \sum_{n=1}^{N} (x_n - t_n)^2 = \bar{x}^T \bar{x} + \bar{t}^T \bar{t} - 2\bar{x}^T \bar{t}$$

Fórmula 3 – Suma de diferencias cuadráticas entre dos vectores.

Un último aspecto sobre el que conviene incidir para explicar el compresor híbrido implementado es una explicación sobre lo que proporciona como salida. Se trata de una única variable que contiene toda la información relevante que obtiene un compresor híbrido. En el siguiente ejemplo que se muestra a continuación esta variable se llama *mpeg*. La variable es una estructura cuyos campos contienen la información del vídeo comprimido. Por ejemplo, *mpeg.picttype* (de valor 'IBBPBBPBBP') contiene los tipos de imagen en el orden de reproducción de la secuencia. *mpeg.frames{i}* contiene toda la información relativa al frame i en el orden de codificación. En el ejemplo se particulariza para i=2, mostrándose que es posible acceder a toda la información de esta imagen: vectores de movimiento *mvf*, un bloque concreto de luminancia (el de posición 20,35), etc. En el Anexo III se proporciona un esquema detallado del contenido de esta estructura.

```
>> mpeg
mpeg =
      params: [1x1 struct]
    picttype: 'IBBPBBPBBP'
      frames: {1x10 cell}

>> mpeg.frames{2}
ans =

    type: 'P'
    quantiser_scale_code: 4
    quantiser_scale_type: 1
      intra_dc_precision: 0
                     mvf: [18x22x2 double]
             blockqdcty: [4-D double]
          blockqdctcbcr: [5-D double]

>> mpeg.frames{2}.blockqdcty(:,:,20,35) % Bloque (20,35) de luminancia
ans =
     4     1     0     0    -3     0     0     0
     0     2     0     0     1    -1     0     0
    -1     0     1     0     0     0     0     0
     0     0     1     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
```

3.3 – CODIFICADOR BINARIO DEL FLUJO ELEMENTAL DE VÍDEO MPEG-2

El objetivo de este módulo es el de facilitar al alumnado la verificación de la compresión realizada en el proceso completo. Para ello el módulo debe ser capaz de generar un flujo elemental de vídeo acorde a MPEG-2 reproducible en cualquier herramienta multimedia compatible con dicho formato. Esto permitirá al alumno revisar el compromiso compresión–calidad a partir del fichero generado en disco, a la vez que le garantiza que el proceso se ajusta en todo momento a la norma MPEG-2.

Las decisiones del diseño de este módulo se obtienen a partir de una primera fase de recopilación de información. La mayor fuente de información ha sido el propio estándar MPEG-2 [1], aunque también ha sido necesaria la consulta de otros recursos. Un tipo de fuentes adicionales ha sido el de libros que documentan el proceso de codificación directa en MPEG-2, aportando además otro tipo de información útil como ejemplos numéricos y gráficos [3]. También se han encontrado implementaciones incompletas o relacionadas con la que se pretendía abordar aquí en Matlab [17], [18]. Se han reaprovechado algunos elementos puntuales concretos de estas implementaciones, como tablas del estándar o bits de cabecera de los diversos niveles. Pero, al margen de estos detalles, la implementación realizada aquí es original e independiente y se basa sobre todo en las fuentes bibliográficas.

Analizando la complejidad del módulo se tomó la decisión de dividir el desarrollo interno en tres partes, con las que se conseguirían los tres hitos más importantes. El primero de ellos se alcanzaría al codificar secuencias únicamente con imágenes tipo I, el segundo con imágenes también de tipo P y el último de ellos, con el que se conseguiría el módulo completo, sería el de lograr codificar una secuencia de vídeo con cualquier tipo de imagen (I, P y B). Para la comprobación de los logros en cada hito se ha dispuesto de tres secuencias de vídeo diferentes: una imagen estática repetida a lo largo de todos los fotogramas, una secuencia en la que se aprecia un barrido diagonal uniforme sobre una imagen de mayor tamaño y una secuencia de vídeo con movimiento natural en la que aparece una persona.

