UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DISCA
DEPARTAMENTO DE INFORMÁTICA DE SISTEMAS Y COMPUTADORES

Estudio de la técnica de codificación de vídeo HEVC y streaming sobre HTTP usando DASH

TRABAJO FIN DE MASTER

Máster en Ingeniería de Computadores y Redes

Julio 2017

Autor: MATEO GONZÁLEZ MORENO

Director: JOSE OLIVER

Resumen

En la actualidad, es una realidad innegable el hecho que los contenidos multimedia se se están apoderando de internet a pasos agigantados, las redes sociales y las plataformas de distribución de contenidos en general han tenido una enorme acogida liderando los mercados de la mayoría de sectores de la economía al rededor del mundo. Es por esto que se ve como los contenidos se producen cada vez en mayores cantidades y con más facilidad; las cámaras y dispositivos portátiles poseen altísimas resoluciones de grabación de video, cuadros por segundo, calidad de audio, filtros, etc, que, si bien son un gran desarrollo que se evidencia algunas veces de forma transparente, el trasfondo de estos se basa en la evolución de los métodos de transmisión de esta información y las formas de hacer que la mayor tasa de bits requerida por estos contenidos de gran resolución espacial (píxeles por imagen) y temporal (cuadros por segundo) tenga un menor impacto en las redes en las que se transmiten y en los servidores que los almacenan, simplificando el alcance a estos y que, por consecuencia, la experiencia del usuario mejore.

Actualmente los codificadores de video H.264 y VP9 lideran el mundo multimedia: El primero, con un gran rendimiento en relación compresión vs calidad ya se ve limitado con las tecnologías de ultima generación de definiciones de video como Ultra HD, 4K, 8K y hasta 10K que aumentaron en tamaño de los contenidos de manera exponencial y que, por consiguiente, la compresión debe mejorar su eficiencia de la misma forma. Por su parte, para superar este problema, VP9 ya ha ingresado al mercado de plataformas como YouTube en donde se suben millones de horas de video diarias en diferentes calidades que, si bien puede codificar de forma eficiente las ultimas tecnologías de video y ademas tiene la ventaja sobre sus competidores por ser software de código abierto, aun puede mejorar más en términos de costo computacional, tiempos de codificación y métodos de compresión entre otros. Es por esta razón que paralelamente desde hace algunos años la Joint Collaborative Team on Video Coding (JCT-VC) ha desarrollado la evolución de H.264, hasta hoy líder en términos de codificación de los reproductores de la mayoría de dispositivos multimedia. HEVC (High Efficiency Video Codec) o también llamado H.265 se encarga entonces de reducir la tasa binaria hasta un 50 % en comparación a su versión anterior brindando mejor calidad de video, aumentando

la complejidad de los algoritmos de codificación y decodificación, traducido en un aumento en el coste computacional.

Este trabajo tiene como fin evaluar las prestaciones que brinda HEVC, sus métodos de codificación y decodificación, se realizaran pruebas de comparación objetivas y subjetivas de rendimiento de este codec, la eficiencia que tiene contra su antecesor H.264 y otros codecs en varios escenarios, se analizará el panorama de este estándar en términos de su inclusión en la actualidad de la reproducción de contenidos multimedia en diferentes plataformas y se propondrá una alternativa de distribución de contenidos en streaming adaptativo (DASH) sobre HTTP que permite un incremento en la calidad de experiencia, adaptándose a las condiciones de conexiones de red que tenga el usuario final.

Palabras clave:

Codificación, compresión, vídeo, HEVC, multimedia, streaming, DASH, HTTP.

Keywords

Coding, compression, video, HEVC, multimedia, streaming, DASH, HTTP

Índice de figuras

Índice general

Capítulo 1

Introduccion

En la actualidad, con la veloz evolución en materia de tecnologías de dispositivos, los contenidos que circulan por las diferentes plataformas basadas en redes de comunicaciones son cada vez más utilizados por usuarios de todo el mundo. Tan solo en materia de equipos móviles según estadísticas a Junio de 2017 de la *Oficina de Censado de Estados Unidos*[1], existen unos 7,398,256.345 habitantes en la tierra, mientras que al mismo mes según *GSMA Intelligence*[2] existen más de 8,135,671,567 conexiones móviles y M2M, esto ratifica que la demanda de dispositivos en el mundo y contenidos accesibles por red, son un tema de importancia para los desarrolladores de tecnologías.

