E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Codificación adaptada al tipo de contenido para el estándar H.264/AVC

Máster Universitario en
Ingeniería de Telecomunicación

Trabajo Fin de Máster

Mikel Blazquez Tatiegi

Silvia Díaz Lucas

Pamplona, 18/09/2020

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Resumen

H.264/*Advanced Video Coding* (AVC) es uno de los *códecs* de video más utilizado en el ámbito de la codificación de video a nivel mundial. Tiene una penetración significativa en los mercados de discos ópticos o en el *streaming* de video, entre otros, y se caracteriza por ofrecer una buena calidad de imagen y una compresión eficiente de video. A pesar de que existe un estándar (el H.265/High *Efficiency Video Coding* (HEVC)) más reciente de la misma familia que dispone de unas mejores prestaciones de compresión, la complejidad computacional de ejecutar el codificador H.264/AVC es de varios órdenes menor. Además, existen multitud de dispositivos que incorporan aceleración hardware para la codificación y decodificación en este formato.

Mediante este Trabajo Fin de Máster se ha pretendido facilitar al centro tecnológico Vicomtech el desarrollo de una aplicación multimedia que, basándose en el estándar H.264/MPEG-4 AVC, selecciona el bitrate adecuado para un flujo de video, en función del perfil de calidad que se desea obtener y el tipo de contenido que se quiera codificar, además de su posterior integración en una librería de propósito general que se emplea en el centro. De este modo, se ha tenido que realizar un análisis previo de varias secuencias pertenecientes a diferentes tipos de imágenes, con el propósito de obtener los rangos operativos en los que se ha basado la aplicación diseñada. Cabe destacar que, para llevar a cabo el proyecto, se ha tenido que hacer uso de los *frameworks* de multimedia FFMPEG y GStreamer.

Palabras clave: H.264/AVC, PSNR, GStreamer, Rate control, Codificación de video

Abstract

H.264/Advanced Video Coding (AVC) is one of the most widely used video codecs worldwide in the field of video encoding. It can be highlighted its significant penetration in the markets for optical discs or video streaming, among others. Being this way, the codec is characterized by offering good image quality and efficient video compression. Although there is a more recent standard (the H.265/High Efficiency Video Coding (HEVC)) of the same family that has better compression features, the computational complexity of implementing the H.264/AVC encoder is about several orders smaller. In addition, there are many devices that incorporate hardware acceleration for encoding and decoding processes in this format. Those are the reasons why this codec has been used to carry out this project.

The aim of this Master Thesis has been to develop a multimedia application based on the H.264/MPEG-4 AVC standard and to integrate it in a general-purpose library used in the Vicomtech technological centre. The developed application will be able to select the appropriate bitrate for a video stream, depending on the quality profile the user wants to obtain and the type of content the user wants to encode. In such manner, several sequences belonging to different types of content had been analysed in order to get the operating ranges (bitrate values) on which the designed application is based. It should be noted that to carry out the project, two different multimedia *frameworks* have been used, FFMPEG and GStreamer.

Key words: H.264/AVC, PSNR, GStreamer, Rate control, Video encoding

Laburpena

Bideo kodifikazioaren esparruan, H.264/Advanced Video Coding (AVC) mundu mailan gehien erabiltzen den estandarretako bat dela esan daiteke. Besteak beste, disko optikoen edo bideo streaming-aren merkatuetan arrakasta handia dauka eta konpresio eraginkor batez baliatuz kalitate egoki bat eskaintzea kontsideratzen da bere puntu indartsuenetako bat. Nahiz eta familia berekoa den estandar berriago bat jada merkatuan egon, zeinak konpresio prestazio hobeak dituen (H.265/High *Efficiency Video Coding* (HEVC)), H.264/AVC kodifikadorearen inplementazioa konplexutasun konputazionalaren ikuspuntutik optimoagoa da. Gainera, kodifikazio eta dekodifikaziorako hardware-aren azelerazioa erantsia duten gailu ugari aurki daitezke merkatuan formatu honetarako.

Master Amaierako Lan honen bitartez, Vicomtech zentro teknologikoarentzat H.264/AVC estandarrean oinarritzen den multimedia aplikazio bat garatzea eta zentroko xede orokorreko liburutegi batean integratzea izan da. Aplikazioak, ezartzea nahi den bideo kalitatearen eta kodifikatzailea lan egiten ari den eduki motaren arabera, bitrate-a aldatzeko aukera eman beharko dio erabiltzaileari. Modu honetan, talde desberdinetan sailkatu diren hainbat bideo sekuentzia analizatu behar izan dira, hauek prozesatuz, garatutako aplikazioa oinarrituko den lan eremua (bitrate balioak) eskuratzeko. Esan beharra dago, proiektua aurrera eramateko, bi multimedia *framework* erabili direla: FFMPEG eta GStreamer.

Hitz gakoak: H.264/AVC, PSNR, GStreamer, Rate control, Bideo kodifikazioa

Índice de contenidos

Lista de figuras

Lista de tablas

Lista de acrónimos

AVC	Advance Video Coding
CBR	Constant Bitrate
CCTV	Circuito Cerrado de Televisión
EC	Ecuación
GOP	Group of Pictures
HVS	Human Visual System
MPEG	Moving Picture Experts Group
MSE	Mean Squared Error
PSNR	Peak Signal-to-Noise Ratio
QP	Quantification Parameter
RGB	Red Green Blue
SSIM	Structural Similarity index
VBR	Variable Bitrate

1 Introducción

1.1 Motivación

La codificación de video es el proceso de comprimir una secuencia de imágenes con el propósito de reducir su tamaño, estableciendo formatos eficaces para el almacenamiento o la transmisión de video. Se trata de un proceso que suele tener pérdidas [1], ya que se elimina información relacionada con la secuencia, por lo que, a la hora de descomprimir la versión codificada, con el propósito de reproducirla, se genera una aproximación del archivo original, que no suele ser la misma. El software que tiene la función de implementar esta tarea se conoce como *códec*.

En el mundo de la codificación de video, hay ciertos elementos o parámetros que han adquirido un gran valor, así pues, hay que tenerlos muy en cuenta. El bitrate y el *rate control* son dos de estos elementos y es importante entender bien la función que cumplen. El bitrate, hace referencia a la cantidad de datos que un codificador emplea por unidad de tiempo para generar el archivo codificado. El *rate control* es el elemento que se encarga de ajustar dinámicamente los parámetros del codificador para lograr un bitrate objetivo [2] y, vale decir que, lo bueno o lo malo que sea una implementación de cualquier *códec* reside en el *rate control*, ya que es el encargado de buscar un compromiso entre la eficiencia de compresión y la calidad resultante. Por una parte, interesa que el bitrate sea lo suficientemente bajo para que se reduzcan al máximo los costes de almacenamiento y transmisión y, por otra parte, es mejor que el bitrate sea lo suficientemente alto para ganar en fidelidad respecto a la original.

Es necesario considerar que, la relación entre el bitrate y la calidad deseada, muestra una dependencia directa con ciertos factores o parámetros del codificador [3] como pueden ser el tipo de imagen con la que se está trabajando, la dinamicidad de las escenas, la resolución de la imagen, el *framerate* o el rango dinámico. Por consiguiente, un mismo valor de bitrate no siempre produce la misma calidad en un flujo multimedia. En este sentido, a pesar de que los enfoques clásicos de codificación no suelen tener en cuenta el tipo de imagen, y emplean una misma codificación predefinida para los diferentes tipos de imágenes, se puede llegar a pensar que aplicando una codificación dependiente del propio contenido de una secuencia de imágenes, se reducirían los costes de almacenamiento y transmisión.

Partiendo de esta idea, Netflix desarrolló en 2015 un algoritmo donde se optimizaba la codificación [4], personalizando el bitrate en función de la complejidad que presentaba el elemento multimedia que se tenía a la entrada de un codificador de video. Hoy en día, esta forma de codificación se conoce como *per-title encoding* y tiene como finalidad garantizar una calidad señalada, seleccionando el bitrate que proporciona la menor cantidad posible de bits al *códec*, con el fin generar una experiencia visual óptima para el espectador a un menor coste para el proveedor del servicio multimedia. De este modo, se consigue mejorar la eficiencia de compresión, manteniendo un buen nivel de calidad y eliminando cualquier información redundante que el ojo humano no vaya a ser capaz de percibirlo.

Para implementar efectivamente este algoritmo, hay que tener claro, en que se basa el concepto de la complejidad en una imagen y la diferencia que existe entre una imagen de baja y de alta complejidad. En primer lugar, una secuencia de video se caracteriza por tener una baja complejidad, si las imágenes que la forman están compuestas mayormente por regiones planas o con un nulo granulado, y si presenta poca variabilidad temporal entre las imágenes consecutivas que la forman.

Los dibujos animados, las imágenes generadas por un sistema informático, como pueden ser los videojuegos, o los videos de vigilancia suelen entrar en esta categoría, y pueden ser comprimidos fielmente reduciendo significativamente el bitrate empleado, sin tener ningún impacto en la percepción visual del espectador.

En cambio, una secuencia de video se considera compleja, si existe una alta variabilidad entre las imágenes que la forman (la cual se traduce en movimiento) y si se dan pocas redundancias espaciales en cada una de ellas. Las películas de acción, los documentales, o los eventos deportivos pueden entrar en esta categoría, ya que presentan escenarios complejos o rápidos cambios de escena, movimientos rápidos de los objetos… Por lo tanto, presentan menos oportunidades a la hora de comprimir la información sin afectar a la calidad percibida.

Inicialmente, este documento va a centrarse en describir el funcionamiento de un *códec* de video concreto, es decir, el ITU-T H.264 (MPEG-4 AVC ISO/IEC 14496-10 – MPEG-4 Part 10, *Advanced Video Coding*) [5], ya que se trata del *códec* que se ha empleado para obtener los objetivos que se van a presentar en el punto 1.2. Asimismo, se podría decir que el reto principal propuesto para este Trabajo Fin de Máster se ha basado en la búsqueda de un conjunto de bitrates fijos que deberían servir de forma general para obtener ciertos umbrales de calidad (alta, media y baja) para diferentes tipos de contenidos (entretenimiento, conducción, realidad aumentada, CCTV, producción y 360º). Posteriormente, se ha buscado configurar un sistema que, en base al tipo de contenido del flujo de entrada y al perfil de calidad que se quiera obtener, codifique el flujo de entrada de una forma dinámica a través de los bitrates previamente establecidos. Cabe destacar que, este proyecto surge a raíz de la estancia en el Centro Tecnológico Vicomtech (en la línea de *Media Workflow Management*) que se encuentra en Donosti y se ha realizado entre los meses de mayo y septiembre del año 2020.

1.2 Objetivos del proyecto

El objetivo de este Trabajo Fin de Máster ha sido el de encontrar los rangos operativos que permitan desarrollar un sistema de codificación que emula el funcionamiento del *Per-Title encoding*. Para ello se ha planteado desarrollar un sistema que se encarga de generar una salida en función del tipo de contenido por la que se caracteriza la imagen que se tiene a la entrada y de la calidad que se desea conseguir a la salida. En la siguiente figura se puede observar una representación gráfica del sistema, donde se muestran las entradas y las salidas que lo caracterizarían.

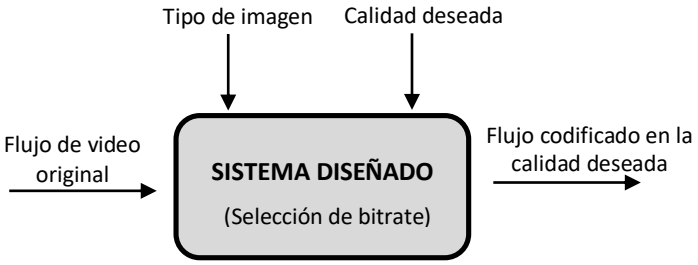


Figura 1: Sistema a diseñar en el proyecto.

El diseño del sistema que se propone se basará en la obtención previa de un conjunto de pares de bitrate-calidad para varios tipos de secuencias de imagen, las cuales se distinguirán entre ellas debido a las distintas características que presentan. Se le prestará especial atención tanto a la complejidad espacial (variabilidad en la propia imagen), como a la temporal (cambios de escena), por las que dichas secuencias están caracterizadas. Con el fin de obtener los datos necesarios para desarrollar el sistema, se deberán seleccionar, analizar y procesar un conjunto de muestras (secuencias de video) que formarán el *dataset*/conjunto de datos del proyecto.

Algunos de los retos a afrontar en este proyecto incluyen:

1. El establecimiento de los rangos operativos que relacionarán los perfiles de calidad deseados con los bitrates apropiados mediante el *framework* FFMPEG.
2. El desarrollo del sistema una vez definidos los márgenes de actuación y su aplicación en vivo configurando los parámetros necesarios (bitrate…) dinámicamente a través de Gstreamer.
3. Su integración en una librería de propósito general que se tiene en Vicomtech, centro tecnológico donde se ha llevado a cabo el proyecto.

Cabe destacar que, desde un principio Vicomtech mostró su interés en la búsqueda de tres perfiles de calidad (correspondientes a alta, media y baja calidad), evitando centrarse solamente en realizar el *Per-Title encodig* para buscar la mejor calidad posible. Esto es de especial relevancia cuando se trata de aumentar la capacidad de transmisión de flujos mediante un mismo interfaz de red, como es el caso de sistemas comunicándose mediante redes móviles.

Obsérvese que, la metodología y los resultados aquí presentados son aplicables al códec H.265/HEVC (*High Efficiency Video Coding* Rec. ITU-T H.265 ISO/IEC 23008-2) [6]. No obstante, se recalca que se ha pretendido que el proyecto se basase en su totalidad en el estándar H.264/MPEG-4 AVC al ser el códec con una presencia predominante en aplicaciones y servicios cuyos contenidos sustentan el análisis, las actividades y las conclusiones de este trabajo.

1.3 Planificación del proyecto

En esta sección se lista un breve resumen de las tareas que se diseñaron y se han llevado a cabo para poder cumplir los objetivos previamente definidos. En el apartado correspondiente a la metodología se analizan detalladamente las diferentes etapas que ha tenido el proyecto

I. Concepción e iniciación del proyecto

- Familiarizarse con el entorno de la codificación.
- Realizar un estudio del estándar H.264/MPEG-4 AVC, prestándole especial atención al funcionamiento de cada uno de los bloques principales que forman el códec que presenta el estándar.
- Aprender a manejar el *framework* de multimedia FFMPEG.
- Completar el conjunto de datos con los videos a procesar.
- Definir qué datos pueden ser de interés y analizar qué gráficas pueden ser generados con dichos datos, para la creación posterior de las tablas de búsqueda.
- Analizar cuáles podrían ser los umbrales de calidad.

II. Pruebas de caracterización y establecimiento de las tablas de búsqueda

- Diseñar los scripts definitivos para obtener los datos previamente definidos.
- Automatizar las pruebas para acelerar los procesos.
- Representar gráficamente los datos obtenidos.
- Definir los umbrales definitivos

III. Ajuste dinámico y desarrollo

- Familiarizarse con el *framework* Gstreamer.
- Entender cómo se configuran dinámicamente los atributos que pertenecen a un elemento.
- Volcar las tablas de búsqueda establecidas en el punto 2 a este entorno.
- Asegurarse del correcto funcionamiento del sistema.
- Integrar el desarrollo en la biblioteca de propósito general del centro

1.4 Estructura de la memoria

Esta memoria está formada por cinco apartados, los cuales se detallarán a continuación:

- Apartado 1: Introducción. Se trata del presente apartado. Realiza una breve introducción sobre el contexto de este proyecto y se presentan los objetivos que se han tenido que ir cumpliendo durante los cuatro meses que ha durado el proyecto. Además, se realiza un breve resumen de las tareas que se han tenido que llevar a cabo.
- Apartado 2: Contexto tecnológico. Se encarga de revisar la literatura básica relacionada con el proyecto. Es recomendable leerlo antes de analizar los siguientes apartados, ya que se definen los conceptos teóricos que sirven para fundamentarse en el proyecto.
- Apartado 3: Metodología. Se describen las tareas, las fases, los procedimientos, los recursos… en las que se ha basado el proyecto para garantizar el cumplimiento de los objetivos.
- Apartado 4: Resultados. Se muestran los resultados obtenidos y se discute sobre ellos.
- Apartado 5: Desarrollo en GStreamer. Se describe de una manera visual el código empleado para el desarrollo de la aplicación y se ilustra la interacción entre los diferentes módulos lógicos del programa.
- Apartado 6: Conclusiones y líneas futuras. Se valora el desarrollo que ha tenido el proyecto y se habla sobre la evolución que pueda tener en el futuro.

2 Contexto tecnológico

2.1 Espacios de color

Desde un punto de vista científico, el color se caracteriza por ser la interpretación que un individuo realiza sobre la reflexión de la luz en una superficie. Un modelo de color se define como una forma matemática abstracta que permite representar un color numéricamente. De este modo, mediante un modelo de color se consigue que los colores se organicen de una forma específica en una imagen o en un video. CIE, RGB, YUV, HSL/HSV y CMYK (y sus variantes) son algunos de los modelos que más se han empleado en los últimos años. Este apartado realiza una breve introducción acerca del modelo RGB (Red Green Blue) y se centra mayormente en el modelo YUV y en sus variantes

2.1.1 Modelo RGB

RGB es un modelo cromático que se basa en la representación de una amplia gama de colores a partir de la mezcla de tres colores primarios, que son el rojo, el verde y el azul. Dicho de otro modo, se trata de un sistema de síntesis aditiva donde mediante la combinación adecuada de cada uno de los tres componentes que forman este modelo, es posible representar cualquier color [7]. De este modo, cada píxel de una imagen se figura mediante 3 valores, los cuales se conocen como canales e indican la proporción relativa de los componentes rojo, verde y azul. En la Figura 2 se describe esto gráficamente, haciendo uso de un plano tridimensional, donde cada canal (R, G y B) es representado en un eje.

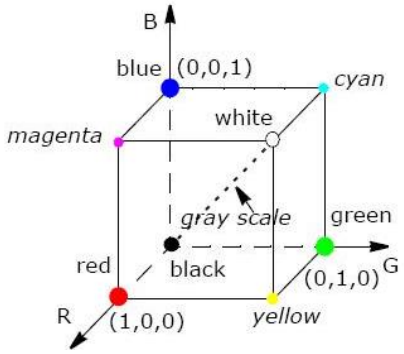


Figura 2: Representación gráfica del modelo RGB [8].

Cabe destacar que, este cubo producido en un plano de coordenadas se conoce como el espacio de color RGB. En él, se observa que si los tres canales se establecen en sus valores máximos (255 si se utiliza 1 byte para representar cada canal), el color resultante que se obtiene es el blanco. En cambio, si los tres canales se establecen en 0, significa que no se emite ninguna luz y el color resultante es el negro (píxel apagado).

No obstante, RGB es un modelo donde cada píxel posee tres canales de información. Sin embargo, en el dominio de aplicación de codificación de video (por ejemplo, en procesamiento de imagen, *broadcasting*, almacenamiento…) se pueden usar otros espacios de color que exploten la menor sensibilidad del ser humano al color y la mayor sensibilidad a la presencia de objetos. En este sentido es importante saber que en codificación de video no se trabaja en RGB, sino que en YUV o en sus diferentes variantes.

2.1.2 Modelo YUV

YUV es otro modelo de color que normalmente se emplea en diferentes disciplinas de procesamiento de imagen y video. El diseño de este modelo se basa en la sensibilidad del ojo humano, ya que codifica una imagen teniendo en cuenta su percepción. A diferencia de como ocurre con RGB, en este caso, cada píxel se representa mediante un vector de tres componentes Y, U y V [9]. Y se refiere al componente correspondiente a la luminancia y se caracteriza por ser el componente más importante de los tres, ya que determina la intensidad de luz que llega al ojo humano, es decir, establece el nivel de gris en una escala de grises. En cambio, U y V son los valores correspondientes a la crominancia y contienen la información acerca del color del píxel.

La principal diferencia entre los modelos RGB y YUV consiste en que si bien en el modelo RGB la información de luminancia y crominancia van mezcladas en cada uno de los tres canales (en R, en G y en B), en el modelo YUV se consigue separar la luminancia de la crominancia en canales independientes. Cabe destacar que, esta separación entre las informaciones correspondientes a la luminancia y crominancia tiene sus razones prácticas, ya que permite realizar operaciones específicas en la información correspondiente al color o en la información correspondiente a la luminancia, según convenga.

A lo largo de la historia de la televisión (o de la distribución del contenido multimedia en general), se podría decir que principalmente ha habido dos razones para el uso del modelo YUV sobre el modelo RGB. Por una parte, a finales de los años 70, cuando se añadió la información de color a la televisión analógica, se procuró mantener la compatibilidad con los transceptores en blanco y negro, ya que se necesitó un tiempo hasta que la televisión en color se consolidará en la sociedad. De este modo, si una televisión en blanco y negro recibía una señal en YUV, sería capaz de ignorar las señales de color y representaría la imagen basándose únicamente en el canal correspondiente a la luminancia. Por otro lado, teniendo en cuenta que el ojo humano es más sensible a la luminancia que a la crominancia, el sistema YUV permite comprimir una imagen perjudicando únicamente a la información de color, asumiendo que la información correspondiente a la luminancia quedaría intacta y que se representaría con una resolución más alta (con más muestras). En este sentido, se consigue minimizar el tamaño de la imagen, reduciendo el número de bits empleados para codificar los componentes correspondientes a la crominancia y sin que el ojo se dé cuenta de la pérdida de calidad a la hora de visualizar la imagen.

A fin de cuentas, un modelo que define un conjunto de colores que se encuentran en un espacio de color a partir de un valor de luminancia y dos valores de crominancia, se conoce como un sistema *luma-chroma* y el modelo YUV entraría en este grupo. Por otra parte, se puede decir que existe mucha ambigüedad en la literatura de la codificación de la imagen con la nomenclatura YUV [10], ya que muchas veces este término se emplea incorrectamente para referirse a otros términos que se basan también en la teoría de color *luma-chroma*. Pueden ser los casos de Y'UV, Y'CbCr o YCbCr, entre otros. A continuación, se intentará aclarar esto.

Teniendo en cuenta que el ojo humano no responde linealmente a la luz, para compensar ciertas propiedades de la visión humana se emplea la corrección gamma [11]. Como se ha comentado previamente, en el modelo YUV la Y hace referencia a la luminancia, pero si se le aplica una corrección a dicho componente se obtiene una luminancia gammificada, el cual se conoce como luma. De este modo, la nomenclatura que se utiliza para referirse al modelo YUV después de aplicarle una corrección

gamma sería Y'UV, donde la Y' significa luma. Como se puede apreciar en la siguiente ecuación la obtención de este componente Y' se realiza a partir un promedio con pesos de los componentes R, G y B gamma corregidos.

$$\mathbf{Y'} = \mathbf{0.299} * \mathbf{R'} + \mathbf{0.587} * \mathbf{G'} + \mathbf{0.114} * \mathbf{B'} \qquad \text{Ec. 1}$$

Por otro lado, Y'CbCr es un sistema de codificación que deriva de YUV y se caracteriza por tener sus propias propiedades matemáticas, los cuales no son intercambiables con las propiedades del modelo YUV. Este esquema Y'CbC, se emplea en la codificación digital de imagen y video, de ahí viene su importancia dentro de la familia YUV. La Y' corresponde al componente luma que se acaba de comentar, y tanto Cb, como Cr, son los componentes correspondientes a la crominancia, donde cada uno de ellos se obtiene a través de la diferencia entre los componentes R' o B' y la luminancia gammificada (luma).

$$\boldsymbol{Cb} = \mathbf{0.564} * (\boldsymbol{B'} - \boldsymbol{Y'}) \qquad \text{Ec. 2}$$
$$\boldsymbol{Cr} = \mathbf{0.713} * (\boldsymbol{R'} - \boldsymbol{Y'}) \qquad \text{Ec. 3}$$

Debido a la alta sensibilidad del ojo a la hora de detectar cambios en la luminancia, se intenta aplicarle una tasa de muestreo más alta (o por lo menos la misma) al componente Y' en comparación con los componentes Cb y Cr. Dicho esto, los formatos más comunes en Y'CbCr se consideran los siguientes: 4:4:4, 4:2:2 y 4:2:0. En el formato Y'CbCr 4:4:4, los tres componentes se muestrean con la misma cantidad de muestras. En el formato Y'CbCr 4:2:2 los dos componentes de crominancia se muestrean horizontalmente con la mitad de las muestras en comparación con el componente de luminancia. Por último, en Y'CbCr 4:2:0 se emplean 4 veces más muestras en la luminancia que en los componentes correspondientes a la crominancia. A continuación, se puede contemplar esto gráficamente.

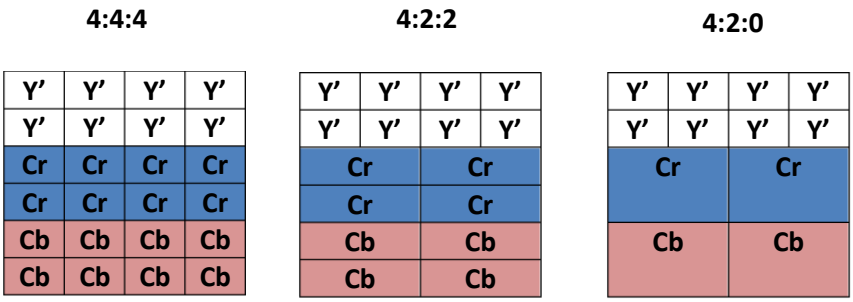


Figura 3: Representación gráfica del muestreo de la crominancia.

El formato más común se considera el Y'CbCr 4:2:0 y como se puede apreciar en la figura anterior, por cada cuatro muestras de luminancias se tiene una muestra de crominancia por cada componente de crominancia. Es digno de señalar que, antes de que un archivo YUV sea reconvertido al modelo RGB, se tiene que realizar un proceso de sobremuestreo en los componentes de crominancia para volver disponer de un formato Y'CbCr 4:4:4. Esto se puede realizar duplicando directamente las muestras correspondientes a la crominancia o utilizando otros métodos para mejorar la exactitud, como puede ser la información de la vecindad. En la Figura 4 se esquematiza este sistema de muestreo.

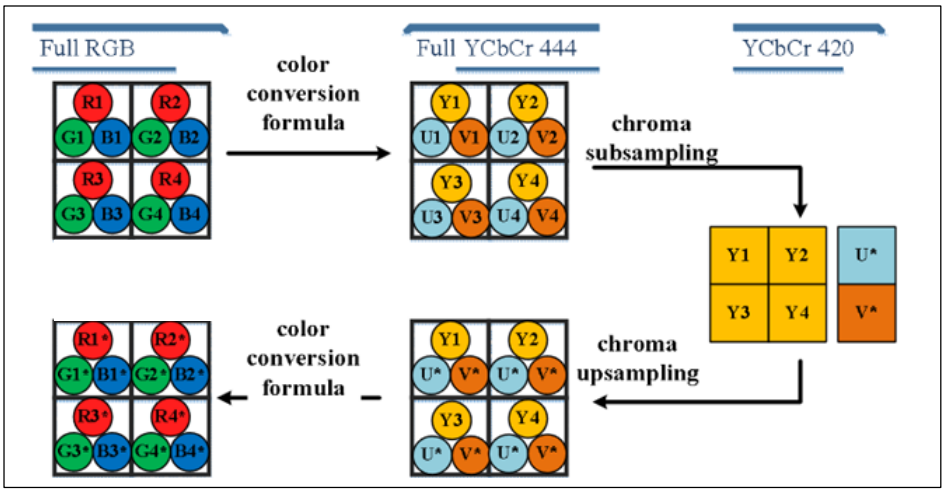


Figura 4: Los procesos de submuestreo y sobremuestreo de los componentes de crominancia [12].

2.2 Códecs

La comunicación vía imagen digital es un campo que se encuentra en pleno desarrollo, especialmente con el progreso que se está realizando en las técnicas de codificación o compresión de video. Hay que tener en cuenta que se requiere una gran tasa de bits para poder representar, almacenar o transmitir un archivo multimedia en su estado original (en *raw*). Es aquí donde entra en juego la compresión, el cual se podría definir como el proceso de convertir un archivo de video digital, en un formato óptimo para su transmisión o almacenamiento, siendo su propósito la reducción del tamaño manteniendo una fidelidad aceptable respecto a la original. Hoy en día, este proceso tecnológico se ha convertido en algo esencial para aplicaciones como la televisión digital, el DVD-Video, la televisión móvil, las videoconferencias o la transmisión de video vía Internet (*streaming*).

Como se contempla en la Figura 5, el proceso de codificación está definido por dos sistemas. En primer lugar, se encuentra el codificador y se encarga de crear una versión comprimida de la fuente multimedia que se tiene a la entrada, para que sea transmitida o almacenada. En segundo lugar, el sistema debe estar completado por un decodificador, el cual es el encargado de recuperar el video original, descomprimiendo la imagen que se encuentra codificada. Es interesante saber que, estos sistemas formados por un codificador y un decodificador se conocen como *códecs*.

Figura 5: Codificador/Decodificador

Siendo este el contexto, se puede llegar a pensar que una compresión sin pérdidas de información en las imágenes puede ser viable a la hora de codificar una secuencia de video antes de que sea transmitida para su futura reproducción ante un espectador humano. Sin embargo, esto no es así, ya que de esta manera solamente se consigue una cantidad moderada de compresión, por lo que la mayoría de las veces se ve necesario recurrir a una compresión con pérdidas [13].

Estos sistemas de compresión de video con pérdidas se fundamentan en el principio de eliminar la redundancia subjetiva, es decir, elementos de la imagen que puedan ser suprimidos sin afectar significativamente en la percepción de la calidad visual del espectador. Eso sí, en un sistema con pérdidas, la entrada y la salida de un *códec* nunca van a ser idénticas, aunque puede que esas diferencias no sean detectables para el ojo humano. Idealmente el modelo utilizado por cualquier *códec* para representar el video original, debe buscar un compromiso entre el tamaño del flujo de datos generado (número de bits empleados) y la percepción visual que se obtiene en la salida (calidad).

A continuación, se puede apreciar de forma visual como afecta el uso de un menor bitrate en la calidad de las imágenes codificadas respecto a la original. De este modo, se demuestra mediante una imagen de testeo estándar (más concretamente utilizando la imagen mundialmente conocida en el campo de procesamiento de imagen de Lena) que el uso de mayores ratios de compresión afecta en la calidad con la que un espectador percibe la imagen.

Figura 6: Ilustración de la imagen Lena decodificada con diferentes bitrates: (i) imagen original, (ii) ratio de compresión de 60 obteniendo una PSNR de 33.9735 dB, (iii) ratio de compresión de 30 obteniendo una PSNR de 30.3385 dB, y (iv) ratio de compresión de 5 obteniendo una PSNR de 22.9062 dB.

Para llevar a cabo este proyecto se ha hecho uso del estándar H.264/MPEG-4 AVC, ya que se trata del *códec* más extendido en soluciones comerciales con amplio soporte para codificar y decodificar.