PRIMER HITO: GENERACIÓN DE IMÁGENES DE TIPO I

La complejidad que requiere alcanzar el primer hito reside en la elaboración de la estructura principal del código. Por primera vez, aunque sólo se fuesen a generar imágenes de tipo I había que generar un archivo m2v reproducible y, para ello, ya era necesario considerar toda la jerarquía, sintaxis y cabeceras del estándar (*sequence*, *GOP*, *picture*, *slice*, *macroblock*, *block*). Algunas de las cabeceras contienen además

parámetros que podría interesar que los propios alumnos pudieran modificar para observar cómo interfieren en el resultado final.

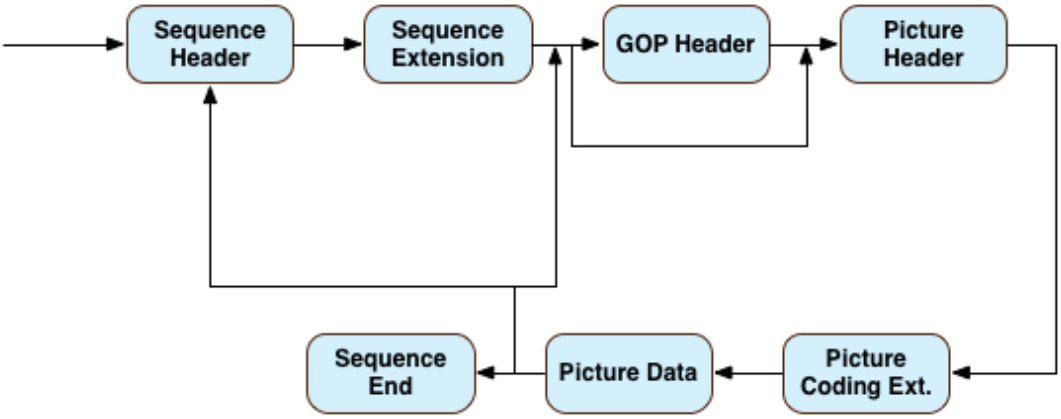

Figura 18 – Sintaxis simplificada de MPEG-2.

Para conocer, y por tanto poder programar, la estructura del flujo de bits se consultaban frecuentemente las fuentes bibliográficas a las que ya se ha hecho referencia. Pero para tener un grado adicional de confianza y revisar si la comprensión era correcta se trabajaba de forma práctica analizando flujos generados por otras herramientas con el software *CodecVisa* y examinándolos además a nivel de bit para comprobar las cabeceras y datos que aparecen en el flujo elemental de vídeo.

Una vez comprobado el progreso y su éxito con las secuencias de vídeo disponibles, el módulo es capaz de flujos elementales de vídeo MPEG-2 con imágenes de tipo I y almacenarlas en disco para visualizarlas con un reproductor multimedia compatible. Con este resultado se comprueba además que la transformación y cuantificación de coeficientes DCT del módulo anterior se estaba haciendo correctamente.

SEGUNDO HITO: GENERACIÓN DE IMÁGENES DE TIPO P

Para alcanzar con éxito el segundo hito es necesario recordar lo que una imagen de tipo P supone: la posible existencia de predicción temporal en los macrobloques de la imagen. Esto significa que en los macrobloques, junto a los coeficientes asociados al residuo de los bloques, aparecerán también los vectores de movimiento.

Los vectores de movimiento se codifican diferencialmente y, ya que MPEG-2 permite una precisión de medio píxel en la predicción temporal, estos aparecerán en el flujo de datos representados por su valor doble, para simplificar la codificación al tratarse así de números enteros. Este hecho limita las opciones de precisión subpíxel añadidas en el primer módulo a un píxel y medio píxel, ya que precisiones más pequeñas no están contempladas en la norma.

Además de implicar la modificación de algunos parámetros en las cabeceras de imagen, la predicción temporal añade una nueva situación que antes no ocurría: la no escritura de bloques o macrobloques en el flujo de datos. Esto es debido a que no siempre es necesario el envío de los coeficientes DCT del residuo de predicción. Si son

nulos o, en general, su magnitud se estima muy baja, el bloque o macrobloque puede reconstruirse sin residuo, simplemente copiando y pegando la información correspondiente de la imagen de referencia.

