PROYECTO FIN DE GRADO

TÍTULO: Creación de una aplicación MATLAB para la representación interactiva de los datos obtenidos durante el proceso de codificación de video.

AUTOR/A: Almudena Estévez Sanz

TITULACIÓN: Grado en Ingeniería de Sonido e Imagen

TUTOR/A: Enrique Rendón Angulo

DEPARTAMENTO: Ingeniería Audiovisual y Comunicaciones

VºBº TUTOR/A

Miembros del Tribunal Calificador:

PRESIDENTE/A: Francisco Javier Jiménez Martínez

TUTOR/A: Enrique Rendón Angulo

SECRETARIO/A: Martina Eckert

Fecha de lectura: 17 de Julio del 2024

Calificación:

El Secretario/La Secretaria,

Agradecimientos

A todas aquellas personas que contribuyeron al desarrollo y éxito de este Proyecto de Fin de Grado.

A mi tutor Enrique por su gran ayuda, orientación y sugerencias a lo largo de todo el proceso. Sin su conocimiento y experiencia no habría sido posible alcanzar todos los objetivos planteados en este proyecto.

A mi familia, por su apoyo, su paciencia y su comprensión incondicional durante todo este tiempo.

A mis compañeros de clase y amigos que me dieron apoyo moral y motivación en los momentos difíciles, compartiendo conmigo este viaje académico.

A la institución educativa por brindarme la oportunidad de llevar a cabo este proyecto y por proporcionar los recursos necesarios para su realización.

Resumen

El objetivo de este proyecto es crear una solución tecnológica para reforzar la comprensión de un tema complejo como es la codificación de video. Con este propósito, se desarrolla una aplicación de escritorio para que los estudiantes puedan fortalecer el conocimiento adquirido en las clases teóricas.

VisualCodecInsight (VCI) es una aplicación educativa desarrollada mediante MATLAB App Designer que permite a los estudiantes visualizar y comprender los procesos fundamentales de la codificación de imágenes. Dicha herramienta facilita el aprendizaje de forma interactiva mostrando las imágenes calculadas en cada parte del proceso de codificación.

En esta aplicación, los usuarios pueden cargar diferentes secuencias y elegir los fotogramas que quieran visualizar, a partir de los cuales pueden: visualizar sus modos de macrobloques, visualizar únicamente el aporte de las cromas, superponer los diferentes tipos de vectores de movimiento, reproducir secuencias de forma automática e interpretar todos sus datos mediante la gestión de tablas y gráficos.

Esta aplicación busca crear un impacto significativo en la educación técnica, sobre todo fomentando el dinamismo y el desarrollo de habilidades tecnológicas cruciales para el futuro de los ingenieros. Esta aplicación puede significar un avance en la manera en que se enseñan y aprenden las tecnologías de codificación de imágenes, puesto que aporta un apoyo visual para aclarar conceptos.

Abstract

The aim of this project is to create a technological solution to reinforce the understanding of a complex subject such as video coding. For this purpose, a desktop application is developed so that students can strengthen the knowledge acquired in the theoretical classes.

VisualCodecInsight (VCI) is an educational application developed with MATLAB App Designer that allows students to visualize and understand the fundamental process in image coding. The tool provides an interactive learning experience by showing computed images at each part of the coding process.

In this application, users can load different sequences and choose the frames they want to visualize, where they can: visualize their macro-block modes, visualize the contribution of the chroma components only, overlay the different types of movement vectors, automatically play sequences and interpret all their data by managing tables and graphs.

This application seeks to encourage dynamism and the development of technological skills that are crucial for the future of engineers. This project can be a breakthrough in the way image coding technologies are taught and learned, as it provides visual support to clarify concepts.

Lista de acrónimos

A continuación, se detalla la lista de acrónimos utilizados a lo largo de esta memoria:

BD – Bidireccional: tipo de predicción en vídeo que utiliza información de cuadros anteriores y posteriores en el tiempo al actual, se utiliza para calificar los modos de macrobloque que utilizan de una referencia pasada y otra futura.

BW – Backward: tipo de predicción en vídeo que utiliza información de cuadros posteriores en el tiempo al actual, se utiliza para calificar los vectores de movimiento que se refieren a una referencia futura y al modo de macrobloque que usa la referencia futura exclusivamente.

CÓDEC – Codificador-decodificador: es un codificador de vídeo predictivo transformacional que basa sus predicciones en las imágenes previamente decodificadas.

Frame – Cuadro de video o fotograma: término inglés para referirse a una imagen concreta dentro de una sucesión de imágenes en movimiento.

FW – Forward: tipo de predicción en vídeo que utiliza información de cuadros anteriores en el tiempo al actual, se utiliza para calificar los vectores de movimiento que se refieren a una referencia pasada y al modo de macrobloque que emplea la referencia pasada exclusivamente con vector de movimiento no nulo.

MBm – Modos de Macrobloque: diferentes técnicas de predicción aplicadas a un macrobloque en la codificación de vídeo

RGB – Red, Green and Blue: espacio de color que reproduce los colores visibles para los humanos mediante la mezcla aditiva de los tres colores primarios: rojo, verde y azul

Tipo B – Imagen Tipo Bidireccional: imágenes codificadas a partir de referencias tanto futuras como pasadas.

Tipo I – Imagen Tipo Intra: imágenes que no utilizan ninguna imagen como referencia.

Tipo P – Imagen Tipo Predictiva: imágenes obtenidas a partir de referencias pasadas.

VCI – VisualCodecInsight: Nombre escogido para la aplicación desarrollada.

VM – Vectores de Movimiento: vectores que indican el desplazamiento de bloques de píxeles entre cuadros próximos en una secuencia de vídeo.

YCbCr – Luma, diferencia de crominancia azul y diferencia de crominancia roja: señales usadas en sistemas de vídeo.

Índice de contenidos

Índice de Figuras

Índice de Tablas

1. Introducción

1.1 Marco y motivación del proyecto

Actualmente en la sociedad en la que vivimos se tiene bastante integrado el intercambio de contenidos audiovisuales, por lo tanto, debido a la gran cantidad de datos que se intercambian y almacenan a la hora de la reproducción de cualquier vídeo, es un factor fundamental que los ingenieros de imagen y sonido comprendan en su totalidad el proceso de codificación y decodificación de video durante su formación. Por esta razón, este proyecto se enfoca directamente en la creación de una aplicación de escritorio educativa que permita a los usuarios visualizar diferentes tipos de imágenes e información generada durante el proceso de codificación.

La codificación de video se refiere al proceso de compresión de datos de video para su posterior transmisión o almacenamiento de una forma mucho más eficiente. En la codificación se pretende reducir/comprimir el tamaño de los archivos a enviar considerando la relevancia perceptual para controlar la calidad de la imagen. Esta reducción posibilita la transmisión de información relevante de una forma más rápida y efectiva, facilitando así la difusión de contenido de video digital. Sin este proceso, resultaría imposible transmitir la gran cantidad de información necesaria para hacer llegar los contenidos audiovisuales a nuestros televisores mediante satélite, cable o Internet, así como en las plataformas como YouTube, Twitch o HBO, y en discos Blu-ray y DVD.

La codificación video está compuesta por tres grandes etapas: muestreo, cuantificación y codificación. En la primera etapa, la señal de video analógica o digital es tomada y convertida en una secuencia de muestras espaciales y temporales durante el proceso de muestreo. Posteriormente, en la etapa de cuantificación, los valores muestreados son clasificados en intervalos y discretizados de acuerdo con el intervalo de valores al que correspondan. Finalmente, en la codificación se comprimen los datos obtenidos en las etapas anteriores y se reduce el tamaño del archivo de video disminuyendo su calidad y eliminando las redundancias e información menos relevante perceptualmente. Una vez se dispone de la información codificada, se envía a los sistemas receptores donde los datos codificados son descomprimidos para convertirse en la secuencia de video del archivo original y reproducirse en la pantalla del dispositivo en uso [1].

Existen varios tipos de codificación de fuente como MPEG-2, MPEG-4, H.264, H.265, VP9, entre otros [1][2]. Todos los estándares tienen como base un concepto común de esquemas predictivos transformacionales; no obstante, la elección de un estándar de codificación de video se realizará según las necesidades particulares de cada proyecto, considerando factores como la calidad requerida, las tasas de bits disponibles y la compatibilidad con los dispositivos receptores.

1.2 Objetivos técnicos y académicos

El proyecto actual surge de la necesidad de realizar una aplicación capaz de ayudar a un usuario final. En este caso, se aplica al ámbito académico donde se decide diseñar una nueva herramienta capaz de ayudar al alumnado a entender el proceso de la codificación de video de una forma visual e interactiva. Una vez obtenida la propuesta e idea general, su desarrollo evoluciona acorde con una serie de objetivos y requisitos definidos. Muchos de esos objetivos nacen de ideas generales que con el paso del tiempo se particularizan y definen de forma adecuada para conducir a la obtención del producto final.

Desde el punto de vista técnico, se diferencian y recogen una serie de objetivos, ordenados a continuación por orden de prioridad:

- Selección del entorno de trabajo.
- Tipos de imágenes a mostrar.
- Datos e información a representar.
- Organización de la herramienta para facilitar la visualización.
- Codificación y decodificación de imágenes

Desde el punto de vista académico y en relación con el Grado de Sonido e Imagen donde se engloba este proyecto, se persiguen la adquisición de competencias relacionadas con el ámbito en el que se encuentra y más específicamente con las recogidas en la Guía Oficial de la asignatura de Tecnologías de Imagen y Vídeo [3].

A continuación, se enumeran brevemente una serie de competencias que aplican en este proyecto: CE SO01, CE SO05, CE TEL01, CG 11 y CG 13.

- CE SO01 – Capacidad de construir, explotar y gestionar servicios y aplicaciones de telecomunicaciones, entendidas éstas como sistemas de captación, tratamiento analógico y digital, codificación, transporte, representación, procesado, almacenamiento, reproducción, gestión y presentación de servicios audiovisuales e información multimedia.
- CE SO05 – Capacidad para crear, codificar, gestionar, difundir y distribuir contenidos multimedia, atendiendo a criterios de usabilidad y accesibilidad de los servicios audiovisuales, de difusión e interactivos.
- CE TEL01 – Capacidad para aprender de manera autónoma nuevos conocimientos y técnicas adecuados para la concepción, el desarrollo o la explotación de sistemas y servicios de telecomunicación.
- CG 11 – Habilidades para la utilización de las Tecnologías de la Información y las Comunicaciones.
- CG 13 – Habilidades de aprendizaje con un alto grado de autonomía.

1.3 Estructura de la memoria

Esta memoria se estructura de manera general acorde con la Figura 1. En los primeros apartados se realiza una introducción sobre el tema del proyecto, posteriormente se trata el marco tecnológico, que se divide en dos partes, el proceso de digitalización y el de codificación.

Figura 1. Estructura básica de la memoria.

El proceso de digitalización se comenta brevemente puesto que es importante conocer cómo se realiza la conversión de las señales analógicas a digitales, sin embargo, el proceso de codificación sí que se explica con más detalle debido a que es el grueso de donde deriva la herramienta creada y es necesario entender bien los elementos, operaciones y pasos implicados a lo largo de dicho proceso.

Una vez aclarados los aspectos teóricos, se procede al desarrollo de las especificaciones y restricciones del diseño, donde se enumeración y explican los objetivos planteados en las primeras etapas del proyecto y su grado de cumplimiento. Tras explicar los requisitos, se describe en detalle la solución obtenida junto con los resultados conseguidos tras la realización de una serie de pruebas, para así entrar en las últimas fases donde se muestra el presupuesto, se exponen las conclusiones, se comenta el impacto del proyecto en el ambiente académico y finalmente se explican las posibles líneas de trabajo futuras concluyendo con los manuales de usuario y del desarrollador.

2. Marco tecnológico

Para entender correctamente el proceso de transmisión de vídeo y poder establecer la procedencia de los diferentes tipos de imágenes y datos a representar en la herramienta a desarrollar con finalidad educativa, es necesario visualizar una idea clara del proceso que sufren las imágenes desde que son tomadas, hasta que son recibidas en los receptores.
Primero se realiza la captación de imágenes, a través de las cámaras, para luego llevar a cabo el proceso de digitalización, donde se convierte analógico a digital, empleando el muestreo y la cuantificación. Posteriormente, una vez se dispone de la información digital se procede a su codificación de fuente, lo cual permite que esos datos se compriman para reducir su tamaño a la hora de enviarlos hacia el receptor.
En el receptor se decodifican los datos y se someten al proceso inverso de digitalización. Es esta última etapa una vez se disponen de los datos decodificados se reconstruyen e interpolan para poder recomponer de nuevo la imagen.

2.1 Obtención de imágenes

La captación de imágenes o video se consigue mediante las cámaras, en la Figura 2 se muestran las partes que de una cámara réflex, donde se identifican los elementos involucrados en el proceso de obtención de una imagen digital.

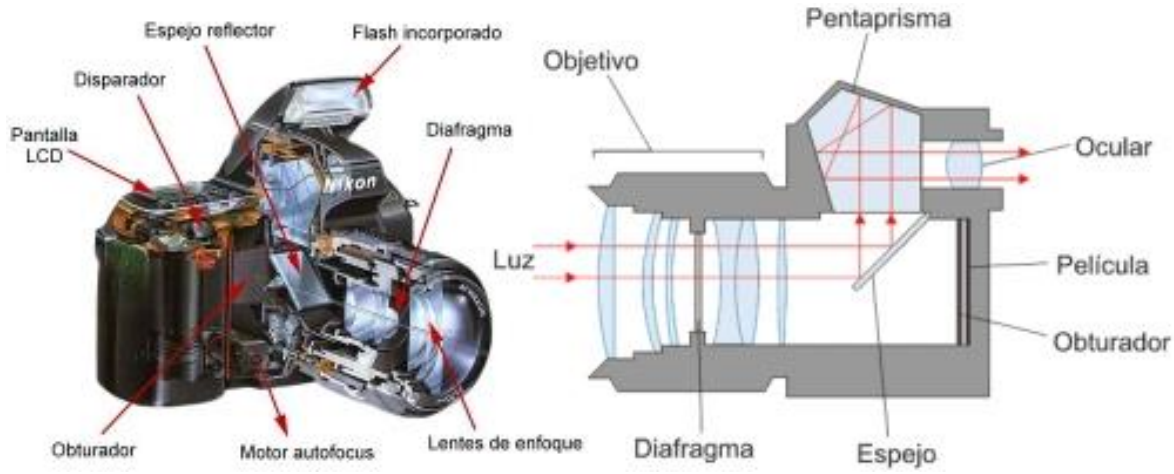

Figura 2. Partes de una cámara y recorrido de la luz en su interior [4].

La luz de cualquier escena a capturar viaja a través del diafragma, donde se modifica y gradúa su caudal mediante la apertura del diafragma. Una vez se dispone de la toma bien enfocada sobre el sensor, gracias a un sistema de lentes, y se presiona el disparador, se activa el obturador que también controla el paso de la cantidad de luz según el tiempo de obturación establecido. La luz que atraviesa incide sobre un sensor electrónico capaz de transformar la luz en tensión eléctrica. Cuando la luz llega al sensor, una malla de píxeles muestrea la imagen para obtener valores discretos de amplitud de la señal eléctrica continua. Al cuantificarse esos valores, se obtienen valores discretos en amplitud los cuales se codifican para ser enviados a los diferentes receptores.

Al recibir toda esa información en los equipos receptores, estos deben ser capaces de aplicar una decodificación y una reconstrucción inversa al proceso de cuantificación y muestreo para poder recuperar la imagen de nuevo. Estos pasos se representan en la Figura 3, donde coloreados de azul se muestran las etapas que tienen lugar en el emisor, mientras que las rojas se llevan a cabo en el receptor.

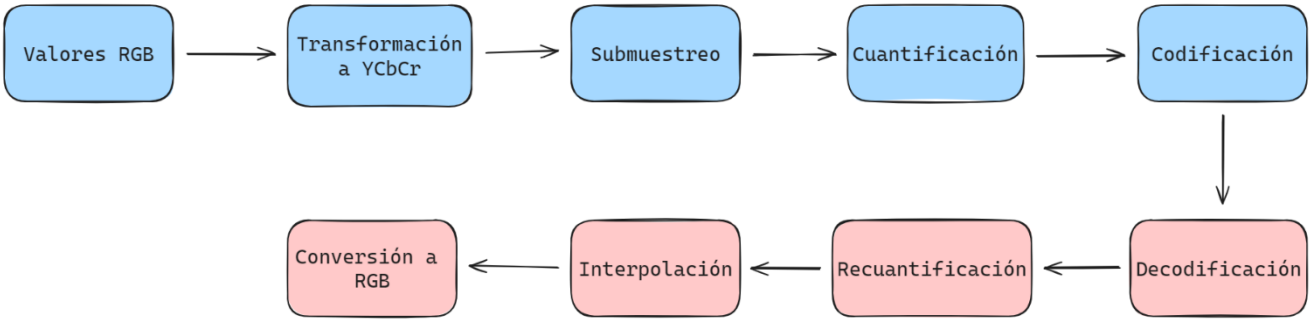

Figura 3. Etapas operacionales del proceso de codificación de video.

2.2 Digitalización

Para conseguir una señal eléctrica, se debe aplicar a la señal analógica de luz un proceso de muestreo y cuantificación, donde se discretiza la señal tanto espacialmente como en amplitud. En la Figura 4, cuando la luz llega al sensor se obtiene información eléctrica para cada píxel, la cual se descompone en las componentes de color RGB (red, green y blue). Cuando se dispone de la señal eléctrica RGB se convierte al espacio de color YCbCr, ya que es el espacio de color utilizado en la compresión de señales. Este espacio de color tiene su origen en el periodo de transición entre las señales en blanco y negro y las señales a color, por este motivo, es compatible con ambos sistemas.

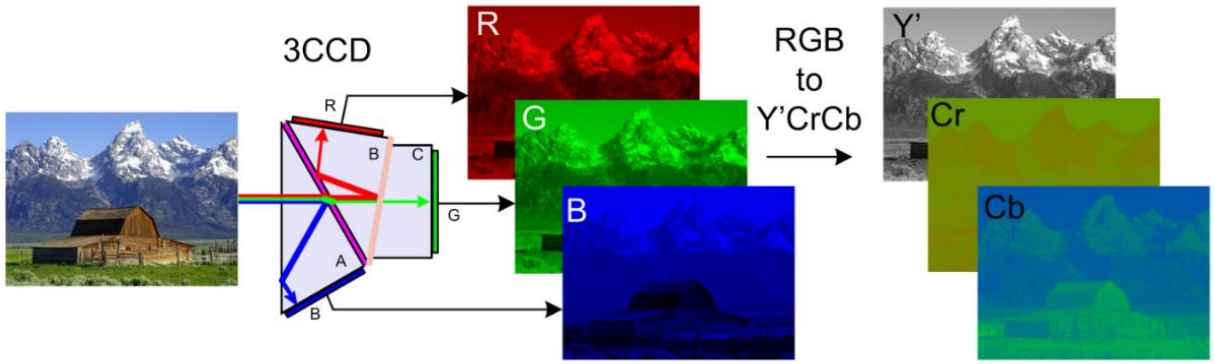

Figura 4. Proceso de obtención de señales YCbCr [5].

La componente Y representa la información de brillo o luminancia en el video. Es similar a una versión en escala de grises de la imagen original. Lleva la información de intensidad de la imagen y se utiliza para representar la imagen en blanco y negro.

Las cromas (Cb y Cr) son las componentes que representan la información del color en el vídeo. Cb es la diferencia en color entre el azul y la luminancia, mientras que Cr se refiere a la diferencia en color entre el rojo y la luminancia.

La separación de la información de luminancia y croma en el espacio de color YCbCr aparte de permitir una compatibilidad entre sistemas también ofrece una mayor eficiencia en la compresión de video, puesto que la señal de luminancia se puede almacenar o transmitir con alta resolución o ancho de banda, mientras que las diferencias cromáticas se pueden reducir en ancho de banda, submuestrear, comprimir o tratar de manera separada para mejorar la eficiencia del sistema. Esto es debido a que el sistema visual humano (SVH) es más sensible a las variaciones espaciales de la luminancia que a las cromas o diferencias de color azul y rojo, lo cual permite reducir la información a transmitir.