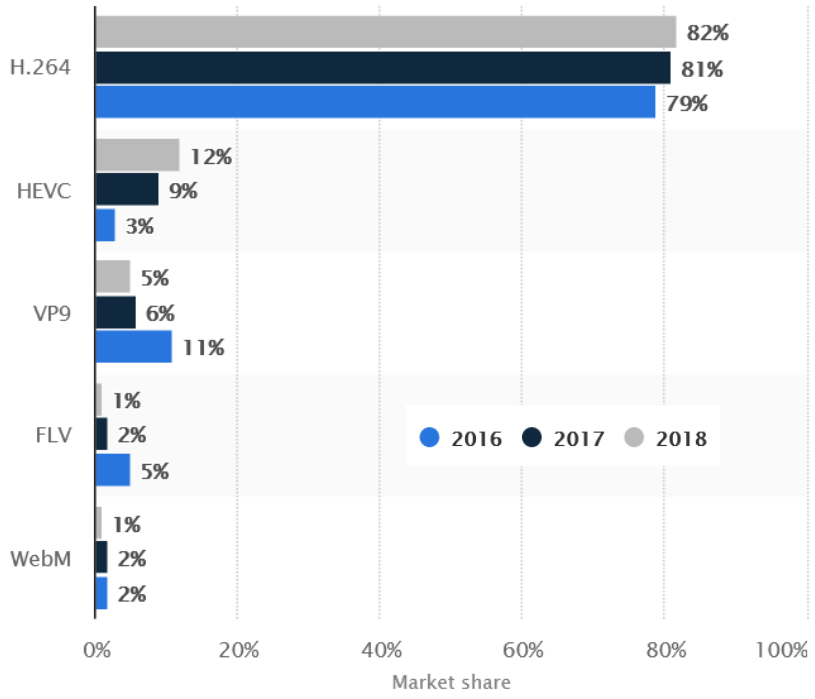


Figura 7: Comparación de los *códecs* más utilizados en 2016, 2017 y 2018 en *broadcasting* [14]

En la Figura 7 se puede observar una gráfica que compara los *códecs* de video más utilizados por los publicadores de contenidos multimedia y los *broadcasters* online de todo el mundo en los años 2016, 2017 y 2018. Se ve claramente el dominio de H.264/MPEG-4 AVC en este ámbito frente a estándares como H.265/HEVC. Estos datos fueron publicados por el portal alemán de estadísticas internacionales Statista.

H.264/MPEG-4 AVC se caracteriza por ser más eficiente que MPEG-2, tiene un mejor ratio de compresión que JPEG o MJPEG-4, es adecuado para contenido de video que presenta movimiento, destaca por su posibilidad de integración y garantiza una alta calidad en aplicaciones en tiempo real, y proporciona una menor complejidad computacional en comparación con H.265/HEVC. Por lo que este proyecto se ha basado completamente en dicho *códec*. Cabe señalar que, esta recomendación corresponde a un documento publicado conjuntamente por los dos siguientes organismos internacionales de normalización: ITU-T e ISO/IEC, y aunque fuera publicado en 2003, el estándar ha ido mejorando y actualizándose desde entonces. Asimismo, se basa en los estándares MPEG-2 y MPEG-4 Visual, ofreciendo un mejor rendimiento a la hora de comprimir: para una misma resolución se obtiene una calidad de imagen dada, utilizando un menor número de bits.

2.3 Fundamentos del estándar H.264/MPEG-4 AVC

El *códec* H.264/MPEG-4 AVC consta de 3 unidades funcionales principales [15]: un modelo de predicción, un modelo espacial y un codificador de entropía. En la Figura 8 se puede observar esto. En esta sección se intenta describir el funcionamiento de cada uno de estos bloques.

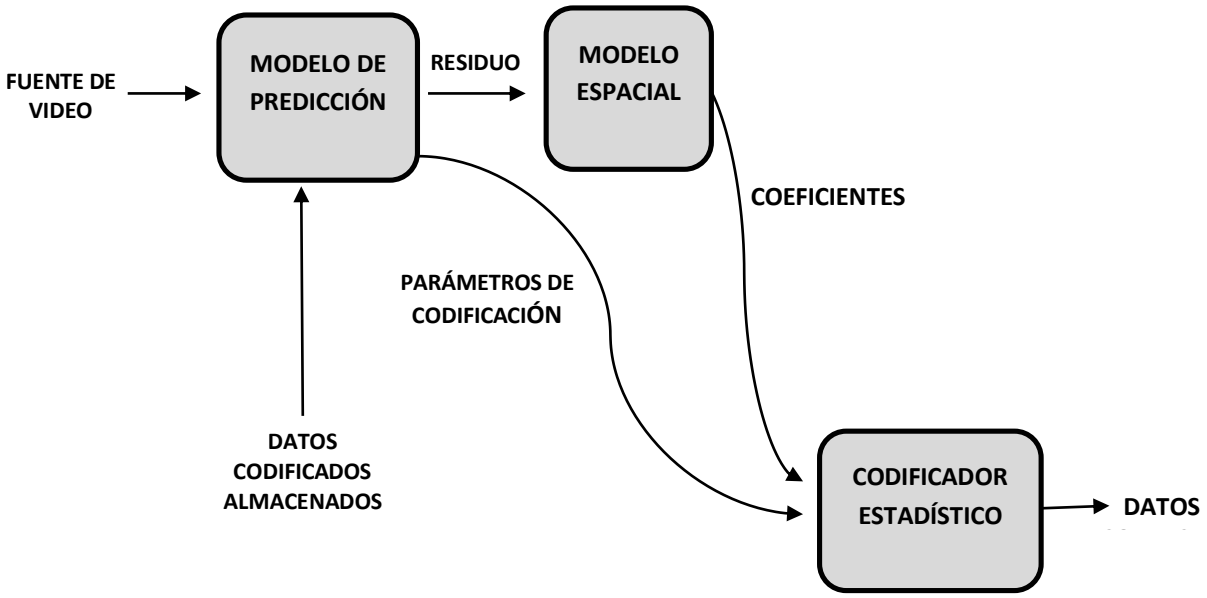


Figura 8: Diagrama de bloques del codificador de video H.264/MPEG-4 AVC

2.3.1 Predicción

El objetivo del proceso de predicción consiste en minimizar la redundancia de información existente en una secuencia de imágenes. Para ello, se parte de la idea en la que los datos correspondientes de una imagen que forma la secuencia pueden ser predichos analizando tanto las muestras ya codificadas de la propia imagen, como las muestras de las imágenes de la misma secuencia que hayan sido previamente codificadas. En los dos casos se genera un residuo de predicción a partir de la información original y de la predicha, y dicho residuo es lo que se codifica y se transmite hacia el decodificador. Asimismo, el codificador también debe transmitir la información necesaria para que el decodificador sea capaz de recrear la imagen original a partir de la predicción realizada y el residuo resultante. Obsérvese que, si la predicción es exitosa, la energía en el residuo tiende a ser menor que en la imagen original, siendo posible representar dicho residuo con un menor número de bits.

Por otra parte, es preciso enfatizar que en H.264/MPEG-4 AVC, todas las imágenes que componen un video suelen dividirse en macrobloques a la hora de realizar la codificación. Un macrobloque es una sección rectangular que se caracteriza por tener un tamaño inferior que la propia imagen donde se encuentra. El macrobloque se considera la unidad básica de procesamiento en el mundo de la codificación de video, trasladando los conceptos de codificación de vídeo a un entorno de computación real con memoria limitada. Así, los codificadores realizan las compresiones trabajando con los macrobloques en vez de con las imágenes enteras, y de este modo se consigue una mayor eficiencia a la hora de codificar.

H.264/MPEG-4 AVC permite utilizar una amplia gama de opciones a la hora de predecir. En este documento se profundizan las opciones correspondientes a la predicción temporal y a la espacial. Asimismo, como se verá más adelante, el residuo se cuantifica y esto hace que el proceso de codificación tenga pérdidas. Estas pérdidas implican que los píxeles decodificados no sean idénticos a los originales. En este sentido, si para realizar las predicciones tanto temporales, como espaciales, el codificador hace uso de los píxeles originales, podría darse un desajuste acumulativo entre el codificador y el decodificador, ya que el decodificador no tendrá a su disponibilidad dichos píxeles originales. Para evitar esta situación y con el fin de realizar las mismas predicciones en ambos extremos del sistema, el codificador debe decodificar el residuo y reconstruir los píxeles para emplearlos a la hora de realizar las predicciones.

2.3.1.1 Predicción temporal

La predicción temporal o la inter-predicción, se conoce como el proceso de predicción de un bloque de muestras de luminancia y de crominancia de una imagen, a partir de la información correspondiente a otra imagen que haya sido previamente codificada. Esto es, en virtud de una o varias imágenes de referencia que se encuentran almacenadas en un búfer de imágenes codificadas en el propio codificador. La compensación de movimiento es una técnica que emplea H.264/MPEG-4 AVC para aumentar la compresión, y es algo que hay que tenerlo muy en cuenta, ya que se centra en eliminar la redundancia temporal que existe entre las imágenes que forman un archivo de video, partiendo de la división de la imagen en macrobloques.

Primero hay que buscar en la imagen de referencia una región de predicción que coincida con el bloque original que se esté analizando. Para esta búsqueda se tiene en cuenta la posición del bloque a inspeccionar en la imagen original. Este proceso se conoce como la estimación de movimiento y un método para poder llevarlo a cabo consiste en restar las regiones candidatas de la imagen de referencia con el bloque original, con el fin examinar las energías de los residuos obtenidos, seleccionando la región que minimice la energía en el residuo. De este modo, la región seleccionada se convierte en el predictor y el codificador codifica el residuo obtenido mediante la resta del predictor con la región original, transmitiendo al decodificador tanto el residuo codificado como el vector de movimiento (el offset entre el bloque en la imagen original y la posición del predictor).

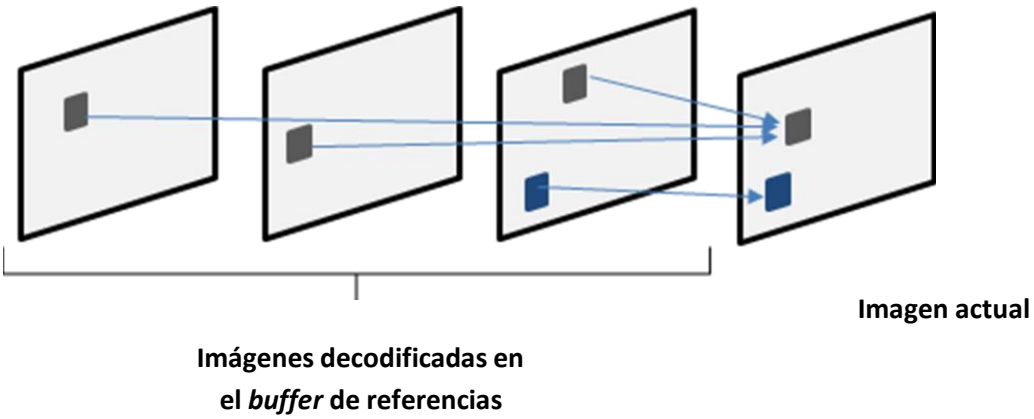


Figura 9: Predicción temporal

Es interesante saber que, el tamaño estándar de un macrobloque suele ser de 16x16 píxeles. Sin embargo, los objetos que se mueven en una secuencia de video raramente se adaptan a esta medida y suele ser más eficiente emplear un tamaño de bloque variable a la hora de estimar y compensar el movimiento (ver Figura 9). La disponibilidad de emplear un tamaño de bloque variable en la estimación de movimiento es un factor que diferencia este *códec* con estándares anteriores. Esta función del *códec* se conoce como la compensación de movimiento estructurada en árbol.

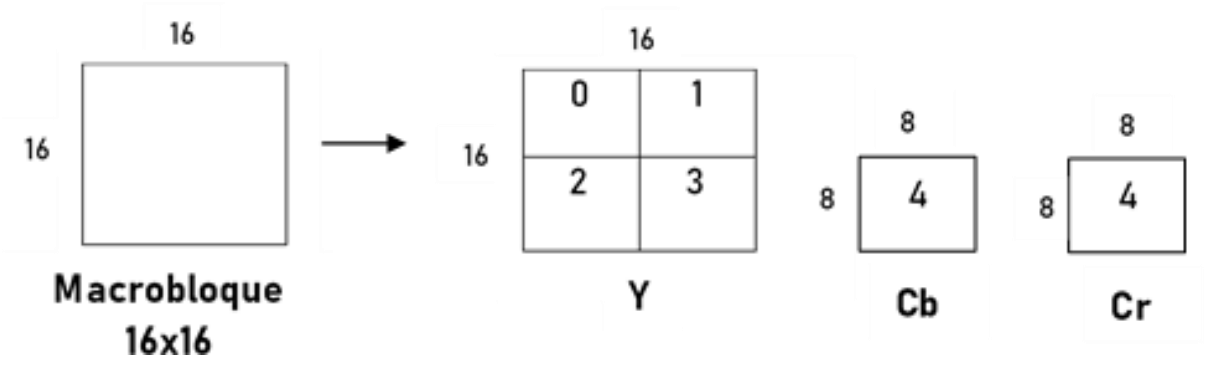


Figura 10: Macrobloque 16x16 4:2:0

Siendo esto así, en la inter-predicción, un bloque puede variar su tamaño desde un macrobloque de 16 x 16 muestras de luminancia y las correspondientes muestras de crominancia, hasta un bloque de 4 x 4 muestras de luminancia y las correspondientes muestras de crominancia. Cuanto menor sea el tamaño del bloque, mejor serán los resultados obtenidos a la hora de compensar el movimiento ya que se mejora la precisión de la predicción, pero al mismo tiempo la complejidad computacional incrementa, ya que aumentan tanto el número de operaciones de búsqueda a completar como el número de vectores de movimiento a transmitir (se necesitan tantos vectores de movimiento como particiones por macrobloque para poder compensar el movimiento). Por lo que hay que buscar un compromiso entre estos dos factores y adaptar el tamaño del bloque a las características de la imagen. Una práctica común suele ser el empleo de bloques más grandes en las regiones homogéneas y de bloques más pequeños en las regiones con muchos detalles.

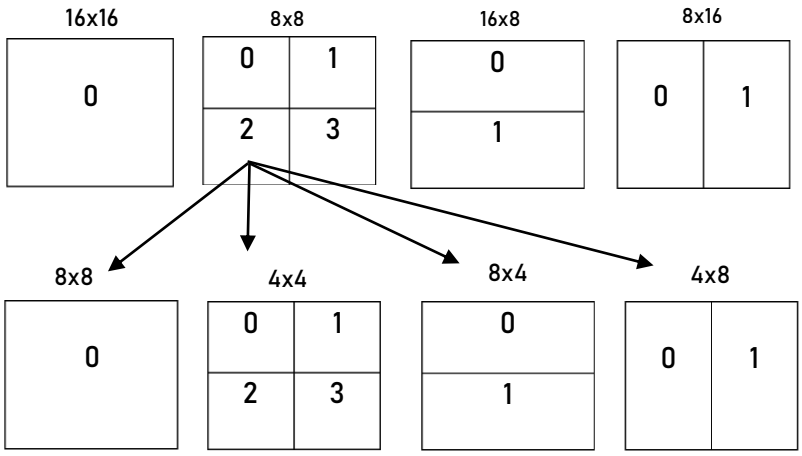


Figura 11: Partición de macrobloques

Igualmente, en ciertos escenarios, conviene realizar una compensación de movimiento a nivel subpíxel, ya que mejora el resultado final. En este caso, hay que tener en cuenta que las muestras de luma y croma no existen en la imagen de referencia en las posiciones correspondientes a los subpíxeles y que es necesario crearlas mediante la interpolación, haciendo uso de las muestras vecinas en la imagen [16]. En general, como ocurre con el tamaño de los macrobloques, una interpolación más fina suele proporcionar un mejor rendimiento a la hora de compensar el movimiento, produciendo un residuo con una menor energía a costa de una mayor complejidad. Por otra parte, la ganancia de rendimiento tiende a disminuir a medida que aumentan los pasos de interpolación. Dicho de otro modo, mediante una interpolación de medio píxel se consigue una ganancia significativa en la compensación de movimiento respecto a la obtenida con un píxel entero (sin interpolar), la cual va a ser más grande que la ganancia obtenida entre una interpolación de un cuarto de píxel y una interpolación de medio píxel… y así sucesivamente.

Para concluir, se puede afirmar que debe existir un compromiso entre la complejidad del modelo utilizado para compensar el movimiento y la eficiencia de compresión que se quiere lograr. Una compensación de movimiento más precisa, requiere el uso de un mayor número de bits a la hora de codificar los vectores necesarios y un menor número bits a la hora de codificar el residuo de interés. Con una compensación de movimiento menos precisa ocurre justo lo contrario.

2.3.1.2 Predicción espacial

La predicción espacial o la intra-predicción, se basa en eliminar la redundancia espacial de los píxeles que se encuentran dentro de la imagen actual. Para ello, el contenido de un macrobloque es predicho utilizando las muestras de los macrobloques adyacentes que se encuentran en la misma imagen y que han sido previamente codificados. Esto ocurre debido a que, las muestras correspondientes a un bloque típico de luma o chroma, se encuentran altamente correlacionadas entre ellas, independientemente de que las muestras se encuentren en los bloques vecinos.

Dentro de los tipos de imágenes que emplea H.264/MPEG-4 AVC, las imágenes i*ntra* (I) suelen estar compuestas por macrobloques también del tipo I, suelen ser las menos comprimibles y suelen basarse únicamente en la predicción espacial. Asimismo, en el estándar H.264/MPEG-4 AVC la granularidad de predicción se reduce desde el nivel de imagen al nivel de segmentos. En este contexto, un *slice* o un segmento, que suele estar formado por varios macrobloques, se caracteriza por ser una región espacialmente distinta en comparación con el resto de la imagen y se codifica independientemente respecto a cualquier otra región de esa misma imagen. De este modo, cada macrobloque en un *slice* I, también suele ser del tipo I y se codifica mediante la predicción espacial, utilizando los datos que hayan sido previamente codificados en el mismo *slice*.

Aunque existen diferentes enfoques para realizar la predicción espacial, H.264/MPEG-4 AVC se basa en la extrapolación espacial. A la hora de codificar los macrobloques que forman una imagen, se sigue una orden *raster-scan* y se forman varias predicciones extrapolando las muestras superiores y las que se encuentran en el lado izquierdo del bloque de interés. Teniendo en cuenta que los píxeles vecinos tienen más probabilidades de estar altamente correlacionados con las muestras del bloque original, se recrea dicho bloque.

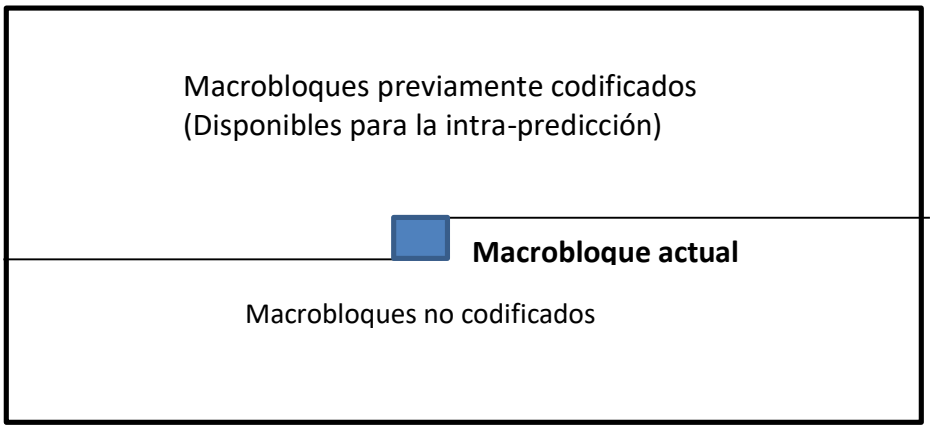


Figura 12: Predicción espacial: muestras disponibles

H.264 establece tres tamaños diferentes de macrobloques para la intra-predicción de los bloques de luminancia los cuales son 16x16, 8x8x y 4x4 y se representan como como I16MB, I8MB e I4MB respectivamente [17]. En cambio, el tamaño de bloques para la intra-predicción de macrobloques de crominancia es de 8x8. Como ocurre con la predicción temporal, mientras que I16MB suele ser un tamaño adecuado para las regiones homogéneas, I4MB se emplea en las regiones con más detalles.

Respecto a las diferentes direcciones de predicción que se emplean en H.264, cabe destacar que existen varios modos dependiendo del tipo de componente y del tamaño del macrobloque. En este sentido, los macrobloques de luminancia con un tamaño 16x16 admiten cuatro modos de predicción, los macrobloques de luminancia con un tamaño de 8x8 o de 4x4 admiten nueve modos de predicción y los macrobloques de crominancia admiten cuatro modos de predicción.

I. **Modos de predicción para los macrobloques de luminancia del tamaño 4x4 y 8x8**
   La predicción espacial se genera a partir de las muestras previamente codificadas y reconstruidas de los macrobloques vecinos que se encuentran respecto al macrobloque de interés: a la izquierda, arriba, arriba a la izquierda, arriba a la derecha o combinando estas opciones dependiendo de la ubicación del macrobloque. Así pues, los nueve modos incluyen un modo de predicción en DC (2), donde un píxel del macrobloque de interés se predice mediante un promedio de los píxeles de referencia, y los ocho modos de predicción direccional: vertical (0), horizontal (1), diagonal abajo-izquierda (3), diagonal abajo-derecha (4), vertical derecha (5), horizontal abajo (6), vertical izquierda (7), horizontal arriba (8).

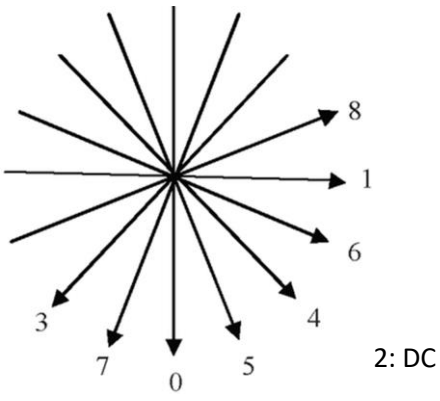


Figura 13: Direcciones de predicción de I4MB y de I8MB y sus modos correspondientes

En la siguiente imagen se representa gráficamente la intra-predicción de un macrobloque de 4x4. En él se pueden observar los píxeles de un macrobloque que deben ser codificados (los cuales están etiquetados por caracteres en minúscula) mediante los píxeles de los macrobloques vecinos ya reconstruidos (los cuales se representan por caracteres en mayúsculas).

M	A	B	C	D	E	F	G	H
I	**a**	**b**	**c**	**d**				
J	**e**	**f**	**g**	**h**				
K	**i**	**j**	**k**	**l**				
L	**m**	**n**	**o**	**p**				

Figura 14: Macrobloque de un tamaño de 4x4 y sus píxeles vecinos reconstruidos

II. Modos de predicción para los macrobloques de luminancia del tamaño 16x16 y de crominancia del tamaño 8x8

La predicción espacial se genera a partir de las muestras previamente codificadas y reconstruidas de los macrobloques vecinos que se encuentran arriba y/o a la izquierda del macrobloque de interés. Los dos casos (luminancia 16x16 y crominancia 8x8) difieren entre ellos únicamente en el orden de los números de los modos y cabe destacar que siempre se emplea el mismo modo de predicción para los dos componentes de crominancia (Cb y Cr). Los cuatro modos de predicción direccional son las siguientes para un macrobloque de luminancia 16x16: vertical (0), horizontal (1), DC (2) y plano (3).

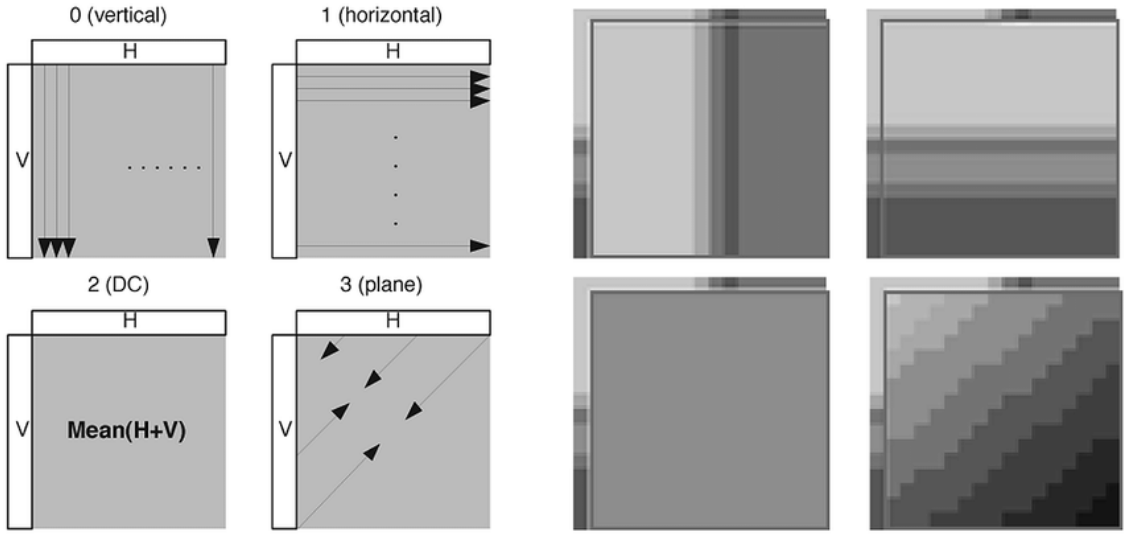


Figura 15: (izquierda) ilustración de los cuatro modos de predicción de I16MB; (derecha) ejemplos con imágenes reales [18]

2.3.2 Transformada

El propósito de esta etapa se basa en trasladar los datos que forman el residuo a un nuevo dominio, con el fin de mejorar la eficiencia de la compresión. Hay que tener en cuenta que, existen varios requerimientos que se deben cumplir a la hora implementar una transformada en un proceso de codificación. Principalmente, la transformada debe conseguir que en el nuevo dominio los datos se encuentren no correlacionados y compactos entre ellos. Vale decir, en el nuevo dominio la información debe estar separada en componentes con una mínima interdependencia entre ellas, y al mismo tiempo, la energía debe estar concentrado en una cantidad pequeña de valores. Además, la transformada debe ser reversible, es decir, tiene que existir la inversa, y computacionalmente debe ser factible realizar el cálculo (bajo requerimiento de memoria, bajo número de operaciones aritméticas…).

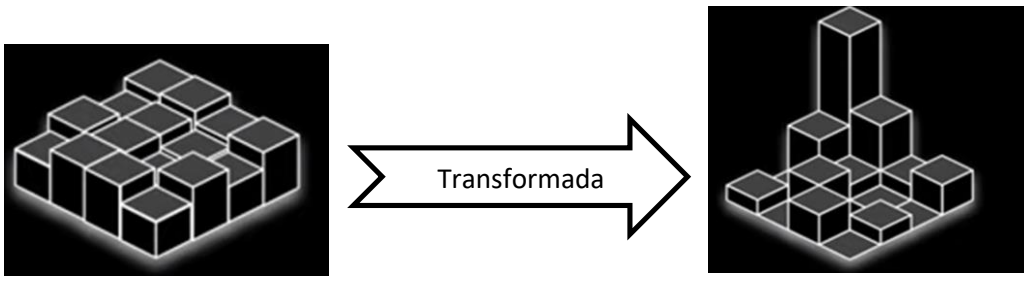


Figura 16: Transformada directa

Cabe destacar que, aunque las transformadas basadas en imágenes (p. ej. *Discrete Wavelet Transform*, DWT) son muy comunes y puede que mejores a la hora de comprimir ciertas secuencias de imágenes que presentan poco movimiento, las transformadas basadas en bloques (p. ej. *Discrete Cosine Transform*, DCT), suponen unos bajos requisitos de memoria y se adaptan fielmente a las características de la información con la que se trabaja en codificación. La transformada básica en el estándar H.264/MPEG-4 AVC se podría decir que consiste en una aproximación escalada de la Transformada discreta del coseno (DCT), el cual opera con bloques de N x N muestras, más concretamente con bloques de 4 x 4 u 8 x 8 muestras.

Matemáticamente hablando, tanto la DCT de un bloque compuesto por N x N muestras, como su inversa, se pueden calcular mediante las siguientes expresiones:

$$\boldsymbol{Y} = \boldsymbol{A} * \boldsymbol{X} * \boldsymbol{A}^{\boldsymbol{T}} \qquad \text{Ec. 4}$$

$$\boldsymbol{X} = \boldsymbol{A}^{\boldsymbol{T}} * \boldsymbol{Y} * \boldsymbol{A} \qquad \text{Ec. 5}$$

X equivale a una matriz de muestras, Y a una matriz de coeficientes y A es una matriz de transformación con una dimensión de N × N valores, el cual se calcula de la asiguiente forma:

$$\boldsymbol{A_{ij}} = \boldsymbol{C_i} * \cos\frac{(2j+1)*i*\pi}{2N} \qquad \text{Ec. 6}$$

$$\text{donde } C_i = \sqrt{\frac{1}{N}} \ (i = 0) \ \ y \ \ C_i = \sqrt{\frac{2}{N}} \ \ (i > 0)$$

Ec. 4 y Ec. 5 pueden ser reescritas de la siguiente manera:

$$Y_{xy} = C_x * C_y * \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} X_{ij} * cos\frac{(2j+1) * y * \pi}{2N} * cos\frac{(2i+1) * x * \pi}{2N} \quad \text{Ec. 7}$$

$$X_{ij} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} C_x * C_y * X_{ij} * cos\frac{(2j+1) * y * \pi}{2N} * cos\frac{(2i+1) * x * \pi}{2N} \quad \text{Ec. 8}$$

A la salida de la transformada se obtiene un conjunto de coeficientes, donde cada uno de ellos se considera un valor de ponderación para un estándar de patrones [15].

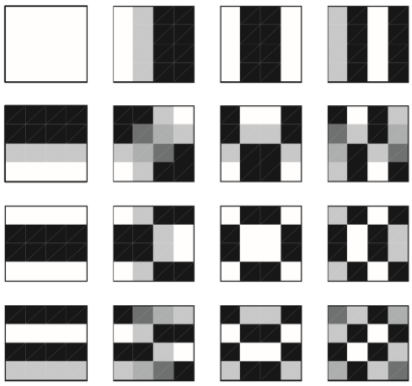


Figura 17: Patrones base DCT 4x4

Cualquier bloque de la imagen puede reconstruirse mediante la combinación de todos los patrones, siempre y cuando cada uno de ellos sea multiplicado por el factor de ponderación (coeficiente de salida de la transformada) correspondiente. En este sentido, la transformada inversa genera los macrobloques de NxN muestras que corresponden al residuo generado en el codificador, ponderando cada patrón base estándar de acuerdo con los valores de los coeficientes reescalados (en el siguiente apartado se habla de la cuantificación y del reescalado de los coeficientes) y combinándolos entre ellos.

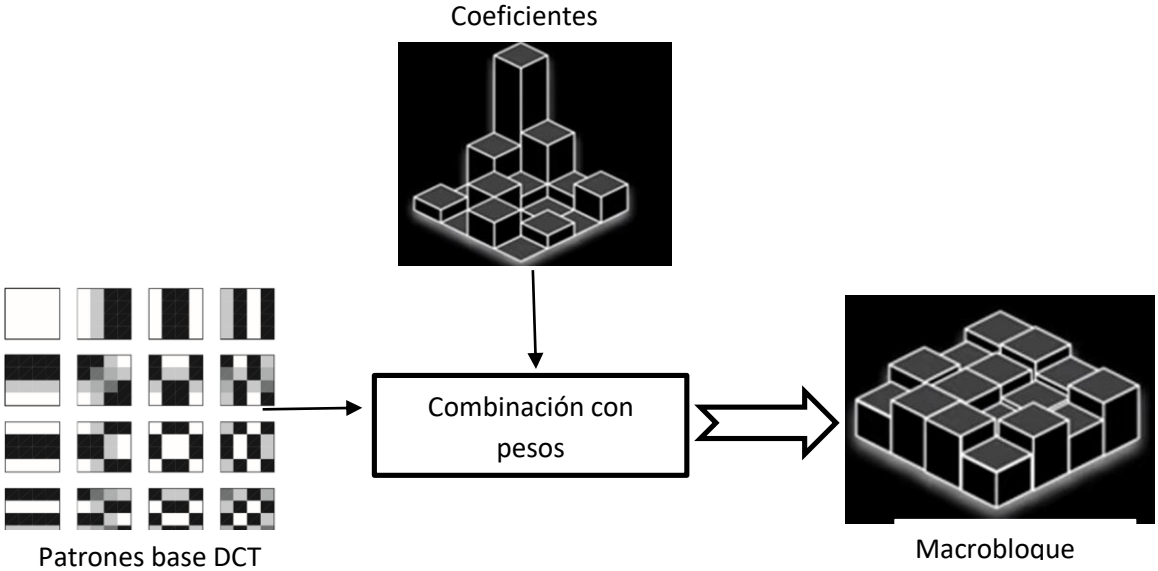


Figura 18: Transformada inversa

2.3.3 Cuantificación

En procesamiento de señales digitales, el proceso de cuantificación se encarga de mapear unos valores de entrada, pertenecientes a un amplio rango, en otros valores de salida, pertenecientes a un rango más reducido. En la mayoría de los *códecs* de compresión de imagen y video, el proceso de cuantificación generalmente se divide en dos partes. Por un lado, el codificador suele estar formado por un cuantificador directo, el cual se encarga del escalado. Por otro lado, en el decodificador se encuentra el cuantificador inverso y este se encarga del reescalado. No hay que olvidar que el sistema tiene pérdidas y que no es reversible, por lo que la entrada del codificador y la salida del decodificador nunca van a ser idénticas.