Se permite así que el codificador pueda omitir cualquiera de los seis bloques que contiene un macrobloque (indicado por la variable *cbp* – *coded_block_pattern* – que describe el estándar), o incluso los seis bloques a la vez, codificando únicamente el vector de movimiento. En la implementación del codificador se añadió un parámetro que regula la cantidad de coeficientes del residuo distintos de cero necesaria para que el bloque aparezca o no en el flujo de datos. Esto permite al estudiante comprobar el valor que tiene el residuo en la compensación de movimiento, incluso si es próximo a cero.

En el caso de que tanto el residuo como el vector de movimiento sean nulos, es decir, que no ha habido una variación de su posición en la imagen, MPEG-2 contempla la posibilidad de que el macrobloque completo no esté presente en el flujo de datos (*skipped macroblock*). Por esta razón es necesario reescribir la implementación de la codificación de macrobloque cuando aparecen imágenes de tipo P en el flujo de datos.

TERCER HITO: GENERACIÓN DE IMÁGENES DE TIPO B

En esta última parte del desarrollo se incluyen imágenes de tipo B que pueden emplear adicionalmente una imagen de referencia futura, lo cual supone un plus sobre el hito anterior. Ahora puede existir predicción sólo hacia delante, sólo hacia detrás o bidireccional. Realmente las implicaciones de esta particularidad (posibilidad de más de un vector de movimiento por macrobloque, distinto orden en la codificación y reproducción de las imágenes) ya se tenían en cuenta desde las primeras versiones del módulo compresor híbrido de vídeo previo. Pero ahora había que introducirlas en la generación del flujo elemental de vídeo comprimido de modo acorde a la norma.

Además de la incorporación potencial de nuevos vectores de movimiento al flujo de datos, la codificación de imágenes de tipo B añade un sentido distinto a los *skipped macroblock*. La norma indica que en imágenes de tipo B un macrobloque que no aparece en el flujo de datos se debe descodificar utilizando los vectores de movimiento y el tipo de predicción del último macrobloque transmitido y sin residuo de predicción.

3.4 – INTERFAZ GRÁFICA

Para facilitar el uso por parte del alumnado, se tomó la iniciativa de presentar la herramienta mediante una interfaz gráfica, con la que poder ajustar de forma sencilla los parámetros que permiten modificar el comportamiento del codificador. Se incorpora además información adicional sobre alguno de estos parámetros para que el usuario pueda conocer cómo puede afectar al resultado la elección de uno u otro valor.

La interfaz gráfica no solo permite generar vídeo comprimido sin tener que realizar las llamadas a las funciones por línea de comandos. También proporciona dos gráficas informativas de los resultados del proceso de compresión una vez que este ha concluido. La primera de ellas muestra la cantidad de bits por píxel necesarios para codificar cada fotograma. Esta gráfica es por tanto representativa de la tasa de compresión alcanzada. La segunda gráfica da la relación señal a ruido por imagen. Se toma como señal la imagen original sin comprimir y como ruido la diferencia entre esta imagen y el resultado de comprimirla y descomprimirla. Esta segunda gráfica da idea por tanto de la calidad del proceso. Se muestra a continuación un ejemplo de la interfaz gráfica tras la codificación de una secuencia de vídeo:

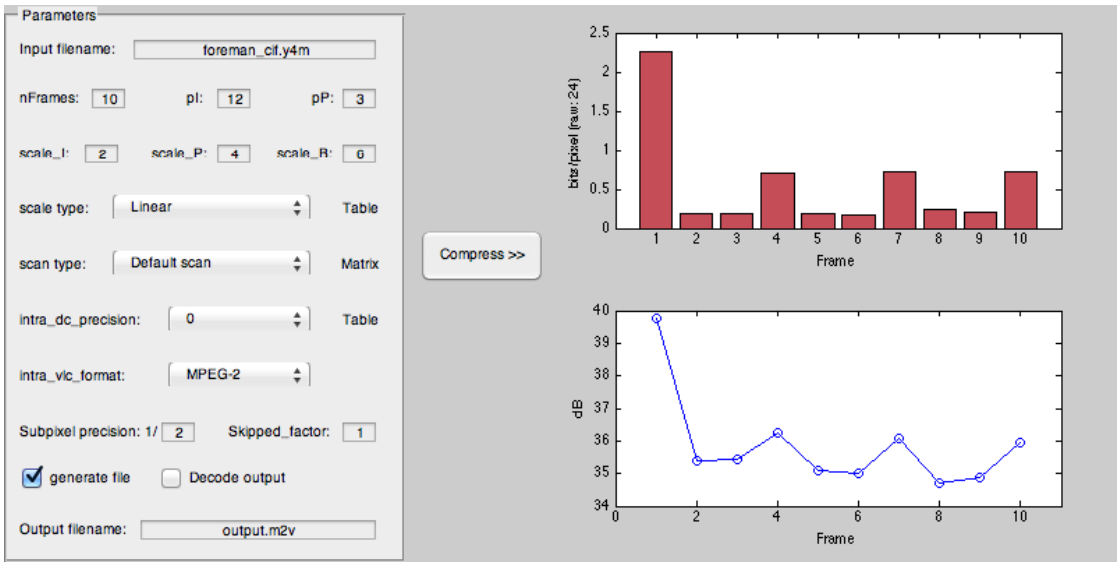

Figura 19 – Interfaz gráfica de la herramienta con información sobre la compresión.

3.5 – VALIDACIÓN DEL DESARROLLO

El flujo elemental de vídeo generado por la herramienta es guardado en disco con la extensión m2v, utilizada por los reproductores multimedia cuando se trata de secuencias de vídeo MPEG-2 sin pistas de audio. Sistemáticamente se iba comprobando que no existían errores al visualizar los archivos generados por la herramienta en reproductores estándar como *VLC*, *QuickTime* o el integrado en *Mac OS X*.

CodecVisa, que ya se ha mencionado en el apartado 3.3, se ha utilizado con la doble finalidad de aprender sobre la composición del flujo binario de MPEG-2 y la de validar el desarrollo. Un software de este tipo como referente externo es muy útil para comprobar que los bits escritos contienen realmente la información que se quería introducir en ellos. Esta herramienta detalla todos los parámetros descodificados del flujo de datos y los representa de una forma más visual. Este es el caso de los vectores de movimiento en la imagen que se muestra a continuación en la que a *CodecVisa* se le pasa uno de los archivos generados en este proyecto:

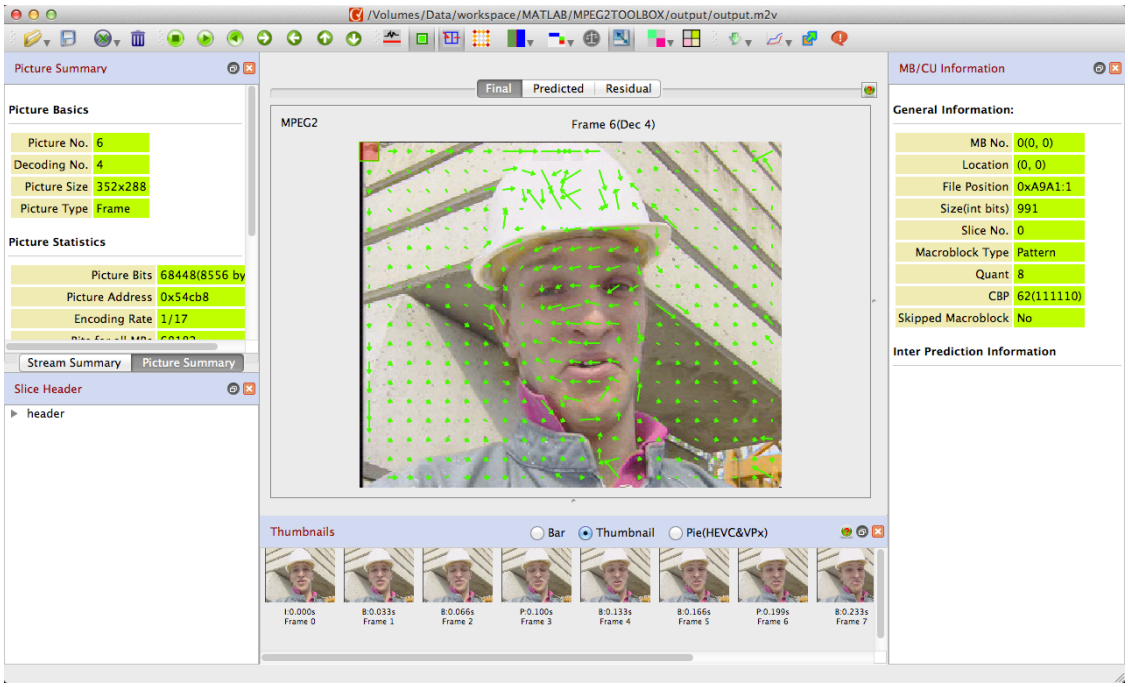

Figura 20 – Visualización en *CodecVisa* de un archivo de vídeo generado por la herramienta.

También se han usado en la validación programas ejecutables sin interfaz gráfica que operan en línea de comandos, pero que realizan la codificación o descodificación de MPEG-2. Este ha sido el caso de la biblioteca genérica de conversión multimedia *ffmpeg* [19]. Pero, en esta línea, la herramienta de validación más útil ha sido un software desarrollado por *MPEG Software Simulation Group* (MSSG) de acuerdo a las recomendaciones en *Test Model 5* [20]. Esta herramienta es popular como referente de facto para la codificación y descodificación de MPEG-2. Se ha hecho uso especial del ejecutable del descodificador *mpeg2decode.exe*. El objetivo era comprobar si, al descomprimir con este ejecutable un archivo m2v generado por la herramienta, el resultado de la descodificación era idéntico al que proporcionaba internamente el compresor híbrido. Esto corroboraría que, en nuestro código, la parte de descompresión está siguiendo la norma. Este tipo de test permitieron realizar ajustes finos sobre nuestro software.

Existen también otros test de validación o compatibilidad oficiales desarrollados en [21], los cuales exploran todas las opciones que contempla el estándar de una forma más exhaustiva. Estos no se han considerado necesarios ya que en el desarrollo de este proyecto no se han incluido otros detalles definidos también en el estándar, que se alejan del valor didáctico que se pretende buscar a la herramienta.

3.6 – RESULTADOS OBTENIDOS CON LA HERRAMIENTA

Este proyecto no estaba enfocado a producir resultados experimentales, sino a la creación de una herramienta que fuese didáctica y cumpliese el estándar MPEG-2 vídeo. En este apartado se describen, a modo de ejemplo, algunos resultados que el alumno podría obtener o constatar trabajando sobre la herramienta diseñada, como la conveniencia de explotar la predicción temporal o un análisis de costes en Matlab para

saber qué partes del proceso de compresión son más exigentes. También se ha desarrollado un guión práctico a desarrollar por el alumno para indagar en el proceso de compresión de vídeo, el cual se encuentra disponible en el Anexo IV.

Para analizar la capacidad de compresión y su relación con el hecho de explotar mecanismos de predicción temporal se toman como referencia los diez primeros 10 fotogramas de tres secuencias de vídeo de 352x288 píxeles. Los bits necesarios para representar cada secuencia de vídeo sin comprimir son los siguientes:

$$288\frac{líneas}{fotograma}\cdot 352\frac{píxeles}{línea}\cdot 3\frac{colores}{píxel}\cdot 8\frac{bits}{color}*10 fotogramas = 24330240\ bits = \ 3.04\ MB$$

Si utilizamos la herramienta desarrollada para comprimir tres secuencias de las mismas características en dos situaciones diferentes, una sin emplear predicción temporal y otra con compensación de movimiento, y le aplicamos los parámetros que aparecen en la Figura 19, obtenemos flujos elementales de vídeo de los siguientes tamaños:

Secuencia	**Sin predicción temp.**	**Con predicción temp.**
foreman	278.92 KB (9.39%)	69.17 KB (2.33%)
crew	218.56 KB (7.35%)	77.50 KB (2.61%)
football	437.60 KB (14.73%)	171.55 KB (5.78%)

Tabla 2 – Resultados de la compresión de tres secuencias de vídeo obtenidas de [11].