2.2.1 Muestreo

El muestreo es el proceso donde se transforma una señal continua a una señal discreta. Partiendo de una matriz bidimensional de píxeles, se toman diferentes valores de amplitud respecto al tiempo a partir de la señal continua recibida. El tiempo empleado en tomar las muestras se conoce como frecuencia de muestreo y debe cumplir el teorema de muestreo de Nyquist-Shannon (*1*) que establece que una señal continua de ancho de banda limitado puede ser completamente reconstruida a partir de sus muestras discretas si la frecuencia de muestreo utilizada es al menos el doble de la frecuencia máxima presente en la señal.

$$f_{muestreo} \geq 2 \cdot f_{max} \qquad (1)$$

La frecuencia de muestreo debe elegirse adecuadamente para no producir efecto Aliasing, provocando que diferentes señales se conviertan en indistinguibles tras la digitalización. Esto se manifiesta visualmente de dos formas principales, mostradas en la Figura 5.

- Efecto Moiré: producido al capturar o mostrar imágenes con texturas finas y repetitivas que interfieren con la matriz de píxeles del sensor de la cámara o del monitor, generando un efecto óptico de patrones ondulantes o repetitivos de baja frecuencia que no existen en la imagen original
- Pixelado de Bordes: producido cuando no hay suficientes píxeles para representar suavemente las transiciones de color y forma en la imagen, lo cual genera bordes y líneas finas pueden aparecer dentados o "escalonados".

Figura 5. Efecto Moiré (Izq.) [6] y Pixelado de bordes (Dcha.) [7].

Del proceso de muestreo realizado sobre imágenes a color se obtiene una asociación de valores para cada uno de los píxeles en los que se divide la imagen. Cada píxel a su vez está compuesto por información de luminancia (Y) y de crominancia (Cb y Cr) ya que es el espacio de color empleado en la codificación de imagen.

2.2.1.1 Submuestreo

Realizar un submuestreo consiste en comprimir la información de color de una señal para dar prioridad a la luminancia, de esta forma se aplica un submuestreo mayor en las cromas reduciendo la información a transmitir y disminuyendo la cantidad de muestras tomadas. Esto repercute en la calidad de la imagen puesto que conlleva pérdidas de información. Existen diferentes formatos de submuestreo que comparten la estructura A:B:C, donde A hace referencia al muestreo horizontal, B se refiere al submuestreo relativo de A para las cromas y C puede referirse al submuestreo igual que B, pero aplicada sobre la otra croma o aparecer con un valor 0 para indicar la existencia de submuestreo vertical.

En el proceso de codificación, la imagen se divide tal y como se muestra en la Figura 6. Se divide en macrobloques (MBs) de un tamaño de píxeles definido (16x16 píxeles en este caso) y posteriormente cada MB es dividido a su vez en bloques de 8x8 valores.

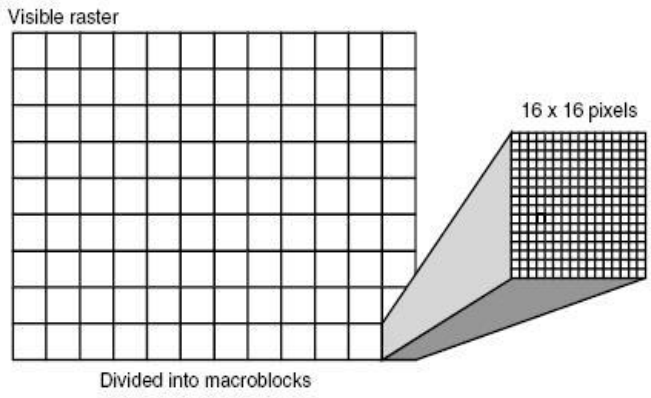

Figura 6. División en macrobloques de una imagen [8].

En la Figura 7 se muestra como ejemplo el submuestreo 4:2:0, donde por cada 4 muestras de Y se obtiene 1 de cada croma. Es el formato de muestreo más común en video y como resultado para cada macrobloque (16x16) se corresponden un total de 4 bloques de Y, 1 bloque de Cb y otro bloque de Cr, todos de 8x8 valores.

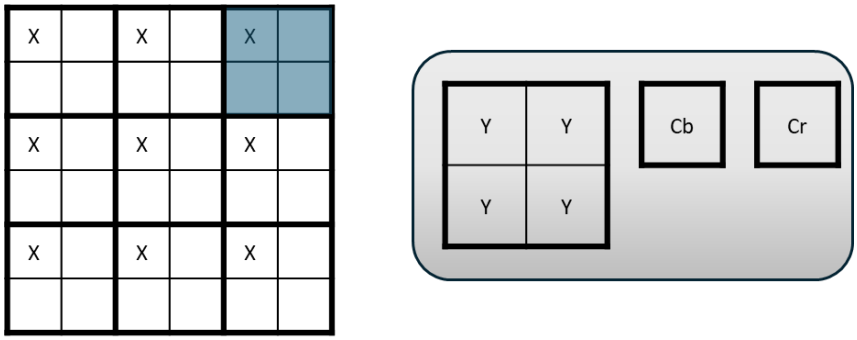

Figura 7. Submuestreo 4:2:0 (Izq.) y Composición de un MB en cada componente (Dcha.)

2.2.2 Cuantificación

El proceso de cuantificación se encarga de asignar un valor discreto por cada muestra obtenida en la fase anterior. Esto implica la asignación de niveles, escalones o rangos de valores numéricos a todas las muestras. La precisión de la cuantificación es lo que determina la calidad de la señal digital resultante.

En la Figura 8 se muestra un ejemplo visual de cuantificación uniforme, donde se establecen 8 escalones en los que agrupar todos los valores de Y comprendidos entre 0 y 255. Por ejemplo, para un valor de 76,5 el escalón al que se añadirá es al de 72.

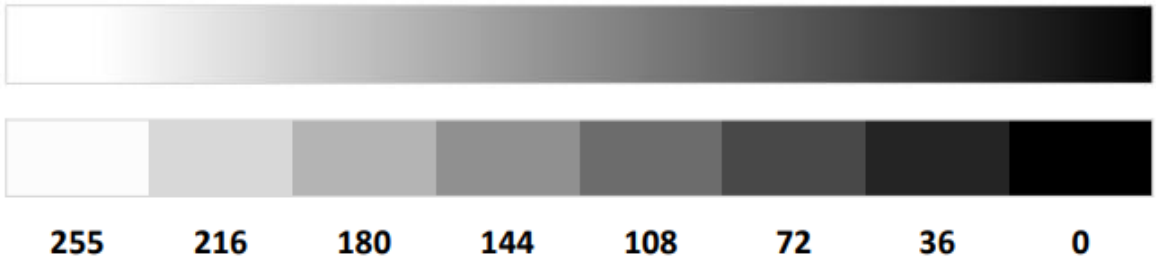

Figura 8. Cuantificación de Y con 8 valores y posterior reconstrucción [9].

Los niveles o escalones disponibles para representar las muestras dependen de la cantidad de bits que se utilicen, a mayor número de bits, mayor será la calidad obtenida en la representación de la imagen. Por ejemplo, con 3 bits para representar un píxel se pueden diferenciar un total de 2^3 niveles.

Cuantos más bits se utilicen, mayor número de niveles se tendrá y, por tanto, la precisión a la hora de encontrar valores semejantes a los muestreados aumentará.

2.3 Codificador

La imagen muestreada que llega al codificador se divide en unidades mínimas para optimizar el proceso de compresión y mejorar la eficiencia de la codificación de vídeo. Algunas de las razones por las cuales se realizan estas divisiones en las imágenes son las siguientes:

- Permiten aplicar diferentes técnicas de compresión en áreas diferentes, optimizando la eficiencia.
- Es más fácil identificar y eliminar redundancias en áreas pequeñas y similares, mejorando la tasa de compresión.
- Permite realizar predicciones más precisas y eficaces.
- Permite aplicar niveles de cuantificación diferentes a diferentes partes de la imagen, basándose en la cantidad de detalle presente

Si en la Figura 6 se adelantaba el proceso de división de la imagen en macrobloques, a continuación, en la Figura 9 se muestra como a partir de esa división en macrobloques (MB) de 16x16 píxeles, se segmenta de nuevo dentro de cada MB para obtener bloques de 8x8 valores.

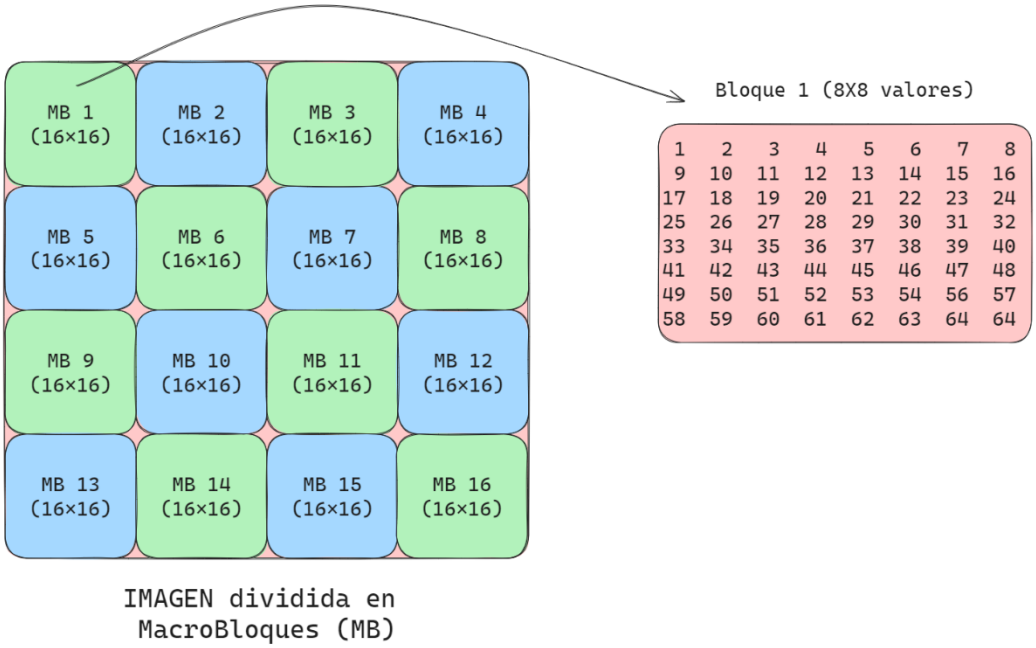

Figura 9. División de imagen en unidades mínimas.

2.3.1 Transformada Discreta del Coseno (DCT)

Cada bloque obtenido en la división de la imagen es sometido a la Transformada Discreta del Coseno (DCT). Este es un paso elemental durante la compresión de imágenes y videos, ya que se encarga de convertir los datos de los píxeles del dominio espacial (valores de intensidad de los píxeles) al dominio de la frecuencia. Así se realiza una mejor compresión ya que se concentra la energía de la imagen en los primeros coeficientes, sobre todo en el coeficiente DC F(0,0), lo que permite una compresión eficiente.

La DCT toma una matriz bidimensional de datos y la transforma en una nueva matriz bidimensional del mismo tamaño siguiendo la fórmula (*2*), donde cada valor representa una frecuencia específica. La DCT es similar a la Transformada de Fourier, pero dispone de la ventaja de no tener parte compleja, lo cual simplifica los cálculos y además genera menos efecto de bordes que la DFT (Discrete Fourier Transform), lo que resulta en una representación más eficiente para señales con bordes y detalles finos, como las imágenes.

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

Donde $f(x,y)$ es el valor del píxel en la posición (x,y) del bloque original, $F(u,v)$ es el coeficiente de DCT en la posición (u,v), N es el tamaño del bloque y $\alpha(u) y\ \alpha(v)$ son factores de normalización definidos como (*3*):

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & si\ u = 0 \\ \sqrt{\frac{2}{N}} & si\ u > 0 \end{cases} \qquad \alpha(v) = \begin{cases} \sqrt{\frac{1}{N}} & si\ v = 0 \\ \sqrt{\frac{2}{N}} & si\ v > 0 \end{cases} \qquad (3)$$

Este paso es útil ya que los coeficientes de alta frecuencia, que generalmente aportan los detalles menos perceptibles, pueden ser eliminados o cuantizados agresivamente sin una pérdida significativa de calidad visual.

Existe una variación de la DCT, conocida como la Transformada Discreta Entera (DIT) que se utiliza en los codificadores modernos, esta variante se adapta específicamente a operaciones con enteros (no con reales), lo que reduce los errores de redondeo y las imprecisiones ocurridas cuando hay decimales. Además, la DIT conserva propiedades útiles de la DCT, como la capacidad de concentrar la energía en unos pocos coeficientes de baja frecuencia y la reducción de efectos de borde, solo que aplicada a bloques de 4x4 valores.

2.3.2 Cuantificación

La cuantificación es el proceso posterior a la DCT que se encarga de reducir la precisión de los coeficientes DCT para disminuir la cantidad de datos que deben almacenarse o transmitirse. Esto se logra, según la fórmula (*4*), dividiendo cada coeficiente DCT por un valor de cuantificación y luego redondeando el resultado al entero más cercano. Cabe destacar que, aunque este paso introduce pérdida de información, permite una significativa reducción en el tamaño de los datos.

$$Q(u,v) = round\left(\frac{F(u,v)}{QMatrix(u,v)}\right) \qquad (4)$$

La cuantificación se realiza empleando una matriz de cuantificación la cual varía según el nivel de compresión deseado y del estándar de codificación aplicado (como JPEG, MPEG, etc.).

Las matrices utilizadas en la cuantificación se guían mediante criterios perceptuales, cuantificando con escalones mayores los coeficientes de frecuencias menos perceptibles para el ojo humano (altas frecuencias). Esto genera menor rango, lo cual facilita un menor uso de bits y una mayor compresión. Por otro lado, los coeficientes de baja frecuencia contienen la mayor parte de la información visual importante y se cuantifican con escalones menores para mantener la calidad visual.

2.3.3 Reordenamiento

La matriz de coeficientes cuantificados o índices se reordena siguiendo un patrón en zigzag como el de la Figura 10, que muestra un bloque de 8x8 empleado en MPEG2, agrupando primero al principio de la secuencia aquellos coeficientes de baja frecuencia (que tienden a ser más significativos) y dejando al final los de alta frecuencia (que tienden a ser menos significativos). Esto genera largas secuencias de ceros que se pueden codificar con muy pocos bits.

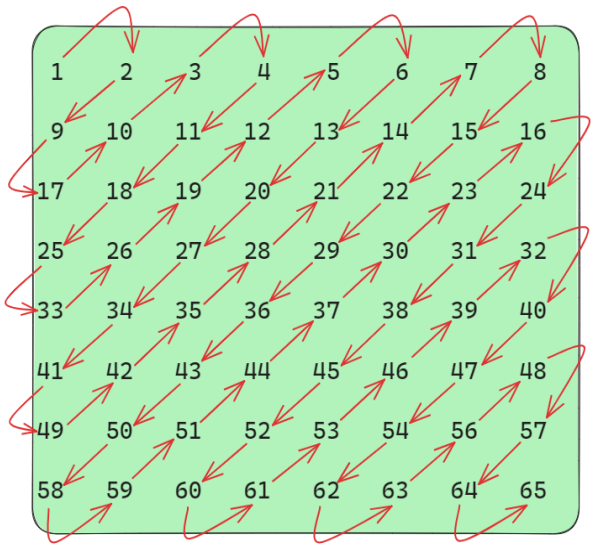

Figura 10. Ordenamiento de índices al serializar un bloque.

2.3.4 Codificación de Entropía

La codificación de entropía es el paso final, previo al empaquetado y transmisión, del proceso de compresión de imágenes y videos. Aquí se aprovechan las propiedades estadísticas de los datos para representar la información de manera más compacta. En este proceso destacan dos técnicas principales: la codificación Run-Length (RLE) y la codificación de Huffman.

- Codificación de Run-Length: esta codificación es utilizada para comprimir secuencias de datos con valores repetidos, que suelen ser los coeficientes de frecuencia más altos donde se obtienen muchos ceros consecutivos tras la cuantificación y el reordenamiento en zigzag.

- Codificación de Huffman: es una técnica de codificación de entropía que asigna códigos de longitud variable a los valores, de manera que los valores más frecuentes obtienen códigos más cortos. Este método reduce el tamaño promedio de los datos codificados.

En este paso se introducen los valores binarios que componen el Código de Longitud Variable (VLC), se deben tener en cuenta la existencia de valores reservados para asegurar la integridad y correcta decodificación de la secuencia como, por ejemplo:

- Valores de sincronización: son marcas que ayudan al decodificador a sincronizarse con el flujo de datos.
- Indicadores de Inicio/Fin de Bloque: valores que delimitan el comienzo y el final de los bloques.
- Control de Errores: señalan y manejan errores en la transmisión de datos.

2.4 Proceso de codificación

Una vez enumeradas todas las operaciones llevadas a cabo por el codificador, tras el proceso de digitalización, conviene conocer la funcionalidad del algoritmo utilizado en la codificación y se encarga de deducir datos de bloques futuros a partir de los recibidos anteriormente. Esto genera una predicción, lo cual permite codificar solamente la diferencia entre lo predicho y lo que realmente se ha recibido, si se realiza una buena predicción los bits a utilizar para codificar la diferencia se reducen. A esa diferencia se la denomina error de predicción o imagen residual.

2.4.1 Predicción

La predicción es una técnica utilizada en la codificación de video para reducir la redundancia temporal y espacial en las secuencias de video. La predicción se basa en la estimación de los valores de píxeles en un cuadro de vídeo basándose en la información disponible de cuadros anteriores o del mismo cuadro. Existen dos tipos de predicción: Intra e Inter.

- Predicción INTRA

  Utiliza la información disponible dentro de un solo cuadro y explota la redundancia espacial. Como en la codificación de video los cuadros se dividen en bloques, se va a utilizar la predicción para explotar las similitudes entre los píxeles vecinos dentro del mismo bloque. Se predicen los valores de cada bloque de píxeles basándose en los valores de los píxeles circundantes que ya han sido procesados. Se pueden aplicar diferentes predicciones como, por ejemplo:

  - Predicción vertical: Predice los valores de los píxeles basándose en los píxeles de arriba.
  - Predicción horizontal: Predice los valores de los píxeles basándose en los píxeles de la izquierda.
  - Predicción DC: Utiliza el valor promedio de los píxeles circundantes.

- Predicción INTER:

  Utiliza la información disponible entre diferentes cuadros de video y explota la redundancia temporal, es decir, las similitudes entre cuadros consecutivos. Se predicen bloques de un cuadro actual basándose en bloques de cuadros anteriores o posteriores, denominados cuadros de referencia. Igual que en el caso anterior, la imagen se divide en bloques, para los cuales se realiza una búsqueda para encontrar el bloque, en el cuadro de referencia, más parecido con el bloque actual, de acuerdo con una distancia definida.

Este algoritmo se basa en la compensación de movimiento de un mismo cuadro o diferentes y es donde se obtiene la imagen predicha a partir de los vectores de movimiento (VM) que han sido previamente calculados a partir de la estimación de movimiento y que describen el desplazamiento de los bloques desde el cuadro de referencia hasta el cuadro actual.

Una vez obtenida la predicción, se realiza la resta entre el bloque original recibido y el bloque predicho, a partir del cual se obtiene el bloque residual que es el que se va a codificar. En la Figura 11 se muestra un ejemplo de resta de bloques tras aplicar una predicción horizontal.