En el estándar H.264/MPEG-4 AVC, la cuantificación reduce la precisión de los coeficientes obtenidos tras la etapa de transformación, utilizando un parámetro de cuantificación conocido como QP (*Quantization Parameter*). Los coeficientes que se consiguen a la salida de la transformada se dividen por un valor de QP entero, que va de 0 a 51. Si se establece un valor de QP alto, un mayor número de coeficientes obtienen un valor de 0 y el rango de los valores cuantificados se reduce mucho, consiguiendo una alta compresión a costa de obtener una imagen de menor calidad tras la decodificación, debido a la mala aproximación realizada en el reescalado (debido a la pérdida de información). En cambio, si se establece un valor de QP bajo, después de realizar la cuantificación en el codificador habrá más coeficientes con un valor distinto a 0 y el rango de dichos valores será superior al caso anterior, consiguiendo que los valores obtenidos en el reescalado sean más similares a las originales, mejor calidad en la imagen decodificada, a costa de perder la eficiencia de compresión.

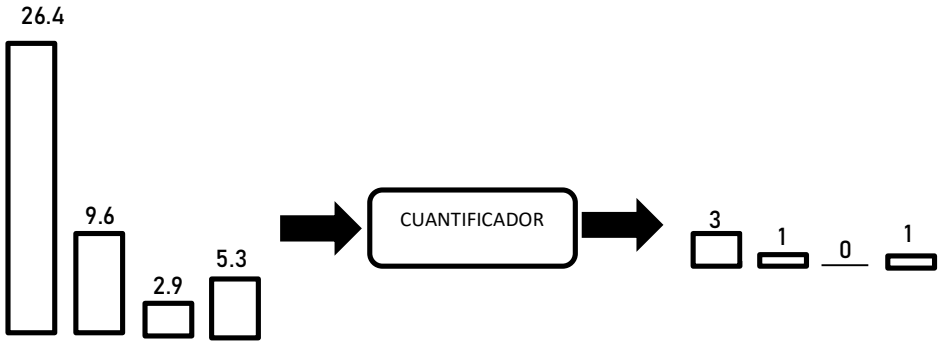


Figura 19: Ejemplo de cuantificación.

Después de cuantificar los coeficientes DCT y antes de realizar la codificación estadística, los datos pertenecientes a cada bloque son examinados y reordenados en un array lineal, con el objetivo de agrupar los coeficientes que contengan un valor diferente a cero al comienzo del array, seguidos de una larga secuencia de ceros, lo cual facilita la codificación posterior. A la hora de codificar, el orden óptimo de los coeficientes que forman un bloque depende de la distribución de los coeficientes distintos a cero en el array. Por ejemplo, en la imagen 3 se puede apreciar cómo se distribuyen las probabilidades de los coeficientes DCT distintos a cero en cada posición. En este caso, un escaneo adecuado consistiría en el *zigzag* empezando por el primer coeficiente que se encuentra en la parte superior del bloque en la zona izquierda.

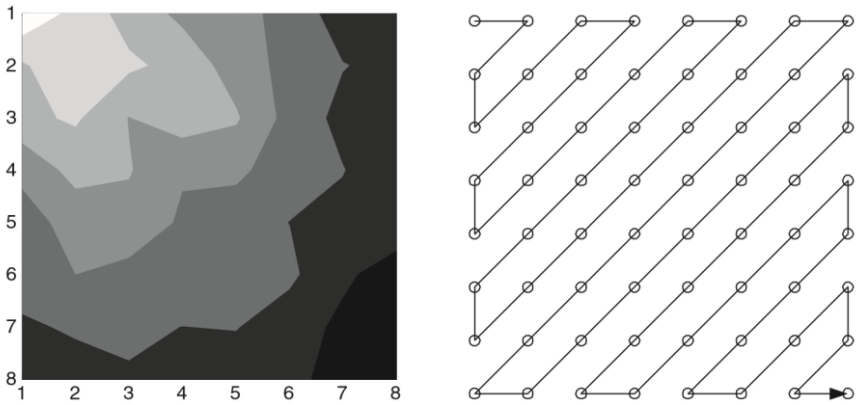


Figura 20: Distribución de coeficientes (DCT 8x8) y un escaneo en zigzag [1]

En el proceso de decodificación estos coeficientes cuantificados tienen que ser reescalados antes de que la transformada inversa sea aplicada. Para ello cada uno de estos coeficientes es multiplicado por un valor entero, con el propósito de restaurar la escala original. Como se contempla en la siguiente figura, mediante el reescalado se consigue aproximar a los coeficientes originales, pero nunca se consigue que sean idénticas, ya que siempre habrá información que haya sido eliminada en el proceso de cuantificación y que sea imposible restaurarla en el reescalado. De este modo, aunque muchas veces se haga referencia al reescalado como la cuantificación inversa, hay que tener presente que la cuantificación no es un proceso completamente reversible e introduce pérdidas.

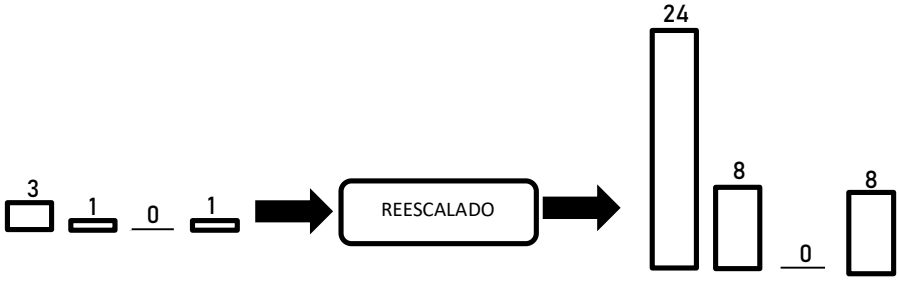


Figura 21: Ejemplo de reescalado

2.3.4 Codificación estadística

La función del codificador de entropía en el estándar H.264/MPEG-4 AVC, consiste en reducir la redundancia todavía existente en los coeficientes cuantificados obtenidos tras la etapa de transformación, para disminuir en gran medida la cantidad de datos que se desean transmitir hacia el decodificador. El proceso consiste en realizar la conversión de una serie de símbolos que representan elementos de una secuencia de video, en un flujo de bits compacto.

Dicho flujo de bits suele contener la información necesaria para que el decodificador pueda descomprimir el archivo codificado sin ningún problema. De este modo, en él, se encuentran los coeficientes cuantificados obtenidos tras la etapa de transformación, la información necesaria para permitir al decodificador recrear la predicción adecuada, los metadatos acerca de la estructura de los datos comprimidos como las herramientas de compresión empleadas a la hora de codificar, y la información sobre la secuencia completa de video.

Una característica importante de las codificaciones estadísticas que emplea H.264/MPEG-4 AVC es que no tienen pérdidas y que por lo tanto siempre se recupera la información original al aplicar el proceso inverso. Los métodos que H.264/MPEG-4 AVC específica para poder codificar los símbolos, se basan en una codificación de longitud variable (CAVLC, *Context-Adaptive Variable Length Coding*) o en una codificación aritmética (CABAC, *Context-Adaptive Binary Arithmetic Coding*). Esto depende de la configuración empleada en el parámetro *entropy_coding_mode*. Es importante tener presente que, la compresión de información suele darse cuando se trabaja con un número de símbolos lo suficientemente grande a la entrada. De igual manera, cuantos más ceros se tengan a la entrada, más compacta va a ser la salida del codificador.

2.4 Métricas de calidad

La mayoría de los algoritmos que emplean los codificadores de video suelen tener pérdidas, y el estándar H.264/MPEG-4 AVC se incluye en dicho grupo de algoritmos (ver apartado 2.2). Esto hace que cualquier proceso de compresión que se le aplica a una imagen, pueda causar alguna perdida de información que afecte a la calidad de la imagen que se obtiene a la salida del *códec*. Por lo que, considerando que el objetivo a la hora de codificar una secuencia de imágenes consiste en la obtención de una correcta percepción visual, teniendo en cuenta el bitrate empleado a la hora de realizar dicha acción, existen ciertas métricas que se emplean para evaluar el resultado de comparar la calidad de las imágenes originales y las codificadas.

Cabe destacar que, la calidad de un video se considera una variable subjetiva y no se puede medir directamente, a diferencia de otros parámetros objetivos que sí se pueden, como es el caso del propio bitrate. Siendo esto así, existen diferentes métodos para poder medirlo de manera subjetiva. Uno de ellos podría ser el siguiente: mostrar a varios observadores una secuencia de video codificada, junto a su versión original, y que estos les asignen una puntuación a la codificada teniendo en cuenta la sensación visual con la que perciben ambas secuencias. Es obvio que, aunque se garanticen los resultados requeridos, este tipo de métodos pueden presentar muchos problemas prácticos, ya que se basan completamente en la percepción humana, no se persigue un criterio explícito a la hora de calificar y se consume mucho tiempo. Además, hay ciertos factores que pueden influir a la hora de obtener la medida: la actitud o el estado del espectador, el ambiente en el que se halla, el tiempo que permanezca observando…

Una alternativa ampliamente utilizada y mucho más eficiente, consiste en emplear ciertas medidas objetivas de calidad que puedan aproximarse a las subjetivas, las cuales se pueden clasificar en tres grandes grupos: métricas de referencia completa, métricas de referencia reducida y métricas sin referencia. Los resultados obtenidos para este proyecto están basados en un par de métricas objetivas de referencia completa y se caracterizan por requerir tanto las imágenes originales que forman un video como sus equivalentes codificadas. Estos algoritmos emulan el funcionamiento del ojo humano, realizando comparaciones que utilizan criterios numéricos explícitos entre ambas imágenes (las originales y las codificadas) [19]. La Proporción Máxima de Señal a Ruido (PSNR) y el Índice de Similitud Estructural (SSIM) son las dos métricas objetivas de referencia completa que se han empleado en este Trabajo Fin de Máster.

2.4.1 Peak signal-to-noise ratio

La Proporción Máxima de Señal a Ruido (PSNR, *Peak signal-to-noise ratio*) define la relación existente entre la máxima energía posible en una señal y el ruido que afecta a su representación fidedigna. Este estimador se representa en unidades logarítmicas, a causa del amplio rango dinámico que puedan tener muchas señales. Una de sus aplicaciones más conocidas se relaciona con la medición de la calidad de los archivos multimedia, siendo la imagen original equivalente a la señal y el error introducido debido a la compresión equivalente al ruido. Cabe destacar que, las dimensiones de ambas imágenes tienen que ser idénticas.

En la siguiente expresión matemática, se puede apreciar que, partiendo de una imagen de referencia “f” y una imagen comprimida “g”, ambas del mismo tamaño de M x N píxeles, cómo se realizaría el

cálculo del valor de la PSNR entre las dos imágenes. La mayor ventaja que ofrece esta métrica es su simplicidad, ya que deriva directamente del error cuadrático medio (MSE) entre los píxeles de ambas imágenes, el cual permite comparar los valores de los píxeles que forman la imagen original con sus equivalentes en la codificada, definiendo como error la diferencia entre los píxeles correspondientes de ambas imágenes.

$$\boldsymbol{PSNR = 20 * \log_{10}(\frac{MAX_f}{\sqrt{MSE}})} \quad \text{Ec. 9}$$

$$\boldsymbol{MSE = \frac{1}{mn}\sum_{0}^{m-1}\sum_{0}^{n-1}||f(i,j) - g(i,j)||^2} \quad \text{Ec. 10}$$

f:	Datos correspondientes a la imagen original
g:	Datos correspondientes a la imagen comprimida
m:	Número de filas en píxeles de f y g i representa el índice de la fila
n:	Número de columnas en píxeles de f y g j representa el índice de la columna
MAX_f:	El valor máximo de un píxel en la imagen original

Se observa que cuando el MSE tiende a 0, la PSNR tiende a ser infinito. Esto demuestra que cuanto mayor sea el valor de la PSNR, la calidad que proporciona la imagen comprimida es más alta. En cambio, un bajo valor de la PSNR implica grandes diferencias numéricas entre las dos imágenes comparadas.

Es verdad, que el uso de esta métrica no suele ser del todo recomendable en el ámbito de la codificación digital, ya que hay estudios que demuestran la baja correlación que presenta con las puntuaciones de calidad subjetivas previamente mencionadas. Sin embargo, la mayoría de estos estudios, se basan en secuencias de videos que cubren diferentes tipos de contenidos. Por otro lado, cuando la evaluación se realiza independientemente para cada tipo de contenido (sin mezclarlos en un único análisis), la situación cambia y se considera óptimo el uso de esta métrica [20]. Este proyecto se ha basado en analizar videos con diferente contenido separadamente, por lo que se ha visto oportuno utilizarla.

Por último, conviene comentar que, la gran limitación del PNR se considera que no estima las diferencias de percepción entre las dos imágenes de interés, es decir, no tiene en cuenta ningún factor biológico del sistema de visión humana y se basa estrictamente en una comparación numérica. Esto puede considerarse un punto a mejorar. De este modo, con el fin de respaldar los resultados y las conclusiones obtenidas, se ha visto interesante utilizar una segunda métrica que se caracteriza por no disponer de esta limitación: el Índice de Similitud Estructural (SSIM).

2.4.2 The Structural Similarity index

El índice de Similitud Estructural (SSIM, *Structural Similarity index*) es una métrica de calidad de imagen que se emplea para medir la similitud entre dos imágenes y, a diferencia de la PSNR, está correlacionado con la percepción de calidad del Sistema Visual Humano (HVS). Es preciso señalar que, el SSIM se fundamenta en la hipótesis de que el HVS está altamente adaptado para extraer información estructural y considera que la degradación de una imagen como un cambio percibido en la información estructural. Antes de avanzar, se puede definir qué, la información estructural se basa en la idea de que existen fuertes interdependencias entre los píxeles que se encuentran espacialmente cercanos [21].

Siendo esto así, en lugar de utilizar métodos tradicionales que se basan en la suma de errores, el SSIM está diseñado para modelar cualquier distorsión en una imagen, mediante una combinación multiplicativa de tres términos: la distorsión de luminancia, la distorsión del contraste y la estructura (o pérdida de correlación). A continuación, se puede observar la expresión matemática que resume lo descrito [22]. Es verdad que en un primer vistazo dicha ecuación pueda parecer simple, sin embargo, está demostrado que el SSIM predice objetivamente puntuaciones subjetivas tan bien como algunos algoritmos sofisticados de evaluación de calidad.

$$\boldsymbol{SSIM(x,y) = l(x,y)^{\propto} * c(x,y)^{\beta} * s(x,y)^{\gamma}} \qquad \text{Ec. 11}$$

Por un lado, X e Y hacen referencia a dos imágenes y se representan en forma de matrices. Por otro lado, x = {xi | i = 1,2,3….,N} e y = {yi | i = 1,2,3….,N} son bloques de píxeles de dimensión N x N que se encuentran ubicados en las mismas posiciones espaciales en las imágenes X e Y, y se representan en forma de sub matrices. De este modo, se contempla que, la luminancia se mide haciendo uso del promedio de los píxeles de interés (μ_x y μ_y), la a distorsión del contraste teniendo en cuenta la desviación estándar (σ_x y σ_y) y la estructura basándose en la correlación entre las dos señales (σ_{xy}).

$$\boldsymbol{l(x,y)\frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}} \qquad \text{Ec. 12}$$

$$\boldsymbol{c(x,y)\frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}} \qquad \text{Ec. 13}$$

$$\boldsymbol{s(x,y)\frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}} \qquad \text{Ec. 14}$$

μx:	El promedio en un bloque de píxeles en la imagen X
μy:	El promedio en un bloque de píxeles en la imagen Y
σ_x:	La desviación estándar sobre un bloque de píxeles en la imagen X
σ_y:	La desviación estándar sobre un bloque de píxeles en la imagen Y
σxy:	La covarianza sobre un bloque de píxeles en la imagen X y en la imagen Y
x:	Bloque de píxeles en la imagen X
y:	Bloque de píxeles en la imagen Y
C1 C2 C3:	Constantes
α, β, γ:	Pesos para cada término

C1, C2 y C3 son constantes que se introducen para evitar inestabilidades que se pueden dar cuando ($\mu_x^2 + \mu_y^2$), ($\sigma_x^2 + \sigma_y^2$) y ($\sigma_x \sigma_y$) se acercan a 0. Por lo que, asignándole un valor de C2/2 a C3 y estableciendo un mismo peso a los tres términos (α = β = γ =1) se obtiene la siguiente expresión matemática para calcular el SSIM:

$$\boldsymbol{SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}} \qquad \text{Ec. 15}$$

Esta métrica se caracteriza por adquirir un valor decimal que se encuentra entre el rango de 1 y -1. Un valor de SSIM = 1 se obtiene únicamente cuando los datos correspondientes a las dos imágenes que se han comparado sean idénticas, indicando que se dispone de una similitud estructural perfecta. En cambio, un SSIM = 0 indica que no existe ninguna similitud estructural entre las dos imágenes.

2.5 Rate control

El *Quantification Parameter* (QP) es el parámetro que regula la cantidad de información espacial que se elimina a la hora de codificar un archivo multimedia. El *rate control* es el elemento que se encarga de ajustar dinámicamente el valor del parámetro QP en el codificador, con el fin de establecer un bitrate apropiado [2]. Además, soluciona el dilema del huevo y de la gallina presente en los codificadores de video, ya que cumple con la función de asignar una cantidad de bits a cada grupo de imágenes (GOP, *Group of Pictures*), imágenes individuales, *slices* o macrobloques, sin conocer de antemano el número de bits requeridos para codificar el flujo de entrada. Por lo que, el *rate control* es un elemento que no se encuentra dentro del estándar H.264/MPEG-4 AVC.

Siendo esto así, el parámetro QP se puede establecer en un valor que se encuentra dentro del rango de valores entre 0 y 51. Observando la siguiente figura, se puede apreciar que utilizando bajos valores de QP se consigue conservar casi toda la información a la hora de codificar, a expensas de una mínima reducción en el bitrate. Por otro lado, al aumentar el valor de QP, se disminuye el valor del bitrate requerido, a costa de una reducción en la calidad respecto a la imagen original. Cabe destacar que, la relación entre la calidad esperada y el bitrate empleado también depende de la complejidad que presenta la imagen que se quiera codificar. Por consiguiente, el tipo de imagen es un factor que hay que tener en cuenta a la hora de codificar una secuencia de video y asignar un valor al parámetro QP.

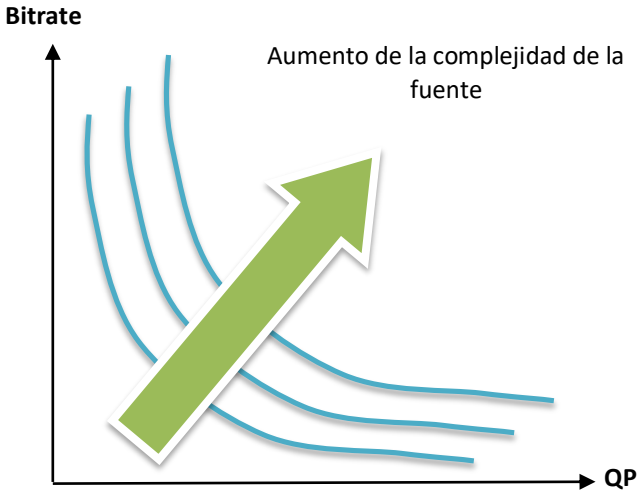


Figura 22: Evolución de la calidad de la imagen codificada

Independientemente del *códec* empleado a la hora de codificar (H.264, HEVC, VP9…), suele ser importante seleccionar el método de codificación que se quiere utilizar. Existen dos métodos principales para codificar video: *Constant bitrate* (CBR) y *Variable bitrate* (VBR). Si se emplea una codificación CBR, se puede predecir el ancho de banda necesario en la red y esto facilita administrar los recursos disponibles en dicha red. No obstante, si la complejidad de una escena aumenta, la calidad de la secuencia puede degradarse sensiblemente. En cambio, el empleo de una codificación VBR permite establecer un nivel predefinido de calidad de imagen, independientemente de la complejidad que caracteriza la escena, a costa de que el consumo del ancho de banda en la red aumente o disminuya en función del bitrate requerido para garantizar dicha calidad. Existen ampliaciones del VBR donde el bitrate puede variar, pero sin superar unos valores límites, garantizando un rango operativo.

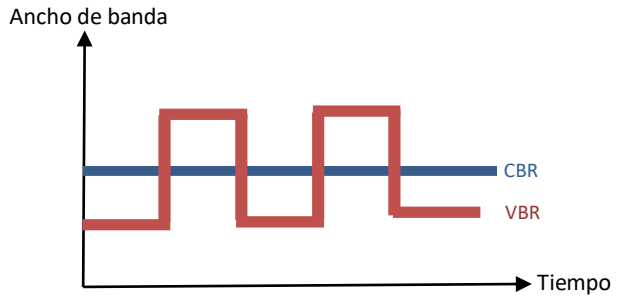


Figura 23: Comportamiento del ancho de banda en los modos CBR y VBR.

La Figura 23 muestra una codificación en lazo abierto, también conocido como *Variable bitrate*. Se ve que se tienen dos entradas, una secuencia video en formato *raw* y un valor de QP. En este caso, se obtiene una calidad constante dependiendo del QP que se haya empleado. No obstante, si la complejidad entre las imágenes que forman la secuencia varía mucho, el bitrate también puede variar notablemente. Esto no es algo que siempre interesa ya que pueden existir limitaciones en el bitrate, impuestos por el tamaño del búfer en el decodificador o por el ancho de banda en la red. Por consiguiente, se puede ver necesario codificar con un bitrate que sea mínimamente constante.

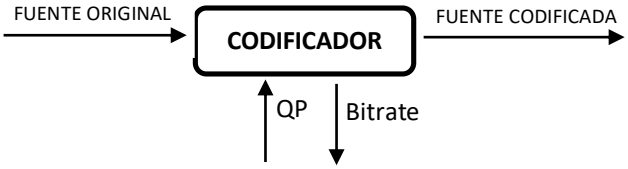


Figura 24: Codificación en lazo abierto (VBR).

En la Figura 24 se puede observar el modo de codificación en lazo cerrado, también conocido como *Constant bitrate*. En este caso, también se tienen dos entradas, un archivo video en formato *raw* y un valor de bitrate pretendido. Cabe destacar que, en este caso se analiza la complejidad de las imágenes (o grupos de imágenes) de la secuencia de video que se tiene a la entrada y en función de ello y del bitrate introducido como entrada, el valor de QP varía, obteniendo una mejor asignación de bits en el codificador. A continuación, se analiza este modo más a fondo.

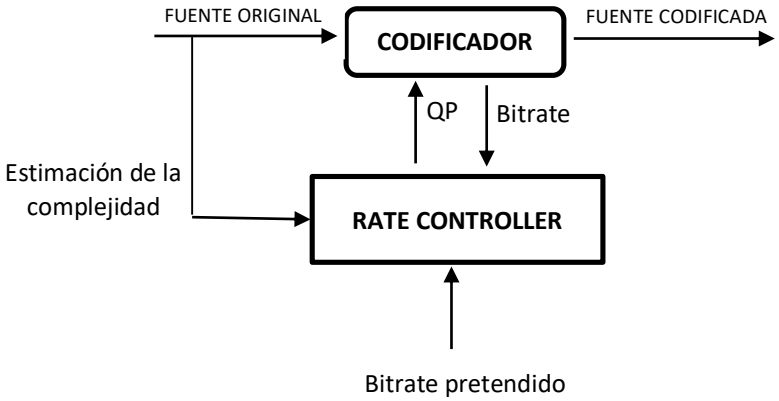


Figura 25: Codificación en lazo cerrado (CBR).

En la Figura 25, se puede observar una posible arquitectura del *rate control*, donde están presentes los bloques más importantes por los que un controlador del bitrate suele estar formado. Es verdad, que se trata de una representación conceptual y no de una representación literal de alguna

implementación de un software dado, ya que hay detalles que se pasan por alto (por ejemplo, que las imágenes B y P se tratan de diferente manera…). Sin embargo, la mayoría de estos elementos se consideran comunes en otros esquemas de *rate control*.

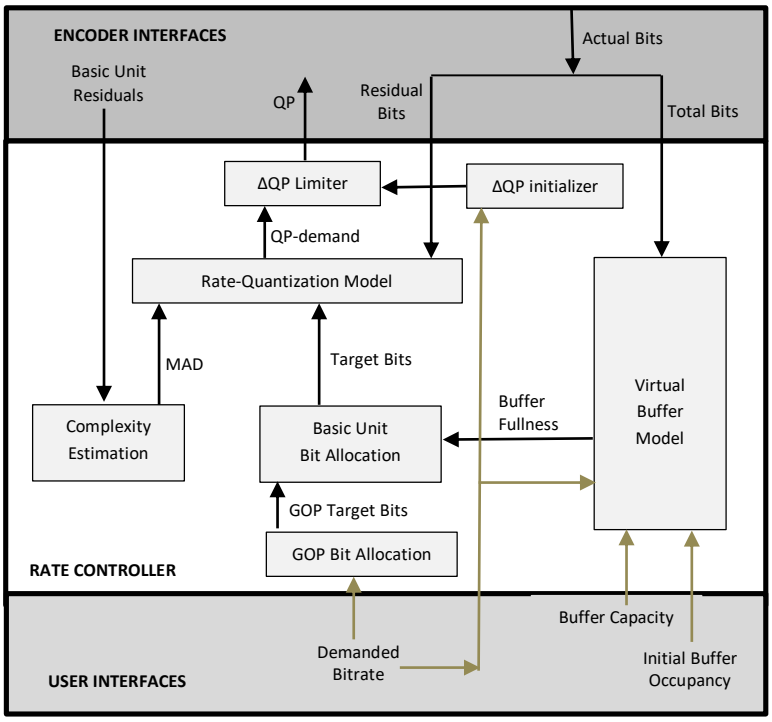


Figura 26: Elementos del *Rate Controller* de H.264 [2]

- Modelo de cuantificación. Se trata de la implementación de un modelo cuantitativo que intenta emular lo que se ha descrito la Figura 21. Se considera el corazón del algoritmo ya que relaciona el parámetro QP, con el bitrate y la complejidad de la imagen.
- Estimador de la complejidad. Realiza el cálculo de la complejidad de codificación asociada con los residuos a nivel de imagen, macrobloque…
- Limitador del QP. En las secuencias de video que presentan constantes cambios de complejidad entre las imágenes, el valor del parámetro QP puede oscilar notablemente. Por lo que para garantizar una estabilidad y minimizar las variaciones perceptibles en la calidad, se emplea un limitador de QP que fija el cambio máximo del QP entre dos imágenes consecutivas en ± 2.
- Modelo de buffer virtual. Un decodificador suele estar equipado por una memoria que se emplea para suavizar las variaciones en el tiempo de llegada de los datos entrantes. Este bloque se encarga de simular esta memoria, ya que idealmente un codificador debe producir un flujo de bits que satisfaga las restricciones del decodificador.
- Inicializador del QP. Se trata del elemento que se encarga de inicializar el parámetro QP al inicio de una secuencia de video.
- Asignador de bits al GOP. Se determina un bitrate objetivo para el GOP a partir del bitrate pretendido y de lo lleno que se encuentra el buffer virtual.
- Asignación de bits a la unidad básica. Se determina la cantidad de bits que hay que emplear a nivel de macrobloque, *slice*, imagen…

3 Metodología

En este capítulo se explica la metodología implementada para desarrollar el sistema propuesto en los objetivos. Esto incluye las fases que se han llevado a cabo tanto a la hora de completar la tabla de búsqueda que asocia los umbrales de calidad deseadas con los bitrates necesarios para alcanzar dichas calidades para cada tipo de imagen analizada, como en el diseño del propio sistema. Asimismo, se examinan las tecnologías que se han necesitado a lo largo del proyecto, se habla sobre el conjunto de datos que se ha formado para llevar a cabo el proyecto y mediante un diagrama Gantt se visualizan los hitos que se han ido cumpliendo en el tiempo que ha durado el proyecto. Se recomienda leer el capítulo anterior, correspondiente al contexto tecnológico, antes de continuar con esta sección.

3.1 Descripción de las fases y tareas realizadas

En este apartado se abordan las fases por las que ha transcurrido el proyecto, analizando fundamentalmente las diferentes tareas que se han realizado en cada una de esas fases. De este modo se comenta la situación inicial al principio del proyecto y la evolución que se ha llevado a cabo para su culminación.

3.1.1 Fase I. Concepción e iniciación del proyecto

La primera tarea de esta fase I se centró en asentar una base sólida de los conceptos y tecnologías de la codificación de video. Para ello, se realizaron reuniones con el tutor del centro, en las que se habló sobre cuáles serían los conceptos clave en las que sería importante enfocarse, además de que se facilitara cierta bibliografía para que el estudio fuera lo más eficiente posible.

Esta tarea se basó básicamente en ir asimilando poco a poco toda la información que se ha comentado en el segundo capítulo de esta memoria. Es decir, se vio imprescindible entender la razón por la que es necesario codificar los flujos de video antes de transmitirlos o almacenarlos, investigar acerca del motivo por el que no se utiliza el modelo RGB a la hora de codificar, analizar los bloques básicos del estándar H.264/MPEG-4 AVC y entender la función que cumple cada uno de ellos. Para después buscar información acerca de la codificación *per-title* y comprender que la relación existente entre el bitrate y la calidad también depende de otros factores como la complejidad de la imagen, familiarizarse con la existencia de las distintas métricas que se emplean para cuantificar la percepción visual que presentan los videos, o ser consciente de la importancia que tiene el *rate control* (el dilema de la gallina y el huevo), entre otras cosas.

Por otro lado, la obtención de los rangos operativos, los cuales debían conseguir asociar las calidades deseadas con sus respectivos bitrates, se ha basado en el *framework* multimedia FFMPEG. Por consiguiente, se vio necesario descargar e instalar esta herramienta, para después documentarse acerca de las opciones que ofrecía y recopilar información acerca de las opciones que pudieran ser útiles para satisfacer las necesidades del proyecto, además de aprender a manejarlo. Para ello, se recurrió fundamentalmente a la documentación [23] que está accesible para cualquier usuario en la web.

En esta fase se realizaron los primeros *tests* con FFMPEG, se investigó sobre cómo lanzar conversiones de video y se inspeccionaron los parámetros que se podían configurar a la hora de codificar un flujo multimedia. Cabe destacar que, FFMPEG es compatible con la mayoría de los *códecs* que se emplean

hoy en día y como se ha comentado en más de una ocasión, este proyecto se ha basado únicamente en el estándar H.264/MPEG-4 AVC. Por lo que, empleando dicho códec, se realizaron un sinfín de pruebas codificando varios flujos con diferentes bitrates, empleando modos de codificación CBR o VBR, analizando lo que ocurría en caso de emplear una codificación VBR con una segunda pasada o comprobando como varía la relación entre el bitrate y la calidad, dependiendo de la resolución en la que se codifica o el tipo de imagen con la que se trabaja.