Esto nos demuestra una compresión de la secuencia de vídeo original de al menos el 85.27%, que puede llegar a alcanzar el 96.43% en valor medio si se emplea predicción temporal y compensación de movimiento.

Una forma de medir la eficiencia del código es mediante la función *profile* de Matlab. Sobre una ejecución de la herramienta, esta función muestra el tiempo requerido en cada una de las partes del código. En la Figura 21 se muestra el resultado para la compresión de una secuencia de vídeo empleando predicción temporal.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
interfaz>GenerateCall	1	74.437 s	0.027 s	
video2ES	1	74.383 s	0.810 s	
mpeg2ES	1	37.719 s	0.984 s	
video2mpeg	1	34.696 s	2.249 s	
PutMacroBlock	3960	33.102 s	1.648 s	
getmeas4tempmatch	15	28.230 s	27.957 s	
code_MB_NI	7312	16.629 s	4.307 s	
dec2bin	145179	13.687 s	13.687 s	
code_MBY	396	7.222 s	2.577 s	
code_MV	5097	4.371 s	3.206 s	

Figura 21 – Resumen de los tiempos de ejecución de la herramienta.

Se puede apreciar cómo la función *mpeg2ES*, que implementa el codificador binario, necesita un tiempo de ejecución mayor que el de la parte correspondiente a la compresión de la señal MPEG-2 (*vídeo2mpeg*). Aunque no sea una gran diferencia, sí que puede indicarnos la idea de que Matlab se encuentra optimizado para operaciones matemáticas de procesado de la señal, y no para operaciones de otra índole como puede ser la conversión de una secuencia a binario, realizada en la función *dec2bin*.

Aunque se pueda pensar que la realización de la transformada DCT sobre cada bloque suponga una gran carga al proceso, se comprueba cómo el mayor tiempo de procesado lo requiere el cálculo de vectores de movimiento: 28,2 segundos (*getmeas4tempmatch*) sobre los 34,7 segundos que dura el proceso de compresión (*vídeo2mpeg*). Se puede concluir con esto que la técnica empleada en el cálculo de vectores de movimiento influye notablemente en el tiempo de ejecución. Como se adelantó, el uso de otras técnicas de búsqueda que sustituyan a la implementada podrían hacer ganar velocidad de cómputo a la herramienta a cambio de reducir su precesión.

4 - CONCLUSIONES

4.1 – RESUMEN DEL PROYECTO Y ANÁLISIS DE LOS OBJETIVOS

El propósito de este proyecto ha sido el desarrollo de una herramienta para el estudio de la compresión de vídeo digital, diseñada especialmente con fines didácticos y para el entorno académico. En la elaboración de la misma se han realizado diferentes tareas para cumplir con los objetivos marcados al comienzo del proyecto:

- Se ha realizado un análisis del estado del arte de la compresión de vídeo mediante codificadores híbridos, en especial sobre aquellas técnicas y métodos empleados por el estándar MPEG-2 implementado en la herramienta.

- El código fuente desarrollado se encuentra accesible para su uso académico. Se ha procurado seguir una estructura de lectura sencilla, elevando la complejidad solo en funciones internas con cometidos específicos. De esta forma el alumno podrá seguir de manera directa el proceso de compresión de vídeo que realiza la herramienta.

- Se ha intentado aprovechar las características de Matlab para aquello para lo que es eficiente. Por ejemplo, todos los coeficientes DCT o todos los vectores de movimiento de una misma imagen se obtienen simultáneamente mediante el uso de indexación para maximizar el rendimiento de la herramienta.