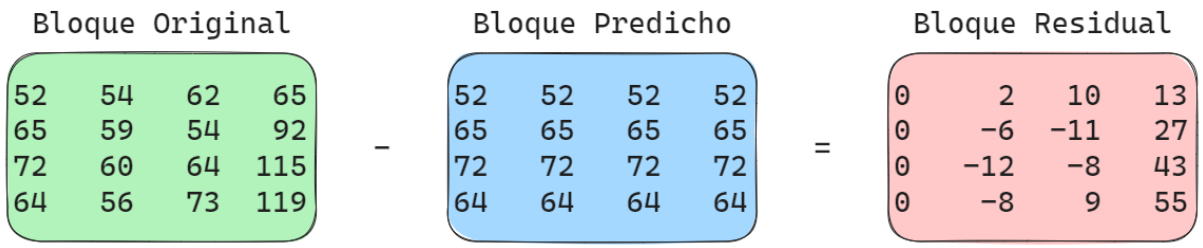

Figura 11. Ejemplo de cálculo de bloque residual o error de predicción para bloques 4x4.

2.4.2 Esquema final de Codificación

A continuación, se muestran a modo de resumen unos esquemas para visualizar el tráfico de las imágenes durante el proceso de codificación de un codificador MPEG2.

Cuando se dispone de una secuencia de vídeo, el primer fotograma a enviar no dispone de información precedente con la cual compararse, por lo que el modo a utilizar en este caso suele ser INTRA. En la Figura 12 se destaca mediante flechas rojas el recorrido realizado por la imagen actual denominada imagen de tipo I (intra), donde primero se somete al cálculo de la DCT y la cuantificación (Q) para transformarse en una imagen codificada. La imagen codificada es enviada a través de dos caminos diferentes:

- Un camino consiste en enviar la información de la imagen codificada en forma de secuencia binaria al receptor, para su posterior decodificación.
- El otro camino consiste en enviar la misma información al decodificador existente dentro del emisor (con las mismas características que el decodificador del receptor) para disponer también de la imagen decodificada.

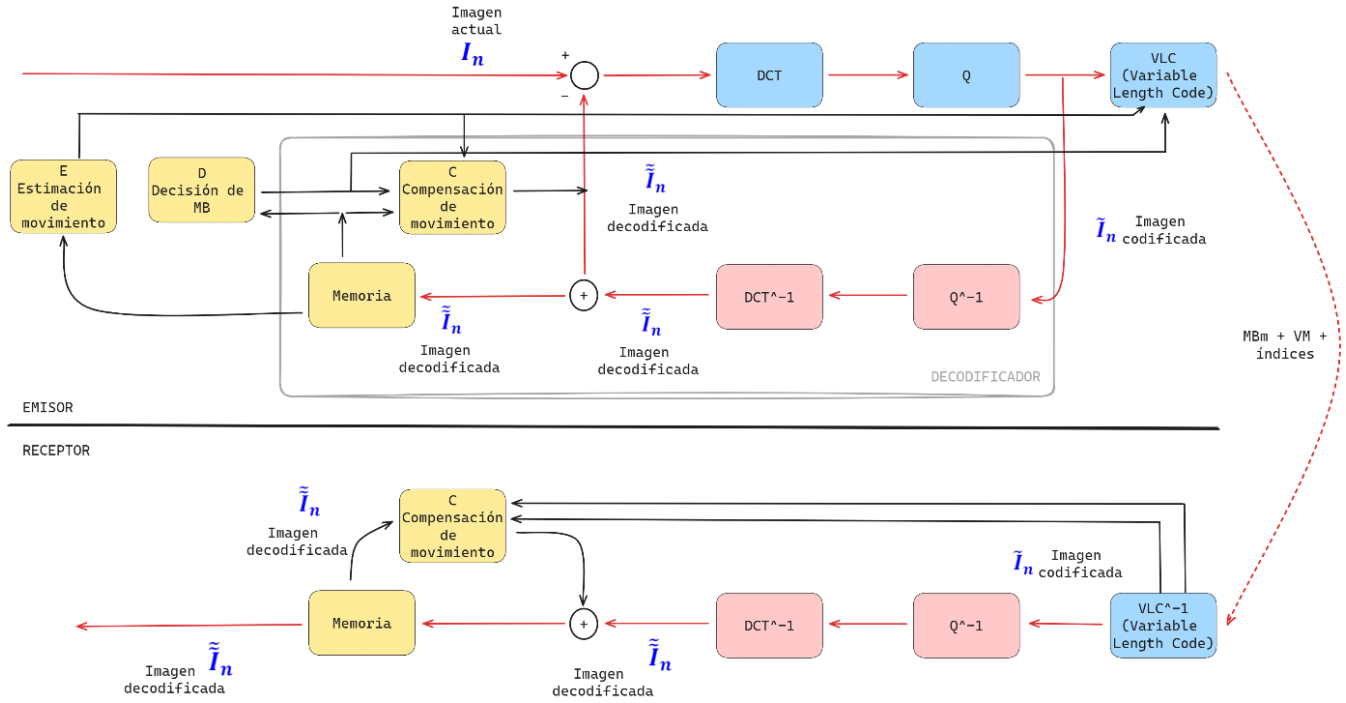

Figura 12. Esquema de circulación de una imagen INTRA.

En el emisor la imagen decodificada se guarda en memoria para convertirse en la imagen referencia de los siguientes fotogramas. En la parte del receptor, la imagen decodificada aparte de mostrarse al cliente, también se almacena en memoria para recomponer el resto de las imágenes a recibir.

Cabe destacar que en el caso de predicción INTRA, todos los macrobloques son del mismo tipo (INTRA) y al no disponer de imagen referencia, se resta con una imagen nula. Esto provoca que no se generen vectores de movimiento (VM).

Continuando con la secuencia de imágenes, al codificador llega una nueva imagen actual donde la imagen que lo era anteriormente pasa a convertirse en la imagen anterior decodificada o también conocida como imagen de referencia. En el momento en el que se dispone de una imagen de referencia ya se puede aplicar la predicción INTER, en este caso para una imagen de tipo P (predicción). En la Figura 13 se visualiza el nuevo proceso llevado a cabo junto con los tipos de imágenes implicadas.

La imagen actual se utiliza para compararse con la de referencia y realizar una estimación de movimiento, donde se obtienen los vectores de movimiento. En este caso como solo hay una referencia anterior, el tipo de imagen es P y tiene como VM, los VM forward (FW) que permiten predecir la siguiente imagen mediante referencias pasadas. Estos VM sirven para conocer la dirección y posición en la que se encuentra el bloque más parecido e la imagen de referencia. Además, también se deciden los modos de macrobloque (MBm), que para imágenes de tipo P pueden ser: Intra, Forward y sin vector de movimiento.

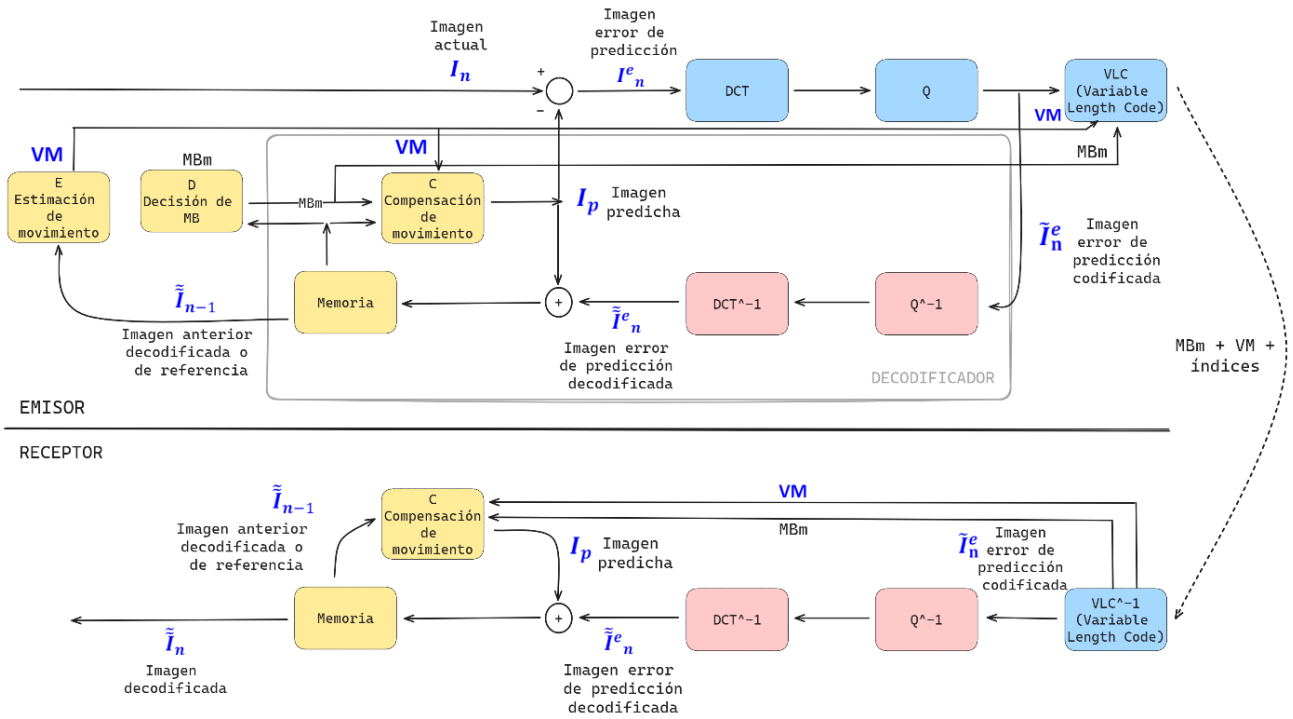

Figura 13. Esquema de circulación de una imagen INTER.

Una vez calculados los VM y los MBm, se puede intentar deducir de forma aproximada la imagen actual, este resultado es lo que se conoce como imagen predicha. Una vez generada la imagen predicha, a la imagen actual se le resta la imagen predicha para obtener directamente la imagen de error de predicción. Esta imagen es la que se va a codificar y enviar a los dos decodificadores, tanto del emisor como del receptor.

En el decodificador del emisor, directamente se recupera la imagen de error de predicción decodificada y se le suma la imagen de predicción calculada para así obtener la imagen actual, la cual se almacena en memoria para servir de referencia para las siguientes imágenes a decodificar. Sin embargo, al receptor llega tanto la imagen de error de predicción codificada, como los modos de codificación de cada macrobloque y sus vectores de movimiento. Con esos datos la imagen de error de predicción codificada se decodifica en el receptor y junto con los vectores de movimiento y modos de macrobloque se envían al compensador de movimiento (C) donde se reproduce la imagen predicha a partir de la imagen anterior de referencia que se tenía almacenada. Al conseguir la imagen predicha ya solo queda sumar el error de predicción para obtener la imagen decodificada actual.

Continuando de forma sucesiva con la aplicación de este proceso de codificación se encuentra un tercer tipo de imagen, tipo B (bidireccional). Estas imágenes tienen referencias tanto futuras como pasadas, por lo que dentro de sus VM se distinguen dos tipos: Forward (FW) y Backward (BW). También incrementan las posibilidades de sus MBm, donde pueden tener modos: Intra, Forward, Backward, Bidireccional y sin vector de movimiento.

3. Especificaciones y restricciones de diseño

El objetivo principal del proyecto es el diseño e implementación de una aplicación que permita al usuario visualizar y representar toda la información implicada en el proceso de codificación de video. Se busca que la herramienta sea didáctica y de fácil manejo para los estudiantes de ingeniería que cursen la asignatura de Tecnologías de Imagen y Vídeo del Grado de Sonido e Imagen de la ETSIST y que deseen comprender en profundidad el proceso de codificación de video, más concretamente la extensión de un codificador MPEG2 utilizado en las prácticas.

3.1 Objetivos del proyecto

A continuación, se agrupan y recogen los objetivos del proyecto según su finalidad, esto permite reflejar la evolución de estos de una forma clara.

3.1.1 Selección del entorno de trabajo

Como punto de partida en cualquier proyecto que implique desarrollo de software, antes de comenzar se debe elegir la plataforma sobre la cual se va a construir. En este caso se valoraron dos tipos de plataforma diferentes, MATLAB y Unity, ya que ambas plataformas permiten desarrollar software mediante una interfaz gráfica de forma visual, aunque cada una de ellas ofrece unas prestaciones diferentes.

- MATLAB: dispone de una herramienta propia llamada “App Designer” la cual permite diseñar de forma rápida y sencilla aplicaciones con interfaz de usuario compleja. Es una herramienta poderosa y versátil con la cual diseñar y crear fácilmente aplicaciones interactivas sin necesidad de tener experiencia previa en programación de interfaces de usuario. Un punto a favor de esta opción es que ya se dispone de un código base previo para la codificación de imágenes, los cuales se pueden utilizar directamente como base de este proyecto.
- Unity: ofrece la posibilidad de crear una interfaz de usuario visual y llamativa, acompañada de gran velocidad de cálculo de operaciones útiles para su uso en los procesos de codificación y decodificación. A la hora de la obtención del ejecutable final se conoce con seguridad que el código interno es inaccesible para el usuario, lo cual cumple con el requisito necesario de aislar al usuario del código interno de la aplicación. Sin embargo, el aspecto negativo aquí es la migración y adaptación del código base previo de los que ya se disponían en MATLAB.

Tras valorar ambas opciones, se decide utilizar MATLAB para mantener la homogeneidad entre el código y las utilidades ya existentes y los nuevos desarrollos, descubriendo el potencial de la herramienta App Designer y aprendiendo sobre su funcionamiento.

3.1.2 Selección de imágenes y datos para visualizar

Una vez escogida la plataforma, el objetivo principal pasa a ser la enumeración de los elementos a mostrar en la aplicación y su presentación conjunta de forma visual, permitiendo la conmutación rápida entre los mismos. Así los alumnos disponen de toda la información implicada en la codificación y son capaces de obtener sus propias conclusiones en base a comparaciones.

Se distinguen tres grupos diferentes para mostrar información:

- Imágenes: imágenes implicadas en el proceso de codificación e informaciones auxiliares sobre los resultados o los datos de entrada.
    - Imagen Original: es una imagen o cuadro de vídeo extraído de la secuencia de video que se va a codificar.
    - Imagen de Predicción: es el resultado de la compensación de movimiento cuando se trata de una imagen Inter.
    - Error de Predicción: es la imagen de error obtenida al restar la imagen original y la imagen de predicción.
    - Imagen Diferencia Forward (FW): es la imagen diferencia entre la imagen actual y la referencia pasada. Representa el cambio entre ambas.
    - Imagen Diferencia Backward (BW): es la diferencia entre la imagen actual y la referencia futura.
    - Bits por bloque Y: es la representación gráfica de la cantidad de información necesaria para codificar cada bloque de la imagen de Luminancia.
    - Bits por bloque C: es la representación gráfica de la cantidad de información necesaria para codificar cada bloque de la imagen de Crominancia.
    - Imagen Decodificada: imagen resultado obtenida en el receptor tras la decodificación.
- Overlay: son los elementos que se superponen sobre las imágenes y que aportan información visual sobre cada una de las imágenes anteriores:
    1. Vectores de movimiento (VM): vectores que indican el desplazamiento de bloques de píxeles entre cuadros consecutivos en una secuencia de vídeo. Según al cuadro de video al que hagan referencia pueden ser: forward o backward.
    2. Modos de Macrobloque (MB): diferentes técnicas de predicción aplicadas a un macrobloque en la codificación de vídeo. Los modos pueden incluir predicción Intra, Forward, Backward, Bidireccional y sin vector de movimiento.
    3. Rejilla: estructura de división de una imagen en bloques o macrobloques durante el proceso de codificación de vídeo.
- Tablas y gráficas: apartado para mostrar y presentar los datos comparativos para cada secuencia de codificación de forma rápida, ordenada y visual.

3.1.3 Comparación de imágenes

A la hora de visualizar una secuencia de imágenes resulta de gran utilidad la comparación de los diferentes fotogramas entre sí de manera simultánea para ayudar a comprender el proceso de codificación. Por tanto, una de las especificaciones más clara de este proyecto era permitir la posibilidad de comparación y conmutación rápida entre los fotogramas y los tipos de datos representados que componen cada secuencia.

3.1.4 Codificación y decodificación

Otro posible objetivo para completar aún más la aplicación es la posibilidad de realizar todos los cálculos de codificación y decodificación en la aplicación, ya que para los objetivos anteriores se aportan ya secuencias con todos los tipos de imágenes precalculadas y se almacenan en un fichero los datos intermedios del proceso de codificación. Llevar a cabo este punto facilita el manejo al usuario, ya que solo necesita aportar una secuencia, para obtener directamente todos los datos a visualizar.

Incluir el proceso de codificación y decodificación en la aplicación, mejora la usabilidad de la herramienta, puesto que se gestionan los tipos de imágenes a calcular de forma interna. Sin embargo, es posible que se alarguen los tiempos de visualización de los resultados.

3.2 Grado de cumplimiento de objetivos.

Para analizar el grado de cumplimiento de los objetivos propuestos, se parte de la elección de entorno de trabajo, puesto que dicha elección es un factor condicionante para el resto de los objetivos. El entorno de trabajo elegido tras la realización de un análisis crítico de las ventajas y desventajas es MATLAB, puesto que se decide priorizar la visualización de imágenes antes que la adaptación del código fuente existente. Empleando MATLAB se pueden codificar las imágenes de forma externa y aportarlas como datos de entrada al programa.

Una vez elegido software base se procede a la realización del proyecto, donde al concluir con el proceso se analizan los objetivos y especificaciones establecidos. En la Tabla 1 se recogen todos los objetivos junto con su grado de consecución final.

Tabla 1. Cumplimiento de Objetivos

	¿Conseguido?
Visualizar tipos de imágenes	✓
Presentar tablas	✓
Presentar gráficas	✓
Comparación de imágenes	✓
Codificación de secuencias	✗
Decodificación de secuencias	✗

La codificación y decodificación de secuencias persigue la funcionalidad de que, partiendo de una secuencia de fotogramas, se realicen todas las operaciones del proceso de codificación y decodificación, los cuales no han sido posibles de implementar por falta de tiempo.

Se da prioridad a otros factores mucho más importantes como la visualización y la conmutación de opciones para permitir la comparación de imágenes, antes que el cálculo interno de dichos procesos.

Para solventar esta ausencia de objetivos, se toma como punto de partida el conjunto de imágenes obtenidas como resultado de la ejecución del proceso de codificación disponible en funciones de MATLAB.

4. Descripción de la solución propuesta

En toda aplicación de escritorio, uno de los puntos más importantes a tener en cuenta es la organización y estructuración del espacio, cuanto más limpia, ordenada y manejable sea la aplicación mejor será la experiencia del usuario. Al desarrollarse una herramienta de visualización se prioriza el uso de pantalla completa para ofrecer la mayor resolución posible, ofreciendo un aumento del espacio empleado en mostrar mayor cantidad de información y permitiendo la comparación simultánea.

Visualizar dos imágenes diferentes al mismo tiempo permite al usuario fijarse en los detalles y comparar diferentes cuadros. También se muestran gráficas y tablas para complementar su aprendizaje. Por este motivo se deben organizar todos los elementos visuales, de forma que el usuario pueda acceder fácilmente a ellos permitiendo libertad y fluidez de cambio entre diferentes imágenes, tipos de imágenes y secuencias.

4.1 Instalación y configuración del entorno

Previo al desarrollo de la aplicación, es necesario preparar el entorno de MATLAB, para ello se debe disponer de un ordenador con al menos 8GB de RAM para poder funcionar con la versión de MATLAB R2023a. Además, dentro de la versión de MATLAB se deben instalar una serie de "toolboxes":

- MATLAB Image Processing Toolbox.
- MATLAB Compiler.

Con MATLAB instalado, se inicia y se añaden todos los "toolboxes" necesarios. Para eso se selecciona en la pestaña "Home"→"Add-Ons"→"Get Add-Ons" y se buscan los "toolboxes" mencionados, siguiendo las instrucciones para su instalación.

La apertura del proyecto de App Designer debe realizarse comprobando que la ruta de trabajo de MATLAB coincide con la ubicación de la carpeta donde se tiene el fichero "VisualCodecInsight.mlapp". Tras esto, se realiza doble click para abrir la aplicación en una nueva pestaña de MATLAB App Designer donde se permite editar y desarrollando la herramienta.