Asimismo, se tuvieron que decidir las métricas que iban a ser utilizadas para cuantificar los umbrales de calidad. Para ello, se realizó un análisis de varios proyectos y artículos publicados [19], con el fin de verificar si las métricas que se tenían identificadas eran adecuadas y ver si era posible emplearlas en este ámbito. Se comprobó que la PSNR era la métrica referencia, con sus pros y sus contras (ver sección 2.4), y que la mayoría de las investigaciones analizadas se fundamentaban en esta métrica. Además, se consideró que el uso del SSIM como métrica secundaria podría ser una buena opción para validar los resultados obtenidos. En este sentido, se vio que FFMPEG ofrecía la opción de calcular estas dos métricas de una forma directa, por lo que no se tuvo ningún problema a la hora de obtener estos valores. Sin embargo, cabe mencionar que, también se percibió que FFMPEG ofrecía más de una alternativa para calcular el valor de estas dos métricas, donde se pueden destacar dos de ellas. A continuación, se muestran varios ejemplos para calcular la PSNR mediante FFMPEG.

1. El codificador realiza el cálculo mediante la instrucción *–psnr* y se obtiene el resultado en la misma sesión de codificación.

```
ffmpeg -i input.mp4 -vf -codec:v libx264 -psnr -f mp4 -b:v 100k -pass 1 output.mp4
```

2. El codificador codifica el archivo de entrada, se muestrea el archivo codificado para que tenga la misma resolución que la original (necesario para realizar la comparación para el cálculo de las métricas) y después se hace uso del filtro *psnr* para calcular el resultado, comparando el archivo original con la codificada.

```
ffmpeg -i input.mp4 -vf -codec:v libx264 -f mp4 -b:v 100k -pass 1 output.mp4
ffmpeg -i output.mp4 -y -pix_fmt yuv420p -vsync 0 -s 1920x1080 -sws_flags lanczos output.y4m
ffmpeg -i output.y4m -i input.mp4 -filter_complex psnr -f null –
```

Se comprobó que los resultados obtenidos no eran idénticos y que el segundo modo no cuadraba con la evolución que debían tener las métricas (la PSNR y el SSIM) cuando se codificaba un flujo en diferentes resoluciones. Esto estaba relacionado con el muestreo necesario para igualar el tamaño de imagen antes de obtener las métricas. Por lo que se decidió emplear la primera opción en los *scripts* que se realizarían más adelante.

Tengo que decir que, una tercera medida fue implementada en un principio para refrendar todavía más los resultados obtenidos. La medida corresponde a una métrica desarrollada por Netflix, conocida como VMAF (*Video Multi-Method Assessment Fusion*) [24], la cual se basa en un algoritmo que evalúa la calidad teniendo en cuenta la percepción visual humana. Sin embargo, se descartó el uso de esta métrica debido a la imposibilidad de realizar su cálculo con secuencias que se encontraban en resoluciones diferentes, sin antes muestrearlos. Dado que además de una métrica principal (PSNR) se tenía otra de respaldo (SSIM), se consideró que no era tan necesario disponer de una tercera métrica para el avance del proyecto.

Por otra parte, desde el principio se consideró fundamental definir un conjunto de datos (*dataset*) con las muestras que se iban a procesar, con el fin de que los datos obtenidos fueran los más coherentes posibles para los intereses del proyecto. El *dataset* creado se clasificó en 6 categorías y cada una de las muestras que lo forman el conjunto de datos final pertenece a una de las categorías, dependiendo del contenido por la que se caracteriza. Cada categoría equivale a un tipo de imagen y se pueden resumir en las siguientes:

	Tipo de imagen
1	Entretenimiento
2	Conducción
3	Realidad aumentada
4	360º
5	CCTV
6	Producción

Tabla 1: Tipos de imágenes analizados en el proyecto

Por cada categoría se han procesado 3 muestras, siendo un total de 18 muestras analizadas. En la sección 3.3 se puede encontrar más información acerca del conjunto de datos y las muestras que lo forman. Observando la siguiente figura se puede hacer una idea del tipo de imágenes que se han empleado.

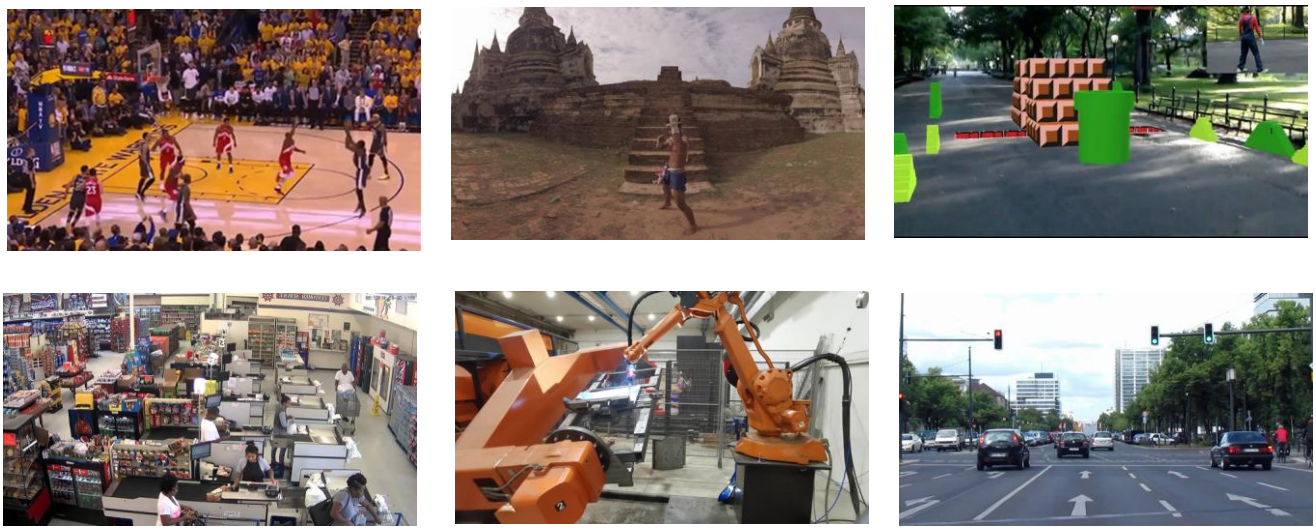


Figura 27: Tipos de imágenes analizados: entretenimiento, 360°, realidad virtual (arriba), CCTV, producción, conducción (abajo)

Antes de pasar a la segunda fase del proyecto y después de analizar los diferentes parámetros de interés que se pudieran calcular mediante FFMPEG, se diseñó un script que se emplearía para procesar las muestras del *dataset* de forma individual, con el fin de obtener los datos, que resultarían de interés en el desarrollo del proyecto, sobre cada muestra. De este modo, además de los valores de las métricas PSNR y SSIM, se consideró necesario anotar tanto el bitrate empleado para alcanzar dichas calidades deseadas por el codificador, como los valores correspondientes a la entropía y al ratio de cambios de escena para parametrizar las diferencias existentes relativas a cada tipo de imagen. Estos dos últimos parámetros han servido para poder aproximar la complejidad espacial y la temporal.

Es digno de señalar que, mientras que FFMPEG permite realizar el cálculo de las métricas de calidad y del bitrate directamente, para obtener los valores referentes a la complejidad se han tenido que hacer unos cálculos adicionales. En primer lugar, la entropía de la secuencia de video se ha calculado obteniendo la entropía de cada *frame* que forma una secuencia de video, haciendo uso del filtro "*entropy*", y realizando posteriormente el promedio entre las entropías calculadas correspondientes a todas las imágenes. Para realizar este cálculo se consideraron únicamente las imágenes en escala de grises (componente Y), con el fin de eliminar las influencias del color.

En segundo lugar, el ratio de cambios de escena se ha calculado obteniendo el número de cambios de escenas mediante la combinación de los filtros "*scene*" y "*showinfo*", y dividiendo dicho valor con la duración del video. Es interesante comentar que, el filtro "*scene*" se basa en la extracción y en el recuento de todas las imágenes que forman una secuencia de video y que difieren del *frame* anterior en dicha secuencia en más de un valor previamente indicado. A continuación, se muestra el uso básico de los dos filtros mencionados.

```
ffmpeg -i input.mp4 -filter:v entropy,metadata=mode=print:file=aEntropy.log -f null -
```

```
ffmpeg -i input.mp4 -filter:v "select='gt(scene,0.4)',showinfo" -f null -
```

Además, para poder diseñar el script, había que tener en cuenta los datos que se necesitarían para generar las gráficas que servirían para sacar las conclusiones necesarias en la siguiente fase. Se concluyó que, con los datos obtenidos de cada muestra del conjunto de datos que había que procesar, se dibujarían unas gráficas donde se representarían las métricas de calidad (la PSNR y el SSIM) en función del bitrate empleado para alcanzar dichas calidades, quedando reflejado así la relación existente entre ambos parámetros (fíjese en cada uno de los puntos de la gráfica de la Figura 28). Igualmente, a la hora de codificar se tuvo que tener en cuenta la resolución en la que se codificaría el flujo, para examinar la variación que se daba en la relación entre la calidad deseada y el bitrate a la hora de codificar dependiendo del tamaño de la imagen. Sobre esto se habla con más profundidad en el siguiente punto.

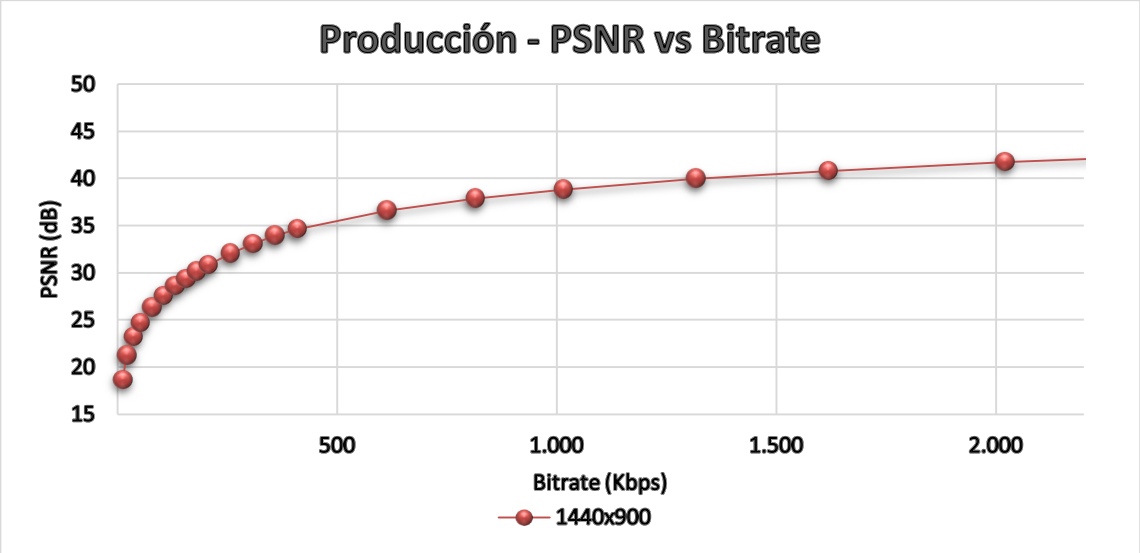


Figura 28: Ejemplo de una gráfica obtenida a partir de los datos obtenidos mediante FFMPEG, correspondientes a una muestra dentro de la categoría de producción.

Por último, después de haber elegido como la métrica referencia la PSNR (y como métrica de respaldo el SSIM) se tuvo que decidir cuáles debían ser los umbrales de alta, media y baja calidad. Es decir, había que definir tres perfiles para cada tipo de imagen, mapeando los valores adecuados de la PSNR, con los tres rangos de calidades subjetivas de video en la que se basa el proyecto. Para establecer estos límites, inicialmente se realizó una búsqueda en la literatura. Se encontraron artículos y publicaciones que se basaban en el cálculo de la calidad subjetiva. Es decir, trataban de investigaciones realizadas teniendo en cuenta la valoración de un conjunto de espectadores aleatorios que tenían como tarea puntuar de una manera subjetiva diferentes muestras de video, para que posteriormente se diseñasen tablas que servirían para realizar los mapeos entre las métricas objetivas (PSNR, SSIM…) y las subjetivas [25, 26, 27, 28]. También se analizaron artículos que se basan en realizar sus propias investigaciones fundamentándose en el mapeo entre las calidades objetivas y subjetivas que se pueden encontrar en la literatura [29]. Teniendo en cuenta todos estos datos pertenecientes a los artículos citados, se estableció la siguiente tabla con umbrales que serviría para diferenciar los tres perfiles de interés.

Calidad de video	PSNR
Alta	**> 40 dB**
Media	**≈ 38 dB**
Baja	**< 35 dB**

Tabla 2: Mapeo de las calidades deseadas con los umbrales de la PSNR

Obsérvese que los valores encontrados en estas fuentes no están estandarizados y varían entre ellos, ya que pertenecen a investigaciones realizadas en entornos independientes. Por lo que, con el propósito de fortalecer el uso de estos valores como umbrales y analizar su coherencia, se consideró como buena opción establecer los umbrales definitivos teniendo en cuenta una metodología visual. Para poder implementar este proceso se vio necesario disponer de las gráficas generadas a partir de los datos obtenidos en FFMPEG, para ver las relaciones existentes entre las PSNR y los bitrates correspondientes a las muestras del conjunto de datos. Siendo esto así, esta tarea se finalizó en la siguiente fase.

3.1.2 Fase II. *Tests* de caracterización y establecimiento de las tablas de búsqueda

Una vez concluidas las labores correspondientes a la fase I, el proyecto se plantó ante lo que se considera su núcleo. En esta fase II se tuvo que materializar todo lo aprendido e diseñado previamente, con el fin de generar unas tablas de búsqueda en las que partiendo de unos parámetros de entrada (valores objetivos de PSNR), se supiera que parámetros de salida se tendrían que utilizar (valores de bitrate) a la hora de realizar las codificaciones de diferentes tipos de video.

En primer lugar, se mejoraron y se automatizaron los *scripts* diseñados en la fase anterior para que en una única sesión se obtuviesen todos los datos correspondientes a cada muestra del *dataset*. Desde un punto de vista realista, era imposible abarcar todo el rango de bitrates y de resoluciones, por lo que se establecieron 6 resoluciones estándares y un conjunto de 22 posibles valores de bitrate que se emplearían a la hora de codificar las muestras. Dicho de otro modo, se estableció que cada muestra

se codificaría en 6 distintas resoluciones, dónde en cada una de ellas se codificaría el flujo original con 22 valores de bitrate diferentes, siendo un total de 132 codificaciones realizadas por cada muestra del *dataset*. De este modo, en cada uno de ellos se obtendrían valores independientes de las métricas de PSNR y de SSIM, que servirían para dibujar las gráficas necesarias. Con la adquisición de esta gran cantidad de datos, se quiso evitar que las curvas que se dibujarían posteriormente dispusieran de suficientes muestras como para aproximar una línea continua.

Resoluciones empleadas
1920 x 1080
1440 x 900
1080 x 720
720 x 480
512 x 384
382 x 288

Tabla 3: Resoluciones analizadas a lo largo del proyecto

Conjunto de bitrates (Kbps)					
10	20	35	50	75	100
125	150	175	200	250	300
350	400	600	800	1.000	1.300
2.000	2.500	3.000			

Tabla 4: Conjunto de bitrates empleados a la hora de codificar las muestras

Igualmente, se pensó que podía ser una buena idea elevar el nivel de automatización diseñando un script que al lanzarlo pudiese recorrer todos los archivos que formaran el conjunto de datos, generando ficheros con nombres propios por cada muestra procesada. Con el fin de extraer y almacenar únicamente los datos de interés que se generasen en el *output* de FFMPEG, se emplearon las siguientes utilidades de la línea de comandos:

- grep (Globally search for a Regular Expression and Print matching lines)
- sed (Stream Editor)
- Awk

Por consiguiente, se puede decir que la tarea correspondiente al procesado de las muestras se basó en extraer y recopilar los datos de interés de diferentes ficheros externos con un formato que permitía importar directamente dichos datos almacenados a un fichero Excel para que posteriormente se ordenasen los datos y se generasen las gráficas. En el capítulo correspondiente a los resultados se describen y se analizan las gráficas generadas a partir de los datos obtenidos por FFMPEG, por lo que en esta sección de la memoria no se van a examinar estás gráficas.

Una vez que las gráficas estaban generadas (aunque no fueran las definitivas) y los datos almacenados de una manera ordenada, se retomó la idea de implementar una metodología visual que permitiese comprobar si los umbrales de las métricas de calidad establecidas acorde a las encontradas en la literatura [25, 29] se consideraban coherentes o no, además de ajustarlos si se viese necesario. De este modo, se creó un nuevo script que hacía uso de FFMPEG para implementar la siguiente metodología visual que cumpliese con ciertos requisitos:

1. Estimación de la duración ideal de la visual.
   Se debía tener en cuenta que cuanto menor fuera la duración de la visual, más probabilidades habría de que el nivel de percepción del espectador no se viera afectado por varios factores como el cansancio o el estado de ánimo. Por lo que se decidió que el video debía durar menos de 3 minutos, con el fin de poder visualizarlo concentradamente en una única sesión.

2. Cálculo de la duración real de la visual.
   Considerando que se partía de un *dataset* formada por 18 muestras, donde cada una de ellas se codificaba en 6 distintas resoluciones y dependiendo del bitrate empleado se obtenía un valor de PSNR distinto, en este caso interesaba visualizar por cada resolución únicamente los archivos codificados que se caracterizaban por tener un valor de PSNR cercana a los umbrales (35 dB, 38 dB y 40 dB). Aun así, si se hubiera extraído un único segundo por cada archivo de interés se obtendría una visual de 5.4 minutos (18 muestras * 6 resoluciones * 3 bitrate * 1 segundo). Por lo que se consideró una opción no viable debido a la condición impuesta en el punto 1.

3. Selección de los fragmentos para la visual.
   Viendo que era necesario recortar la duración de la visual se consideraron las siguientes 3 medidas:
   - Se visualizarían los videos codificados en 3 resoluciones (1080x720, 720x480, 382x288)
   - No se visualizarían las 18 muestras del *dataset*. La visual estaría formada por 10 muestras, donde habría videos de todas las categorías.
   - Se extraerían 2 segundos de cada video codificado para insertarlos en la evaluación visual.

   De esta manera se creó un vídeo con diferentes cortes de las muestras codificadas para realizar una evaluación visual rápida en 3 minutos, en la que la que los primeros 60 segundos correspondían a las secuencias de alta calidad, los siguientes 60 segundos a las secuencias de media calidad y los últimos 60 segundos a las secuencias de baja calidad. Esto permitiría observar la coherencia en la calidad subjetiva de cada nivel para todos los tipos de secuencia.

4. Extracción de los fragmentos para la visual.
   Cabe destacar que, antes de extraer las imágenes de un archivo codificado, se realizaba la búsqueda de un *frame* I a mitad del video y se extraían los dos segundos posteriores a partir de la marca de tiempo de dicho *frame* I.

Después de generar, reproducir y evaluar atentamente la visual:

- Se tuvo una percepción homogénea durante cada uno de los tres tramos de 40 segundos que duraba la secuencia, donde cada tramo se caracterizaba por ofrecer video de una calidad concreta (alta, media y baja).
- Se percibían diferencias entre las secuencias de los distintos tramos.

Es verdad que puede haber mejores metodologías para ajustar los umbrales, sin embargo, la que se utilizó fue diseñada con el propósito de compensar el esfuerzo que se tendría que haber hecho a la

hora de evaluar la visual si se hubieran reproducido todas las opciones posibles con sus duraciones completas (más de 2 horas de duración). Siendo esto así, se dieron por válidos los umbrales que se habían establecido anteriormente y se generaron las gráficas que contenían los tres perfiles de calidad para cada tipo de contenido mediante los datos que se habían volcado al Excel Igualmente, es importante añadir que, las curvas referentes a cada tipo de contenido están dibujadas a partir de los datos de las muestras que completan cada categoría (3 muestras por cada categoría como se ha comentado previamente), por lo que con el fin de realizar los cálculos entre dichas muestras se tuvo que tener cuidado a la hora de realizar transformaciones entre la escala lineal y la logarítmica. . A continuación, se puede apreciar una de estas gráficas.

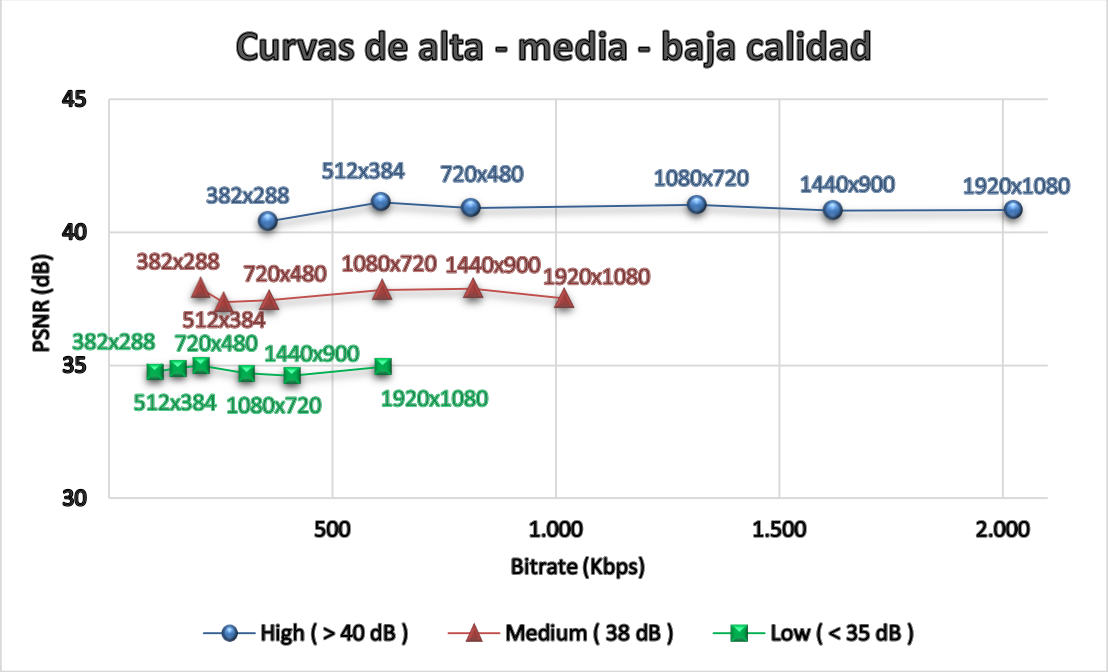


Figura 29. Ejemplo de una gráfica con las curvas de alta/baja/media calidad.

3.1.3 Fase III. Configuración dinámica y desarrollo

Después de definir los márgenes de actuación y representar las curvas correspondientes a cada tipo de imagen, el proyecto se encontraba ante su última fase. En este punto se pretendía integrar todo lo establecido previamente en una aplicación de GStreamer desarrollada en C, que dinámicamente tomase el valor adecuado del bitrate dentro de los rangos de operación preestablecidos. Se decidió que el programa tendría que obtener desde un fichero externo la calidad que se deseaba alcanzar y el tipo de imagen con la que se quería trabajar, y a partir de esta información obtener el bitrate.

Antes que nada, se tuvo que instalar GStreamer y familiarizarse con el *framework*, para entender en la mejor medida posible cómo se desarrollan las aplicaciones multimedia en este entorno y averiguar cómo se creaba un *pipeline*. Para ello, se hizo uso de la documentación que está disponible en la web [30] y se analizaron algunos tutoriales que se ofrecían en él, tanto los prácticos como los teóricos. En este sentido, se aprendieron conceptos relacionados con los *pipelines* dinámicos, el manejo de las señales y de los eventos, la función que cumplen los *pads* o la necesidad de los *probes* a la hora de detectar los *key-frames* o eventos, entre otras cosas.

A la hora de comenzar a desarrollar la aplicación, el primer paso consistió en crear un *pipeline* por el que viajaría el flujo de interés. En él, los elementos principales serían una fuente y un codificador H.264/AVC. Una vez creada el *pipeline* fue imprescindible entender bien la configuración dinámica de los atributos que pertenecieran a un elemento que se encontrara dentro del *pipeline*. Para ello, inicialmente se investigó sobre el uso de los temporizadores para la actualización de los atributos mediante los *callbacks* que generasen, haciendo uso de la función *g_timeout_add()* que facilita GStreamer. Empleando esta función, se pretendió desarrollar una aplicación que cada cierto tiempo preestablecido (parámetro de entrada del temporizador), leyera un fichero externo donde se tenían almacenados el tipo de imagen que se quería codificar y el perfil de calidad que se quería alcanzar, y fuera capaz de actualizar el bitrate en el elemento codificador que se tenía en el *pipeline* de forma dinámica.

Una vez que se logró implementar la configuración dinámica del *pipeline* mediante los temporizadores, la siguiente tarea se fundamentó en entender el funcionamiento de los eventos en GStreamer, ya que en el proyecto no interesaba el uso de los temporizadores. En este sentido, en GStreamer los eventos son objetos que se transmiten en paralelo al flujo de datos (*buffers*) para notificar a los elementos que forman el pipeline de ciertas incidencias. En este caso, la incidencia se basaba en que se debía realizar un cambio de bitrate a la hora de codificar cuando se actualizaban los parámetros de interés.

Por un lado, se modificó el programa para que cuando se detectase que se quería cambiar de escenario (tipo de imagen y perfil de calidad deseado), se lanzase un evento personalizado al pipeline. Por otro lado, la aplicación diseñada debía ser capaz de procesar dichos eventos que se lanzaban al pipeline, para que cuando se lanzase un evento, la propia aplicación identificase dicho evento y realizase el cambio de bitrate al instante. Cabe señalar que, GStreamer facilita unas sondas que se añaden a los interfaces de los elementos que forman el pipeline, conocidas como *probes* y que sirven para analizar el flujo de datos, descartar buffers del flujo o escuchar eventos que se han lanzado al *pipeline*, entre otras funciones. Por lo que, para realizar la detección de los eventos lanzados al *pipeline* se utilizó un *probe*.

Por último, con el fin de evitar la aparición de artefactos en el propio flujo y evitar el mal funcionamiento del sistema, se vio necesario que el elemento codificador dentro del *pipeline* realizase el cambio del bitrate, cuando se disponía de un *key-frame*. Para ello, se decido añadir un segundo *probe* en el pipeline cada vez que se detectase un evento correspondiente al cambio de bitrate, con el propósito de analizar los buffers del flujo y actualizar el bitrate en el momento que el codificador tuviera un *key-frame*. Una vez que se hubiera realizado el cambio se eliminaba el *probe* y se dejaba de analizar el flujo, hasta que se detectase un nuevo evento.

En el capítulo 5 se proponen dos diagramas, uno de secuencias y otro de flujos para que se entienda mejor y de una manera más visual el funcionamiento de la aplicación desarrollada.

3.2 Tecnología utilizada

Esta sección se centra en describir los dos *frameworks* de multimedia que se han tenido que utilizar para poder desarrollar este TFM. Por un lado, a lo largo de las fases I y II del proyecto se ha hecho uso de *FFMEG*. Por otro lado, la tercera fase del proyecto se ha basado en la creación de una aplicación multimedia mediante GStreamer. A continuación, se van a comentar estos dos frameworks.

3.2.1 FFMPEG

FFMPEG es una herramienta de código abierto formado por un gran conjunto de bibliotecas y programas que permiten codificar, decodificar, transcodificar, multiplexar, transmitir, editar, o reproducir video, audio u otros archivos multimedia [23]. Este *framework* está diseñado para el procesamiento basado en línea de comandos y aunque originalmente fuera desarrollado para ser implementado en un entorno Linux, hoy en día se considera un software altamente portable que se puede compilar o ejecutar en varios sistemas operativos, incluyendo en Ubuntu, que es el sistema donde se ha desarrollado el proyecto

Figura 30: Logo de FFMPEG.

Cabe destacar que, se trata de una herramienta que es compatible con la mayoría de los *códecs* de audio o de video existentes, por lo que no ha habido ningún problema de implementar el códec H.264/MPEG-4 AVC. Asimismo, admite casi todos los tipos de contenedores que se emplean en la actualidad, de este modo en la fase de aprendizaje se hicieron pruebas con secuencias mp4, mkv, mp3…. Sin embargo todas las muestras vinculadas al proyecto eran del formato mp4. En cualquier caso, el contenedor no es algo que altere la utilidad del estudio del códec.

A continuación, se contempla un diagrama de bloques que describe los procesos de *encoding* que se han realizado a lo largo del proyecto de una forma simple. Partiendo de un archivo de entrada, *FFMPEG* hace uso de una biblioteca que contiene el *demuxer* (*libavformat*) para leer dicho archivo y obtener los datos codificados que se encuentran ordenados en paquetes. Estos paquetes, se introducen en el decodificador para obtener los *frames* descomprimidos (*raw* video/PCM audio/...) y se procesan de manera que se interese. Posteriormente el codificador se encarga de generar los paquetes codificados y finalmente el *muxer* crea el archivo de salida.

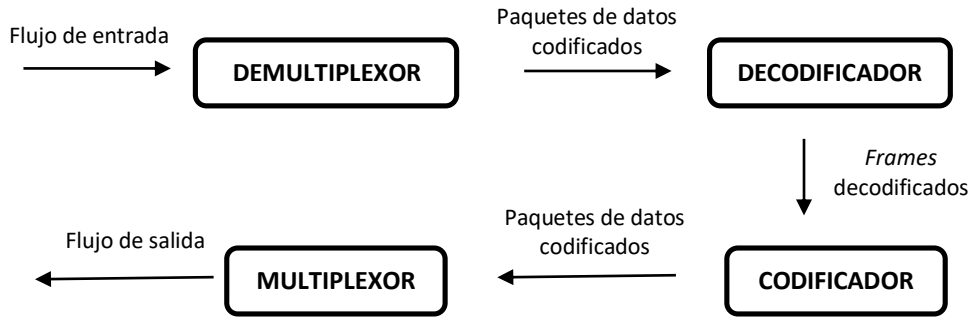


Figura 31: Proceso de codificación FFMPEG

FFMPEG ofrece al usuario la oportunidad de calcular y obtener ciertos parámetros en el proceso de codificación de un archivo multimedia, los cuales pueden ser de gran uso en un posterior análisis que pueda realizar dicho usuario. En este proyecto, se ha empleado este framework para obtener datos relacionados con la calidad de los videos que se han codificado (métricas de calidad: SSIM, PSNR, VMAF), el bitrate, el tamaño de los archivos, información acerca de los cambios de escena o la entropía de las imágenes que forman una secuencia de video.

3.2.2 GStreamer

GStreamer es un *framework* de multimedia [30], libre y multiplataforma, basada en tuberías para crear diferentes tipos de aplicaciones como editores de video, *transcoders*, *broadcasters* de *streaming* o reproductores de audio. En mi caso, GStreamer me ha valido para diseñar una aplicación basada en un pipeline que me permite cambiar el bitrate de un flujo de interés en tiempo real. Uno de los puntos fuertes del *framework* es que admite una amplia variedad de componentes conectables (*pluggable*) que sirven para manejar datos multimedia, los cuales pueden ser mezclados y combinados en tuberías arbitrarias.

Figura 32: Logo de GStreamer

GStreamer está diseñado para funcionar en diferentes sistemas operativos, incluyendo en Ubuntu, que ha sido el sistema operativo donde se ha desarrollado la aplicación. Cabe destacar que, el software emplea una arquitectura basada en *plug-in*s el cual permite que su funcionalidad sea implementada en librerías compartidas. Estas librerías pueden ser cargadas dinámicamente en el momento en el que sean necesarios, siendo capaz de este modo, de trabajar con un amplio espectro de *códecs*, en el que se incluye H.264/MPEG-4 AVC, formatos de contenedor, controladores de entrada/salida y efectos. Por otro lado, es obvio que esta modularidad aumenta la versatilidad a la hora de desarrollar nuevas aplicaciones. Los *plug-ins* de GStreamer podrían clasificarse de la siguiente manera:

- Protocolos de manejo
- Fuentes de audio y video
- *Códecs* (codificadores y decodificadores)
- Filtros (convertidores, mezcladores, efectos…)
- *Sinks* de audio y video

Asimismo, es interesante mencionar que GStreamer se adhiere a Gobject, el modelo de objetos GLib 2.0, y utiliza el mecanismo de las señales y de las propiedades de los objetos. Por consiguiente, todos los objetos de GStreamer se pueden ampliar mediante los métodos de herencia de GObject.