En el campo de la multimedia, desde los años 80's se comenzó una migración generalizada del formato en el que se trataban las señales de video y audio, lo anterior debido al auge que tenían las redes de comunicaciones como internet en donde los contenidos análogos no eran adecuados para su emisión y transmisión. Para ello la ITU-T en union con el proyecto *JCT-VC* comenzaron a trabajar en la creación de estándares de compresión de video para reducir el flujo de bits total en los archivos, que cada vez eran de mayores tamaños, mediante el uso de codificadores que comprimen la señal de video digital, y poder así almacenarlos y transmitirlos de manera más eficiente. Además se desarrollaron decodificadores que lograran transformar estos bits y reconstruir las secuencias y cuadros de videos y proyectarlos en cualquier plataforma de tecnología emergente.

En la actualidad los desarrollos en el campo de la calidad y resolución de videos han llegado a alcanzar resoluciones de UHD, 4K, 8K, 10K, entre otros, esto se traduce en la elaboración de codificadores que puedan conservar calidades visuales extremadamente altas y reducir al máximo el flujo de bits.

En el pasado, la familia *ITU-MPEG* desarrollaba la mayoría de estándares de codificación de video, entre ellos se encuentra uno de los más extendidos en el mundo para reproducción de contenidos en la web, el *H.264*. Aunque este estándar tuvo una buena acogida, con los modernos dispositivos de creación de contenidos los algoritmos de codificación deben ser cada vez más potentes, por lo que se ha optado por la utilización de de arquitecturas multinucleo, procesamiento paralelo, instrucciones multihilo o mediante el uso GPU's. Para el caso de los decodificadores se puede hacer énfasis en los basados en procesadores de propósito general *(GPP)*, basados en procesadores digital de señales *(DSP)* y basados en arquitecturas específicas.

Una vez se aumenta la complejidad de tratamiento de video y se aumenta la capacidad de procesamiento de los equipos encargados de la compresión, se pueden crear algoritmos mas complejos, con mayor tolerancia a fallos e interferencias y mejorando la calidad de los codificadores actuales. Es entonces cuando en Abril de 2013 *ITU-T* crea el estándar evolutivo a *H.264*, llamado *High Efficiency Video Coding (HEVC)* o simplemente *H.265*, cuyas características logran reducir a la mitad el régimen binario que generaba *H.264* pero a su vez se genera un breve aumento en el tiempo de codificación de los archivos. Estas y otras características se estudiarán en detalle en las siguientes secciones de este documento, el cual se encuentra enmarcado en la línea de trabajo de estudio de este nuevo estándar, de su complejidad, el proceso de codificación y decodificación, pruebas de de eficiencia y creación de una pequeña plataforma de red de distribución de contenidos en internet.

1.1 Objetivos

Este trabajo fin de master tiene como fin el estudio de la eficiencia del codificador HEVC y su funcionamiento en internet bajo redes HTTP usando tecnología de streaming adaptativo, para esto se tienen dos objetivos específicos:

- Definir a profundidad los métodos empleados por HEVC para el funcionamiento de su proceso de codificación y decodificación de videos, realizar pruebas del desempeño del estándar con videos de diferentes resoluciones, duración, calidad y complejidad, asimismo realizar un estudio de las principales técnicas de codificación de video en la actualidad y comparar la eficiencia de HEVC respecto a H.264 y a los otros codificadores , determinando las razones por las cuales HEVC ofrece mejores prestaciones.
- Implementar un sistema de distribución de contenidos bajo HTTP en internet haciendo uso de streaming adaptativo (DASH) y streaming almacenado para videos codificados en HEVC y otros codificadores, mostrando las ven-

tajas que este tipo de streaming ofrece y determinar en qué plataformas actuales puede llevarse acabo su reproducción.