4.2 Generación de ficheros de entrada

La obtención de los datos de entrada necesarios a la hora de la utilización del programa se calcula mediante una serie de funciones. De las prácticas del laboratorio de la asignatura de Tecnologías de Imagen y Video (TIV) se dispone de un conjunto de scripts que se comportan como una extensión de un codificador MPEG2.

Solo se necesita aportar a las funciones existentes una secuencia de imágenes junto con el grupo de imágenes (GOP – Group of Pictures). Una vez se ejecuta, se calculan las operaciones de submuestreo, cuantificación, codificación, estimación de movimiento, compensación de movimiento, decodificación y el resto de las operaciones inversas a las anteriores.

De este proceso se obtienen una serie de imágenes recurso agrupadas en carpetas que siguen la estructura mostrada en la Figura 14.

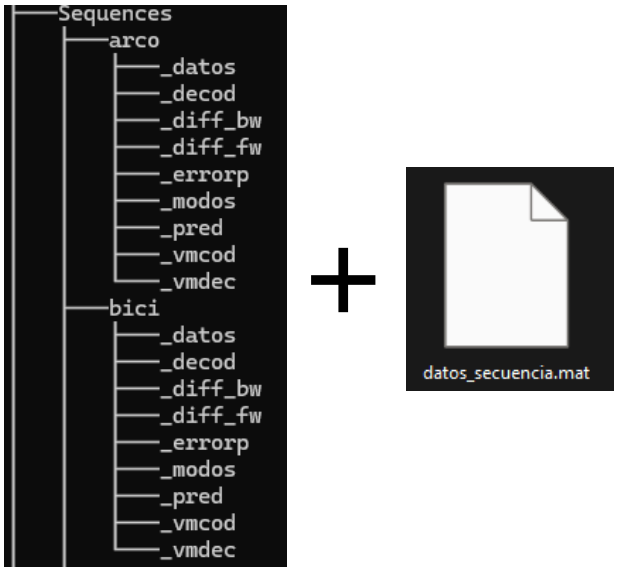

Figura 14. Estructura de carpetas de las secuencias

La relación entre las carpetas y los tipos de imagen presentadas es la siguiente:

- Imagen Original: se encuentran dentro de la propia carpeta de la secuencia sin agruparse en ninguna carpeta.
- Imagen de Predicción: están en la carpeta _pred.
- Error de Predicción: se guardan en la carpeta de _errorp.
- Imagen Diferencia Forward (FW): se almacenan en la carpeta de _diff_fw.
- Imagen Diferencia Backward (BW): se almacenan en la carpeta de _diff_bw.
- Imagen Decodificada: se guardan en la carpeta de _decod.

También se genera un fichero "datos_secuencia.m" propio por cada secuencia que recoge los siguientes datos obtenidos tras el proceso de codificación:

- Datos comunes a todas las imágenes como el alto y el ancho de la imagen, las dimensiones de cada macrobloque o el número de macrobloques totales.
- Datos genéricos de la secuencia como el número de imágenes total, el número de imagen de cada tipo (I, P o B), los bits medios empleados en cada tipo o los modos de la secuencia también conocido como GOP.
- Los vectores de movimiento compuestos por las puntas y las líneas de desplazamiento.
- Array de calidades PSNR.
- Array de bits totales, de luminancia o de crominancia
- Array con todos los modos de macrobloques de la imagen para poder representarlos acorde con el código de colores establecido.
- Dimensiones en píxeles de la imagen.
- El valor de precisión en bits de los coeficientes DC en bloques intra-codificados.

- El tipo de factor qscale que puede ser lineal o exponencial, el qscale empleado en intra y el empleado en inter. Este factor indica la escala de cuantización utilizada y afecta a la calidad visual. Cuanto mayor sea este valor mayor será la cuantificación realizada.

Es importante tener en cuenta lo anterior ya que la aplicación la lee todos esos datos de entrada internamente. Se basa en directorios autocompletados según el "path" recibido al principio para cargar las imágenes obtenidas y utiliza el fichero "datos_secuencia.m" para interpretar sus datos y gestionar dicha información en las gráficas y tablas.

4.3 Diseño base previo

En relación con lo anterior, se plantea una base organizativa mediante un sistema de pestañas ("Tab groups"). Dividir en pestañas la información ofrece esa fluidez, claridad y orden que se busca conseguir para el usuario, permitiendo la conmutación rápida entre ellas. Tener diferentes pestañas hace aumentar el espacio útil ya que es como si la información se encontrara apilada una sobre otra y solo es visible en caso de que se seleccione una pestaña en concreto.

En la Figura 15 se muestra la disposición de las pestañas, ocupando el máximo de la pantalla posible y permitiendo agregar un pequeño logotipo en el lado izquierdo.

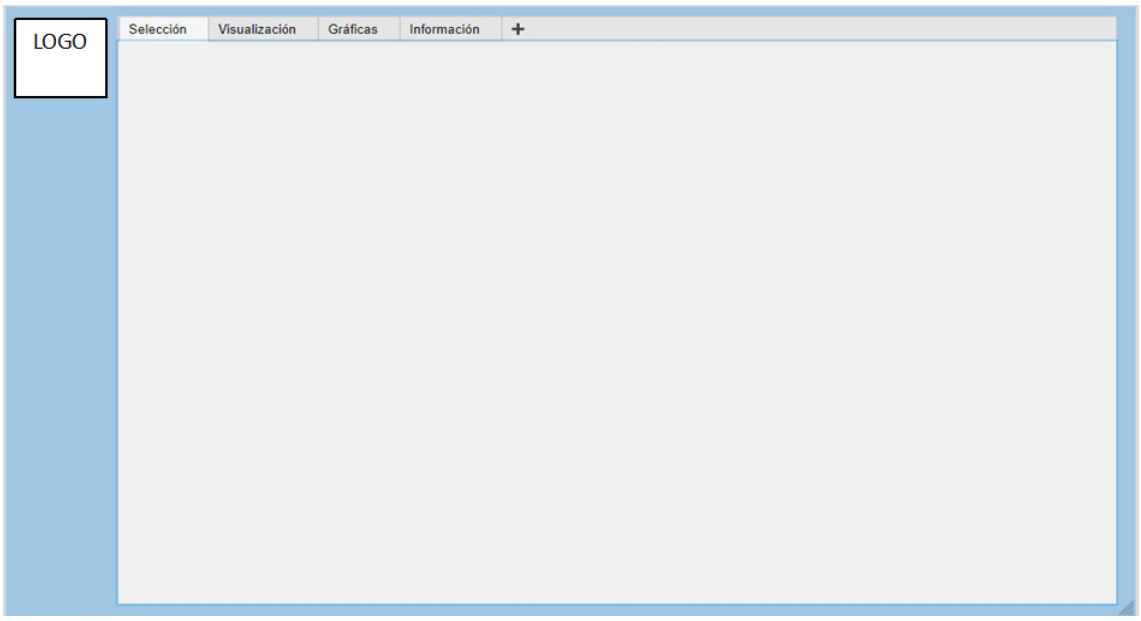

Figura 15. Organización de interfaz base.

4.4 Pestañas establecidas

La información que maneja la aplicación se engloba en imágenes y datos. Al visualizar las imágenes se persigue el objetivo de la comparación y superposición de elementos, esto supone que al menos dos imágenes deben mostrarse de forma simultánea y para ello se requiere mucho espacio útil. De forma análoga sucede con los datos, los cuales deben ser representados en gráficas, tablas y diagramas. Teniendo en cuenta estos aspectos, se decide agregar un total de 4 pestañas: Selección, Visualización, Gráficas e Información.

4.4.1 Pestaña Selección

En esta pestaña se muestran instrucciones iniciales y se permite al usuario elegir la ubicación de la carpeta a utilizar. Al principio, esto se plantea de forma básica, donde el usuario selecciona una secuencia y obtiene los tipos de imágenes a visualizar en una lista, sin embargo, se descubre que no es funcional valorando los siguientes aspectos:

- Si el usuario cambia de tipo de imagen a visualizar se le obliga a conmutar entre varias pestañas.
- Al seleccionar el tipo de imagen, este es único y ambas comparaciones se hacen entre diferentes cuadros de un mismo tipo, lo cual no es útil porque igual en algún caso se quiere ver el 5º fotograma de imagen predicha y de error de predicción.
- Si se cambia la secuencia a visualizar, se debe buscar de nuevo el directorio y esto resulta tedioso y ralentiza el proceso.

La solución final se muestra en la Figura 16 donde a partir del directorio general se agrupan y listan todas las secuencias para que el usuario cambie de secuencia con solo una pulsación.

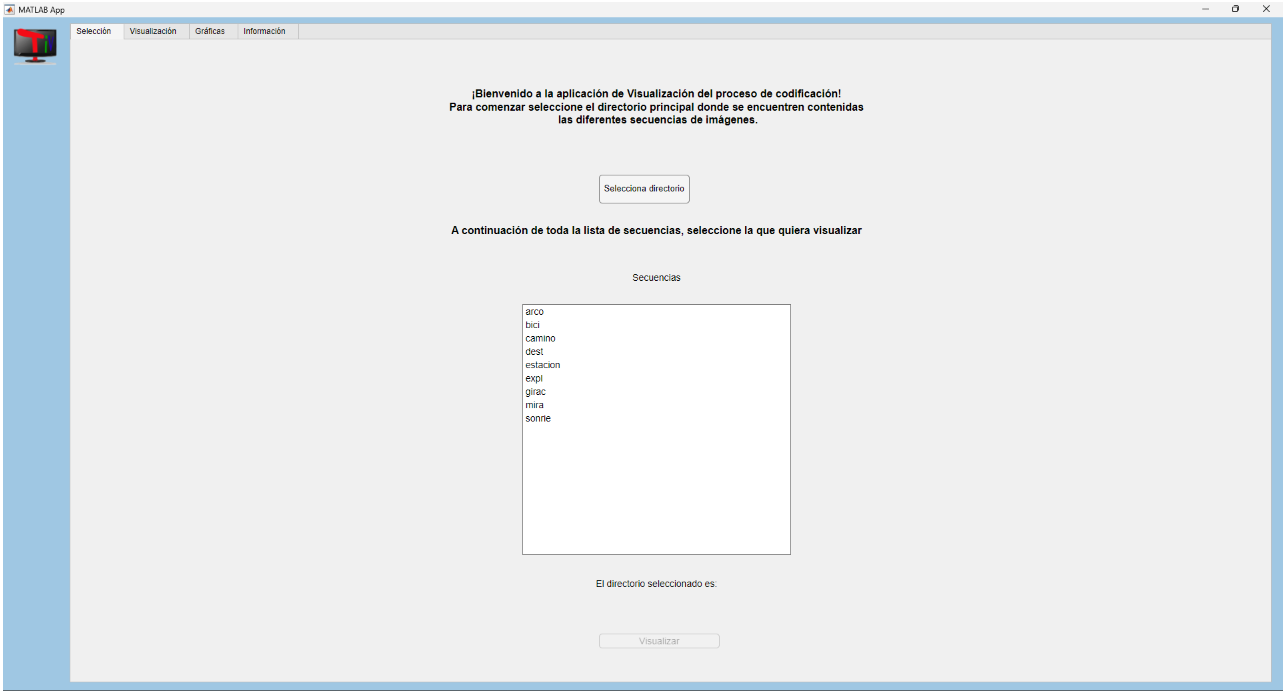

Figura 16. Pestaña de selección del directorio de trabajo.

4.4.2 Pestaña Visualización

En este apartado es donde se agrupan los componentes necesarios para la mostrar al usuario los fotogramas de una secuencia de video. El usuario puede elegir el tipo de información de imagen a visualizar y el instante concreto en la secuencia de vídeo. Además, se pueden comparar dos imágenes de forma simultánea gracias a la existencia de dos secciones independientes, pero que mantienen un mismo funcionamiento.

En el funcionamiento de las listas interactivas se puede elegir el tipo de imagen junto con un número de "frame" a visualizar al que posteriormente se le puede querer cambiar por cualquier otro tipo de imagen y mantiene el índice de secuencia escogido.

Mediante el uso de "check boxes", se puede habilitar o deshabilitar rápidamente la superposición de diferentes overlays, que además son superponibles entre sí. Los tipos de "check box" son:

- Hide luma: elimina la aportación de la Y a la imagen que se esté visualizando.
- Rejilla: crea una cuadrícula que divide los MBs que conforman la imagen.
- Modos de MB: muestran acorde con un código de color, recogido en la Tabla 2, el modo con en el que se procesa y comprime cada MB.

Tabla 2. Código de colores para cada modo de compresión

Intra	Forward	Sin VM	Backward	Bidireccionales

- Vector Forward: vectores calculados sobre imágenes anteriores que ofrecen información de referencia pasadas.
- Vector Backward: vectores calculados sobre imágenes posteriores que ofrecen información de referencia futuras.

Aparte, los overlays disponen de un repertorio prefijado de colores, lo cual permite elegir los colores más adecuados que contrasten sobre el contenido de la imagen representada.

Finalmente se añaden dos funcionalidades nuevas que pueden ser útiles para los usuarios:

- Copiar el zoom de una imagen a la otra, esto facilita la comparación de detalles entre las dos imágenes, ya que, si en una imagen se visualiza una zona, con realizar un click se puede aplicar ese mismo zoom en la otra imagen. También se permite restablecer el zoom original en caso de querer visualizar la imagen de nuevo al completo.
- Se añaden los botones de play y stop que permiten reproducir una secuencia, esto sirve para cuando se quieren ver los cambios secuenciales sin necesidad de estar pulsando todas las imágenes.

La Figura 17 es un ejemplo donde se muestra en la parte superior la imagen de predicción nº5 tipo P, en la cual se han marcado la rejilla y los modos de MB para visualizar, acompañados de los vectores forward visualizados en cada macrobloque que comparta ese modo. En la imagen inferior se ha seleccionado una imagen de tipo B donde se muestran los dos tipos de vectores. En aquellos MBs verdes hay presencia de ambos vectores, tanto forward como backward.

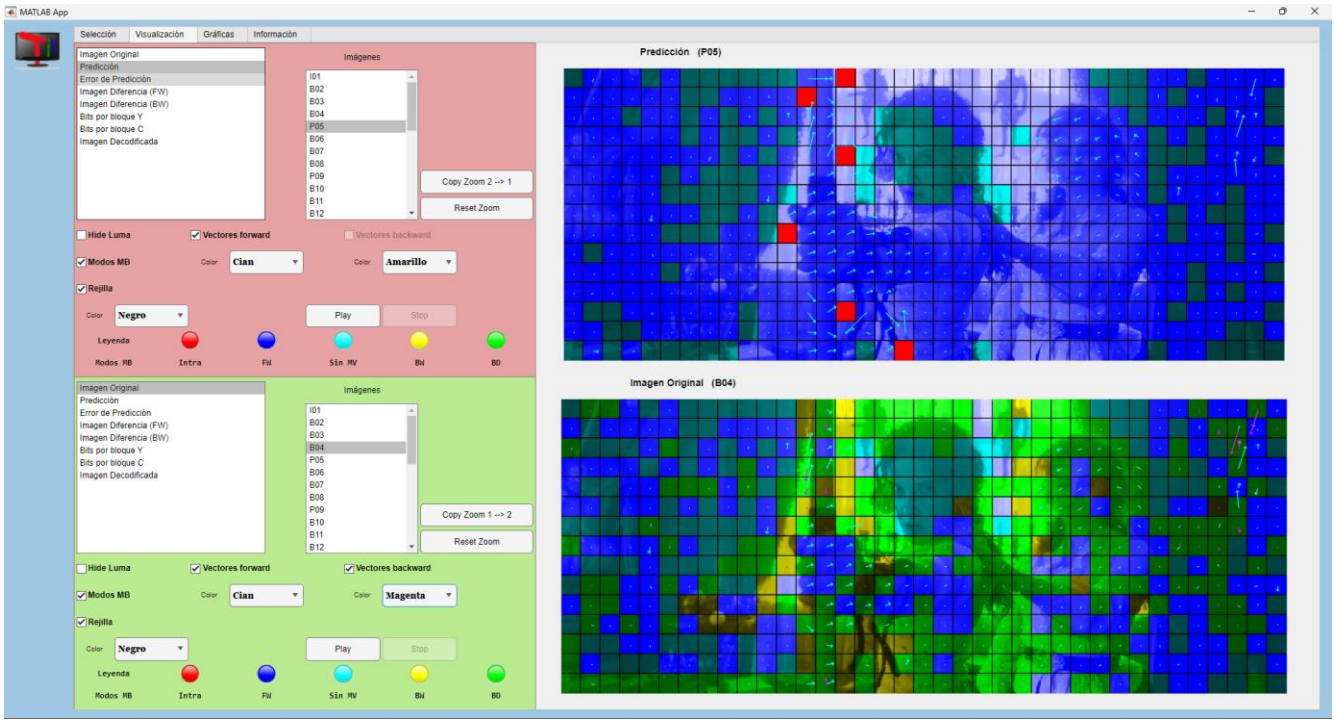

Figura 17. Pantalla de visualización con "check boxes" activados.

4.4.3 Pestaña Gráficas

Esta pestaña es donde se agrupan todas las tablas y gráficas que aportan al usuario una información genérica de una imagen elegida o de la secuencia total codificada. En la Figura 18 se muestra la pestaña en su conjunto.

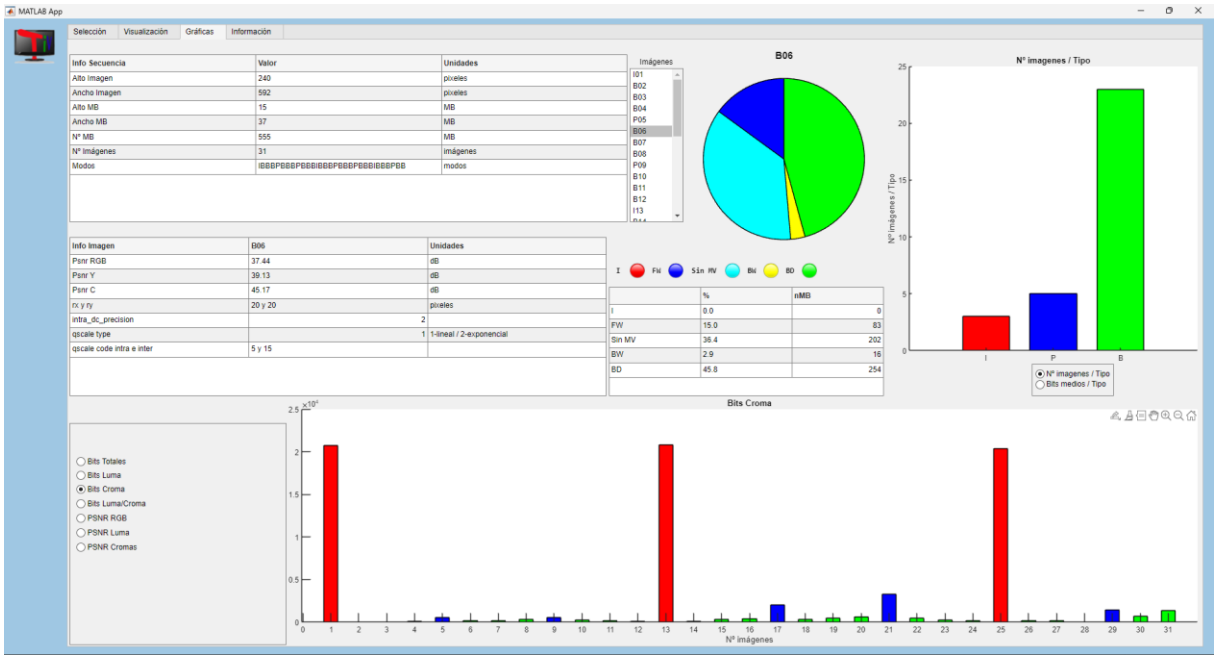

Figura 18. Tablas y gráficas.

- Tabla Información de Secuencia (parte superior izquierda): aporta los datos de la Secuencia, informa del alto y ancho de la imagen, las dimensiones medidas en MB y

por ende el número de MB en una imagen. También aparece el número de fotogramas de los que dispone la secuencia y de los modos de codificación de cada imagen.