Un elemento se considera la clase de objetos más importante dentro de GStreamer y un *pipeline* suele estar compuesto por una cadena de elementos conectados entre sí (fuente, convertidor, codificador,

sink…*),* por el cual fluyen los flujos multimedia. Cada elemento suele cumplir con una función específica que puede ser la lectura de datos de un archivo, la codificación, la decodificación o la conversión, entre otros. De forma predeterminada, GStreamer dispone de una gran colección de elementos lo que hace posible el desarrollo de un gran abanico de aplicaciones.

Vale la pena señalar que, los *pads* son las entradas y las salidas de los elementos y sirven para realizar las conexiones entre diferentes elementos que forman un *pipeline*. Más concretamente, se utilizan para negociar los enlaces y los flujos de datos, y solo se enlazan dos elementos cuando los tipos de datos permitidos de sus *pads* sean compatibles.

Por último, en relación con el intercambio de datos y la comunicación entre la aplicación y el *pipeline* donde se encuentran los elementos, GStreamer proporciona varios mecanismos: buffers, eventos, mensajes, consultas. Por ejemplo, en la aplicación desarrollada se ha basado en realizar cambios de bitrate, y para ello se han tenido que detectar *key-frames* para realizar los cambios en ellos, evitando así artefactos. Para ello se ha tenido que hacer uso de los buffers.

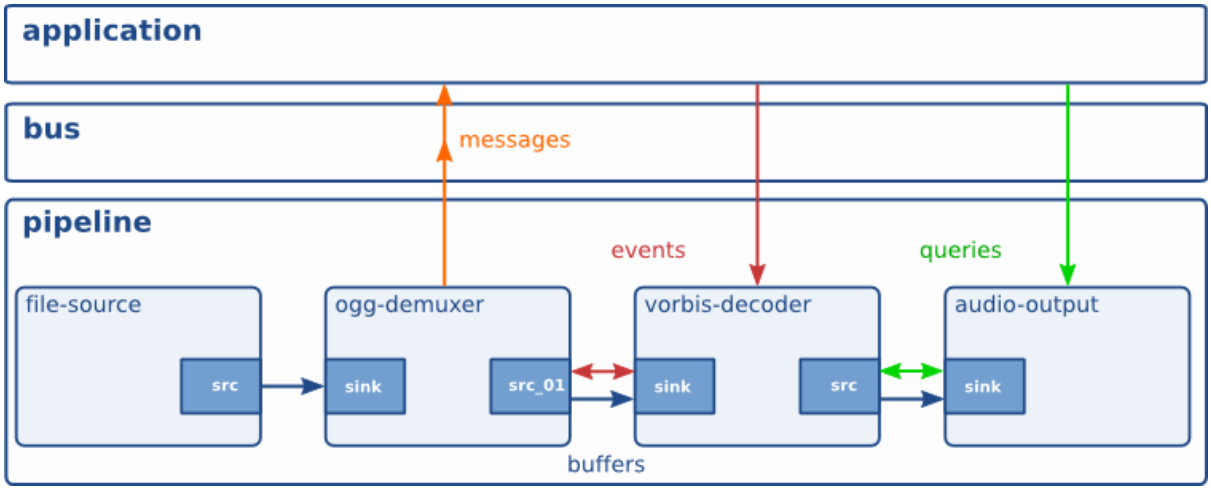


Figura 33: Ejemplo de un *pipeline* [24]

3.3 Formación del conjunto de datos

Esta sección describe cada una de las muestras empleadas para obtener los datos necesarios, a partir de los cuales se han representado las curvas correspondientes a los tres perfiles de calidad (alta, media y baja calidad) para cada tipo de imagen. Debido a la imposibilidad de incluir las secuencias completas en una memoria, se ha incluido un *frame* de cada una de las muestras empleadas.

El proyecto se ha basado en el análisis de 6 tipos de imágenes, donde por cada tipo de imagen se han procesado 3 muestra, siendo un total de 18 muestras las que completan el *dataset* final. En la siguiente tabla se resumen cuáles han sido estos 6 tipos de imágenes:

Tipo de imagen
Entretenimiento
Realidad aumentada
Conducción
CCTV
Producción
360°

Figura 34: Los 6 tipos de imágenes analizados en el proyecto

Cabe destacar que, las muestras empleadas no han sido secuencias estándares, sino que pertenecen a secuencias que han sido facilitadas por el propio centro u obtenidas directamente desde la web. Se proporciona un resumen donde se explican los contenidos que caracterizan cada tipo de imagen.

- **Entretenimiento**: las tres muestras empleadas corresponden a los *highlights* de un partido de fútbol, a un resumen de las mejores jugadas de un partido de baloncesto y a una escena de una película que se encuentra dentro del género de fantasía medieval.

- **Realidad aumentada**: las tres muestras empleadas corresponden a demostraciones de realidad virtual de Súper Mario Bros, de unos animales de un zoo y de una demo de Minecraft.

- **Conducción**: las tres muestras empleadas corresponden a secuencias de imágenes de prueba adquiridas desde un automóvil moviéndose en diferentes lugares y velocidades.

- **CCTV**: las tres muestras empleadas corresponden a fragmentos de videos de seguridad obtenidos desde unas cámaras implantadas en diferentes establecimientos (un supermercado, una oficina y una carretera junto a un parque).

- **Producción**: las tres muestras empleadas corresponden a grabaciones de robots industriales en acción ubicados en diferentes plantas industriales.

- **360°**: las tres muestras empleadas corresponden a fragmentos de videos grabados en 360° en diferentes vistas.

Mediante las siguientes tablas e imágenes se pretende ampliar la información acerca de las muestras empleadas con el propósito de que el lector tenga claro a que imágenes se refiere el documento con cada tipo antes de que se meta con el apartado correspondiente a los resultados.

Entretenimiento					
Muestra	Tamaño (MB)	Duración	Resolución	Formato	Contenido
1	47.2	1’ 30’’	1920 x 1080	Mp4	Fútbol
2	164	5’ 20’’	1920 x 1080	Mp4	Baloncesto
3	124.2	5’ 10’’	1920 x 1080	Mp4	Batalla película

Tabla 5: Resumen de las muestras analizadas dentro de la categoría de entretenimiento

Figura 35: *Frame* correspondiente a la muestra 1 dentro de la categoría de entretenimiento

Figura 36: *Frame* correspondiente a la muestra 2 dentro de la categoría de entretenimiento

Figura 37: *Frame* correspondiente a la muestra 3 dentro de la categoría de entretenimiento

Realidad aumentada					
Muestra	Tamaño (MB)	Duración	Resolución	Formato	Contenido
1	126.4	3’ 58’’	1920 x 1080	Mp4	Mario Bros
2	80	4’ 06’’	1920 x 1080	Mp4	Animales
3	54.4	3’ 00’’	1920 x 1080	Mp4	Minecraft

Tabla 6: Resumen de las muestras analizadas dentro de la categoría de realidad aumentada

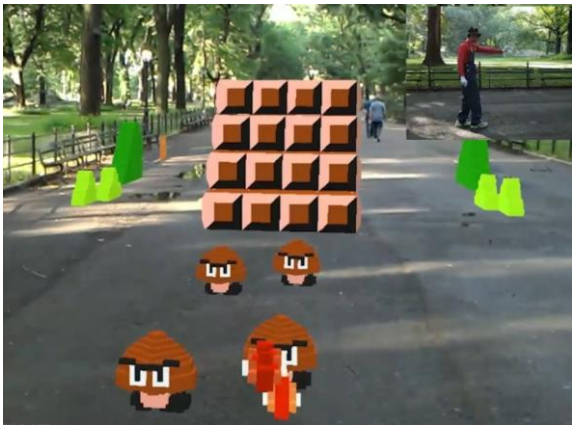


Figura 38: *Frame* correspondiente a la muestra 1 dentro de la categoría de realidad aumentada

Figura 39: *Frame* correspondiente a la muestra 2 dentro de la categoría de realidad aumentada

Figura 40: *Frame* correspondiente a la muestra 3 dentro de la categoría de realidad aumentada

Conducción					
Muestra	Tamaño (MB)	Duración	Resolución	Formato	Contenido
1	67.6	2’ 01’’	1920 x 1080	Mp4	Pueblo
2	161	5’ 01’’	1920 x 1080	Mp4	Noche
3	91.6	3’ 00’’	1920 x 1080	Mp4	Ciudad

Tabla 7: Resumen de las muestras analizadas dentro de la categoría de Conducción

Figura 41: *Frame* correspondiente a la muestra 1 dentro de la categoría de conducción

Figura 42: *Frame* correspondiente a la muestra 2 dentro de la categoría de conducción

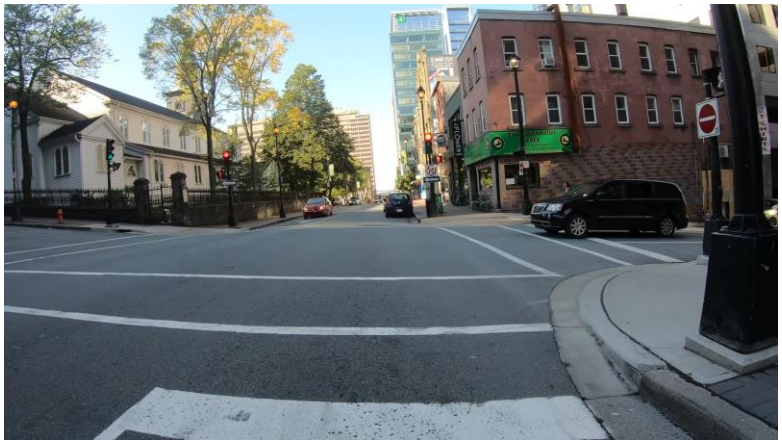


Figura 43: Frame correspondiente a la muestra 3 dentro de la categoría de conducción

CCTV					
Muestra	Tamaño (MB)	Duración	Resolución	Formato	Contenido
1	112.8	4’ 59’’	1920 x 1080	Mp4	Supermercado
2	53	1’ 34’’	1920 x 1080	Mp4	Calle
3	14.9	52’’	1920 x 1080	Mp4	Oficina

Tabla 8: Resumen de las muestras analizadas dentro de la categoría de CCTV

Figura 44: *Frame* correspondiente a la muestra 1 dentro de la categoría de CCTV

Figura 45: *Frame* correspondiente a la muestra 2 dentro de la categoría de CCTV

Figura 46: *Frame* correspondiente a la muestra 3 dentro de la categoría de CCTV

Producción					
Muestra	Tamaño (MB)	Duración	Resolución	Formato	Contenido
1	121.2	6’ 21’’	1920 x 1080	Mp4	Robot industrial
2	42.7	2’ 13’’	1920 x 1080	Mp4	Robot industrial
3	230.4	6’ 55’’	1920 x 1080	Mp4	Robot industrial

Tabla 9: Resumen de las muestras analizadas dentro de la categoría de producción

Figura 47: *Frame* correspondiente a la muestra 1 dentro de la categoría de Producción

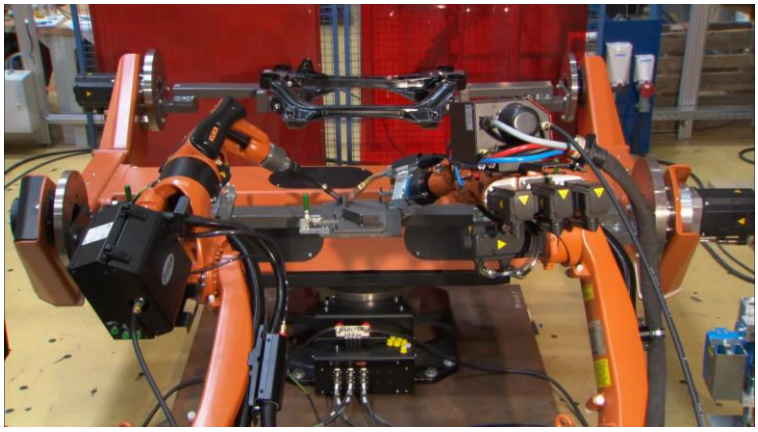


Figura 48: *Frame* correspondiente a la muestra 2 dentro de la categoría de Producción

Figura 49: *Frame* correspondiente a la muestra 3 dentro de la categoría de Producción

360°					
Muestra	Tamaño (MB)	Duración	Resolución	Formato	Contenido
1	157.3	25''	1920 x 1080	Mp4	Vista
2	363.3	2' 00''	1920 x 1080	Mp4	Vista
3	124.2	29''	1920 x 1080	Mp4	Vista

Tabla 10: Resumen de las muestras analizadas dentro de la categoría de 360°

Figura 50: *Frame* correspondiente a la muestra 1 dentro de la categoría de 360°

Figura 51: *Frame* correspondiente a la muestra 2 dentro de la categoría de 360°

Figura 52: *Frame* correspondiente a la muestra 3 dentro de la categoría de 360°

En la misma línea, en relación con lo que ha costado codificar cada tipo de contenido, en la siguiente tabla se pueden apreciar la entropía y el ratio de cambios de escena de las muestras del conjunto de datos que se han empleado para obtener los rangos operativos. Cabe destacar que, estos dos parámetros pueden servir para disponer de cierta información sobre la complejidad que presenta una secuencia.

Mediante la entropía se ha pretendido adquirir información sobre la variabilidad espacial de las secuencias analizadas (complejidad, movimientos de cámara, actividad en la escena) y ver, por ejemplo, cuanto varían las entropías de las secuencias que se han categorizado dentro del mismo tipo de imagen. De este modo, se ve que las imágenes pertenecientes a conducción, producción, CCTV y 360° son las que menos variaciones presentan en la entropía. En segundo lugar, mediante el parámetro correspondiente a los cambios de escena, se ha pretendido calcular información sobre la variabilidad temporal. Como era de esperar, las imágenes de entretenimiento son las que mayor ratio presentan en comparación con las demás.

Tipo de imagen	Secuencia	Entropía	Ratio cambios de escena
Entretenimiento	1	6.43749	0.0333
	2	7,27474	0,0812
	3	5.94691	0.0548
Conducción	1	7,02562	0
	2	7,01050	0
	3	7,23252	0
Realidad Aumentada	1	6,36872	0,0042
	2	7,08790	0,0121
	3	6,78357	0,0555
Producción	1	7,47990	0
	2	6,63913	0,0636
	3	7,44948	0,0096
CCTV	1	7,67918	0
	2	7,29464	0
	3	7,72913	0
360°	1	7,28407	0
	2	7,06166	0
	3	7,01710	0

Tabla 11. Análisis de la complejidad temporal y espacial de los distintos tipos de imágenes

3.4 Diagrama Gantt

Responsables del proyecto	Ángel Martín/Silvia Díaz	Vicomtech	Universidad Pública de Navarra

Nº de la tarea	Título de la tarea	Responsable de la tarea	Porcentaje de trabajo completado	MAYO				JUNIO				JULIO				AGOSTO				SEPTIEMBRE			
				1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	**Concepción e iniciación del proyecto**																						
1.1	Opciones & Tutoriales	Angel	100%																				
1.2	Capacitación	Mikel	100%																				
1.3	Primeras pruebas con FFMPEG	Mikel	100%																				
1.4	Métricas	Mikel	100%																				
1.5	Dataset	Angel	100%																				
1.6	Metodología & Validación	Mikel	100%																				
2	**Tests de caracterización y establecimiento de las tablas de búsqueda**																						
2.1	Pruebas FFMPEG & Captura de datos	Mikel	100%																				
2.2	Metodología & Scripting	Mikel	100%																				
2.3	Tablas & Gráficas	Mikel	100%																				
2.4	Rangos operativos	Mikel	100%																				
3	**Configuración dinámica y desarrollo**																						
3.1	GStreamer: dearrollo de aplicaciones	Mikel	100%																				
3.2	GStreamer: dynamic pipelining	Mikel	100%																				
3.3	Gstreamer: captura de datos y eventos	Mikel	100%																				
3.4	Gstreamer: rate control dinámico	Mikel	100%																				
3.5	*Gstreamer: slicing basado en H.264*	Mikel	0%																				
3.6	*GitLab Streame! Integración SDK*	Mikel	0%																				
4	**Documentación**																						
4.1	Plantilla	Mikel	100%																				
4.2	Anotaciones de las actividades diarias	Mikel	100%																				
4.3	Redacción de la memoria	Mikel	100%																				
4.4	Revisión	Angel/Silvia	100%																				

Mediante el diagrama Gantt se pretende describir de manera más visual la planificación que se ha tenido a lo largo de la evolución del proyecto. Se pueden contemplar cómo se organizaron las diferentes tareas que se tenían que realizar cuando empezó el proyecto y los hitos que se han ido cumpliendo con el avance del tiempo. Además, se diferencian claramente las tres fases en las que se ha dividido el proyecto y que se han comentado en la sección 0.

Se han cumplido todas las tareas que se planificaron al inicio del proyecto excepto las siguientes dos:

- 3.5 Gstreamer: *slicing* basado en H.264
- 3.6 GitLab Streame! Integración SDK

Sin embargo, hay que tener en cuenta que estas dos tares cuando se definieron en su momento se pusieron en cursiva porque se consideraron opcionales.

Respecto a la primera de ellos (el 3.5), se ha visto que no ha habido la necesidad de implementar la parte correspondiente al *slicing*, debido a que se requeriría demasiada computación para el beneficio que se obtendría. En cuanto a la segunda tarea (el 3.6), se va a realizar una vez que se presente el proyecto. Este punto se incluyó en el proyecto pero desde el principio se vio que dependiendo de las vueltas que se le diera a la obtención de los rangos operativos (Fase II. *Tests* de caracterización y establecimiento de las tablas de búsqueda), se llegaría a implementar o no. Con el avance del proyecto se ha visto que no ha dado tiempo.

4 Resultados y discusión

En este capítulo se ilustran y se comentan las diferentes gráficas que se han dibujado con los datos que se han adquirido a lo largo del proyecto con el propósito de desarrollar el sistema propuesto en el capítulo 1.2. De esta manera, se describen las diferencias existentes entre los tres perfiles de calidad calculados para cada uno de los tipos de imágenes analizados, los cuales se clasifican en producción, entretenimiento, realidad aumentada, CCTV, conducción y 360°. Además, se analizan las gráficas que ilustran la evolución de las métricas de calidad PSNR (referencia) y SSIM (respaldo) en distintas resoluciones en función del bitrate de para cada tipo de imagen. Cabe destacar que, las gráficas se han obtenido a partir de la combinación de los datos obtenidos en el procesado de las muestras de interés que se han clasificado en la misma categoría.

Por otro lado, se discute también sobre la complejidad que han presentado dichas muestras que forman el *dataset*, examinando según los cálculos realizados cual es el tipo de imagen que más y menos bitrate necesita para alcanzar los umbrales deseados. Así pues, se ha incluido una tabla donde se resume para las distintas resoluciones analizadas en cada uno de los contenidos, los rangos operativos para alcanzar las calidades referentes a los tres perfiles establecidos. Ésta ha sido la tabla en la que se ha basado la aplicación implementada mediante GStreamer.

4.1 Análisis del tipo de imagen de Producción

Esta sección se basa exclusivamente en analizar los resultados obtenidos relacionados con el tipo de imagen de producción. En este sentido, se pueden encontrar tanto la gráfica de producción que contiene las curvas bien definidas de alta, media y baja calidad como las gráficas que relacionan la PSNR y el SSIM con el bitrate para diferentes resoluciones.

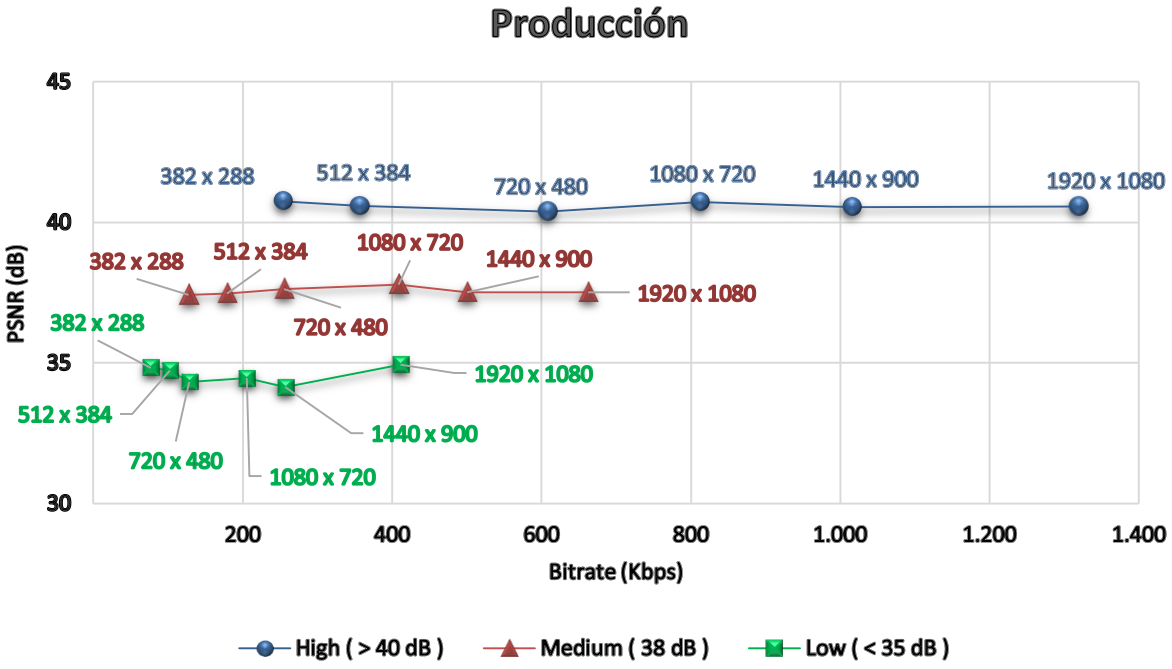


Figura 53: Curvas de alta, media y baja calidad correspondientes a las muestras de producción.

Efectivamente se ve que las curvas están bien definidas y que representan la coherencia esperada, ya que en todos los casos a la hora de codificar un flujo una misma imagen con una menor resolución requiere un menor bitrate para alcanzar la misma calidad deseada. Indistintamente, para un mismo tamaño de imagen, es decir, para una misma resolución, alcanzar una mayor calidad requiere el uso de un mayor bitrate.

Idealmente las tres curvas que se han representado en la gráfica deberían ser planas y deberían encontrarse justo en los umbrales establecidos (40 dB, 38 dB y 35 dB). Sin embargo, se puede observar que en la práctica esto no se ha cumplido, debido a que las pruebas realizadas para obtener los datos necesarios para representar estas gráficas se han realizado con un conjunto de bitrates finito, y no utilizando todos los posibles valores de bitrate (ver Metodología). Cabe destacar que, esto no resta validez a la hora de valorar los objetivos propuestos, ya que la idea desde un principio ha sido la de obtener unos márgenes de actuación para implementar en la aplicación que se ha diseñado en GStreamer.

La siguiente gráfica ha sido dibujada haciendo uso de los datos obtenidos a partir del procesamiento de las tres muestras del conjunto de datos que se encuentra dentro de la categoría de producción. Como se ha comentado previamente, esta gráfica relaciona los valores obtenidos de PSNR para las distintas combinaciones de resoluciones y bitrates. Siendo esto así, en él se pueden apreciar seis curvas con un comportamiento logarítmico, donde cada uno de ellos corresponde a una de las resoluciones analizadas. Por un lado, se puede observar que cuanto mayor sea la resolución de la imagen, se obtiene un menor valor de PSNR a la hora de codificar con un mismo bitrate, lo cual era algo esperado. Cabe destacar que, la gráfica que se ha ilustrado en la Figura 53, equivalente a los tres perfiles de calidad para producción, se ha dibujado a partir de los valores de esta gráfica.

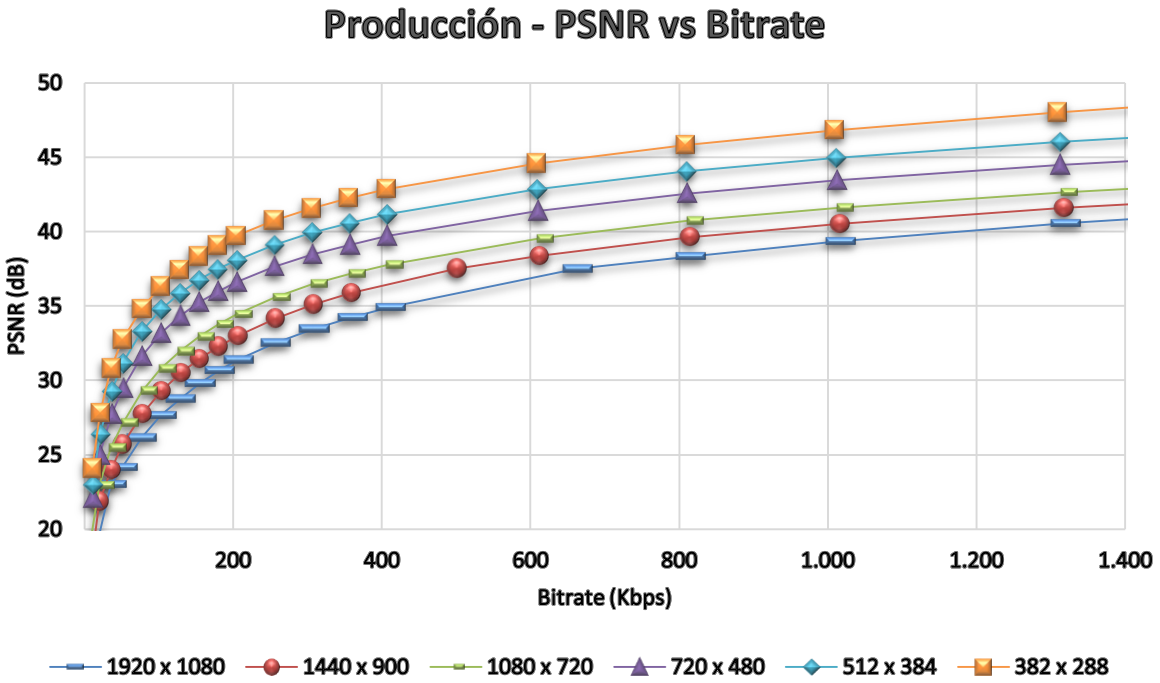


Figura 54: Gráfica que relaciona el bitrate con la PSNR para las distintas resoluciones analizadas y que ha sido dibujada a partir de los datos correspondientes a las muestras de producción

Por otro lado, es interesante comentar que a la hora de codificar cada una de las secuencias de imágenes que completan el *dataset*, se ha visto que la complejidad de la codificación tiende a variar en función de la resolución en la que se pretende codificar la secuencia de interés. En la Figura 54 se contempla que cuanto menor sea la resolución que se haya empleado a la hora de codificar una muestra, el comportamiento de las observaciones obtenidas tiende a ser menos lineal y más abrupto, sobre todo cuanto el bitrate es inferior a 500 Kbps.

Partiendo de la idea de que el tamaño de un macrobloque viene establecido por el estándar (H.264/MPEG-4 AVC) y es constante independientemente de la resolución, cuanto menor sea el tamaño de la imagen con la que se trabaja, se consigue cubrir más imagen con cada macrobloque, lo que supone un aumento en la variabilidad de la información que contiene cada uno de ellos. Por lo que se podría decir que cuanto más baja sea la resolución de una imagen, más compleja es para codificarla.

Esto significa que el número de macrobloques, tanto por ancho como por alto, suele ser menor en una imagen con una resolución más pequeña en comparación con la misma imagen con una resolución más grande, debido al número de píxeles que contiene cada uno de los macrobloques. Así pues, existe una mayor variabilidad en la información que contiene cada macrobloque en una imagen más pequeña, que hace que dicho macrobloque sea más difícil de codificarlo. En resumen, hay que quedarse con la idea de que cuanto menor sea la variabilidad que haya en los macrobloques de una imagen (lo cual se consigue codificando en resoluciones más grandes), más fácil será para el codificador estimar el movimiento en dicha imagen. En cambio, cuanta más variabilidad exista en la información que abarca un macrobloque, más difícil será de codificarlo.

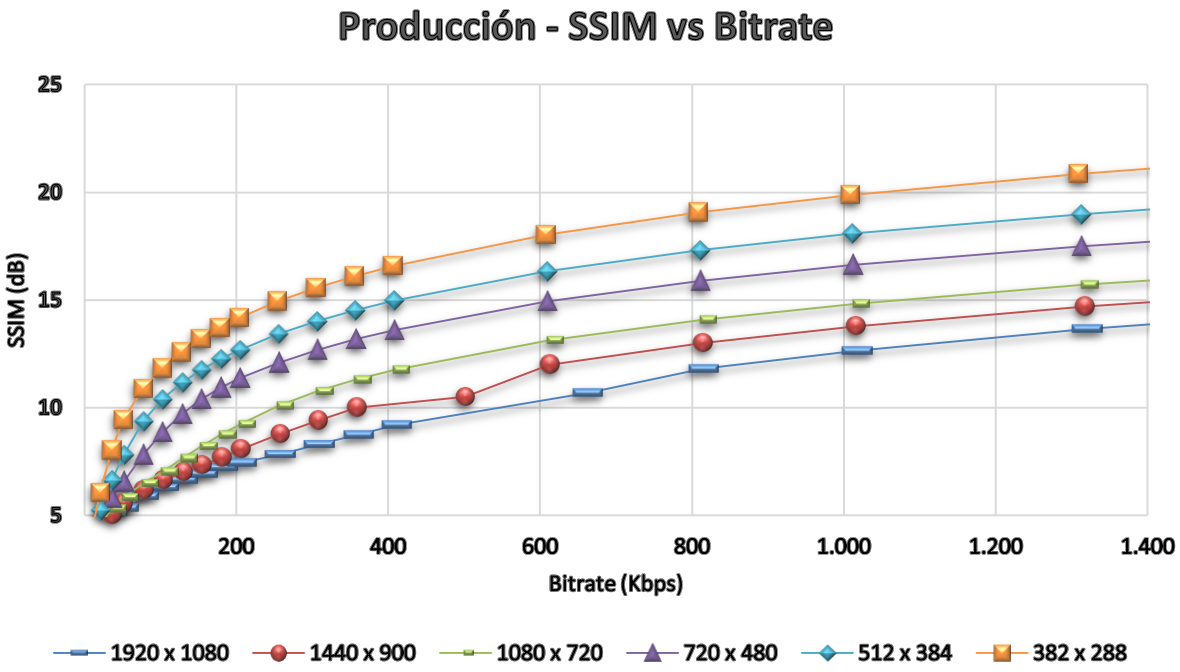


Figura 55: Gráfica que relaciona el bitrate con la SSIM para las distintas resoluciones analizadas y que ha sido dibujada a partir de los datos correspondientes a las muestras de producción

Como se ha comentado a lo largo del documento, para que los resultados fueran más consistentes además de buscar unas gráficas que se basaran en la PSNR, se buscó una segunda métrica de respaldo con el fin de darle más consistencia al proyecto y ver que las curvas dibujadas en los dos casos presentaban un mismo comportamiento (independientemente de las escalas de los ejes de las gráficas en las que se encuentran). De este modo, en la Figura 55 se puede observar la gráfica que relaciona el valor calculado de la SSIM (dibujada a partir de los datos obtenidos tras procesar las tres muestras correspondientes a la categoría de producción) para las distintas configuraciones (variando el bitrate y la resolución).