1.2 Organización de la memoria

En el primer capítulo se realizará una introducción a la situación actual del campo de la multimedia y los contenidos digitales, así como los objetivos que se tienen para cumplir a lo largo del proyecto desarrollado. Posteriormente en el segundo capítulo se detallan los conceptos fundamentales que se deben tener en cuenta para comprender la totalidad del trabajo y la terminología utilizada en el mismo. A continuación en el tercer capítulo se dará una breve descripción de los codificadores de video más importantes que se utilizan actualmente en el mercado. El cuarto capítulo detallará profundamente el modo de funcionamiento de la codificación HEVC, las partes del codificador y decodificador, así como los tipos de predicción para los frames, los filtros utilizados para mejora de la calidad, entre otros aspectos de importancia. En el capítulo 5 se precisan las herramientas y conceptos necesarios para el funcionamiento del streaming de video llevado a cabo en este proyecto. El sexto capítulo muestra las pruebas de comparación de HEVC contra H.264 y los respectivos resultados obtenidos. El capítulo 7 por su parte refleja el desarrollo que se llevó a cabo para la construcción del entorno de streaming de video HEVC usando DASH. Para finalizar este documento se presentan las conclusiones del proyecto en el capítulo 8.

Capítulo 2

Conceptos claves y funcionamiento de un codificador de video

Este capitulo se dividirá en dos partes, en la primera se definirán los conceptos básicos [3][4] que se deben tener en cuenta para la comprensión general del funcionamiento de un codificador, y una segunda que mostrará el esquema general de funcionamiento de un codificador y decodificador de video [4].

2.1 Conceptos Claves

2.1.1 Compresión de video

Hace referencia al proceso mediante el cual se manipulan las señales de video para reducir su tasa de bits; es decir el ancho de banda / espacio que los videos ocupan en formato digital. La compresión de video se basa principalmente en la búsqueda y eliminación de datos que sean redundantes en los archivos conservando los datos entrópicos (escenciales) para que no se afecte la calidad percibida por el usuario, manteniéndola al máximo posible y así poder utilizarlos más eficazmente para propósitos de emisión o almacenamiento en cualquiera de las plataformas o dispositivos tecnológicos relacionadas con contenidos multimedia.

2.1.2 Macrobloque

Un video contiene diferentes tipos de secuencias, muchas de estas poseen redundancia, es decir datos iguales o casi iguales en cada espacio de tiempo; principalmente en escenas en las cuales el fondo es estático, o mantiene los mismos elementos durante un determinado tiempo. Para el proceso de compresión de video, se analizan las imágenes secuenciales una a una, se dividen en cortes; o *slices* y estos se dividen a su vez en macrobloques.

Un macrobloque es la unidad básica de video en donde se realiza un proceso denominado compensación de movimiento, es decir que mediante algunos algoritmos se hace un análisis de la redundancia en el video y el movimiento de una secuencia entre cada cuadro. Estos bloques se componen de información de luminancia y crominancia (luminancia Y y un par de crominancia Cb, Cr).

Cuando se realiza un proceso de codificación básico se aplican diferentes transformadas, la más común en la actualidad es la DCT, a través de la cual se convierte una secuencia de muestras y se obtienen coeficientes que pasarán a ser valores cuantificados una vez perdamos precisión en su representación. En la cabecera de un macrobloque se puede obtener ademas de datos de luminancia y crominancia, información acerca de tipo de macrobloque (I, P o B), valor del cuantificador, dirección del bloque en la imagen, entre otros.

2.1.3 Frames

Los fotogramas o *"frames"* son cada una de las imágenes instantáneas que hacen parte de un archivo de video, la puesta secuencial de éstas generan una sensación de movimiento produciéndose así un video, en la actualidad en el cine se utilizan de 24 *frames per second (fps)*, mientras que en archivos digitales de alta calidad se suelen utilizar 25 o 30 *fps*.

A continuación se explican los 3 tipos principales de frames luego de aplicar algún proceso de compresión en un video.