- Tabla Información de Imagen (bajo la tabla de información de secuencia): ofrece datos específicos para una imagen seleccionada junto con datos relevantes de variables MPEG2 empleadas en el proceso de codificación.
- Diagrama de sectores (parte superior central): muestra la cantidad de macrobloques de un mismo tipo o modo que aparecen en la imagen. También se aportan los porcentajes en la tabla inferior.
- Gráfica derecha: muestra el número de imágenes de cada tipo y los bits medios en cada modo.
- Gráfica inferior: son comparaciones de valores generales a nivel de secuencia donde en el eje de abscisas se enumeran todas las imágenes.

4.4.4 Pestaña Información

En este apartado no consta ningún elemento interactivo, simplemente se crea a modo de créditos, donde se aporta una breve información sobre la aplicación y se nombran a los autores junto con la institución para la que ha sido desarrollada.

4.5 Elementos utilizados en la interfaz de usuario

En esta aplicación se requiere una interfaz de usuario simple y funcional, por ello se utilizan "Tab Group" como contenedores que agrupen y organicen la información en su interior. Dentro de cada "Tab Group" es necesario aplicar un "Grid Layout", esto se debe a que la aplicación se ha creado para funcionar en ventana maximizada y por tanto se necesita que todos los elementos de su interior se redimensionen de forma adecuada al tamaño de la pantalla. Esta rejilla permite que los elementos se coloquen de forma relativa sin superponerse. Estos dos ítems se encuentran en el apartado de "containers" tal y como se ve en la Figura 19.

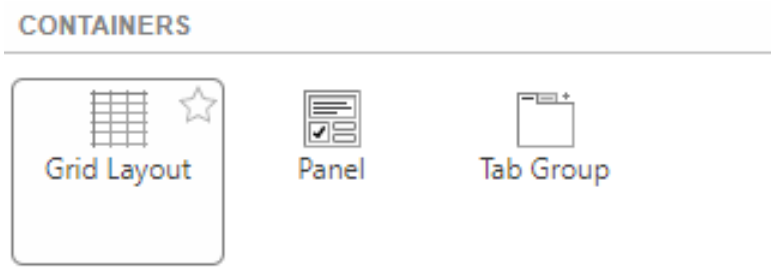

Figura 19. Tipos de "containers".

Al generar el "Grid Layout" es importante dividir de forma adecuada el espacio destinado a los diferentes elementos. Para ello se debe tener en cuenta las dos opciones de configuración:

- Configuración "Weighted": permite asignar pesos relativos a filas y columnas. Por ejemplo, si una fila tiene peso 3 y otra tiene peso 1, la primera fila ocupará el triple de espacio que la segunda fila. Esto es útil si se quiere que las filas o columnas se expandan y contraigan dinámicamente según el tamaño de la ventana o pantalla.

- Configuración "Fixed" : asigna una malla con tamaño fijo, especificando el tamaño de las filas y columnas en unidades de medida como píxeles. Esto significa que las filas y columnas tendrán un tamaño fijo y no cambiarán, independientemente del tamaño de la ventana o pantalla.

Debido a las complejas estructuras de las pestañas se opta por configurar la malla en su mayoría con "Fixed" para que tenga un tamaño adecuado en relación con un número de píxeles.

El propio App Designer ofrece un apartado de componentes interactivos comunes presentado en la Figura 20. De entre los cuales se han utilizado:

- "Axes": sobre los que representar imágenes, overlays y gráficas.
- "Buttons": permiten ejecutar comandos con un solo click.
- "Check boxes": permiten la selección simultánea de varios elementos.
- "List boxes": listan elementos de los que se desconoce su longitud total.
- "Tables": agrupan información sobre la secuencia en formato tabla.
- "Radio button groups": permiten escoger una elección dentro de un grupo de opciones.
- "Labels": permiten añadir texto fijo.

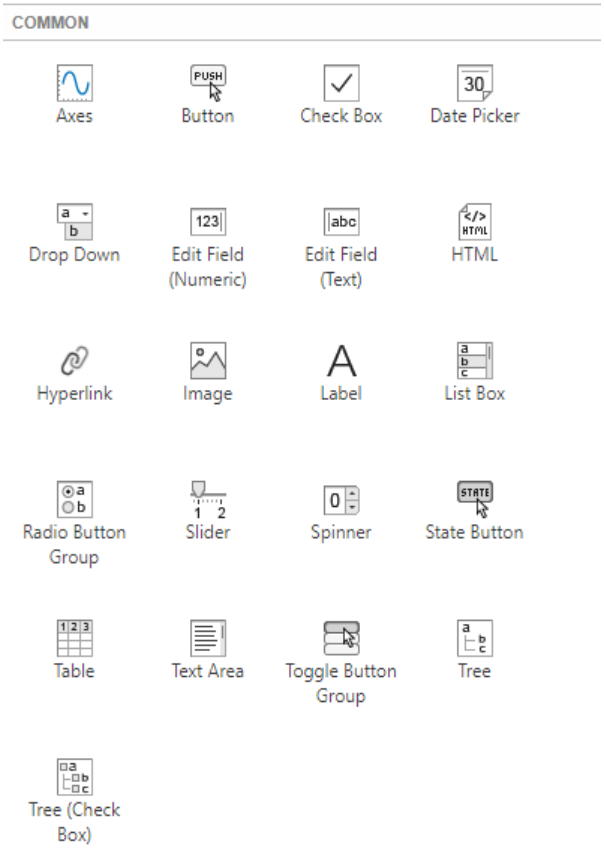

Figura 20. Componentes interactivos comunes en App Designer.

4.6 Callbacks y funciones generales

Los callbacks son funciones ejecutadas como respuesta a un evento en la interfaz gráfica de usuario, como por ejemplo la pulsación de un botón o la elección de un elemento de una lista. Estos callbacks a su vez pueden llamar a otras funciones internas del programa, lo cual permite realizar llamadas asíncronas mientras se ejecutan otras funciones.

Del funcionamiento básico de la herramienta, a continuación, se destacan algunas funciones que explican a rasgos generales el flujo de trabajo llevado a cabo entre eventos y funciones:

1. Al pulsar sobre el botón: Seleccionar directorio, se lanza un evento:

```
function SelectDirectory_ButtonPushed(app, event)
```

   Esta función se encarga de abrir una ventana del explorador de archivos de Windows donde se elige la ubicación de la carpeta de Secuencias, también guarda ese directorio en una variable y llama a la función encargada de listar las carpetas de secuencias:

```
function mostrarCarpetaGeneral(app)
```

   Esta función enlista todas las carpetas y las enumera en el "list box" ubicado en la ventana de Selección.
2. Al seleccionar una opción de secuencia del "list box" se lanza el evento asociado:

```
function SequencesName_ListBoxValueChanged(app, event)
```

   En este punto se realizan un conjunto de acciones:
   - Se lee el fichero de "datos_secuencia.m" mediante:

```
function cargar_fichero_mat(app)
```

   - Se cargan todos los tipos de imágenes disponibles llamando a la función:

```
function carga_tipos_imagenes(app)
```

   - Se obtienen todos los nombres de las imágenes de la secuencia al llamar a:

```
function crear_nombres_imagenes_simples(app)
```

     Estos nombres se componen de un índice y del modo de predicción empleado en cada imagen.
   - Se calcula la rejilla a emplear mediante la llamada de:

```
function crea_rejilla(app)
```

   - Se crea una imagen de luminancia 128 mediante la siguiente llamada:

```
function crea_imagen_luma128(app)
```

Esta imagen se utiliza como imagen de error para mostrarla cuando no exista algún tipo de imagen en la secuencia.

- Se crea la tabla de datos relacionada con la información global de la secuencia llamando a la función:

```
function crear_Tabla_Datos(app)
```

- Para inicializar las gráficas que dependen de la selección de una opción, se establece la primera opción posible y se fuerza el evento asociado a esos elementos para que se actualice la información.
- Se inicializan los dos índices utilizados para almacenar el número de fotograma elegido en cada sección independiente del apartado Visualizar.
- Se gestiona un valor de comprobación para distinguir si es la primera vez que se inicializa la aplicación o no, en cuyo caso si ya se tenía abierta y se cambia de secuencia se mantiene el tipo de imagen seleccionado.

3. Una vez se pasa a la pestaña de Visualización se tiene una función genérica llamada:

```
function mostrar_imagen(app, arg)
```

Esta función es de las más importantes del programa ya que es llamada por muchos eventos y se encarga, gracias a un argumento (1 para la zona superior y 2 para la zona inferior), de diferenciar la sección donde realizar los cambios y visualizaciones.
En ella se comprueban si las casillas están activadas o no, el tipo de imagen y es donde se gestionan las imágenes de error para aquellos frames de los que no se disponga información.
Esta función se llama:

- Cambios entre tipos de secuencias: se tiene en cuenta una sección superior (Up) y otra inferior (Down) y por cada sección se detectan un cambio de valor de manera independiente y lanza su propio evento, que, tras comparar el tipo seleccionado entre sus posibilidades, termina llamando a la función de `mostrar_imagen(app, arg)`:

```
function TypeUp_ListBoxValueChanged(app, event)
      ...
      mostrar_imagen(app, 1);
}
function TypeDown_ListBoxValueChanged(app, event){
       ...
       mostrar_imagen(app, 2);
}
```

- Cambios entre fotogramas: funciona de igual forma que el punto anterior, también se detectan los cambios de valores por cada sección y se lanzan sus eventos asociados llamando a `mostrar_imagen(app, arg)`:

```
function ImagesNamesUp_ListBoxValueChanged(app, event)
function ImagenesNamesDown_ListBoxValueChanged(app, event)
```

- Marcación de overlays: al seleccionar cualquier casilla se lanzarán los eventos asociados a las mismas, si en total en la pestaña de visualización hay un total de 5 opciones de overlay por cada sección se tendrán 10 eventos de los cuales la funcionalidad de cada opción se repite según la sección en la que esté.
    - Hide Luma:

```
function HideLumaUp_CheckBoxValueChanged(app, event)
function HideLumaDown_CheckBoxValueChanged(app, event)
```

    - Modos de MB:

```
function ModosMBUp_CheckBoxValueChanged(app, event)
function ModosMBDown_CheckBoxValueChanged(app, event)
```

    - Rejilla:

```
function RejillaUp_CheckBoxValueChanged(app, event)
function RejillaDown_CheckBoxValueChanged(app, event)
```

    - Vectores de movimiento:

```
function VectorForwardUp_CheckBoxValueChanged(app, event)
function VectorBackwardUp_CheckBoxValueChanged(app, event)
```

```
function VectorForwardDown_CheckBoxValueChanged(app, event)
function VectorBackwardDown_CheckBoxValueChanged(app, event)
```

  Tanto para la rejilla como para los vectores se permite el cambio de color a gusto del usuario entre unas opciones establecidas en un desplegable. Esto funciona de manera análoga a lo anterior donde por cada sección se lanza un evento y se actualiza con el color elegido. Al haber 3 opciones por cada parte se tienen un total de 6 eventos asociados a sus funciones.

4. En la reproducción de imágenes se crean dos timer, uno para cada parte. Esto permite una reproducción asíncrona según la voluntad del usuario.
5. En la pestaña de gráficas y tablas, todas se inicializan al principio mediante un forzado donde se selecciona un valor inicial y se fuerzan sus eventos.
    - En caso de que se modifiquen cualquiera de los grupos de "radio buttons" se llaman a sus eventos asociados encargados de actualizar las gráficas:

```
function InfoTotal_ButtonGroupSelectionChanged(app, event)
function Opciones_ButtonGroupSelectionChanged(app, event)
```

  Cada función actualiza según la opción elegida la gráfica a la que se refiere.

    - En el caso de que se modifique la selección de cualquier fotograma se crea un flujo de trabajo en el que el cambio de valor del "list box" lanza un evento que llama a:

```
function Imagenes_ListBoxValueChanged(app, event)
```

Esta función a su vez actualiza la tabla de información asociada a cada imagen, el diagrama de sectores y la tabla que complementa con datos numéricos al diagrama de sectores:

```
function crear_Tabla_Calidades(app)
function grafica_circular(app, index)
function crear_Tabla_graficacircular(app)
```

En la Figura 22 se recogen el conjunto de funciones y callbacks empleados en la aplicación, entre los cuales se encuentran las funciones que ya han sido explicadas en los apartados anteriores.

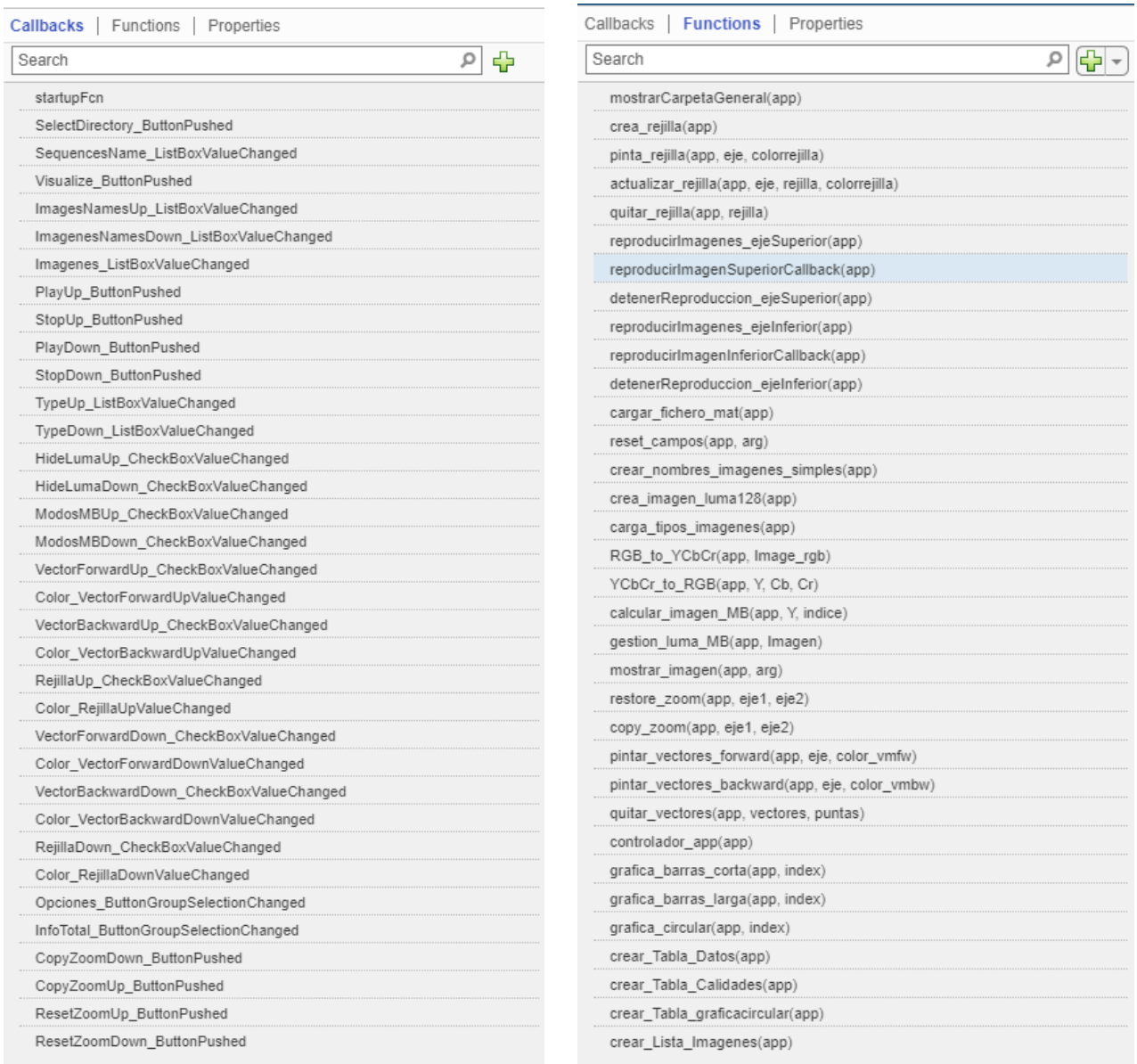

Figura 21. Callbacks y funciones creadas en la aplicación.

4.7 Elecciones relevantes

Durante el desarrollo de la aplicación se descubren aspectos relevantes que influyen en la toma de decisiones. A continuación, se recogen varios aspectos destacables a tener en cuenta para próximos desarrolladores.

- La elección de "list box" permite listar elementos de los que se desconoce su longitud total y ofrece una barra de "scroll" en caso de que el número de elementos supere el rango adaptado para ese "list box". Se prefiere usar esta alternativa antes que la generación de "Toggle Button Group" ya que, aunque se pueda generar un botón por cada elemento de la lista, estos requieren de mucho espacio y pueden salirse de los rangos de la aplicación.
- Diferenciación entre "Check Box" y "Radio Button Group", el primero posibilita la selección de varios "Check Boxes", es decir son opciones compatibles, mientras que en los grupos de "Radio Button" solo se puede seleccionar uno de entre todas las opciones mostradas, es decir son opciones excluyentes. Es importante tener en cuenta este funcionamiento clave para emplearlos de forma adecuada.

5. Resultados

5.1 Pruebas realizadas

Durante el desarrollo de la aplicación, la mayoría de las pruebas se realizan a medida que evolucionaba el proyecto, para asegurar un crecimiento controlado previendo la aparición de errores y permitiendo crear un código estructurado.

5.1.1 Lectura de imágenes y datos de codificación

Las secuencias de imágenes a visualizar en la herramienta son obtenidas previamente tras la ejecución de unos scripts ya existentes de MATLAB. Estos scripts se utilizan en la asignatura de Tecnologías de Imagen y Video (TIV) y recogen los cálculos simplificados de una implementación de las operaciones llevadas a cabo en un codificador de video MPEG2 utilizado en las prácticas del laboratorio. Para cargar todas las imágenes y representarlas en la herramienta, se ha adaptado el código para mantener la topología de carpetas creadas tras la ejecución de los scripts. En la Figura 22, se tiene la jerarquía donde aparece una carpeta compuesta por subcarpetas para cada secuencia, que a su vez cada secuencia contiene los datos generales de la codificación y subcarpetas con imágenes de resultados intermedios de la codificación.

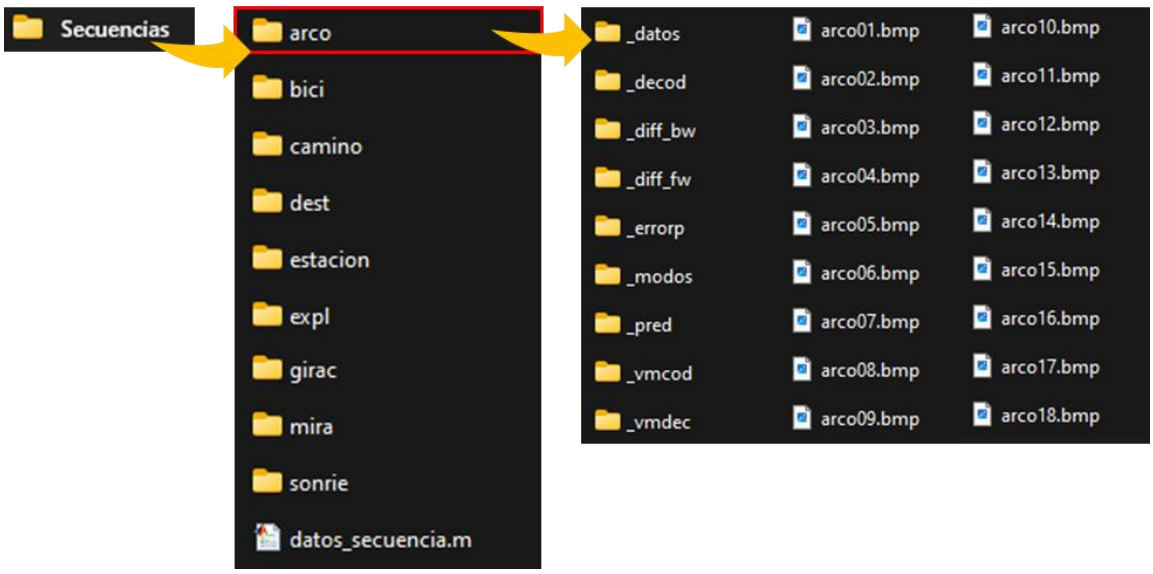

Figura 22. Topología de carpetas esperada.