Las curvas dibujadas en esta gráfica tienden a tener el mismo comportamiento logarítmico que las curvas de la gráfica previamente mostrada (correspondiente a la PSNR). En este sentido, cuanto mayor sea la resolución requerida, codificando con un mismo bitrate se alcanza un menor valor de SSIM, lo mismo que pasaba con la PSNR. Asimismo, se ve que cuanto menor sea el tamaño de imagen que se esté analizando, la curva que lo caracteriza tiende a ser más lineal, cuando el bitrate es inferior a 500 Kbps (después todas las curvas se comportan de una manera parecida).

Por otro lado, hay que tener cuenta que los rangos operativos establecidos deberían funcionar en un alto porcentaje, pero que no son universales (tanto para el tipo de imagen de producción como para los restantes que se analizan en el siguiente apartado). Es decir, se ha visto que algunas secuencias pertenecientes a las categorías fijas+ pueden variar a la hora de alcanzar los umbrales mediante dichos bitrates establecidos.

Asimismo, se ha hecho un desarrollo mediante GStreamer, donde se ha conseguido implementar el cambio de bitrate de una manera dinámica, dependiendo del tipo y tamaño de imagen con la que se esté trabajando y la calidad que se quiera conseguir. Considerando que se está dando solución a una librería del centro que es de propósito general, la aplicación desarrollada tiene que ser capaz de cubrir diferentes posibilidades que todavía no hayan surgido (y puede que ni estén previstas). Lo que se pretende decir es que, del mismo modo que se ha conseguido implementar el cambio del bitrate dinámicamente haciendo uso de los *presets* según las necesidades, puede que sea interesante que se tenga un proceso que analice cual es la calidad que se está obteniendo a la salida del codificador y en función de ello comprobar si se aproxima a la esperada, para reestablecer el bitrate si se ve necesario.

4.2 Análisis de los restantes tipos de imágenes

En este apartado se pueden encontrar tanto las curvas correspondientes a los tres perfiles de calidad (alta, media y baja) para cada tipo de imagen, como las gráficas que relacionan la PSNR y el SSIM con el bitrate para distintas resoluciones, dibujadas mediante los datos obtenidos a partir del procesamiento aplicado a las muestras del *dataset*. En la sección 4.1 se ha analizado el tipo de imagen equivalente a producción, aquí se enseñan las gráficas correspondientes de entretenimiento, conducción, realidad aumentada, producción, CCTV y 360°, con el propósito de examinar como varía el bitrate a la hora de obtener las calidades deseadas dependiendo del tipo de contenido.

Para empezar, lo primero que se ha visto ha sido que efectivamente el bitrate varía dependiendo del tipo de imagen que se quiera codificar. Es decir, no es lo mismo emplear un valor de bitrate específico para codificar secuencias que presentan diferentes tipos de imagen, ya que se obtendrían valores de PSNR o SSIM diferentes, dependiendo de la complejidad que presentan dichas imágenes. Dicho esto, las muestras correspondientes a la categoría de entretenimiento han sido las que mayor bitrate han necesitado para alcanzar las calidades deseadas. Esto era de esperar, debido a la cantidad de movimientos, cambios de escena o variaciones espaciales en la propia imagen que presentaban las muestras empleadas, en comparación con las muestras correspondientes de las categorías restantes. En la siguiente gráfica se aprecian los tres perfiles de la categoría de entretenimiento.

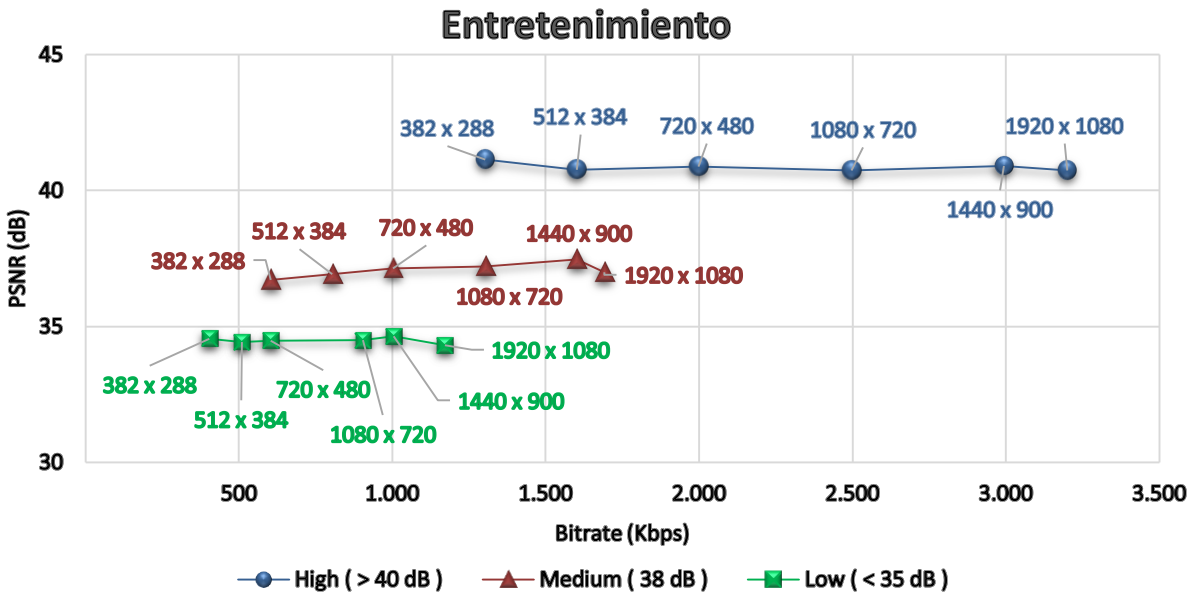


Figura 56: Curvas de alta, media y baja calidad correspondientes a las muestras de Entretenimiento.

Se ve que para todos los casos (combinación de tamaño de imagen y calidad deseada), el bitrate a emplear es muy superior en comparación con al bitrate de los flujos categorizados como producción (ver Figura 53). Obsérvese que, en este caso, codificando un flujo con un bitrate de 1.400 Kbps, solamente se podría obtener una alta calidad si el tamaño de imagen es de 382 x 288, mientras que en el caso de producción con ese bitrate no habría ningún problema en alcanzar el perfil de alta calidad en una imagen de 1920 x 1080. Ocurre exactamente lo mismo con los otros dos perfiles (modificando el bitrate referencia claro). Se podría decir que en caso de no querer sacrificar nada, sería adecuado utilizar estos parámetros a la hora de codificar cualquier tipo de imagen, debido a que siempre se obtendrían las calidades deseadas, sin preocupaciones algunas.

El contenido categorizado como conducción sería la que se encontraría en el siguiente escalón, el cual necesitaría también un mayor bitrate en comparación con la media para llegar a los umbrales de calidad. En este tipo de secuencias es verdad que no ha habido ningún cambio de escena, ya que los flujos analizados se basaban en secuencias grabadas en un único plano. Sin embargo, al ser el tipo de contenido que presenta un mínimo cambio constante entre *frames* consecutivos (es decir, en la información espacial de cada *frame*), el bitrate requerido es mayor que en otros tipos de contenidos.

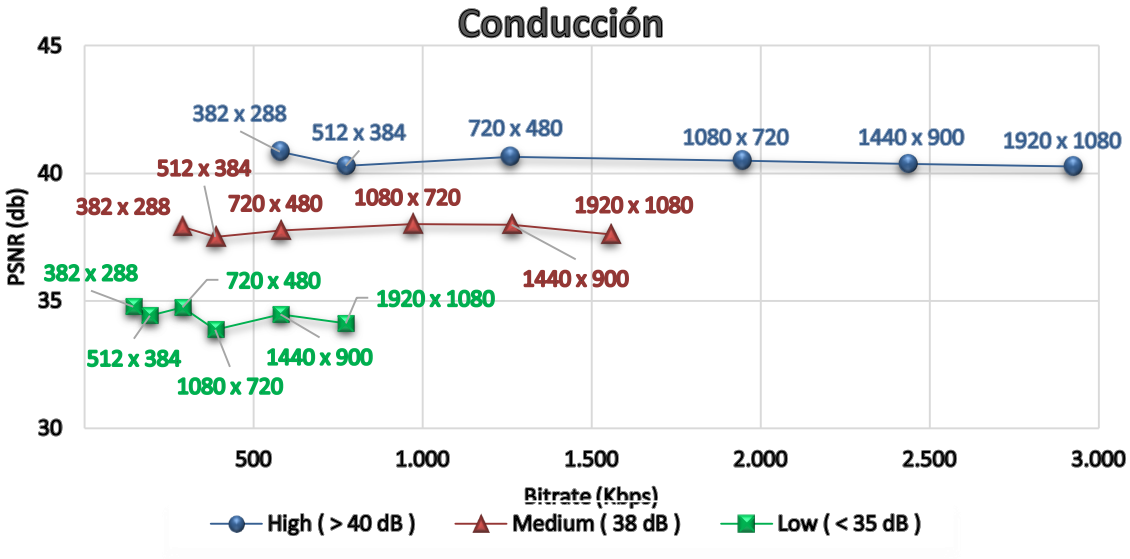


Figura 57: Curvas de alta, media y baja calidad correspondientes a las muestras de conducción

Respecto a las secuencias de realidad aumentada, si lo comparamos con las curvas de conducción, se ve que para alcanzar el perfil de alta calidad en este caso se necesita un menor valor de bitrate. Sin embargo, esto cambia si se busca un perfil de media o baja calidad, ya que ambas imágenes empiezan a comportarse de una manera similar y sus rangos operativos no varían demasiado.

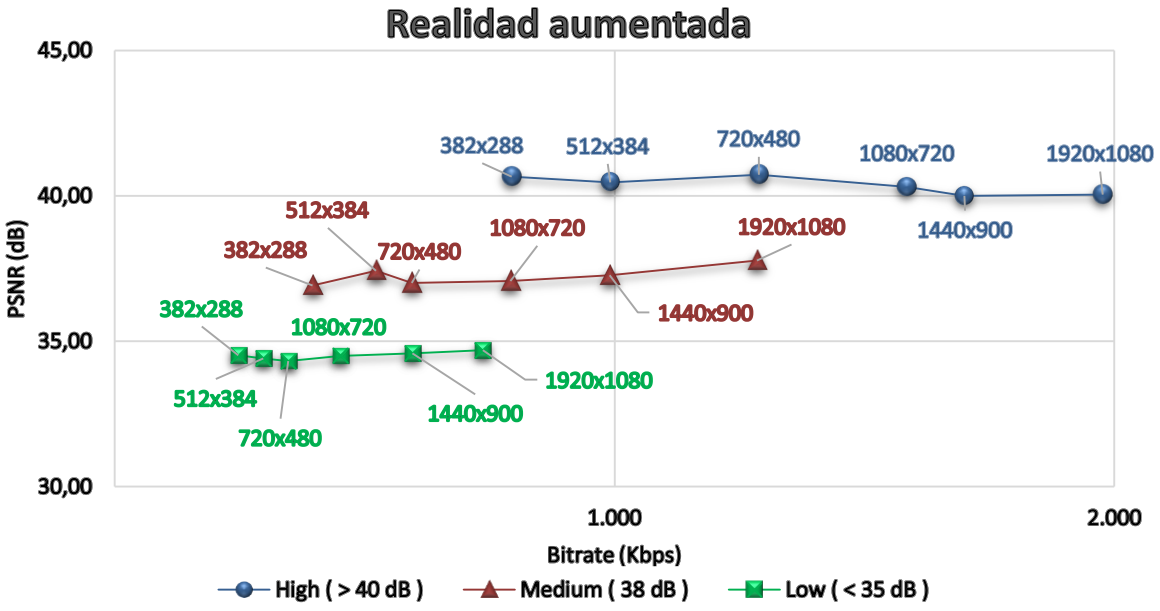


Figura 58: Curvas de alta, media y baja calidad correspondientes a las muestras de realidad aumentada

Por último, se presentan las dos gráficas que faltan, correspondientes a los contenidos de 360° y CCTV. En ambas imágenes, se ve que en los tres perfiles, el salto del bitrate que se debe dar para mantener la calidad al aumentar el tamaño de imagen, va creciendo respecto al salto anterior.

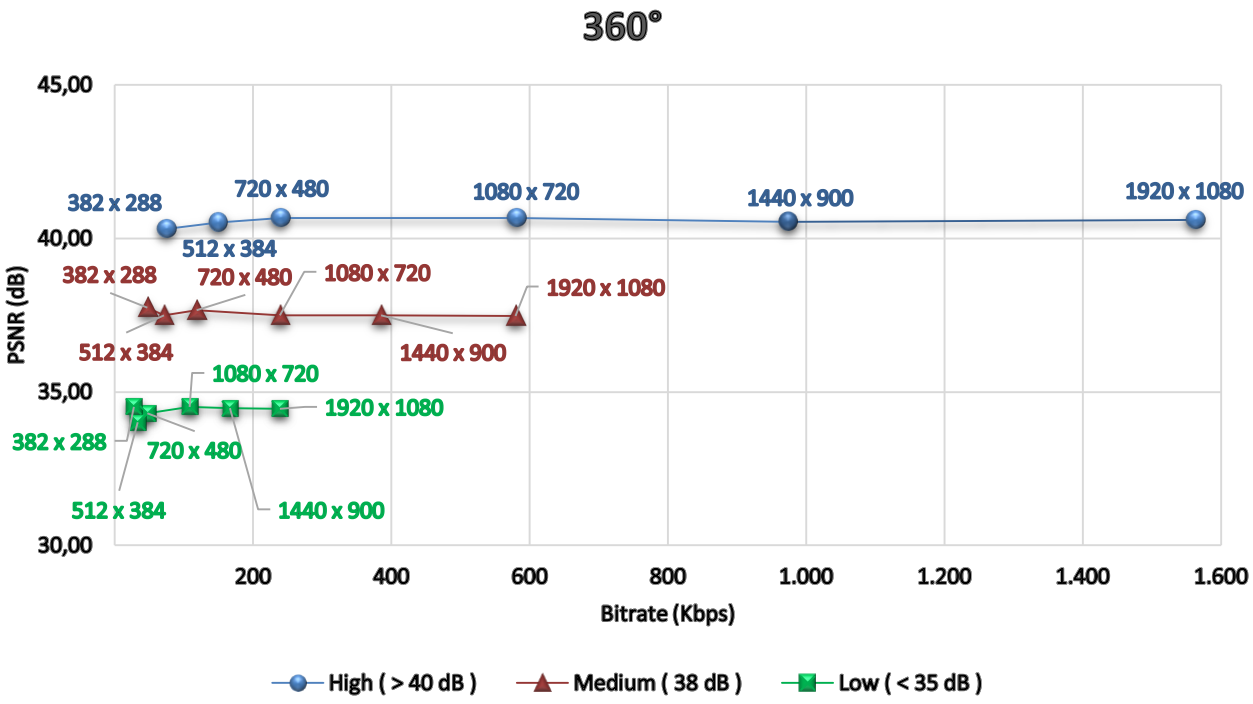


Figura 59: Curvas de alta, media y baja calidad correspondientes a las muestras de 360°

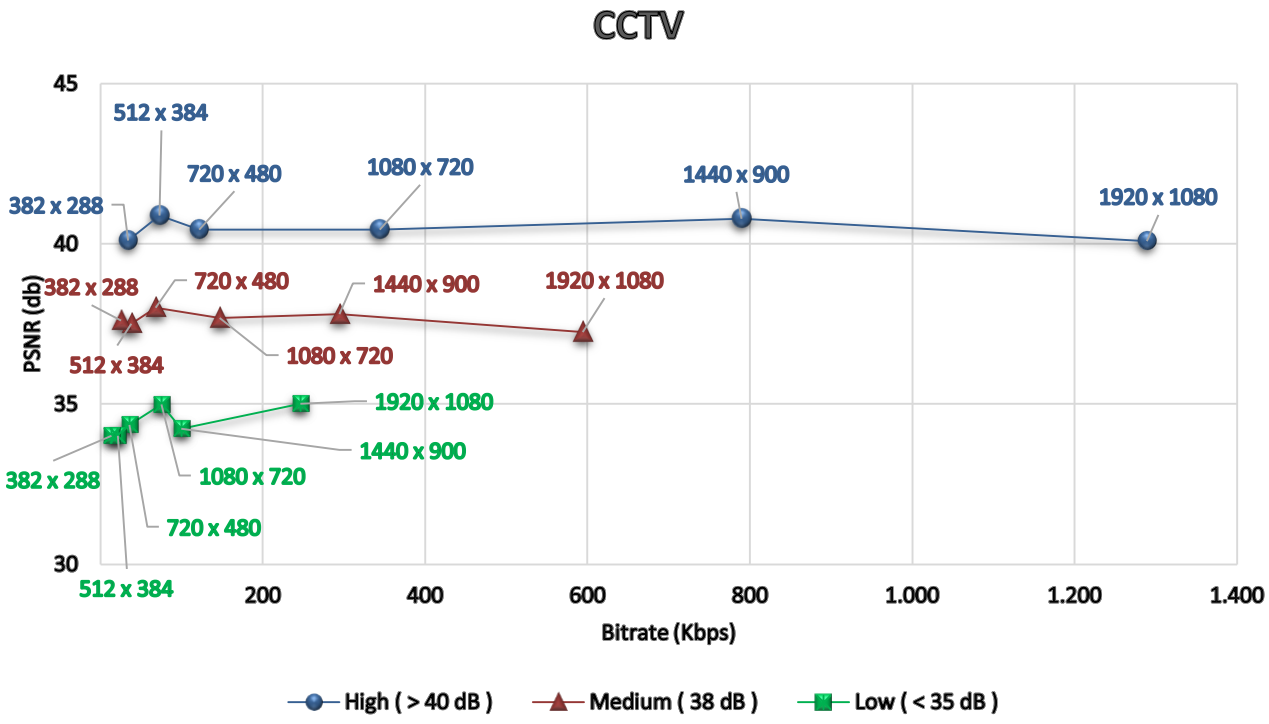


Figura 60: Curvas de alta, media y baja calidad correspondientes a las muestras de CCTV

En resumen, se podría decir que aunque cada tipo de imagen presente sus propios rangos operativos, todos los puntos mencionados en el apartado anterior (ver punto 4.1), donde se ha examinado la categoría de producción, en estas también se cumplen. Es decir, las curvas dibujadas están bien definidas y representan la coherencia esperada. Además, como se ha comentado previamente, las tres curvas deberían ser planas y deberían encontrarse justo en los umbrales establecidos (40 dB, 38 dB y 35 dB), no obstante, debido a que las pruebas realizadas para obtener los datos se han realizado con un conjunto de bitrates finito, y no utilizando todos los posibles valores de bitrate, en la práctica esto no se ha conseguido de manera exacta.

Respecto al análisis de todos los datos obtenidos a través de FFMPEG en la segunda fase, a continuación se ilustra la gráfica de entretenimiento que relaciona la PSNR con el bitrate para las distintas resoluciones empleadas. Se ve que, aunque el rango operativo varíe respecto al de producción, como se ha comentado anteriormente, el comportamiento y la tendencia de las curvas es la misma. Se observa que se necesita un mayor bitrate para mantener la calidad deseada, de manera que el tamaño de una imagen que se quiera codificar aumente. Igualmente, cuanto menor sea la resolución, codificando con un bitrate inferior a 500 Kbps, más complejo será para el codificador realizar la codificación debido al tamaño de los macrobloques (lo mismo que se ha comentado en el punto 4.1). Ocurre lo mismo con todos los tipos de contenidos, en el anexo se pueden examinar las gráficas restantes.

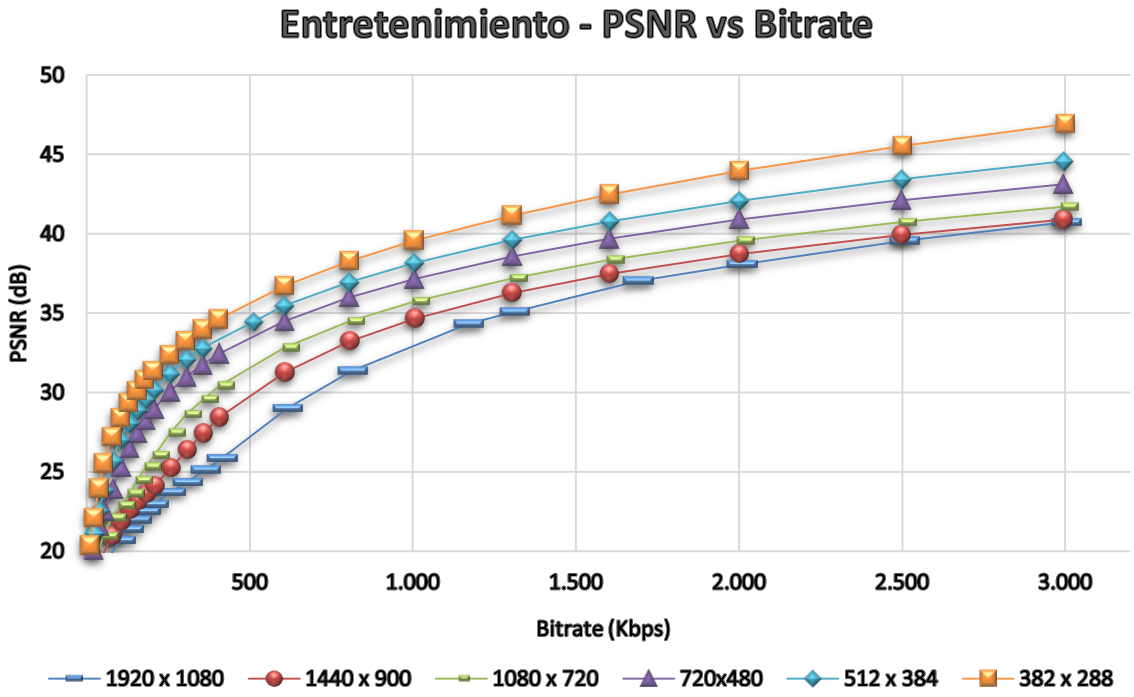


Figura 61: Gráfica que relaciona el bitrate con la SSIM para las distintas resoluciones analizadas y que ha sido dibujada a partir de los datos correspondientes a las muestras de producción

4.3 Tabla resumen de los rangos operativos

Tipo	1920 x 1080			1400 x 900			1080 x 720			720 x 480			512 x 384			382 x 288		
imagen	Alta	Media	Baja	Alta	Media	Baja	Alta	Media	Baja	Alta	Media	Baja	Alta	Media	Baja	Alta	Media	Baja
1	3.200	1.690	1.170	2.993	1.602	1.004	2.496	1.303	905	1.997	1.301	603	1.601	803	510	1.302	603	403
2	2.925	1.554	773	2.435	1.263	581	1.945	970	387	1.259	581	290	773	386	193	578	289	144
3	1.976	1.285	736	1.700	989	596	1.584	791	450	1.288	593	346	992	522	297	793	396	247
4	1.563	579	238	974	384	191	581	239	119	239	118	47	150	70	37	75	47	26
5	1.319	612	411	1.015	500	257	812	408	205	608	255	128	356	178	102	254	127	76
6	1.289	593	246	789	294	99	343	146	74	121	68	35	72	38	21	33	25	14

Tabla 12: Rangos de operación en Kbps por resolución y calidad de cada tipo de imagen

Índice	Tipo de imagen
1	Entretenimiento
2	Conducción
3	Realidad aumentada
4	360°
5	Producción
6	CCTV

Tabla 13: Mapeo entre el tipo de imagen y número de la tabla 11

5 Desarrollo en GStreamer

Mediante este capítulo se intenta describir en que consiste la aplicación final que se ha implementado en GStreamer. En la sección 3.1.3 se ha descrito el programa que se ha diseñado y se han comentado las diferentes dificultades que se han podido tener a la hora de completar su desarrollo, centrándose básicamente en la configuración dinámica del bitrate. Aquí se pretende visualizar mediante un diagrama de flujos (ver Figura 62) la lógica que sigue el código correspondiente al programa que se ha diseñado e ilustrar a través de un diagrama de secuencias (ver Figura 63) la interacción existente entre los diferentes módulos lógicos que se pueden distinguir en la aplicación.

Lógica del código diseñado

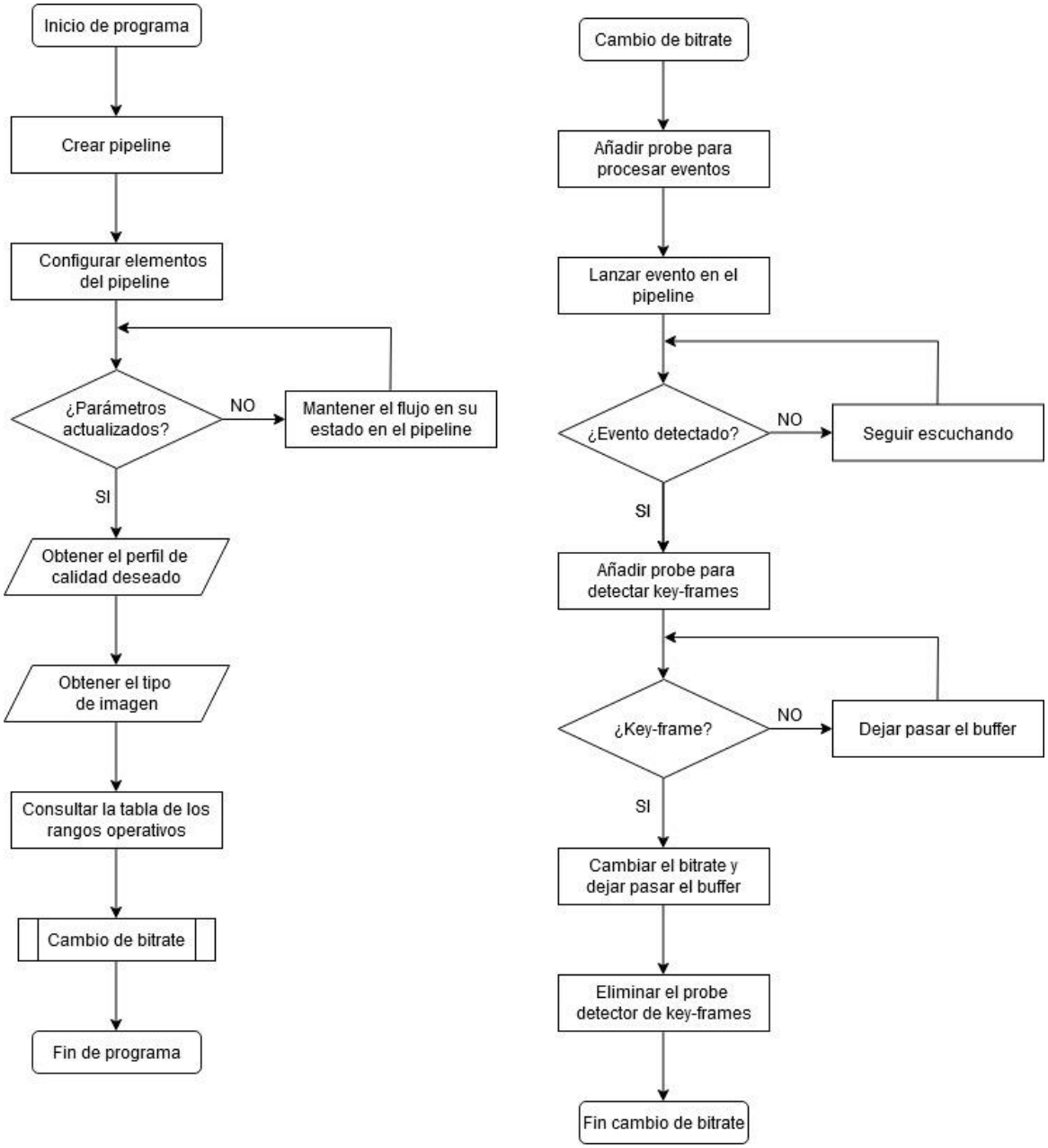


Figura 62: Diagrama de flujos que describe el código de la aplicación multimedia diseñada

Al iniciar el programa se crea el pipeline y se añaden los elementos que lo van a formar. Entre los elementos, se encuentran principalmente una fuente para cargar el flujo de interés y un codificador H.264/AVC para realizar el cambio del bitrate cuando sea necesario. Además, se han incluido un decodificador H.264/AVC y un *sink* para visualizar los flujos después de aplicar el cambio del bitrate y comprobar el correcto funcionamiento de la aplicación. Posteriormente, se vinculan y se configuran los elementos que forman el *pipeline*.

En este momento el flujo de interés estará viajando por el *pipeline*. A su vez, el programa estará esperando a que se actualicen los parámetros correspondientes al tipo de imagen que caracteriza el flujo y al perfil de calidad que se quiera alcanzar, para aplicar el cambio de bitrate. Esto se implementa a través de un fichero externo. Una vez que se actualicen ambos parámetros, el programa los obtendrá y consultará en la tabla de los rangos operativos que tiene a su disposición, el bitrate que corresponde a dicho escenario. Si los datos coinciden y no ha habido ningún problema, se efectuará el cambio de bitrate.

Por un lado, el programa lanzará un evento personalizado al *pipeline* cuando detecte que hay que realizar un cambio de bitrate. Por otro lado, antes de lanzarlo se añadirá en la fuente una sonda virtual que servirá para escuchar dichos eventos, para después procesarlos. Estas sondas se conocen como *probes* y como se ha comentado en la Metodología pueden servir para analizar el flujo que viaja por el pipeline o para escuchar eventos lanzados al pipeline. Por último, antes de realizar el cambio de bitrate se añadirá otro *probe* de tipo *buffer* también en la fuente, con el fin de analizar el flujo y realizar el cambio del bitrate cuando se disponga de un *key-frame*, evitando así posibles artefactos.

Interacción de los diferentes módulos lógicos de la aplicación

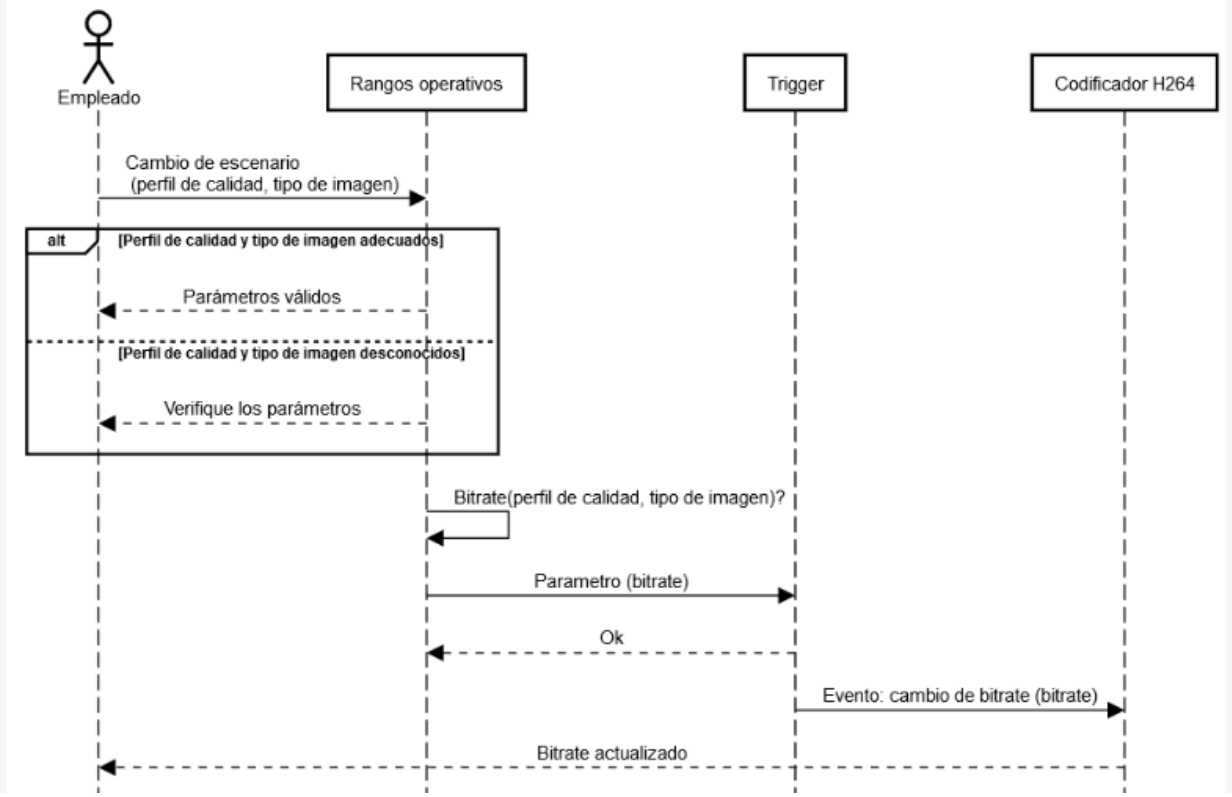


Figura 63: Diagrama de secuencias

En la Figura 63 se puede observar cómo se organizan las interacciones entre los módulos lógicos que se pueden diferenciar en el programa desarrollado, en una secuencia de tiempo. Por un lado, tenemos como actor un programador que quiere cambiar el bitrate de un flujo multimedia con el que se está trabajando. Para ello, actualiza en un fichero externo, el tipo de contenido por la que se caracteriza dicho flujo y el perfil de calidad que quiere alcanzar.