Frame Intra (I)

Es un fotograma que se codifica solo referenciandose a si mismo, por esta razón posee mas información en su contenido entrópico y su proceso de codificación se hace mas lento, sin embargo son utilizados como puntos de referencia para localización que los decodificadores requieren dentro de una secuencia minimizando así la tasa y propagación de errores.

Frame Predicted (P)

Para este tipo de frames la codificación es un poco más compleja que en el caso anterior, ya que no solo depende de sí mismo sino de un proceso de predicción o estimación del movimiento y además de tomar una decisión acerca de cuál es la mejor forma de codificar el macrobloque partiendo de los frames anteriores.

Los frames *P* poseen una propiedad que permite saltar la codificación de algunos macrobloques que tengan información que sea casi igual a los de los frames anteriores, en ese caso se ignoran y se sustituyen los pixeles por los del pasado dando paso una compensación de movimiento, es decir se empieza a reducir el tamaño de bits del video y como consecuencia aumenta el tiempo de codificación en comparación a los *I*, pero se va logrando mayor compresión.

Frame Bi-predicted/Bi-directional (B)

En los frames B, se realiza una codificación similar a la de frames *P*, debido a que se usan frames del pasado, pero también se usan diversos métodos de predicción para obtener información de frames futuros, realizando un promedio entre estos y dando un posible mejor resultado. En los primeros estándares de codificación como H.261 no se utilizaban este tipo de frames por lo que el resultado era de menor calidad. En codificadores como MPEG 1/2 y *H.264* se comenzaron a referenciar y en H.264 se permite la utilización de 1, 2 o más imágenes codificadas anteriormente para realizar la predicción, sin embargo este tipo de frames toman aún más tiempo de codificación pero se requiere menor flujo de bits para su almacenamiento que los *P*.

2.1.4 Cuantificación digital

La técnica de cuantificación hace referencia al procesamiento de imágenes para su compresión hacia un único valor. Esta se define como compresión con pérdida debido a que cuanto mayor sea el factor que se tiene, se obtendrán menos niveles de cuantificación, es decir que existirán más errores: se perderá algo calidad pero se reducirá el tamaño del archivo.

La cuantficación minimiza el número de los colores de una imagen mediante procesos de interpolación para aumentar la impresión que existen más tonos de los que posee una imagen dando como resultado a esta minimización un archivo que tendrá un menor tamaño.

Los procesos de cuantificación y compresión se basan en la capacidad que tiene el ojo humano para percibir las componentes de las imágenes, normalmente se tiene una alta capacidad de diferenciar el cambio de brillo de una imagen pero no es sensible a cambios pequeños en la intensidad de variación del brillo a altas

frecuencias, de esta forma mediante la cuantificación se puede reducir el tamaño de la información que posee altas frecuencias y llevarlas hacia valores negativos o cero sin que se perciban cambios drásticos en la imagen. Ademas de ello, el ojo humano es más sensible a los cambios en luminancia que de crominancia, por lo que si se manejan modelos de cuantificación en espacios de colores como Y'UV puede facilitarse este proceso.

Los codificadores básicos realizan un proceso de división de las imágenes en bloques de diversos tamaños, y mediante la aplicación de una DCT se separan los componentes de bajas y altas frecuencias tanto en las componentes verticales como en las horizontales, estos bloques se dividen o multiplican por una matriz especial de cuantificación dependiendo del estándar utilizado; en el caso de *MPEG1* es una matriz establecida, luego, en estándares como *MPEG2* y *H.264* estas matrices se pueden personalizar. Cada valor resultante se redondea de forma tal que no se generen demasiadas perdidas de calidad en las frecuencias visibles y altamente diferenciables. Estas matrices se dividen en dos secciones, la parte superior izquierda representa los valores de frecuencias bajas y visibles, mientras que la inferior derecha las de frecuencias altas que normalmente luego del proceso de división se reducen y se aproximan a cero.

2.2 Esquema general de un codificador de video

Un codificador de video se basa en diversas técnicas que se han mantenido durante el tiempo y poco a poco se han mejorado y se han ido añadiendo más para obtener resultados como los que se estudiaran en el protocolo HEVC.

El esquema básico de un codificador se muestra en la figura 2.1. El estándar *H.261* fue el primero que estableció estos pasos para lograr comprimir las tramas de bits de video.