En esta prueba se comprueba la correcta lectura de imágenes, asegurando que todas visualizan. Para determinar que la prueba ha resultado exitosa se deben llevar a cabo las siguientes acciones, obteniendo como resultado final el de la Figura 23 y Figura 24:

- Seleccionar un directorio donde se encuentre secuencias a visualizar.
- Comprobar que se listan todas secuencias a visualizar.
- Seleccionar una secuencia de prueba, por ejemplo “arco”.
- Pasar a la etapa de visualización y comprobar que las imágenes mostradas se corresponden con las esperadas

Para completar este apartado se comprueba que, al volver a cambiar de secuencia, los tipos de imagen elegidos se mantienen pero que el índice de imagen seleccionada se resetea.

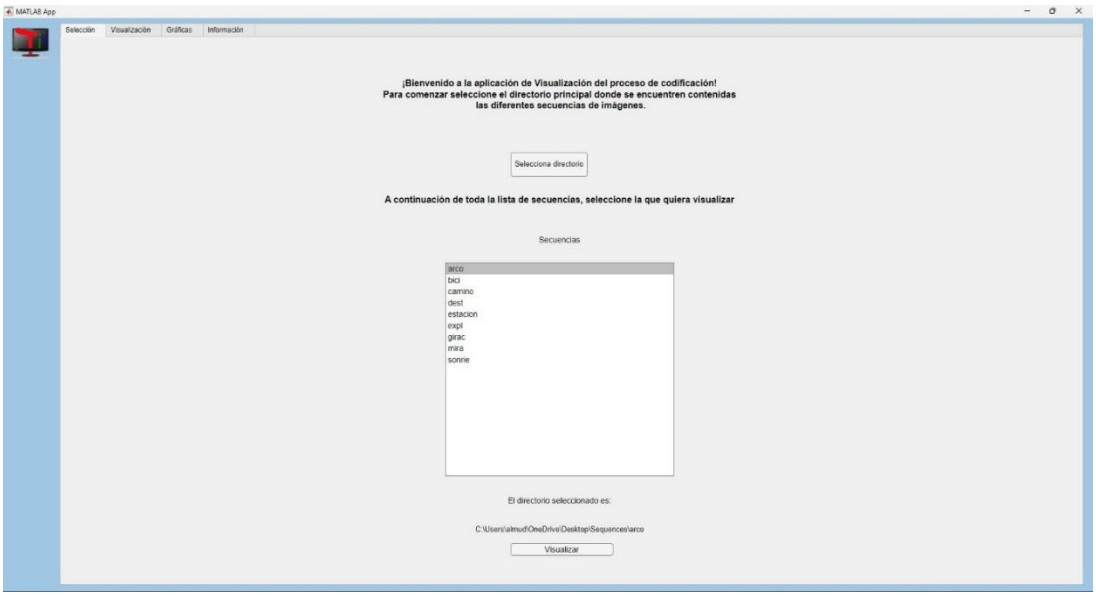

Figura 23. Resultado pestaña de selección.

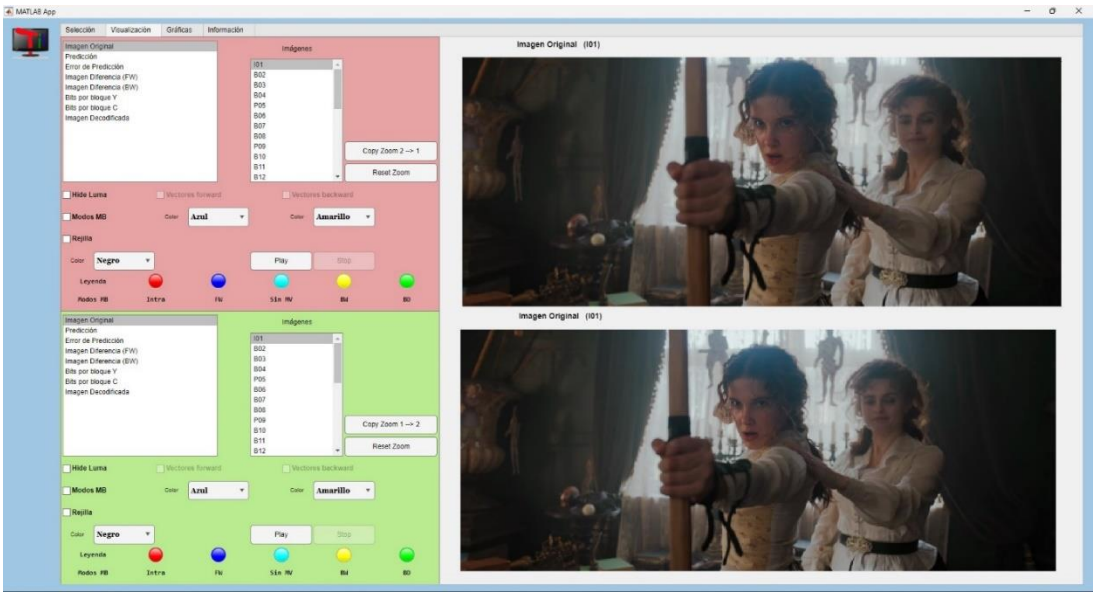

Figura 24. Resultado pestaña de visualización.

5.1.2 Superposición de overlays

Una vez comprobada la visualización de las imágenes, se continua con la gestión de overlays. Los overlays ofrecen información adicional representada en capas superpuestas sobre las imágenes. Se realizan una serie de pruebas para comprobar su funcionamiento.

- Hide luma: al marcar la casilla, las imágenes pierden su componente de luminancia y se muestran solo las cromas.
- Modos de MB: marcando esta casilla junto con la rejilla se comprueba que cada macrobloque recibe un color de los definidos.
- Vectores de movimiento: para probar esta funcionalidad se debe seleccionar una imagen tipo Intra y asegurarse de que ambas opciones salgan deshabilitadas, posteriormente con una imagen tipo P comprobar que solo se pueden marcar los vectores Forward. La última comprobación es asegurarse de que ambas casillas salgan habilitadas cuando se selecciona una imagen tipo B.

Para finalizar, se comprueba la coherencia entre datos visuales, esto se refiere a comprobar lo siguiente:

- Al pintar un MB de rojo (Intra), no se tiene ningún vector asociado a él.
- Al pintar un MB de azul oscuro (Forward), solo se le asocian vectores del mismo tipo.
- Al pintar un MB de cian (Sin MV), son bloques donde hay vectores de movimiento nulos, que no se representan.
- Al pintar un MB de amarillo (Backward), debe tener asociados vectores del mismo tipo.
- Al pintar un MB de verde (Bidireccional), deben aparecer dos vectores asociados al MB, el vector forward y backward.

5.1.3 Reproducción y Zoom

En caso de querer ver la progresión de una secuencia se añade un botón de reproducción para evitar que el usuario tenga que estar pulsando sobre todas las imágenes una por una. Por correspondencia se añade un botón de pausa para parar la reproducción mostrado en la Figura 25. En este caso se van a realizar las siguientes pruebas:

- Reproducir la secuencia de una sección, comprobar que avanza y pararla teniendo en cuenta que la imagen seleccionada es la última reproducida. Realizar lo mismo en la otra sección.
- Reproducir ambas secuencias a la vez en cada sección, partiendo de un índice de imagen diferente y comprobar que la reproducción se hace de forma independiente en cada zona.

Figura 25. Botones de Reproducción y pausa.

En los ejes de MATLAB se permite aplicar un zoom sobre una imagen. En caso de que se quiera comparar la zona seleccionada de una imagen con otra, si disponen de un zoom diferente, se habilita la opción de seleccionar automáticamente el zoom y replicarlo en la imagen que no tiene el mismo zoom. En la Figura 26 se muestra la opción de copiar el zoom aplicado en la imagen 2 a la imagen 1.

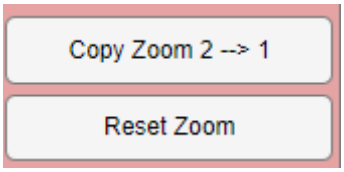

Figura 26. Botones para copiar Zoom y Resetearlo.

5.1.4 Tablas y gráficas

En este apartado se comprueban que los datos representados son coherentes con lo obtenido en el proceso de codificación y que los tamaños de cada elemento son los adecuados al mostrar toda la información sin cortar. Algunas comprobaciones fueron:

- En la gráfica que muestra el número de imágenes de cada tipo: I, P y B, se puede o bien calcular la suma total para comprobar que coincide con el número de imágenes de la secuencia o aplicar los conocimientos adquiridos sobre la codificación, donde se conoce que en una secuencia el tipo de imagen más recurrente es el tipo B, luego el tipo P y por último el tipo I.
- En la gráfica que muestra el número de bits empleados en media en cada tipo de imagen, solo se necesita conocer que en las imágenes Intra es donde más información se manda, por eso utilizará más bits, luego iría la imagen tipo P y finalmente la imagen tipo B.

El resultado esperado en ambas gráficas debe salir parecido al mostrado en la Figura 27.

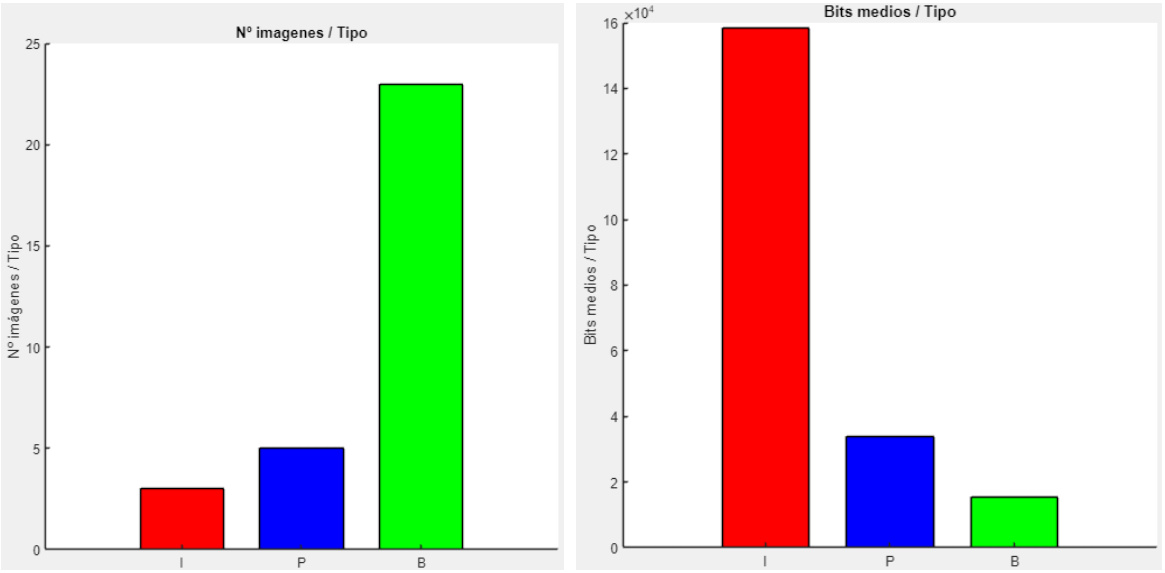

Figura 27. Izq: Grafica de nº de imágenes por tipo. Dcha: Gráfica de bits medios por tipo.

- Para la gráfica inferior donde se muestran muchos tipos de información, lo principal es comprobar que el número de barras y los modos de codificación que aparecen coincidan con el total de imágenes de la secuencia estudiada como es el caso de la Figura 28.

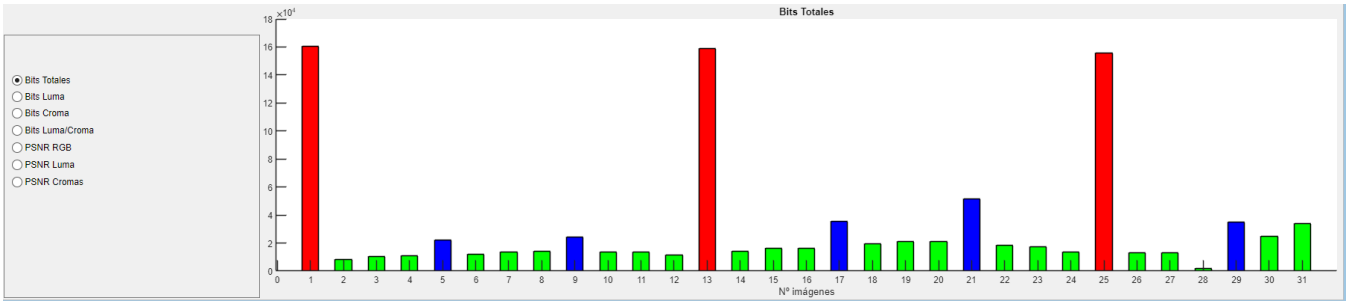

Figura 28. Gráfica multiseleccionable.

- En el sector circular se debe comprobar que los modos de macrobloque posibles en cada tipo de imagen son los únicos presentes y obtener algo similar a la Figura 29:

    - Si se elige una imagen tipo I, todo el gráfico debe ser rojo.
    - Si se elige una imagen tipo P, en el gráfico pueden aparecer los colores rojo, azul y cian.
    - Si se elige una imagen tipo B, en el gráfico pueden aparecer los colores rojo, azul, cian, amarillo y verde.

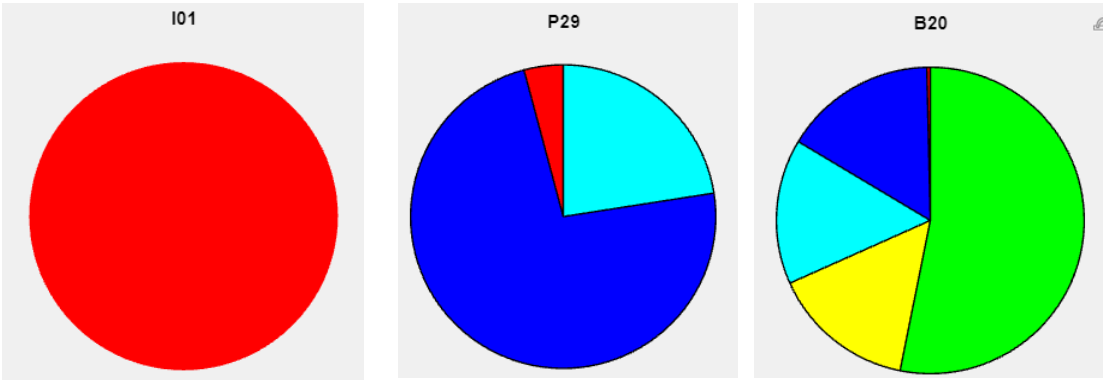

Figura 29. Diagrama de sectores por cada tipo de imagen. De izq. a dcha. Intra (I), Predicción (P) y Bidireccional (B).

5.1.5 Resolución en pantalla completa

Durante el desarrollo del software se ha trabajado y ejecutado la herramienta siempre a pantalla completa en una misma resolución y dimensión, sin embargo, aunque la aplicación todavía no ha sido testeada por el alumnado si se han hecho pruebas de resolución en otra pantalla diferente, la cual resultaba más restrictiva.

Al encontrar un segundo entorno más restrictivo, se decide adaptar la interfaz a esa visualización para permitir que los elementos establecidos sean legibles en la mayoría de las pantallas donde se ejecute y no se realicen cortes inesperados en texto o componente.

En la Figura 30 aparece a modo de comparativa la pestaña de Visualización ejecutada desde un portátil (imagen superior) y la ejecutada desde un monitor de sobremesa (imagen inferior). En estas imágenes se puede apreciar que las pruebas se hicieron sobre una versión intermedia a medida que se iba avanzando y que a partir de las modificaciones surgidas se ha llegado al resultado final.

En la vista del portátil, se aprecia que los títulos informativos donde se especifica el tipo de imagen y el número de secuencia aparecen cortados y casi no se ven, también ocurre lo mismo con la leyenda en la parte inferior.

Figura 30. Comparación de resolución: vista portátil (superior) y vista pantalla sobremesa (inferior).

5.2 Obtención del ejecutable

Como todo software, la aplicación debe tener un ejecutable que posibilite su utilización por el usuario final al que va dirigido, en este caso se tienen en cuenta varios factores previos a la obtención del ejecutable.

- El usuario no debe tener acceso al código fuente.
- Es posible que el usuario no disponga de una instalación de MATLAB.

MATLAB ofrece de un apartado de APPS donde se encuentra la aplicación: Application Compiler, que tras su instalación permite la compilación de código para generar los programas desarrollados con App Designer. En la Figura 31 se muestra en favoritos el símbolo del Application Compiler.

Figura 31. APPS: Application Compiler.

Una vez se dispone del Application Compiler, se permite compilar y compartir aplicaciones mediante 3 métodos diferentes recogidos en la Figura 32.

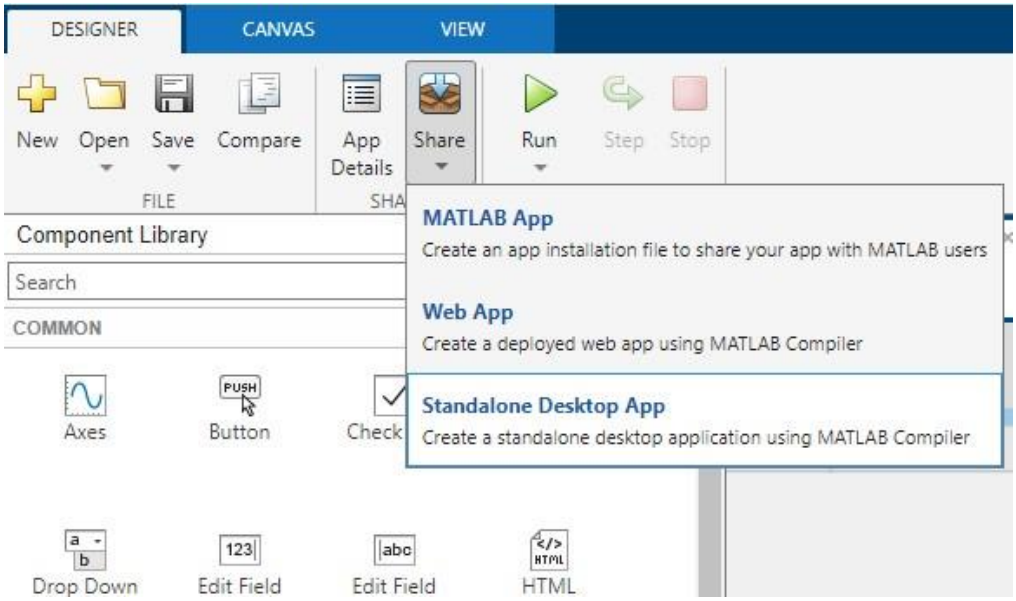

Figura 32. Métodos de compilación posibles.

La opción de MATLAB App se descarta puesto que requiere una instalación de MATLAB y, además, el usuario final debe disponer de un fichero necesario para la correcta interpretación de los datos, lo cual contradice al apartado en el que se establece que el usuario no debe tener conocimiento de los requerimientos propios de la aplicación.

La segunda opción no cumple con la idea general de desarrollar una aplicación de escritorio, y además requiere de un navegador y de conexión a internet para su funcionamiento, por lo que también se descarta esta posibilidad.

Finalmente, el método empleado para la obtención del ejecutable es: Standalone Desktop App, ya que permite generar un ejecutable independiente de Matlab y sin acceso al código fuente. Ofrece dos métodos diferentes de generación mostrados en la Figura 33 :

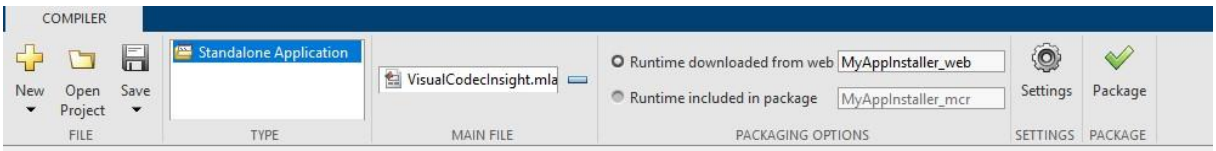

Figura 33. Diferentes tipos de runtime.