Dicho fichero es controlado por el módulo que dispone la tabla correspondiente a los rangos operativos. Por lo que, al detectar una modificación en el fichero, leerá los parámetros actualizados, verificará si son correctos y responderá al actor para informarle de si los datos introducidos han sido válidos o no. En caso de que la información sea válida, obtendrá el bitrate apropiado para dicho escenario consultando la tabla y le enviara el valor de bitrate al módulo *trigger*.

Cuando el módulo *trigger* reciba el valor del bitrate, entenderá que debe lanzar un evento por el pipeline, con el fin de que el codificador realice el cambio. Junto al evento también se enviará el valor del bitrate con el que se debe codificar el flujo. Por último, este evento va a ser proceso por el módulo correspondiente al codificador, por lo que recibirá el bitrate y empezará a codificar el flujo con un nuevo bitrate. Se le indicará al programador que se está realizando la codificación con un nuevo bitrate.

6 Conclusiones y líneas futuras

Este proyecto ha servido para obtener unos rangos operativos de bitrate que se utilizarían a la hora de codificar diferentes flujos multimedia, siempre y cuando se encuentren categorizados dentro de los seis tipos de imágenes que se han analizado en el proyecto. Además, para cada tipo de imagen se han creado tres posibles perfiles, correspondientes a alta, a media y a baja calidad. De este modo, se ha desarrollado una aplicación mediante GStreamer, que basándose en los rangos operativos previamente calculados, emula el funcionamiento de la codificación *per-title*. Es decir, en función del tipo de imagen que se quiera codificar y la calidad que se desea alcanzar, la aplicación selecciona el bitrate óptimo para alcanzar dicha calidad deseada.

Es verdad que, no se ha tenido el tiempo suficiente para llevar más allá el proyecto e integrar la aplicación desarrollada en la librería de propósito general que utiliza Vicomtech, para que la aplicación pueda ser utilizada por cualquier empleado del centro. Sin embargo, se espera que esto suceda en un futuro cercano, ya que el proyecto va a tener continuidad.

Por otro lado, se podría decir que sería interesante realizar el mismo trabajo pero partiendo de un conjunto de datos más amplio. Por ejemplo, dentro del tipo de imagen de entretenimiento se podrían distinguir varias categorías como pueden ser los deportes (fútbol, baloncesto…) o géneros de películas. Igualmente, dentro de la categoría de conducción podría haber distinciones tanto entre las secuencias de conducción por la noche o las secuencias de día, como entre las secuencias de conducción por la autopista o por el campo, ya que los escenarios no siempre son los mismos. Así para cada uno de los tipos de imágenes analizados. Asimismo, por cada tipo de imagen se podrían analizar más secuencias, para que los rangos sean todavía más consistentes.

La idea que se ha tenido a lo largo del proyecto ha sido la de indicarle a la aplicación el tipo de imagen por la que se caracteriza el flujo que se está codificando a la hora de aplicar el cambio del bitrate. Sin embargo, sería interesante que la propia aplicación fuera más inteligente, siendo capaz de detectar el tipo de imagen con la que se trabaja en tiempo real, y en función de ello realizar el cambio de bitrate, únicamente teniendo como parámetro de entrada el perfil de calidad que se quiera alcanzar.

Por último, hay que tener claro que los rangos obtenidos están restringidos a que sean utilizados únicamente cuando se esté realizando la codificación mediante el estándar H.264/MPEG-4 AVC. No sería efectivo emplear estos rangos para codificar, por ejemplo, con el códec H.265/HEVC. Es decir, se tendrían que calcular unos nuevos rangos operativos e integrarlos de nuevo en la aplicación. Por lo que, si en un futuro H.264/MPEG-4 AVC dejará de ser utilizado o bajara significativamente su dominio comercial debido a la aparición de nuevos estándares más efectivos, se tendrían que calcular de nuevo los bitrates. Sin embargo, el volumen de trabajo requerido sería mucho menor, ya que se partiría de una metodología, unos programas y unos scripts ya desarrollados.

7 Bibliografía

[1] I. Richardson (2010): "Video coding concepts", *The H.264 Advanced Video Compression standard*, Reino Unido, John Wiley & Sons, pp. 57-112.

[2] Pixel Tools, Video Processing Experts (2017): *Rate Control and H.264*. Disponible en: https://www.pixeltools.com/rate_control_paper.html [Consulta: 02 de agosto 2020].

[3] Bitmovin (2020): Encoding Software: *What Is Per-Title Encoding?*. Disponible en: https://bitmovin.com/per-title-encoding/ [Consulta: 18 de agosto 2020].

[4] A. Aaron, Z. Li, M. Manoharam, J. De Cock, D. Ronca (2015): "Per-Title Encode Optimization". *Netflix Technology Blog*, 14 de diciembre. Disponible en: https://netflixtechblog.com/per-title-encode-optimization-7e99442b62a2 [Consulta: 4 de agosto 2020].

[5] Recommendation ITU-T H.264 | ISO/IEC 14496-10 (2019): "Advanced Video Coding for generic audio-visual services".

[6] High Efficiency Video Coding. ITU-T Recommendation H.265 and ISO/IEC 23008-2 (HEVC), ISO/IEC and ITU-T, Noviembre de 2013.

[7] M. Podpora, G. Pawel, A. Kawala-Janik (2014): "YUV vs RGB – Choosing a Color Space for Human-Machine Interaction", *Federated Conference on Computer Science and Information Systems*, 3, pp. 29–34.

[8] M. Sahu, K.M. Bhurchandi (2016): "Color Image Segmentation using Genetic Algorithm", *International Journal of Computer Applications*, 140 (5).

[9] I. Richardson (2010): "Video formats and quality", *The H.264 Advanced Video Compression standard*, Reino Unido, John Wiley & Sons, pp. 39-56.

[10] Wofcrow (2020): *What's the difference between YUV, YIQ, YPbPr and YCbCr?*. Disponible en: https://wolfcrow.com/whats-the-difference-between-yuv-yiq-ypbpr-and-ycbcr/ [Consulta: 4 de agosto de 2020].

[11] Wofcrow (2020): *What is Display Gamma and Gamma Correction?*. Disponible en: https://wolfcrow.com/what-is-display-gamma-and-gamma-correction/ [Consulta: 4 de agosto de 2020].

[12] Y. Yu, J. Jhang et al (2017): *Chroma Upsampling for YCbCr 420 Videos*, *IEEE International Conference on Consumer Electronics*.

[13] P. Pawłowski et al. (2018): "Selectionand tests of lossless and lossy video codecs for advanced driver-assistance systems".

[14] S.Liu (2019): "Market share of top online video codecs and containers worldwide from 2016 to 2018". *Statista*, abril de 2019. Disponible en: https://www.statista.com/

[15] I. Richardson (2010): “What is H.264?”, *The H.264 Advanced Video Compression standard*, Reino Unido, John Wiley & Sons, pp. 113-130.

[16] T. Wedi y H. G. Mussman (2003): “Motion- and aliasing-compensated prediction for hybrid video coding”, *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7), pp. 577–586.

[17] D. Mathew y M.C. Binish (2016): “A Fast Intra Prediction for H.264/AVC Based on SATD and Prediction Direction”, *Procedia Technology*.

[18] Y. Huang et al. (2005): “Analysis, Fast Algorithm, and VLSI Architecture Design for H.264/AVC Intra Frame Coder”, *Transactions on circuits and systems for video technology*, 15(3).

[19] A. Horé y D. Ziou (2010): “Image quality metrics: PSNR vs. SSIM”, *International Conference on Pattern Recognition*.

[20] J. Korhonen y J. You (2012): “Peak signal-to-noise ratio revisited: Is simple beautiful?”, *Fourth International Workshop on Quality of Multimedia Experience*

[21] P. Artal (2017): “Image quality metrics III: Similarity between object and image”, *Handbook of Visual Optics: Instrumentation and Vision Correction*, CRC Press, pp. 288-290.

[22] D.M. Rouse y S. S. Hemami (2008): “Understanding and simplifying the structural similarity metric”, *IEEE International Conference on Image Processing*.

[23] FFMPEG (2020). Disponible en: https://ffmpeg.org/ [Consulta: 18 de agosto de 2020]

[24] C. Bampis et al. (2018): “VMAF: The Journey Continues”. *Netflix Technology Blog*, 25 de octubre de 2018. Disponible en: https://netflixtechblog.com/vmaf-the-journey-continues-44b51ee9ed12 [31 de agosto de 2020].

[25] J. Ozer (2019): *Buyers' Guide to Video Quality Metrics*. Disponible en: https://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=130675 [Consulta: 18 de agosto de 2020]

[26] C. He y A. Ziviani (2008): “An Evaluation Framework for More Realistic Simulations of MPEG Video Transmission”, *Journal of Information Science and Engineering*, 24, pp. 425-440.

[27] T. Zinner et al. (2010): “Towards QoE Management for Scalable Video Streaming”.

[28] A. Moldovan et al. (2016): “A Novel Mechanism for Mapping Objective Video Quality Metrics to Subjective MOS Scale”.

[29] A. Moldovan et al. (2017): “QoE-aware Video Resolution ThresholdsComputation for Adaptive Multimedia”, *International Symposium on Broadband Multimedia Systems and Broadcasting*.

[30] GStreamer (2020). Disponible en: https://gstreamer.freedesktop.org/ [Consulta: 18 de agosto de 2020]

[31] MPEG (2020): Disponible en: https://mpeg.chiariglione.org/ [Consulta: 19 de agosto de 2020]

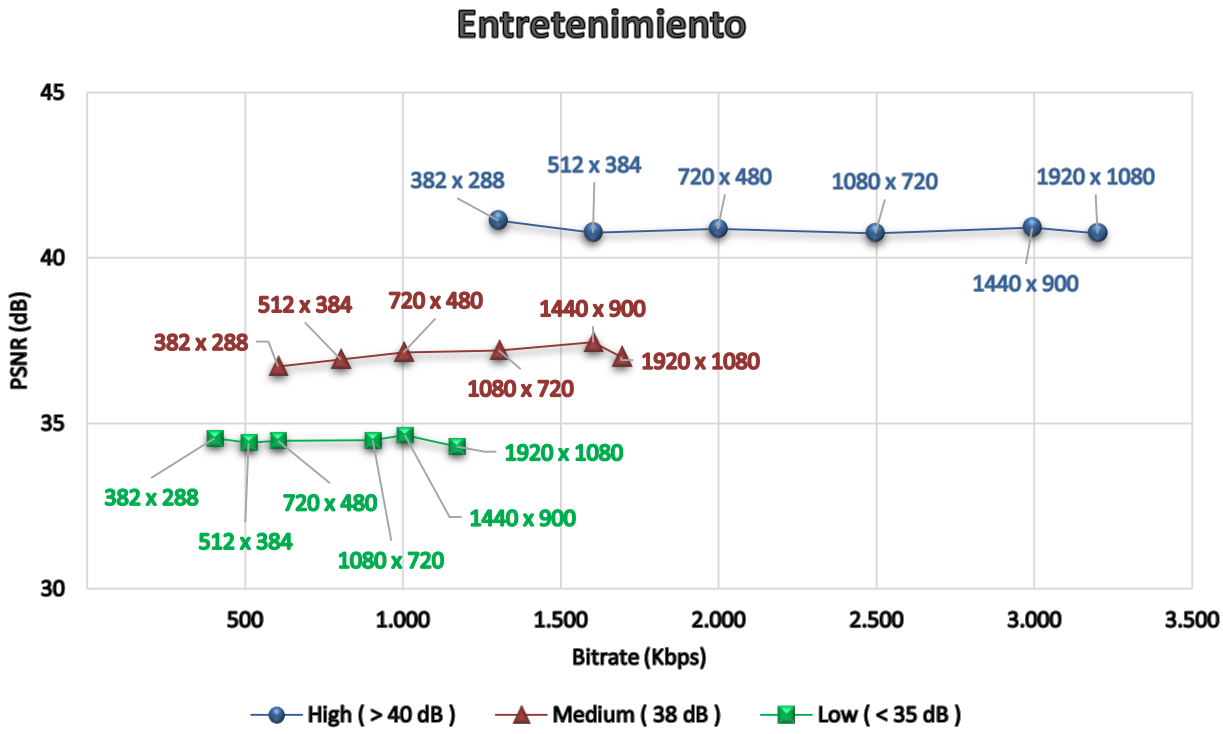
Anexos

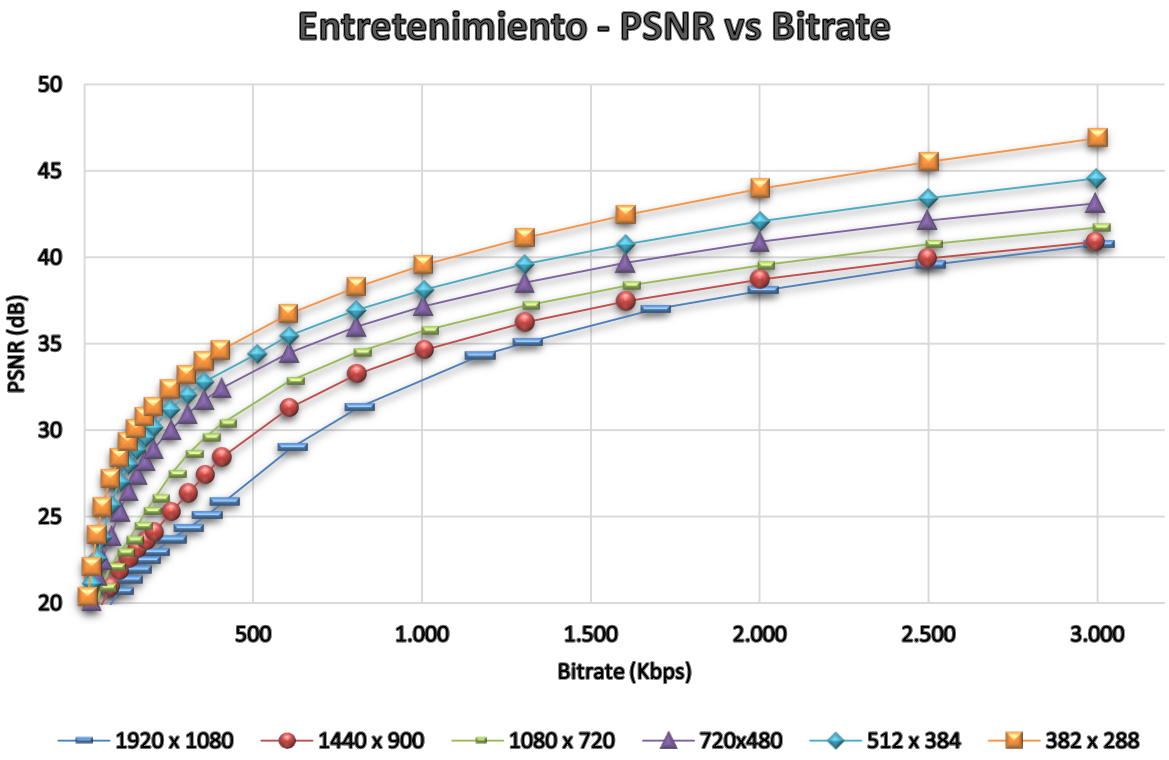
En este capítulo se pueden apreciar las gráficas que se han dibujado para los seis tipos de contenidos analizados:

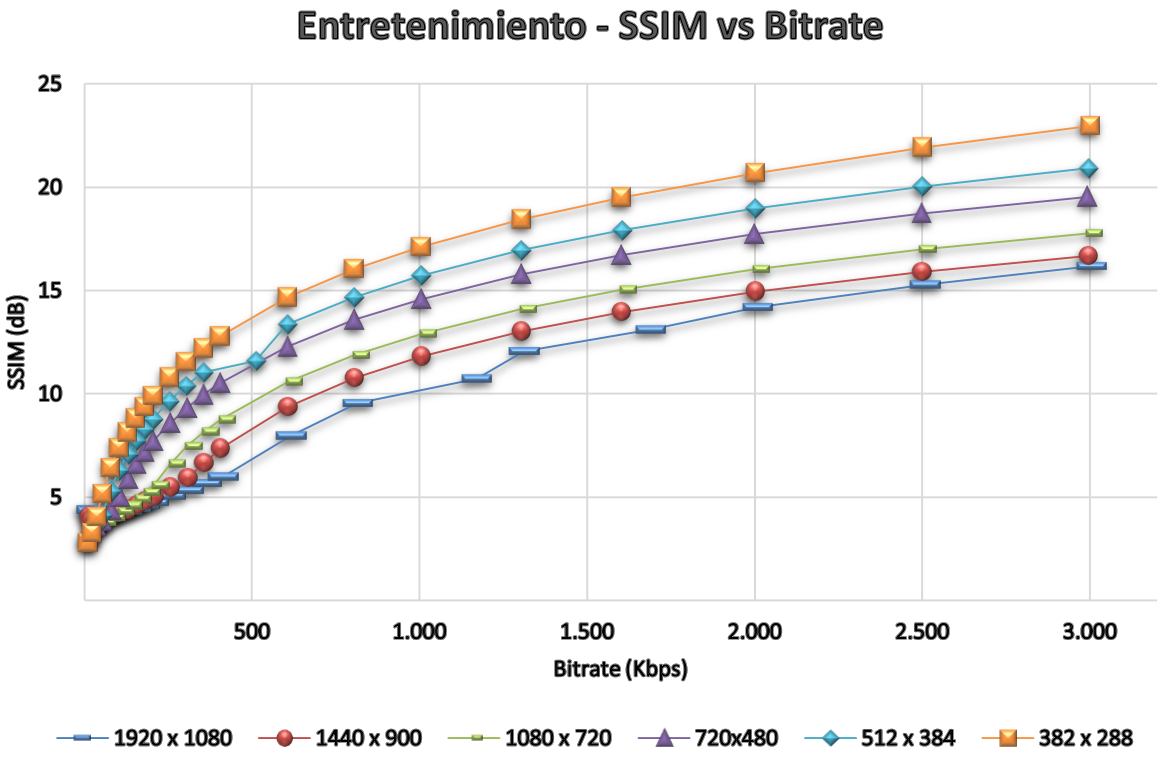
1. Gráficas con las curvas correspondientes a los tres perfiles de calidad (alta, media y baja).
2. Gráficas que relacionan el bitrate con la PSNR dibujadas con los datos obtenidos a partir del procesado de las muestras de cada categoría.
3. Gráficas que relacionan el bitrate con la SSIM dibujadas con los datos obtenidos a partir del procesado de las muestras de cada categoría.

Los tipos de imágenes analizados durante el proyecto han sido las siguientes: entretenimiento, conducción, realidad aumentada, 360°, producción y CCTV.

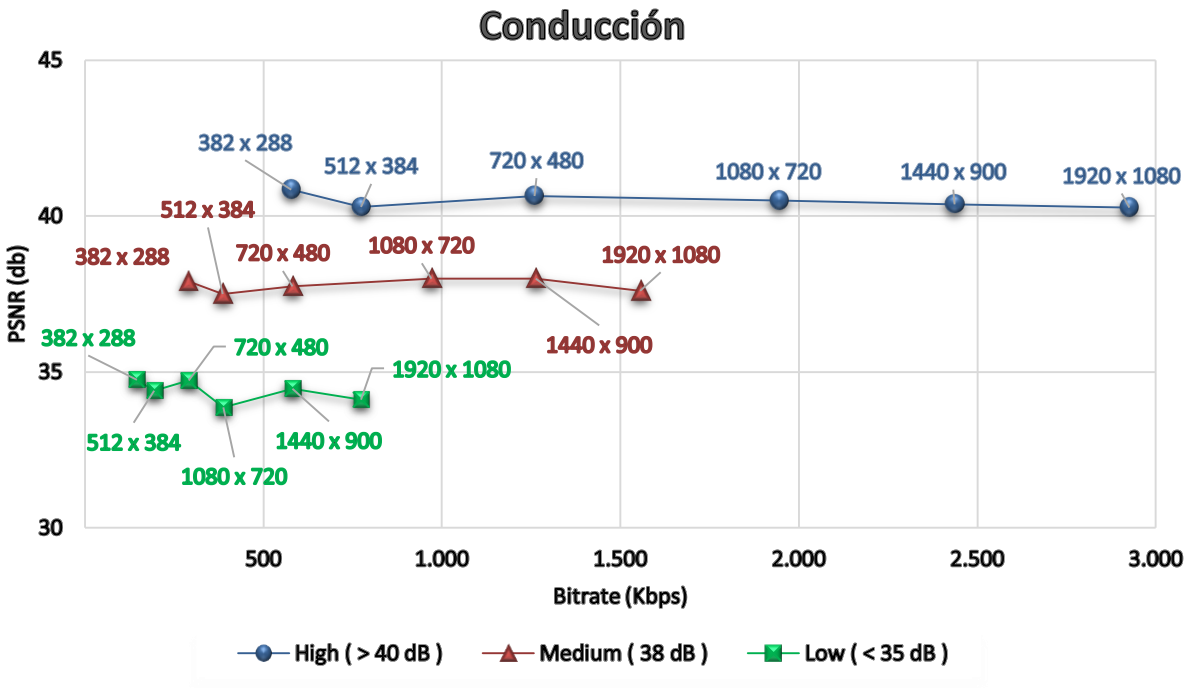
1. Entretenimiento

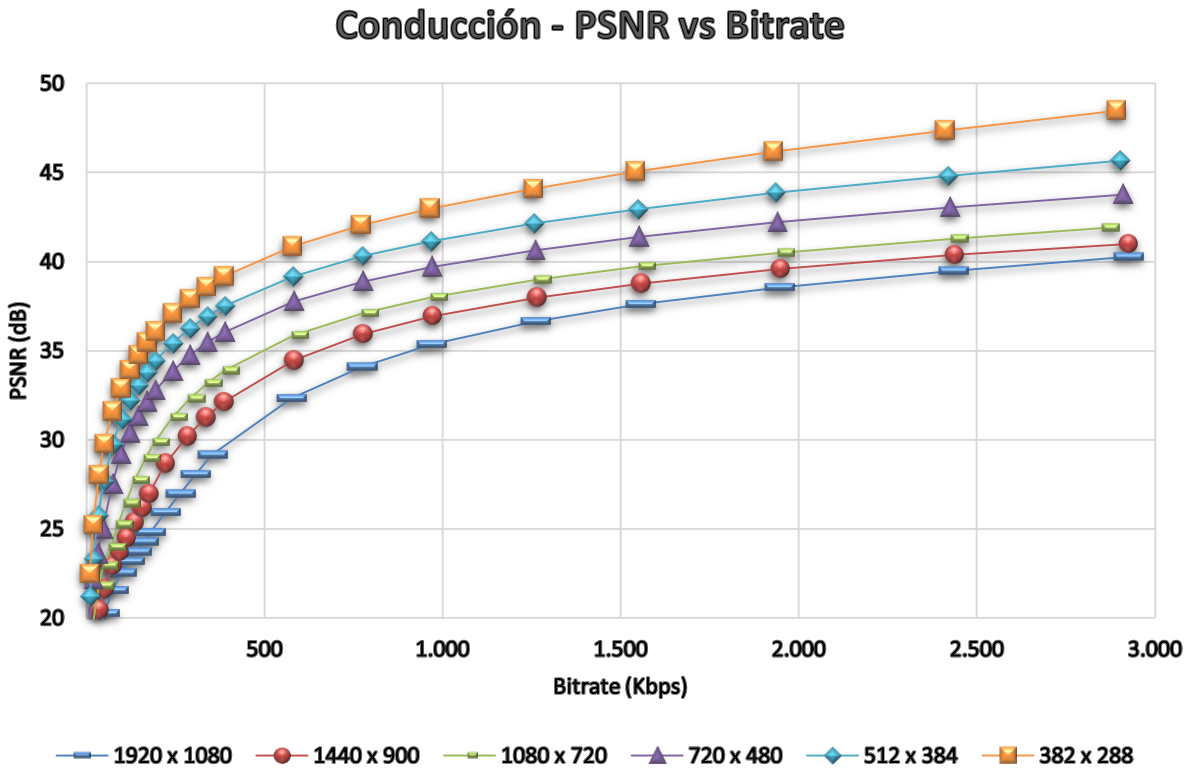


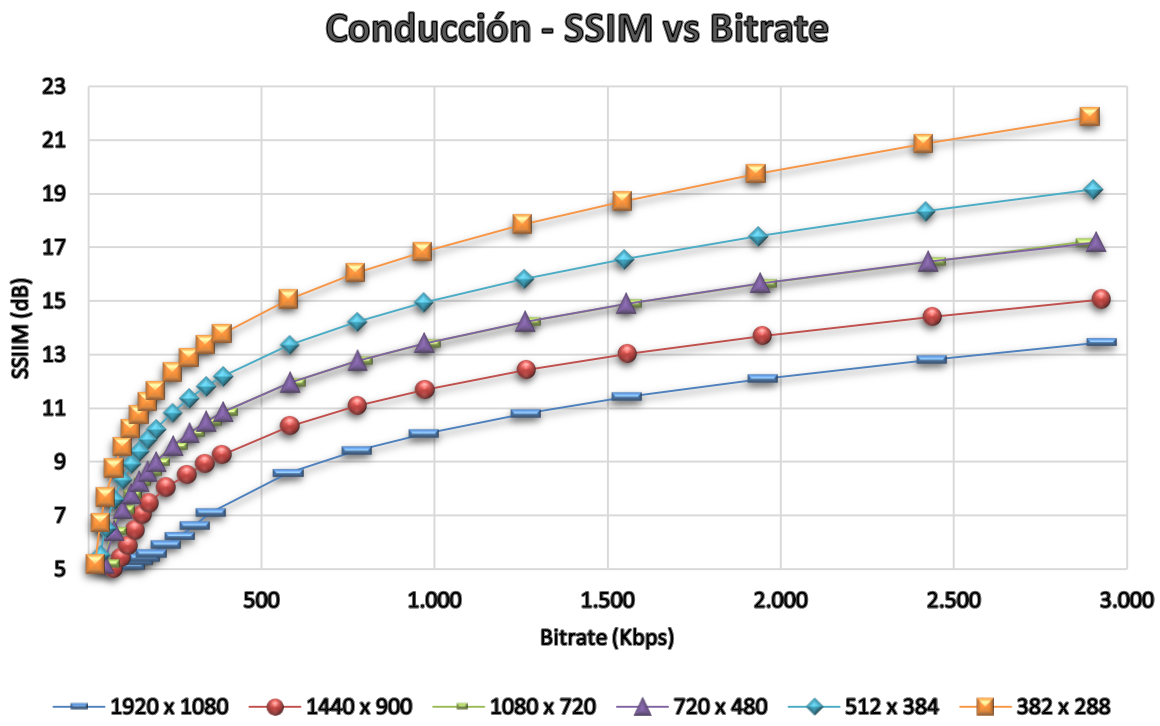

Entretenimiento - PSNR vs Bitrate
PSNR (dB)
50
45
40
35
30
25
20
500
1.000
1.500
2.000
2.500
3.000
Bitrate (Kbps)
1920 x 1080
1440 x 900
1080 x 720
720x480
512 x 384
382 x 288


Entretenimiento - SSIM vs Bitrate
SSIM (dB)
25
20
15
10
5
500
1.000
1.500
2.000
2.500
3.000
Bitrate (Kbps)
1920 x 1080
1440 x 900
1080 x 720
720x480
512 x 384
382 x 288