- Para minimizar la dificultad de un primer uso de la herramienta se ha creado también una interfaz gráfica. Así el usuario puede introducir todos los parámetros de configuración necesarios desde un mismo punto y obtener los resultados. Además se ha preparado una fuente de recursos de secuencias en formato y4m y el código necesario para su lectura, para poder ser utilizados en la herramienta.

- Se ha incluido también un enunciado de una posible práctica, que indicará al alumno los pasos a seguir para entender el funcionamiento de la herramienta y del proceso de compresión de vídeo implementado (Anexo IV).

- Se ha incorporado un análisis de los resultados del proceso, presentado en la interfaz una vez haya terminado la ejecución de la herramienta. Dos gráficas ubicadas en esta ventana muestran información al usuario sobre la cantidad y la calidad de la compresión de vídeo realizada.

- Además de dicho análisis el alumno puede utilizar el flujo elemental de vídeo comprimido generado, el cual se guarda en disco y es reconocido por programas de reproducción multimedia de terceros.

En ningún momento se ha querido competir con las herramientas disponibles en el mercado en términos de rendimiento, sino ofrecer como ventaja la eliminación del gasto en licencias y el enfoque al entorno académico. Esto ha llevado a limitar las

compatibilidades del desarrollo en algunos ámbitos, como ha podido ser el vídeo entrelazado o la gestión de la tasa de bits generada. En cambio, tras el desarrollo, se permite al alumno conocer toda la información que genera el proceso de compresión de vídeo, las técnicas y métodos utilizadas internamente y un análisis del proceso junto con un archivo de vídeo como resultado final de la cadena.

Gracias a este desarrollo se ha puesto en valor la compresión de vídeo, se ha facilitado la usabilidad de la herramienta por parte del alumnado, se han verificado los resultados generados con ella y además se ha analizado su eficiencia tanto en términos de compresión como dentro del entorno que proporciona Matlab.

4.2 – DIFICULTADES ENCONTRADAS

Aunque el estándar MPEG-2 vídeo ofrece todos los detalles por los que cualquier descodificador de vídeo acorde a la norma debe regirse, no se trata de un documento de lectura sencilla en el que uno pueda sumergirse sin precauciones. Aun teniendo como referencia siempre el estándar, se ha tenido que buscar información externa o aclaraciones adicionales para una mejor comprensión del mismo. Esto se ha acentuado aún más en partes del estándar relacionadas con la codificación directa, ya que en el estándar únicamente describe la descodificación. También ha ocurrido especialmente en el tratamiento de la predicción temporal, un terreno que el estándar deja más abierto para fomentar la competencia entre las herramientas del mercado, y en la cuantificación directa de los coeficientes DCT.

Por otro lado, el desarrollo estructurado de la herramienta junto con una implementación modular han facilitado la búsqueda y resolución de los errores que han aparecido a lo largo del proyecto. La interfaz gráfica era también un campo desconocido y, tras saber de las diferentes formas de generarla, se tomó la decisión de implementarla vía código para tener más control sobre la misma en futuras modificaciones.

4.3 – LÍNEAS FUTURAS DE TRABAJO

El desarrollo de la herramienta en la que se centra este proyecto cumple todos los objetivos y requisitos establecidos al inicio del mismo, pero no por ello se trata de una herramienta cerrada a incluir nuevos módulos o desarrollos adicionales, manteniendo siempre el enfoque académico que se pretende tener desde el primer momento. Como han ido apareciendo a lo largo de este documento, existen huecos en los que un desarrollo adicional podría aportar al alumno análisis de otros métodos o técnicas usados en la compresión de vídeo, o incluso nuevos codificadores de flujos elementales de vídeo asociado a otros estándares.

Uno de estos desarrollos podría ser el que contemplara la compatibilidad con el vídeo entrelazado, y explorar así de qué forma afecta a la compresión de las secuencias de vídeo. Esto se podría realizar dentro de la norma MPEG-2 o implementarlo en otros módulos codificadores correspondientes a estándares más modernos.