- Runtime downloaded from web: donde las bibliotecas necesarias que utiliza MATLAB para ejecutar el código se descargan de forma automática desde internet cuando el código se ejecuta por primera vez o cada cuanto sea necesario. En este caso al instalar la aplicación se detectan los componentes específicos necesarios y se descargan en tiempo real para asegurar que el código se ejecute correctamente.
- Runtime included in package: donde las bibliotecas o componentes software necesarios para la ejecución del código se incluyen directamente en el paquete de instalación sin necesidad de disponer de MATLAB. Esto significa que todas las dependencias requeridas para ejecutar el código se encuentran empaquetadas y distribuidas junto con el software en sí mismo. No requiere de conexión a Internet para descargar componentes adicionales durante la ejecución del código.

A la hora de decidir entre estas dos opciones se tienen en cuenta varios factores como: la disponibilidad de conectividad a Internet, el tamaño del paquete de instalación, la frecuencia de actualización de las bibliotecas y las preferencias del desarrollador en cuanto a la gestión de dependencias.

Incluir todas las dependencias en el paquete de instalación permite al usuario una instalación más sencilla y si además no se prevén actualizaciones frecuentes de las bibliotecas esta opción es bastante viable. Por otro lado, descargar las dependencias según sea necesario puede ser más eficiente en términos de espacio en disco (ya que la aplicación pesa menos) y ancho de banda, pero puede requerir una conexión a Internet estable y requerir más tiempo en la primera ejecución del código.

Para permitir una mayor flexibilidad debido a que el software no se plantea que realice actualizaciones constantes, se van a realizar los dos empaquetados diferentes. Así se adapta según las características del usuario. En la Figura 34 se muestran los datos e información añadidos a la hora de crear el ejecutable.

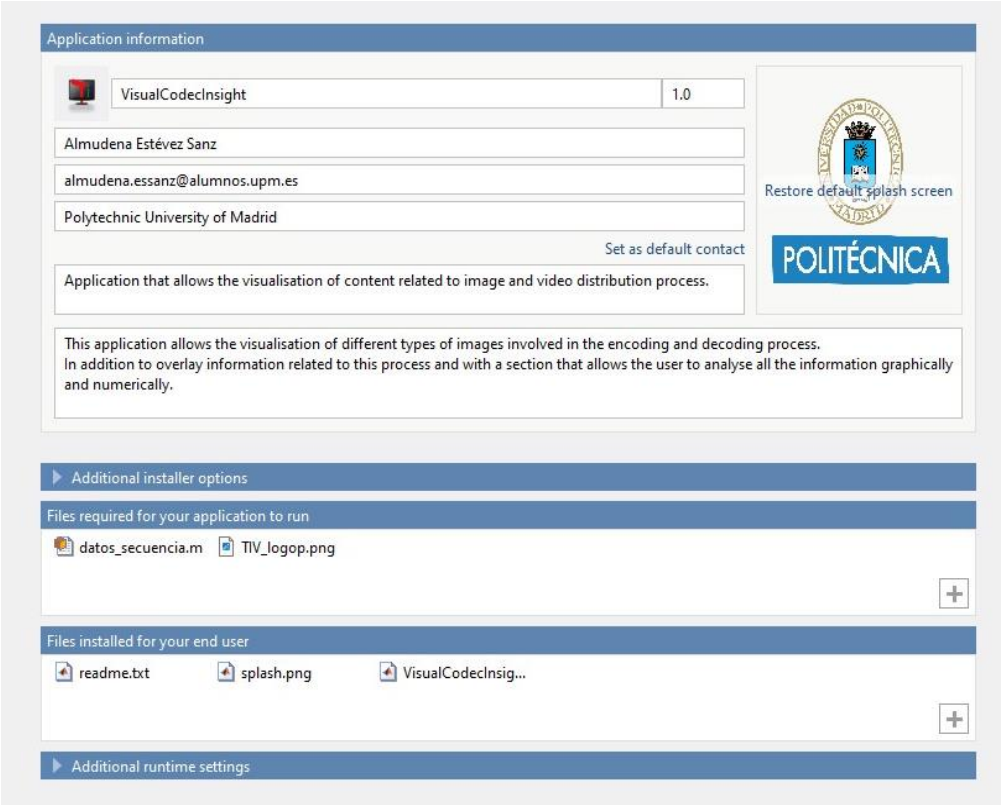

Figura 34. Datos previos a la generación del ejecutable.

Una vez se empaquetan los ejecutables se genera una carpeta general con el nombre de la aplicación y tres carpetas en su interior, esto se aprecia en la Figura 35. La carpeta for_testing permite al desarrollador realizar pruebas para comprobar que la compilación ha resultado exitosa y que la aplicación funciona correctamente. En la carpeta for_redistribution_files_only se tiene el ejecutable que permite su apertura sin tener que instalarlo y finalmente la carpeta que interesa es la de for_redistribution, donde se tienen los dos ejecutables, tanto el que descarga las bibliotecas mediante la web como el que ya las tiene empaquetadas en su interior

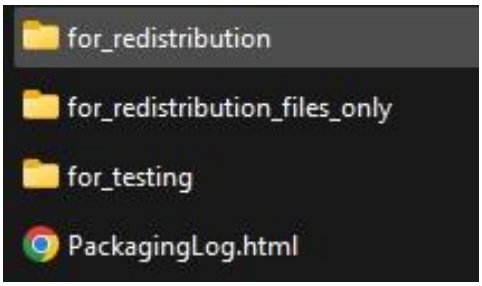

Figura 35. Ficheros obtenidos.

6. Presupuesto

Debido a la naturaleza del proyecto se procede a realizar un presupuesto correspondiente con un proyecto de diseño y desarrollo de software, donde se recojan todos los recursos utilizados para la realización de este proyecto.

6.1 Costes personales

Al ser un trabajo de fin de grado individual se ha llevado a cabo por un ingeniero de sonido e imagen junior con la ayuda de su tutor. Se estima que el tiempo total empleado ha sido de un total de 350 horas de trabajo total descompuesto en diferentes actividades acorde con la Tabla 3.

Tabla 3. Desglose de horas dedicadas.

Actividades realizadas	N.º de horas dedicadas
Reuniones, búsqueda de documentación y videos	30 horas
Desarrollo de software	260 horas
Redacción de memoria	60 horas
TOTAL	350 horas

Tras una búsqueda activa en la red [10] se estima que el sueldo medio de un ingeniero junior en España ronda los 22.000 € para una jornada completa donde se trabajan un total de 40 horas semanales y 1820 horas anuales aproximadamente. Por tanto, realizando una cuenta rápida (*5*) se llega a la conclusión de que el coste personal ronda los 4.231 €.

$$\frac{22.000\ €}{1.820\ horas} \approx 12{,}09\ {}^{€}/_{hora} \rightarrow 12{,}09\ € \cdot 350\ horas = 4.231{,}5€ \qquad (5)$$

6.2 Coste licencias

El software empleado en este proyecto es MATLAB, este programa requiere de la disposición de una licencia para poder usar dicha herramienta. En este caso, por pertenecer a la Universidad Politécnica de Madrid (UPM) se cuenta con la ventaja de una licencia gratuita anual para todos sus alumnos, lo cual reduce costes. En la Tabla 4 se recogen los precios para dos licencias diferentes.

Tabla 4. Precio de licencias de MATLAB.

Tipos de licencias	Precio anual de licencia
MATLAB Academic	262 €
MATLAB Student	69 €
MATLAB (UPM license)	GRATIS

6.3 Coste hardware

El hardware empleado en el desarrollo de la herramienta es un ordenador portátil y un ratón. El portátil que se vaya a utilizar debe superar unas especificaciones mínimas para poder ejecutar adecuadamente el programa:

- Procesador: Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz
- CPU 1.80GHz
- 4 núcleos
- Disco SSD
- 16 GB de RAM
- Windows 11
- NVIDIA GeForce MX250

Un portátil de estas características puede rondar un precio de 700€, sin embargo, se pueden abaratar costos adquiriendo solamente el hardware que cumple con las especificaciones o utilizando ordenadores públicos en bibliotecas.

Los componentes electrónicos tienen un coeficiente lineal máximo del 20% y se pueden amortizar por un máximo de 10 años [11]. En este caso, amortizando el material nos daría lo siguiente: 700€·20% = 140€ anuales. Pagando 140€ de cuota anual, la amortización se irá a un total de 5 años.

6.4 Coste Total

Finalmente, en la Tabla 5 se recogen dos tipos de presupuestos, el genérico que se necesitaría para la replicación del proyecto sin disponer de los medios y el real empleado en este proyecto.

Tabla 5. Costes generales.

Descripción	Unidades	Coste Genérico	Coste Real
Personal			
Ingeniero Junior	x1	4.231€	4.231€
Licencias Software			
MATLAB	x1	262€	0€
Hardware			
Ordenador Portátil	x1	700€	0€
	Total	5.193€	4.231

7. Impacto del proyecto

Este proyecto no solo tiene el potencial de mejorar la educación en ingeniería y tecnología para los estudiantes, sino que también contribuye significativamente con varios Objetivos de Desarrollo Sostenible [12], especialmente aquellos relacionados con la educación de calidad, la innovación tecnológica y la reducción de la desigualdad.

Esta aplicación puede tener un impacto duradero ya que influye sobre la preparación de las próximas generaciones de profesionales de la tecnología y permite crear un entorno de aprendizaje más accesible.

Respecto a los ODS anteriormente mencionados, a continuación, se detallan de forma más profunda las partes con las que este proyecto está relacionado:

- ODS 4: Educación de calidad
    - Mejorar las habilidades técnicas.
      Al tener una finalidad educativa, los estudiantes desarrollan habilidades prácticas sobre el campo de la codificación de video y el procesamiento de imágenes.
    - Facilitación del aprendizaje visual.
      La posibilidad de visualizar imágenes, vectores y modos de macrobloque mejora la comprensión conceptual y técnica, haciendo que el aprendizaje sea aún más efectivo.
- ODS 9: Industria, Innovación e Infraestructura
    - Innovación en la enseñanza.
      La aplicación innova en la forma de enseñar la codificación de video, empleando métodos interactivos y visuales que pueden ser adoptados por otras instituciones educativas.
    - Colaboración y desarrollo.
      La herramienta dispone de líneas futuras de trabajo, por tanto, puede servir como una plataforma para colaboraciones futuras entre educadores, estudiantes e investigadores, promoviendo un ecosistema de aprendizaje y desarrollo continuo.
- ODS 10: Reducción de las Desigualdades
    - Accesibilidad y equidad.
      Al ser una herramienta educativa accesible, se ayuda a reducir las barreras de entrada para la educación técnica y tecnológica, fomentando la equidad educativa.
    - Fomento del interés en STEM.
      Esta herramienta puede inspirar a más estudiantes a interesarse por carreras de ciencia, tecnología, ingeniería y matemáticas (STEM), áreas cruciales para el desarrollo económico y social.

8. Conclusiones

El desarrollo de la aplicación educativa, VisualCodecInsight, basada en la visualización de imágenes que componen el proceso de codificación de vídeo representa una actualización en la enseñanza de la compresión de vídeo digital. Esta herramienta proporciona a los estudiantes una comprensión profunda y visualmente intuitiva de los complejos procesos involucrados en la codificación de vídeo, como la división en macrobloques, los procesos de estimación y compensación de movimiento o los tipos de predicciones, entre otros.

8.1 Conclusiones finales

La idea principal del proyecto es crear una solución tecnológica que permita reforzar la comprensión de un tema complejo como es la codificación de video. Para ello se buscó un escenario donde se pudieran mostrar las imágenes, incluyendo los procesos fundamentales de la compresión de vídeo MPEG2 en una aplicación de escritorio. Con el desarrollo de dicha aplicación, los alumnos pueden reforzar y afianzar los conocimientos adquiridos en las clases teórica.

Gracias al carácter visual e interactivo proporcionado por la herramienta al mostrar imágenes, tablas y gráficas, se permite al usuario comparar resultados, interpretarlos y razonar sobre el proceso llevado a cabo.

La aplicación se basa en un estándar de codificación MPEG2 y aunque en la actualidad hay más tipos de estándares, MPEG2 incluye los conceptos fundamentales incluidos en todos los codificadores de vídeo predictivo transformacionales más modernos. Es posible que a nivel de futuro se implemente otro tipo de estándar para tenerlo más actualizado, o que se vaya actualizando la aplicación en general puesto a que todavía hay varios puntos que se pueden mejorar.

8.2 Líneas futuras de trabajo

Una vez finalizado el proyecto actual y evaluados los objetivos conseguidos y los que no han podido llevarse a cabo, aparecen una serie de tareas para mejorar y aumentar el desarrollo de la aplicación en posibles líneas de trabajo futuras.

- Control de errores: a medida que se vaya usando la aplicación por los estudiantes es posible que aparezcan nuevos errores que no se hayan valorado hasta la fecha, por este motivo se debe someter a la aplicación a un proceso de corrección de errores, bugs e incidencias.
- Optimización del código: según se avance en el proceso de codificación se podrán añadir nuevas funcionalidades que permitan la simplificación de código y el aumento de velocidad.
- Mejora de la interfaz gráfica: actualmente se compone de una interfaz simple y funcional, en un futuro puede hacerse más llamativa para el usuario, implementando

además un reajuste de tamaño para poder trabajar en resoluciones diferentes a la pantalla completa.

- Mejora de la eficiencia del procesamiento de imágenes: actualmente las imágenes se cargan al seleccionar la secuencia, en un futuro igual se puede encontrar una solución mejor que no haga esperar al usuario.
- Añadir análisis detallado por cada macrobloque: por cada macrobloque se podría poner información propia del mismo, esto podría hacerse mediante un cuadro de texto que apareciera al superponer o pulsar sobre el macrobloque.
- Implementar la decodificación parcial: sería un paso previo donde se utilice solo información necesaria para poder generar el resto de información de la que no se disponga.
- Implementar la codificación de las imágenes: sería el paso final, donde la aplicación sería autocontenida y se le podría pasar una secuencia y obtener directamente toda la información e imágenes. Esto pude ocupar mucho espacio y tardar tiempo en cargar, debido al gran número de operaciones que debe realizar.

9. Referencias

[1] C. Pérez Vega, "Compresión de Vídeo". [En línea]. Disponible en: https://personales.unican.es/perezvr/pdf/compresion%20de%20video.pdf [Accedido: 2-marzo-2023].

[2] "Formatos de video más utilizados". *WeAreVideoContent*. [En línea]. Disponible en: https://video.wearecontent.com/blog/video/formatos-de-video-mas-comunes/ [Accedido: 9-marzo-2023].

[3] "GUÍA DE APRENDIZAJE: Tecnologías De Imagen Y Video". Universidad Politécnica de Madrid. [En línea]. Disponible en: https://www.upm.es/comun_gauss/publico/guias/2023-24/1S/GA_59SO_595000128_1S_2023-24.pdf [Accedido: 15-junio-2024].

[4] "Cámaras de fotos | Guías Prácticas". Guías Prácticas. [En línea]. Disponible en: https://www.guiaspracticas.com/camaras-de-fotos [Accedido: 15-junio-2024].

[5] *RGB to YCbCr conversion - Wikipedia*. [Imagen]. Disponible en: https://en.wikipedia.org/wiki/File:CCD.png [Accedido: 15-junio-2024.].

[6] Colaboradores de los proyectos Wikimedia. "Patrón de muaré - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Patrón_de_muaré [Accedido: 15-junio-2024].

[7] "Los bordes de sierra no cortan." Of Coins And Life. [En línea]. Disponible en: https://ofcoinsandlife.wordpress.com/2014/03/11/los-bordes-de-sierra-no-cortan/ [Accedido: 15-junio-2024].

[8] "ELO-330 Algoritmos de compresión de Video." Departamento de Electrónica. [En línea]. Disponible en: http://profesores.elo.utfsm.cl/~agv/elo330/2s06/projects/EspinozaAlarcon/ELO-330%20Algoritmos%20de%20compresion%20de%20Video..html [Accedido: 15-junio-2024].

[9] "Transparencias Tema 2" apuntes de clase, Departamento de Ingeniería Audiovisual y Comunicaciones, Universidad Politécnica de Madrid, primavera 2023.

[10] "Sueldos de Ingeniero Junior". Glassdoor. [En línea]. Disponible en: https://www.glassdoor.es/Sueldos/ingeniero-junior-sueldo-SRCH_KO0,16.htm [Accedido: 15-junio-2024].

[11] "Agencia Tributaria: 1. Por coeficientes de amortización lineal." [En línea]. Disponible en: https://sede.agenciatributaria.gob.es/Sede/ayuda/manuales-videos-folletos/manuales-practicos/irpf-2020/capitulo-7-rendimientos-actividades-economicas-directa/fase-1-determinacion-rendimiento-neto/amortizaciones-dotaciones-ejercicio-fiscalmente-deducibles/requisitos-generales/coeficientes-amortizacion-lineal.html [Accedido: 15-junio-2024].

[12] "Objetivos de Desarrollo Sostenible". Naciones Unidas. Disponible en: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/ [Accedido: 15-junio-2024].

10. Bibliografía

- Colaboradores de los proyectos Wikimedia. "Teorema de muestreo de Nyquist-Shannon - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Teorema_de_muestreo_de_Nyquist-Shannon [Accedido: 15-junio-2024].

- "Muestreo". Universitat de València. [En línea]. Disponible en: https://www.uv.es/masefor/PAGINAS/muestreo.html [Accedido: 15-junio-2024].

- "Qué es 4:4:4, 4:2:2 y 4:2:0 o color subsampling". Profesional Review. [En línea]. Disponible en: https://www.profesionalreview.com/2019/11/09/que-es-444-422-y-420-subsampling/#4_2_0 [Accedido: 15-junio-2024].

- "Cuantificación y Codificación". Departamento de Teoría de la Señal y Comunicaciones - UC3M. [En línea]. Disponible en: https://www.tsc.uc3m.es/~hmolina/wp-content/uploads/2010/02/lsyc_ittst_p6.pdf [Accedido: 15-junio-2024].

- Sergio A. Castaño Giraldo. *Cómo crear un programa EJECUTABLE con Matlab [.EXE] #039*. (11 de agosto de 2020). [Video en línea]. Disponible en: https://www.youtube.com/watch?v=RrRlVqBE8lw [Accedido: 15-junio-2024].

- "MATLAB". MathWorks - Creadores de MATLAB y Simulink - MATLAB y Simulink - MATLAB & Simulink. [En línea]. Disponible: https://es.mathworks.com/help/matlab/ [Accedido: 15-junio-2024].

ANEXO I: Manual de usuario

A continuación, se resumen brevemente los prerrequisitos necesarios para la ejecución de la herramienta, acompañado con una guía de uso explicativa. La finalidad de este apartado es dar a conocer a cualquier usuario las opciones que ofrece la aplicación y como utilizarla de forma adecuada.

A.1 Instalación del ejecutable.

En la Figura 36 se representa un diagrama que muestra los aspectos relevantes a la hora de instalar la herramienta: VisualCodecInsight. Esta herramienta dispone de varias posibilidades de ejecución e instalación del programa según el entorno del que disponga el usuario. Si el usuario no dispone de MATLAB, se permite compilar el proyecto para obtener una aplicación de escritorio independiente. La obtención del ejecutable permite que a la hora de la instalación de la herramienta se realice descarga previa de las librerías necesarias, las cuales puede ser de dos tipos, mediante web o por paquetes.