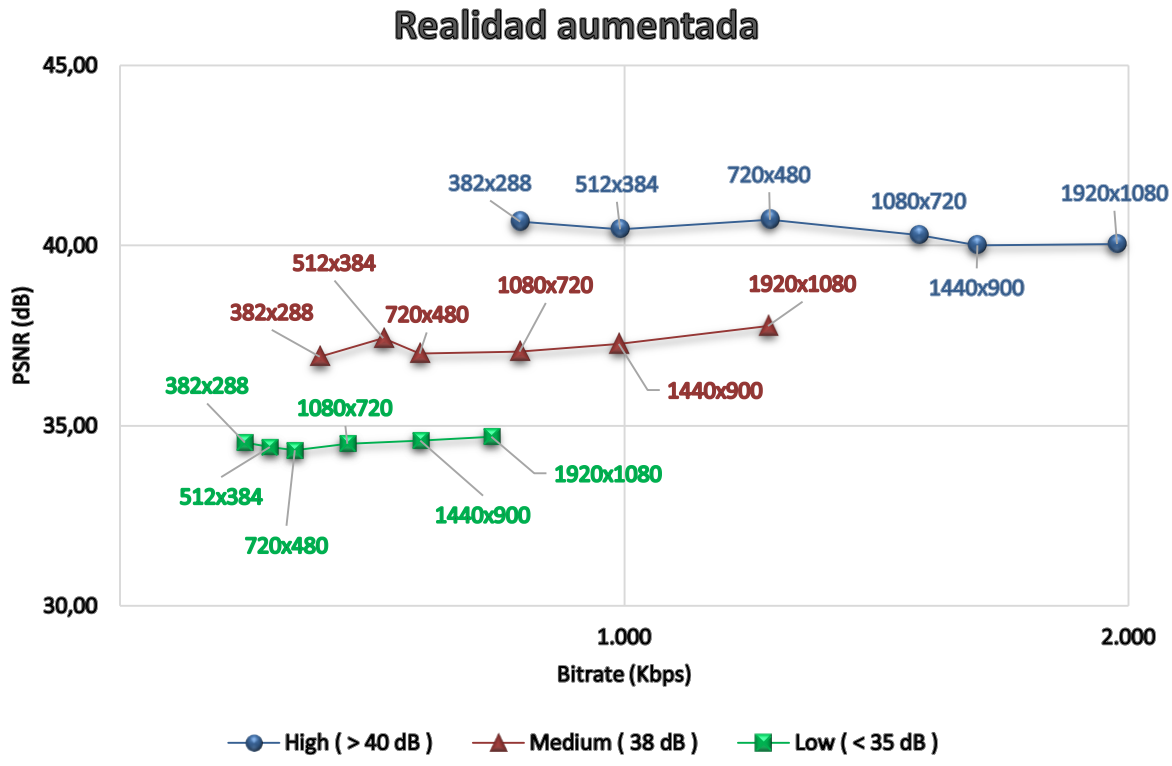
2. Conducción

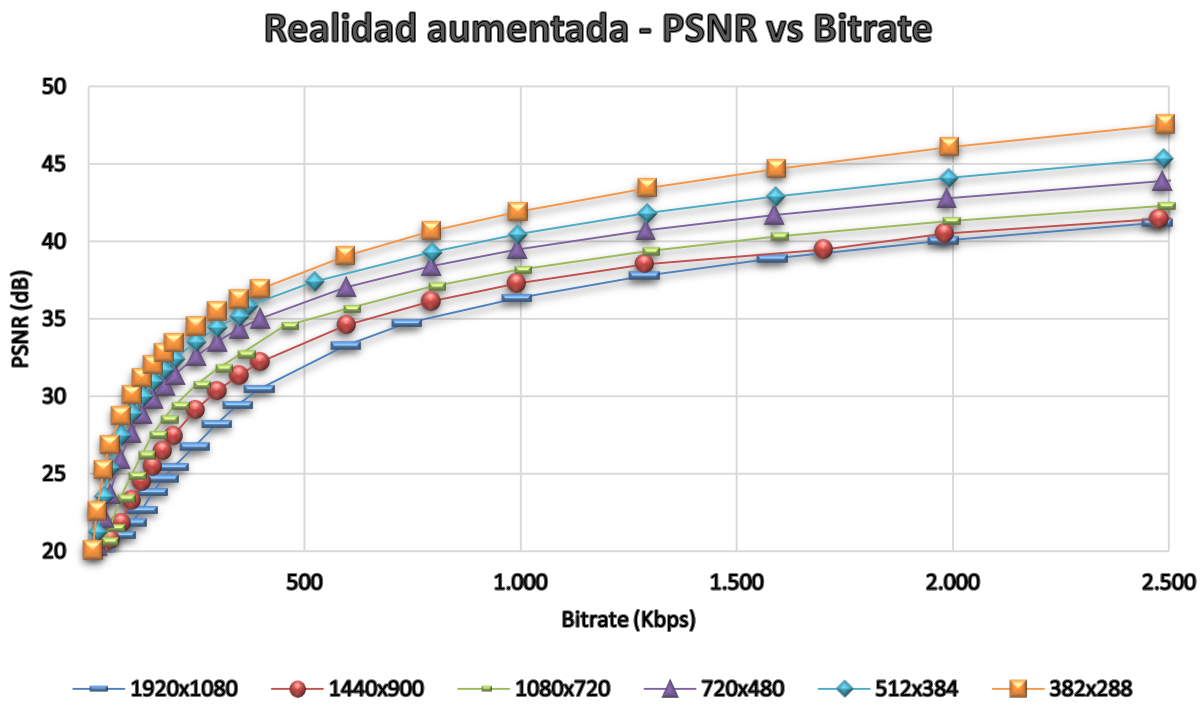


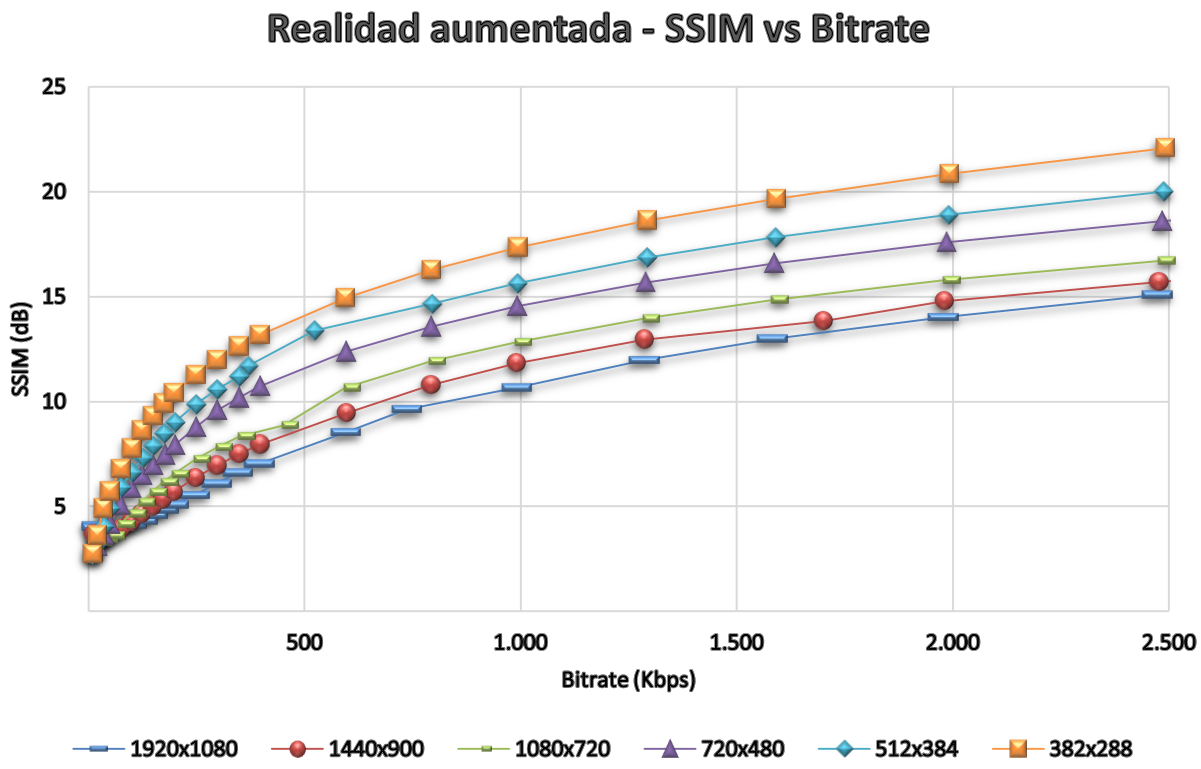




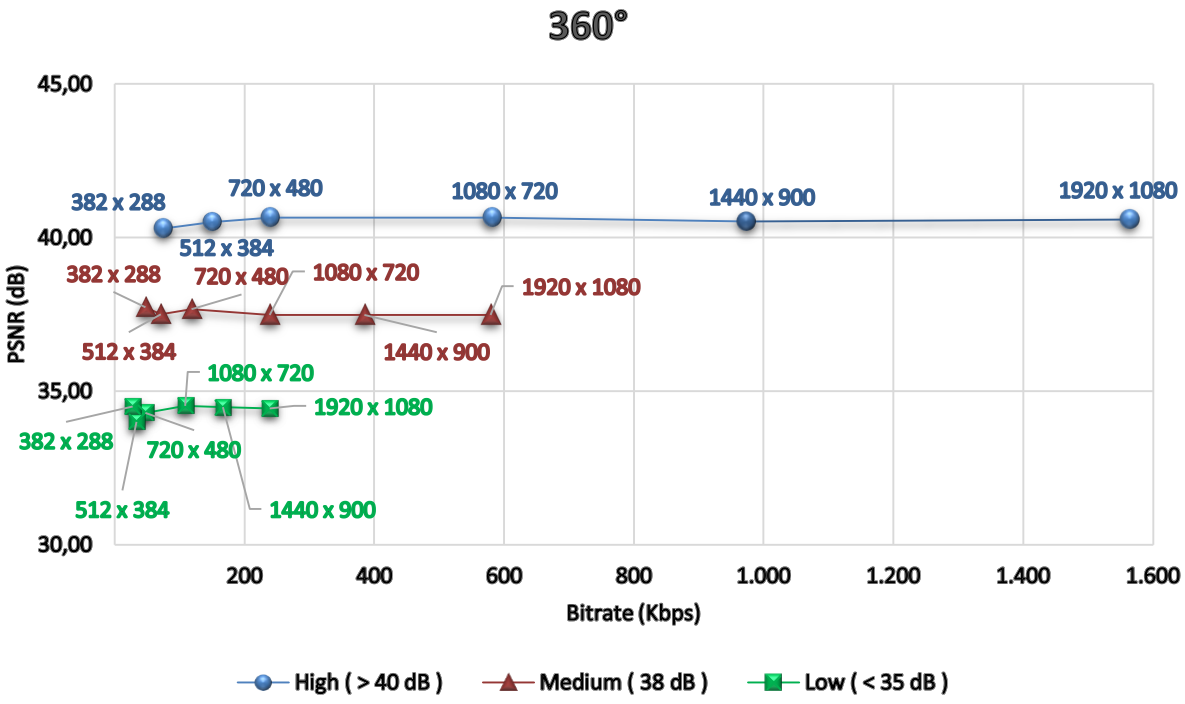
3. Realidad aumentada

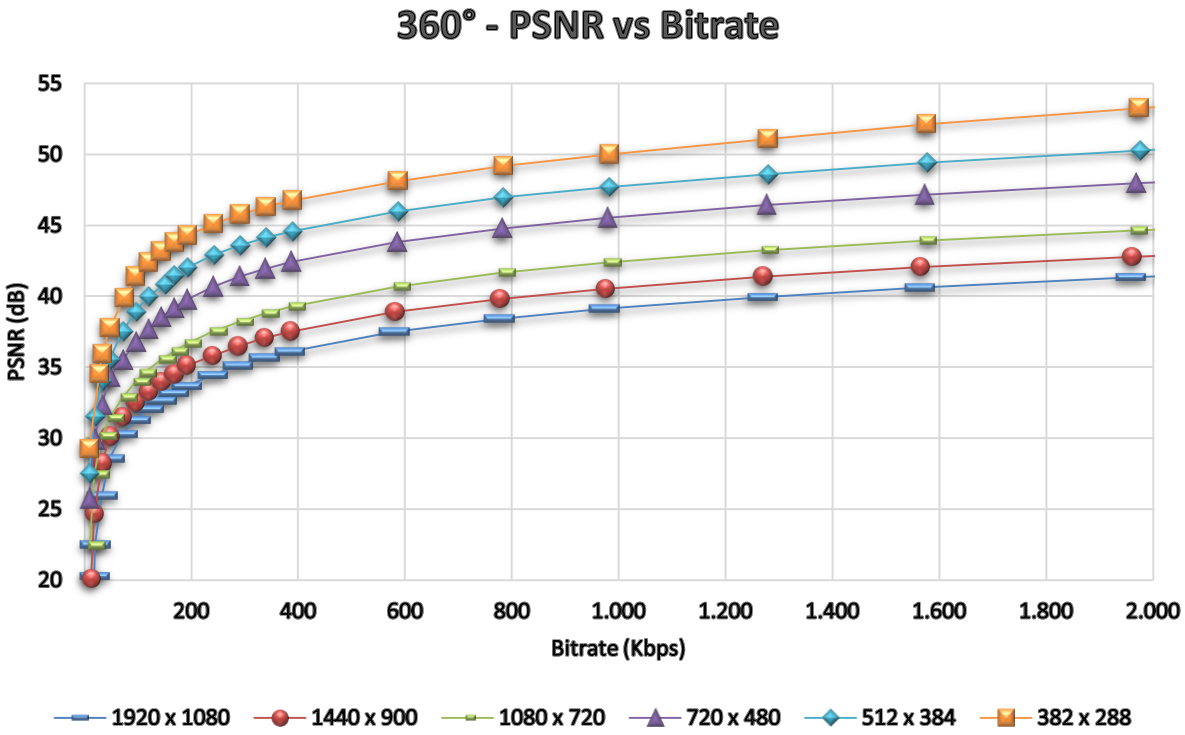


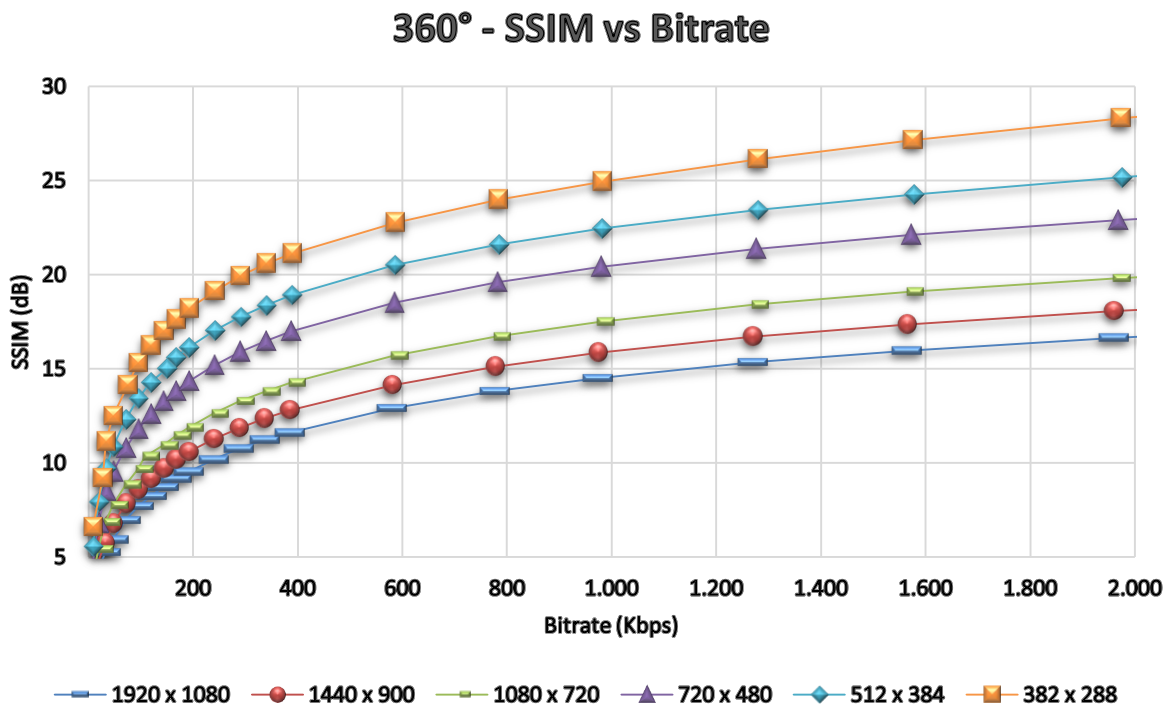

Realidad aumentada - PSNR vs Bitrate
50
45
40
35
30
25
20
PSNR (dB)
500
1.000
1.500
2.000
2.500
Bitrate (Kbps)
1920x1080
1440x900
1080x720
720x480
512x384
382x288


Realidad aumentada - SSIM vs Bitrate
25
20
15
10
5
SSIM (dB)
500
1.000
1.500
2.000
2.500
Bitrate (Kbps)
1920x1080
1440x900
1080x720
720x480
512x384
382x288

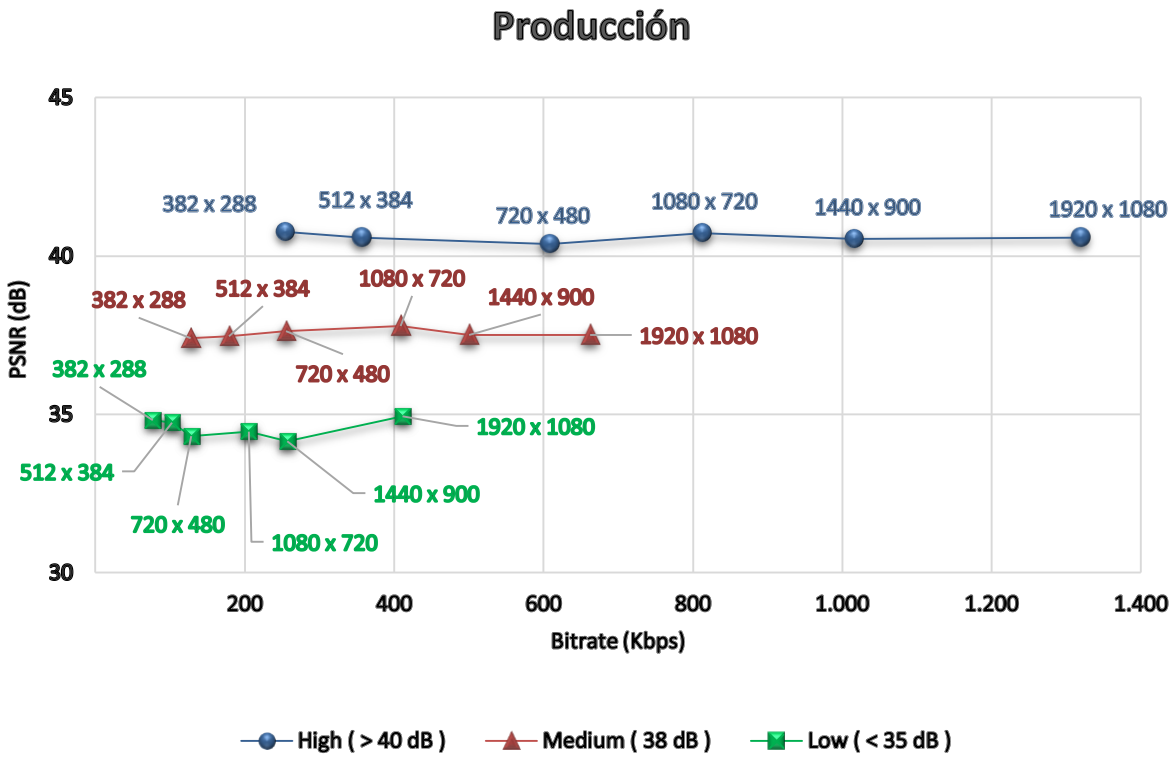
4. 360°

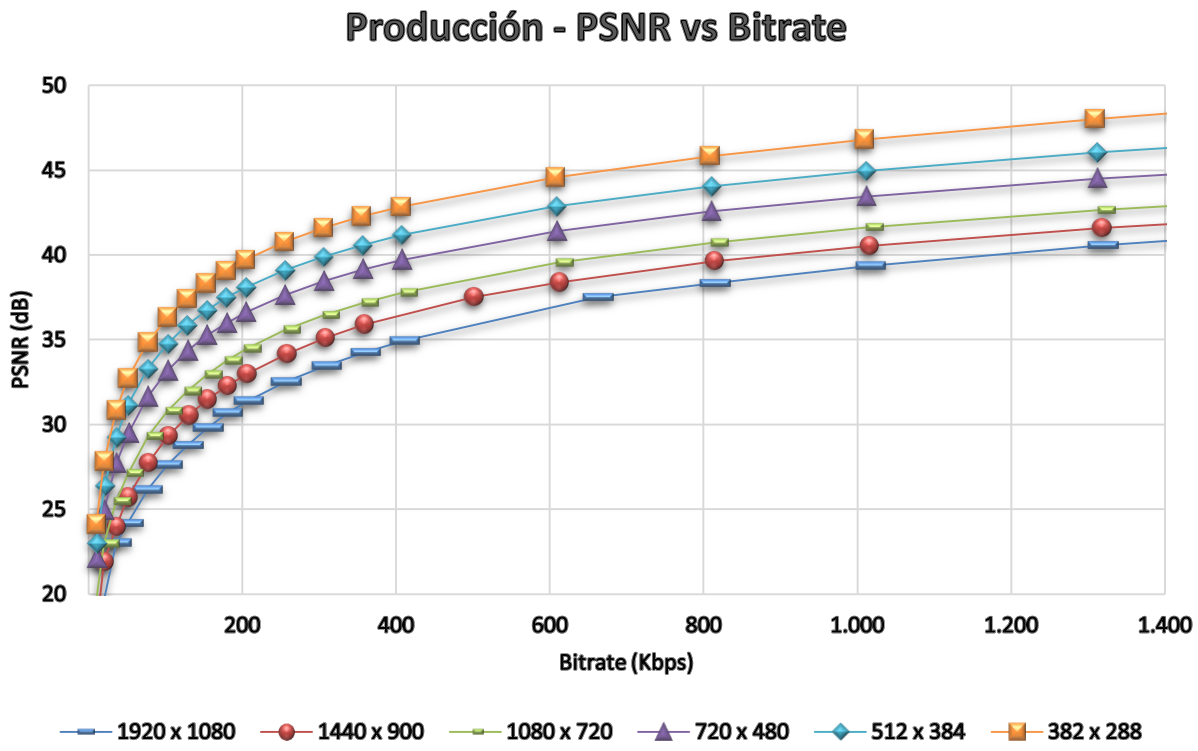

360°
45,00
40,00
35,00
30,00
PSNR (dB)
382 x 288
720 x 480
1080 x 720
1440 x 900
1920 x 1080
512 x 384
200
400
600
800
1.000
1.200
1.400
1.600
Bitrate (Kbps)
High (> 40 dB)
Medium (38 dB)
Low (< 35 dB)

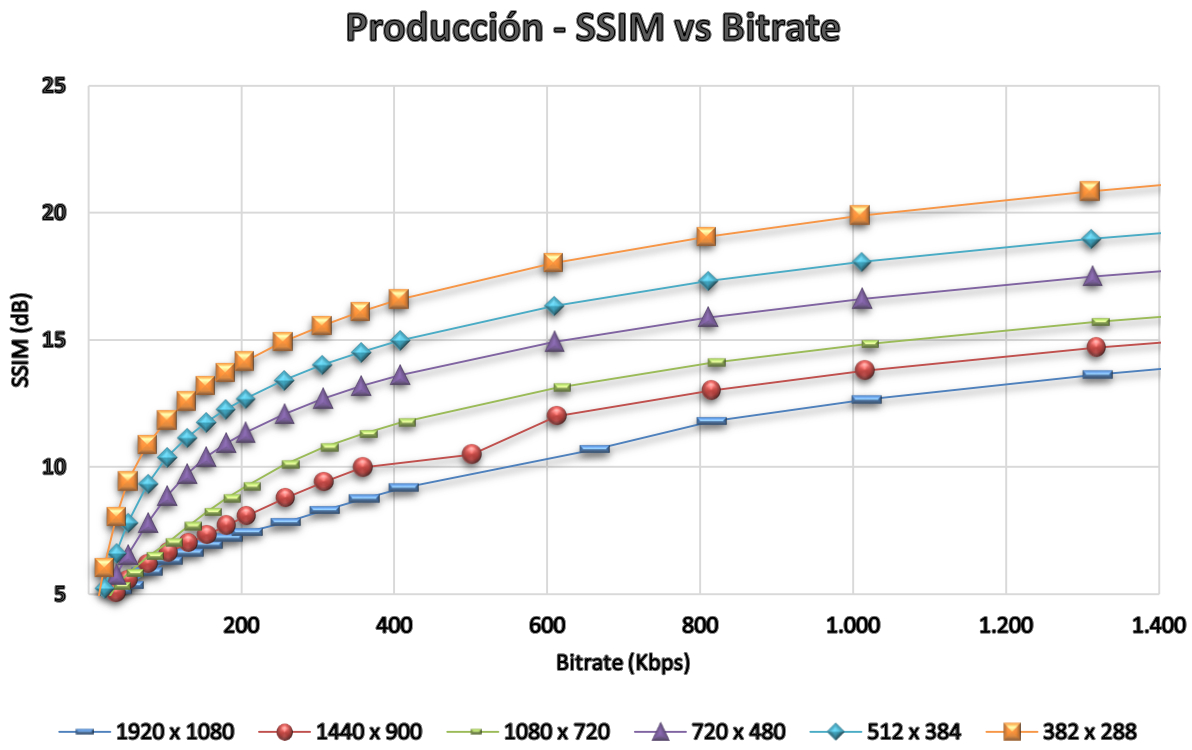

360° - PSNR vs Bitrate
55
50
45
40
35
30
25
20
PSNR (dB)
200
400
600
800
1.000
1.200
1.400
1.600
1.800
2.000
Bitrate (Kbps)
1920 x 1080
1440 x 900
1080 x 720
720 x 480
512 x 384
382 x 288



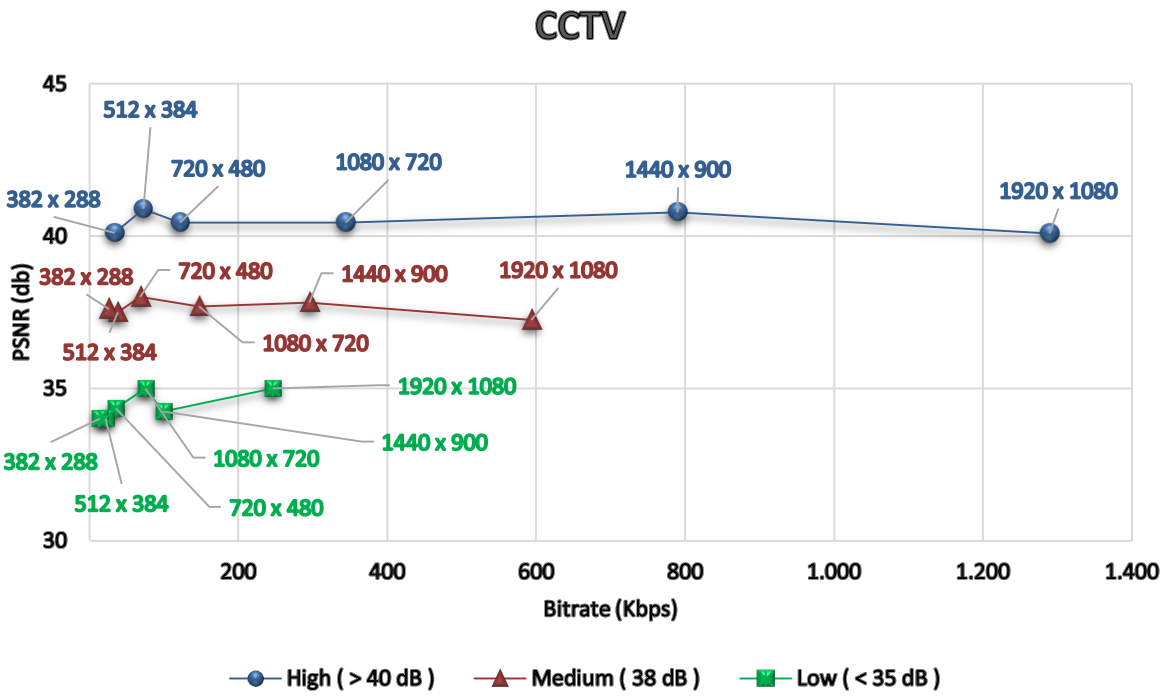
5. Producción

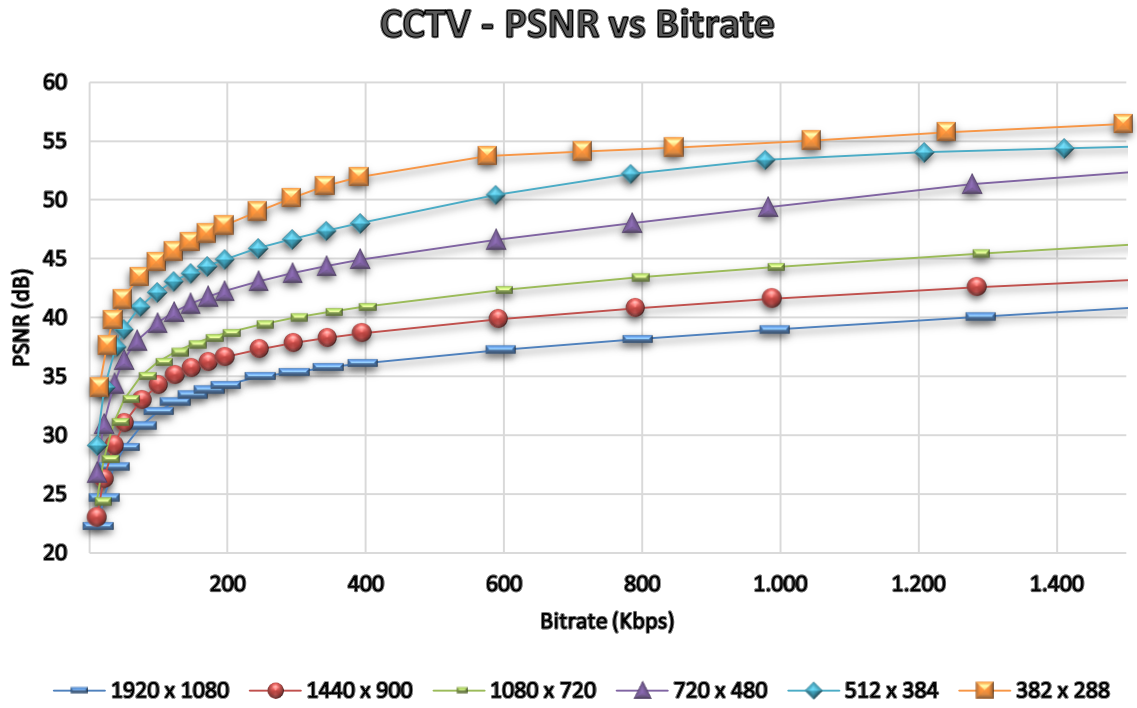


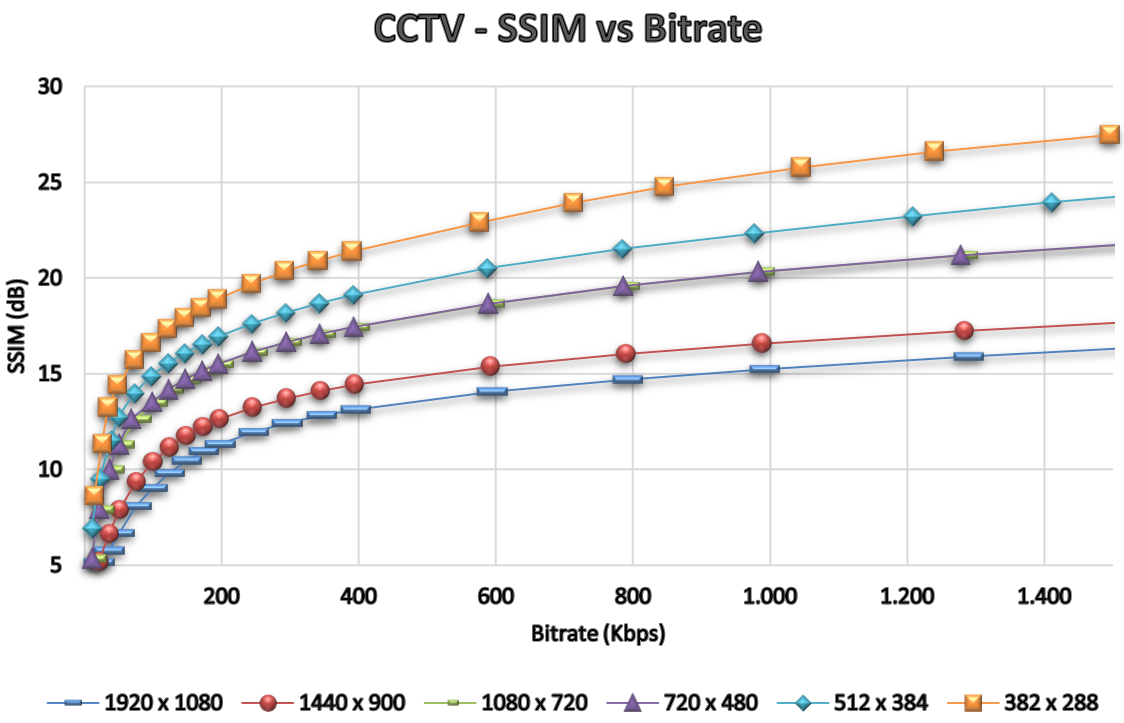

Producción - PSNR vs Bitrate
PSNR (dB)
50
45
40
35
30
25
20
200
400
600
800
1.000
1.200
1.400
Bitrate (Kbps)
1920 x 1080
1440 x 900
1080 x 720
720 x 480
512 x 384
382 x 288


Producción - SSIM vs Bitrate
SSIM (dB)
25
20
15
10
5
200
400
600
800
1.000
1.200
1.400
Bitrate (Kbps)
1920 x 1080
1440 x 900
1080 x 720
720 x 480
512 x 384
382 x 288

6. CCTV


CCTV
PSNR (db)
Bitrate (Kbps)
512 x 384
720 x 480
1080 x 720
1440 x 900
1920 x 1080
382 x 288
High (> 40 dB)
Medium (38 dB)
Low (< 35 dB)


CCTV - PSNR vs Bitrate
PSNR (dB)
Bitrate (Kbps)
1920 x 1080
1440 x 900
1080 x 720
720 x 480
512 x 384
382 x 288


CCTV - SSIM vs Bitrate
SSIM (dB)
30
25
20
15
10
5
200
400
600
800
1.000
1.200
1.400
Bitrate (Kbps)
1920 x 1080
1440 x 900
1080 x 720
720 x 480
512 x 384
382 x 288