También podría contemplarse la posibilidad de añadir a la herramienta un descodificador de un flujo elemental de vídeo MPEG-2. Serviría así para analizar flujos elementales de vídeo MPEG-2 y comprobar sus características, pero su complejidad sería elevada ya que requeriría ser compatible con todas las peculiaridades reflejadas en el estándar.

Cabría incluso la posibilidad de que el uso de la herramienta se reorientara a investigación en procesado de vídeo para compresión, de forma que el potencial investigador usuario añadiera nuevos módulos para incluir aquellos métodos objeto de su estudio, como otras transformadas o nuevas técnicas de estimación de movimiento, entre otros.

BIBLIOGRAFÍA

[1] ISO/IEC 13818-2: Information technology -- Generic coding of moving pictures and associated audio information -- Part 2: Vídeo. 1995

[2] B.G. Haskell, A. Puri, A.N. Netravali. Digital Vídeo: An Introduction to MPEG-2. Editorial Kluwer Academic Publishers. 2002

[3] Alfonso Martín Marcos. Televisión Digital (Vol. I): Compresión MPEG. Editorial Ciencia 3. 2006

[4] Matías Garrido González. Arquitectura versátil para la codificación de vídeo multi-estándar: Aportaciones metodológicas para el diseño de sistemas reutilizables y sistemas en un chip. Tesis Doctoral UPM-ETSIT. 2004

[5] Cisco Systems. Global Mobile Data Traffic Forecast Update 2015–2020. 2016 http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html

[6] Telestream. 2014 http://www.avsforum.com/forum/286-latest-industry-news/1528750-comparing-mpeg-2-h-264-h-265-vídeo-codecs-nab-2014-a.html

[7] ITU-R BT.601-7: Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios. 2011

[8] ITU-R BT.709-6: Parameter values for the HDTV standards for production and international programme Exchange. 2015

[9] N. Roma, L. Sousa. A tutorial overview on the properties of the discrete cosine transform for encoded image and vídeo processing. Signal processing (Ed. Elsevier). Vol: 91, número: 11, páginas: 2443-2464. 2011

[10] Explicación sobre el formato Y4M: https://wiki.multimedia.cx/index.php?title=YUV4MPEG2

[11] Fuentes de vídeo en formato Y4M para docencia e investigación: https://media.xiph.org/vídeo/derf/

[12] Q. Tang, P. Nasiopoulos, R.K. Ward. Compensation of Requantization and Interpolation Errors in MPEG-2 to H.264 Transcoding. IEEE Trans. on Circuits and Systems for Vídeo Technology. Vol: 18, número: 3, páginas: 314-325. 2008

[13] K.R. Rao y J.J. Hwang. Techniques and Standards for Image, Vídeo, and Audio Coding. Editorial Prentice Hall. 1996

[14] M. Störring, T. Mocslund. An Introduction to Template Matching. Computer Vision and Media Technology Laboratory (CVMT), Aalborg University, Denmark., Tech. Rep. 01-04. 2001

[15] L.M. Lopes-Teixeira, y A.P. Alves. Block Matching Algorithms in MPEG Vídeo Coding. Proceedings of the 13th International Conference on Pattern Recognition, Volumen 3, páginas 934-938. 1996

[16] H.-M. Hang, Y.-M. Chou, S.-Ch. Cheng. Motion Estimation for Vídeo Coding Standards. Journal of VLSI Signal Processing. Vol: 17, páginas: 113-136. 1997

[17] Elena Aguilar Fernández. Decodificador de vídeo MPEG-2 en Matlab y análisis del bitstream. Proyecto Fin de Carrera de titulación Ingeniería Telecomunicación de la Universidad de Sevilla. 2008

[18] Ravi Lakkundi. MPEG2 Vídeo Encoder. 2010. http://www.mathworks.com/matlabcentral/fileexchange/27051-mpeg2-vídeo-encoder

[19] FFMPEG. https://ffmpeg.org/

[20] Test Model 5. http://www.mpeg.org/MPEG/vídeo/mssg-free-mpeg-software.html

[21] ISO/IEC 13818-4: Information technology -- Generic coding of moving pictures and associated audio information -- Part 2: Conformance testing. 2004.