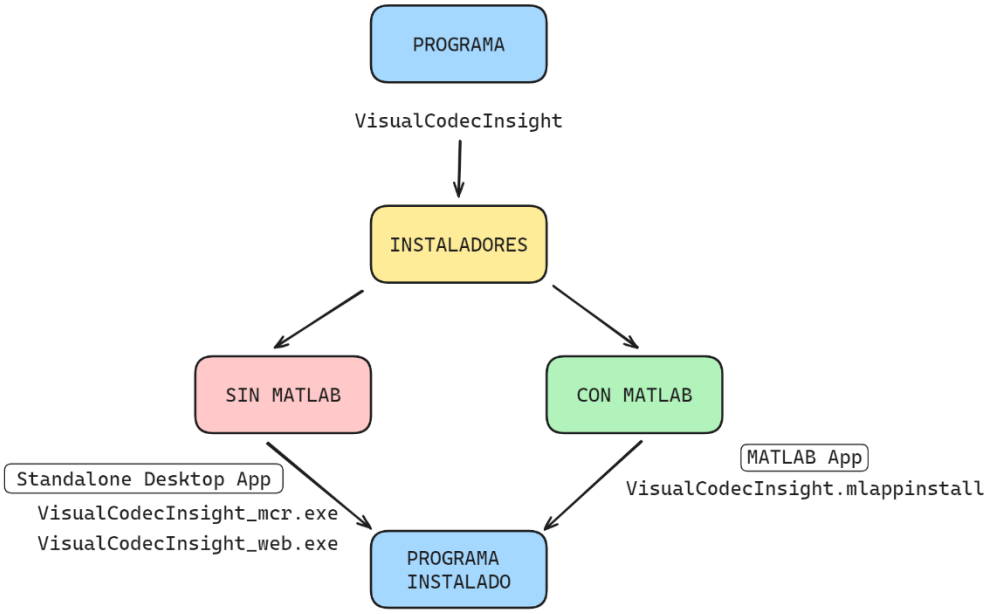

Figura 36. Posibilidades de instalación de VisualCodecInsight (VCI).

Finalmente, de entre las posibilidades de generación del ejecutable se escoge aquella que no requiere de disponer de MATLAB. Además de que permite a la hora de la instalación, una adaptación acorde con las características del usuario. Si no se dispone de internet para descargar las librerías en vez de instalar VisualCodecInsight_web.exe se instala el ejecutable VisualCodecInsight_mcr.exe que ya lleva autocontenidas todas ellas.

A.2 Estructura de recursos.

Para una correcta visualización de imágenes, el usuario debe asegurarse de que el directorio principal seleccionado siga la siguiente jerarquía de carpetas de la Figura 37 donde aparece una carpeta compuesta por subcarpetas para cada secuencia, que a su vez cada secuencia contiene los datos generales de la codificación y subcarpetas con imágenes de resultados intermedios de la codificación. Dentro de cada carpeta de secuencias, a parte de la información en modo de imagen que se ha obtenido también debe aparecer un fichero de datos_secuencia.m donde se almacena el resto de información representada en tablas y gráficas.

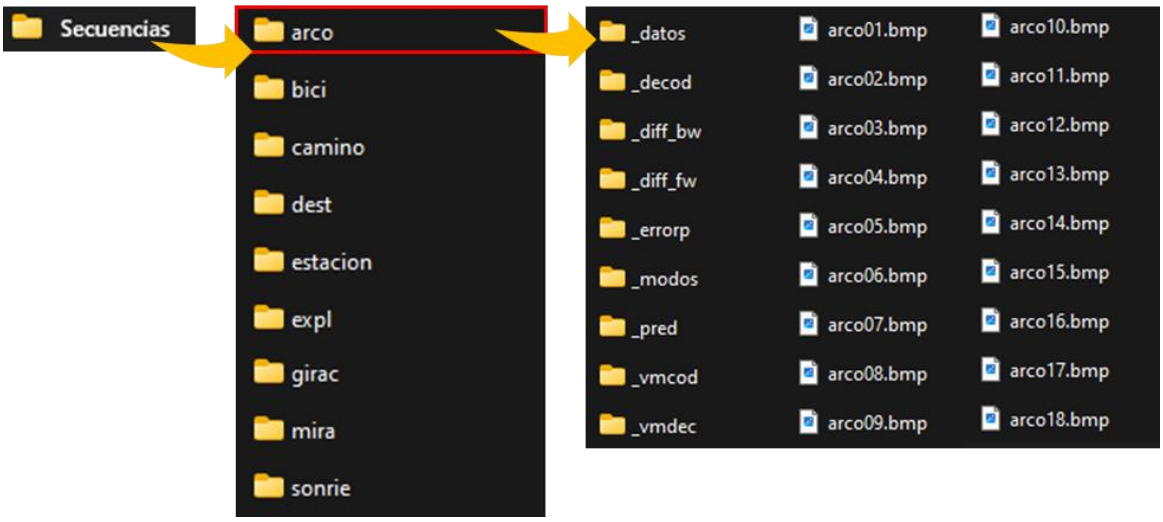

Figura 37. Jerarquía de carpetas para las secuencias.

A.3 Incorporación de imágenes.

Una vez se ejecute el VCI, aparece por defecto la primera pestaña, donde se le da al usuario una pequeña bienvenida junto con las instrucciones para seleccionar el directorio donde el usuario dispone del conjunto de secuencias agrupadas en diferentes carpetas. La aplicación detecta toda la información posible mediante un fichero denominado “datos_secuencia.m”, dentro de la carpeta general de cada secuencia. Tras seleccionar la carpeta raíz, se listan todas las carpetas de las diferentes secuencias disponibles. A partir de este paso solo es necesario seleccionar una de las secuencias para que se carguen sus imágenes y se habilite el botón “Visualizar” como se muestra en la Figura 38.

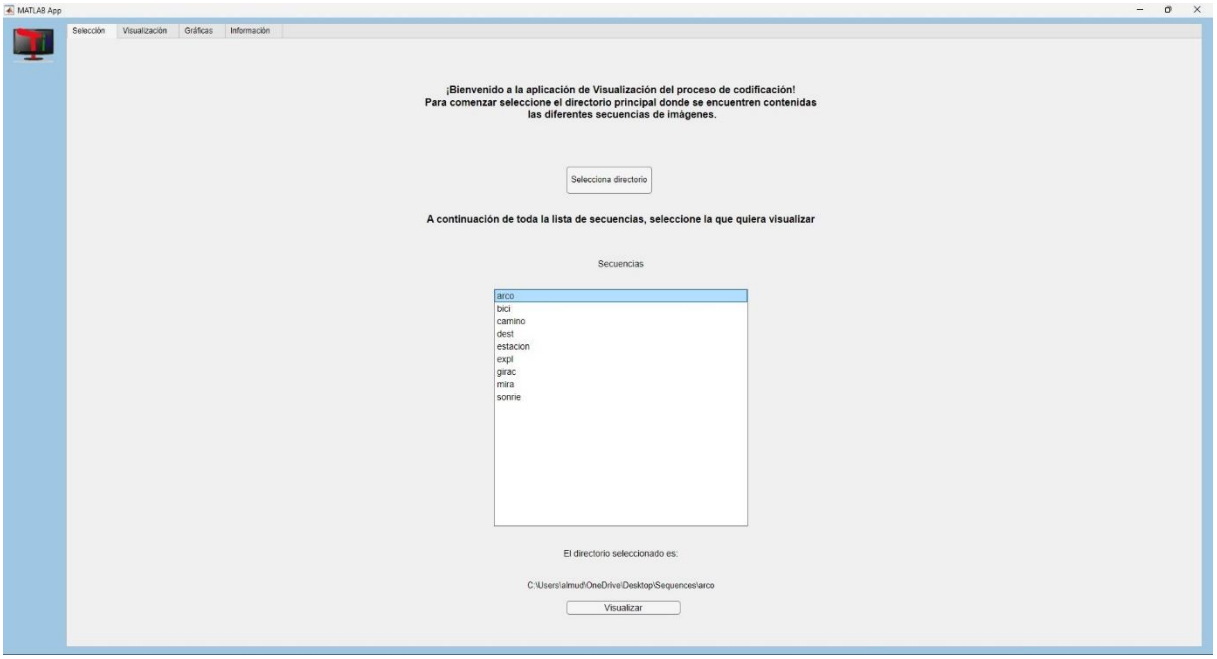

Figura 38. Pestaña principal para seleccionar el directorio de trabajo.

A.4 Pestaña de Visualización

Una vez seleccionada una secuencia y pulsado el botón, el programa cambia a la pestaña de "Visualización". En esta pestaña destacan, gracias a los colores, dos secciones (una superior y otra inferior como se muestra en la Figura 39) que permiten la visualización de dos imágenes de forma simultánea. Esto es muy útil para permitir al usuario la comparación de diferentes "frames" de la secuencia.

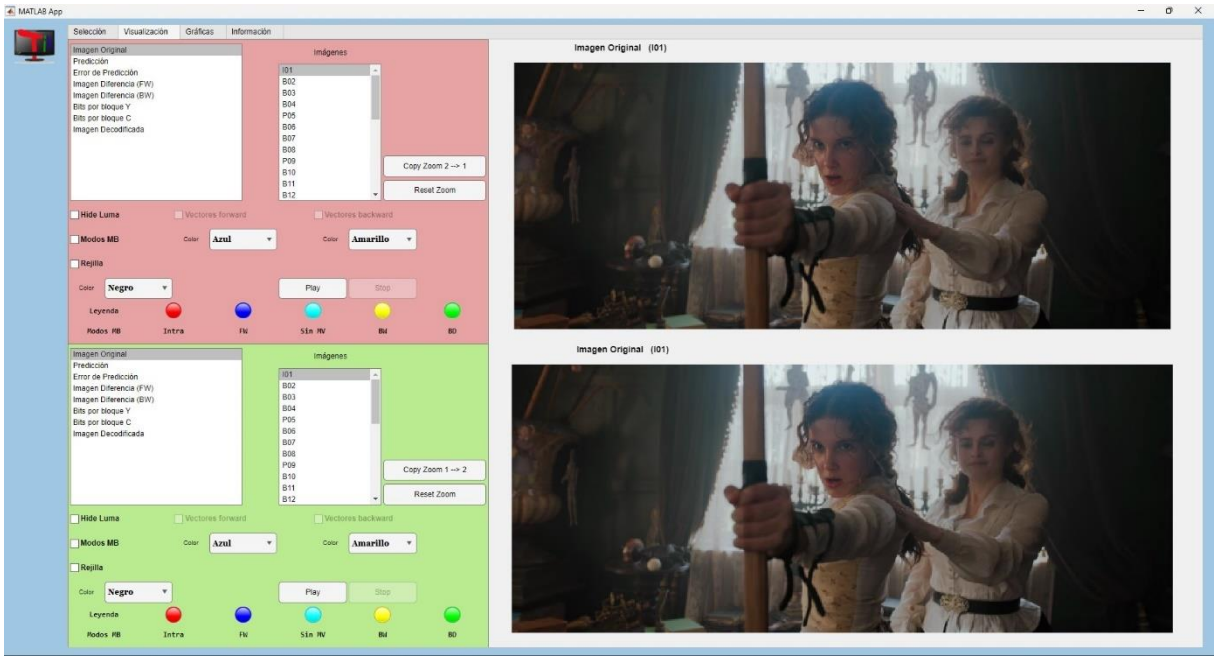

Figura 39. Pestaña de Visualización con sus dos secciones.

Las posibilidades de visualización ofrecidas en esta pestaña son iguales en ambas secciones, sin embargo, funcionan de forma independiente entre sí. En la Figura 40 se muestra el panel de opciones que funciona de forma análoga para ambas secciones y que se explica a continuación.

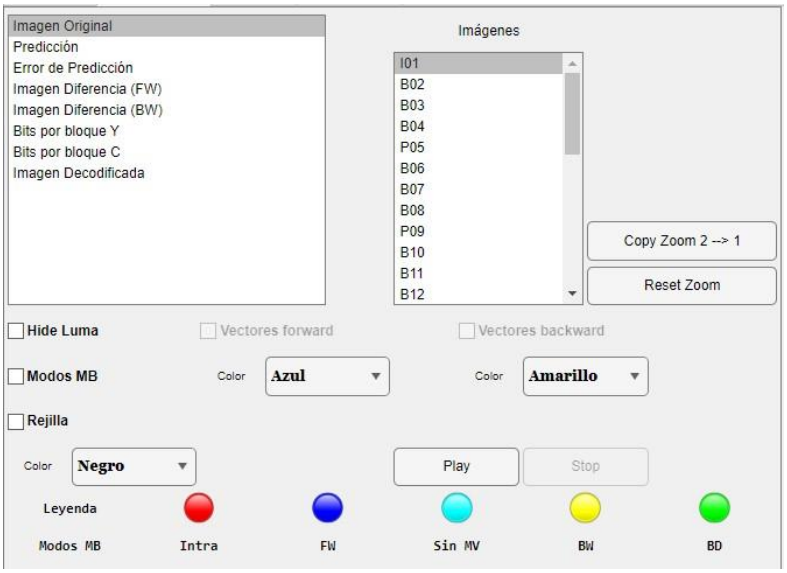

Figura 40. Opciones de visualización.

El primer "list box" de la esquina izquierda superior permite la elección del tipo de imagen a visualizar de entre las que se puede elegir entre:

- Imagen Original: es una imagen o cuadro de vídeo extraído de la secuencia de video que se va a codificar.
- Imagen de Predicción: es el resultado de la compensación de movimiento cuando se trata de una imagen Inter.
- Error de Predicción: es la imagen de error obtenida al restar la imagen original y la imagen de predicción.
- Imagen Diferencia Forward (FW): es la imagen diferencia entre la imagen actual y la referencia pasada. Representa el cambio entre ambas.
- Imagen Diferencia Backward (BW): es la diferencia entre la imagen actual y la referencia futura.
- Bits por bloque Y: es la representación gráfica de la cantidad de información necesaria para codificar cada bloque de la imagen de Luminancia.
- Bits por bloque C: es la representación gráfica de la cantidad de información necesaria para codificar cada bloque de la imagen de Crominancia.
- Imagen Decodificada: imagen resultado obtenida en el receptor tras la decodificación.

Posteriormente en segundo "list box" de la derecha se listan todos los "frames" del tipo de imagen seleccionado que componen la secuencia, acompañados con el modo de predicción que hayan sufrido.

En la parte de la derecha aparecen dos botones de gran utilidad:

- Copy Zoom X → Y: este botón se encarga de aplicar el mismo zoom que tiene una imagen en la otra.
- Reset Zoom: es un botón que resetea el zoom aplicado y devuelve a la imagen su resolución inicial.

Por último, en la parte inferior se encuentran una serie de "check boxes", los cuales se explican a continuación. Algunas de estas casillas están acompañadas por desplegables, que permiten cambiar el color de los trazos. se explican a continuación:

- Hide luma: elimina la aportación de la Y a la imagen que se esté visualizando. Esto permite visualizar la información aportada por las cromas y distinguir mejor los modos de macrobloque cuyos colores pueden falsearse por la información de brillo.
- Modos MB: se pinta por cada MB el color del modo aplicado sobre él. Los modos pueden ser de los siguientes tipos: Intra (pintado de rojo), Forward (pintado de azul), Backward (pintado de amarillo), Bidireccional (pintado de verde) y sin vector de movimiento (pintado de cian).
- Rejilla: crea una cuadrícula de división para segmentar los MBs que conforman la imagen. Además, se ofrece la posibilidad de cambiar el color con el que se dibuja para que se distinga mejor según que secuencias.
- Vectores forward (FW): vectores calculados sobre imágenes anteriores que ofrecen información de referencia pasadas. En este caso también se añade un desplegable que ofrece la posibilidad de cambiar el color para una mejor visualización sobre el fondo de la imagen.
- Vectores backward: vectores calculados sobre imágenes posteriores que ofrecen información de referencia futuras. También disponen de un desplegable para cambiar el color sobre el fondo de la imagen.

A.5 Tablas y Gráficas

En la pestaña de "Gráficas" es donde se agrupan todas las tablas, gráficas y diagramas encargados de presentar información para el usuario.

- Tabla Información de Secuencia: muestra datos de la Secuencia como: el alto y ancho de la imagen, las dimensiones medidas en MB, el número total de MB, el número de fotogramas de los que dispone la secuencia y los modos de codificación de cada imagen.
- Tabla Información de Imagen: muestra datos específicos para una imagen seleccionada junto con datos relevantes de variables MPEG2 empleadas en el proceso de codificación.
- Diagrama de sectores: muestra la cantidad de macrobloques de un mismo tipo o modo que aparecen en la imagen.

- Gráfica derecha: muestra el número de imágenes de cada tipo y además también presenta los bits medios empleado en cada tipo.
- Gráfica inferior: son comparaciones generales a nivel de secuencia donde en el eje de abscisas se enumeran todas las imágenes. Las opciones de gráficas son:
    - Bits totales: representa la cantidad de bits totales codificados en cada imagen de la secuencia.
    - Bits luma: representa la cantidad de bits de luminancia codificados en cada imagen de la secuencia.
    - Bits croma: representa la cantidad de bits de crominancia codificados en cada imagen de la secuencia.
    - Bits luma/croma: es la relación entre los bits dedicados a luma y a croma en cada imagen.
    - PSNR RGB: ofrece la relación señal-ruido que nos muestra la calidad de reconstrucción de la imagen a color, cuanto mayor sea ese valor mejor calidad tiene.
    - PSNR luma: muestra la relación señal-ruido que nos muestra la calidad de reconstrucción de la luminancia en este caso.
    - PSNR croma: muestra la relación señal-ruido que nos muestra la calidad de reconstrucción de las crominancias.

La Figura 41 es lo que vería un usuario tras seleccionar una secuencia y donde se le permite intercambiar las imágenes sin necesidad de ir hacia pestañas anteriores.

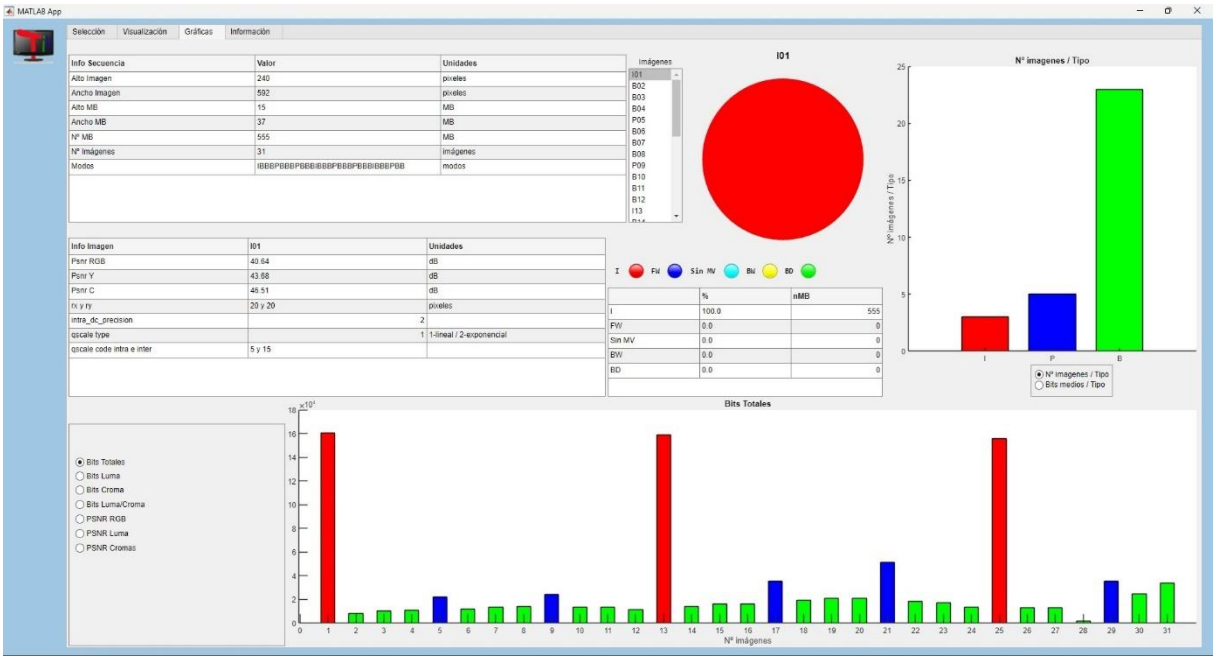

Figura 41. Pestaña de Gráficas.