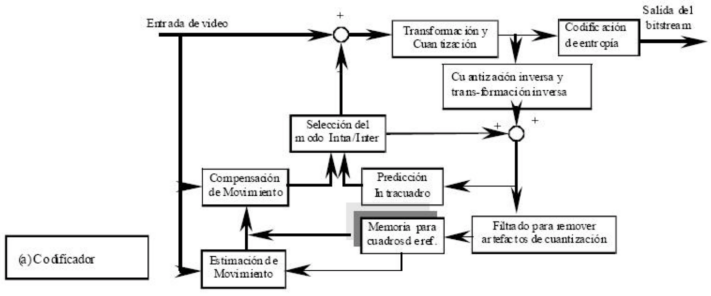

Figura 2.1: Esquema básico de un codificador de video

La trama original se compone de multiples frames trabajados bajo algún modelo de color como el *YUV* debido a que el codificador de video se debe centrar más en los cambios que se generarán en as componentes de luminancia que en los cambios de color, esto partiendo del principio mencionado anteriormente acerca del ojo humano.

2.2.1 Modelo de color YUV

Se utiliza el modelo de color *YUV* en reemplazo del antiguo modelo *RGB* ya que este conoce de antemano las limitaciones del ojo humano y realiza un proceso de eliminación de señales de alta definición de color suprimiendo elementos que no sean perceptibles y no generen diferencias al receptor, evitando así el uso excesivo de ancho de banda. Este proceso es conocido como *codificación perceptiva*.

El modelo de color *YUV* se basa en tres matrices, al igual que *RGB*. La diferencia radica en que son dos elementos de crominancia (señales que transportan el color de la imagen) *U* y *V*, la fase y amplitud de estas crominancias corresponden aproximadamente a la saturación (cantidad de color) y el matiz (de que color es) de la imagen, y la tercer señal es de luminancia *Y*, que contiene la intensidad luminosa "brillo"que va a ser percibido por el ojo.

El modelo *RGB* anterior estaba formado por 3 matrices de color (rojo, verde y azul), en donde cada pixel de las matrices ocupa 8 bits, cuando se superponen las tres matrices se forma la imagen de color de 24 bits. En *YUV* también se realiza la superposición de las tres matrices para obtener la imagen final a color.

La figura 2.2 se muestra un ejemplo del modelo de color *YUV* y sus representaciones en crominancia y luminancia.

Posterior a esta división de la imagen se realiza otro submuestreo en el que se da menos importancia a las dos matrices de crominancia reduciendo su tamaño y dejando sin modificación alguna a la luminancia, generando esto casi ningún cambio para la calidad de la señal. Existen 5 tipos de submuestreos, denotados así:

8:4:4: Las componentes de color se muestrea a frecuencia completa, mientras que la componente de luminancia tiene del doble de la frecuencia.

4:4:4: En esta la frecuencia de muestreo en luminancia y crominancia es la misma.

4:2:2: Las crominancias son muestreadas a la mitad de la frecuencia de la luminancia en dirección horizontal.

4:1:1: Las crominancias son muestreadas a un cuarto de la frecuencia de la luminancia en dirección horizontal.

Figura 2.2: Descomposición de imagen en modelo YUV

4:2:0: Las crominancias se muestrean a la mitad de frecuencia de la luminancia en dirección horizontal y vertical.

La figura 2.3 muestra gráficamente los procesos de submuestreo.

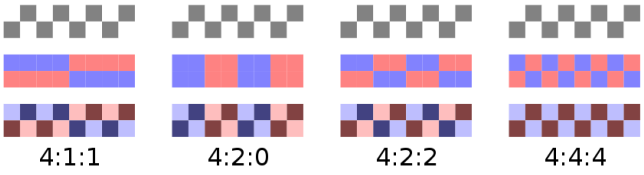

Figura 2.3: Diferentes tipos de submuestreo de imágenes

2.2.2 Codificación Predictiva

La predicción en la codificación se realiza principalmente para disminuir la redundancia espacial y temporal; es decir la información que esta repetida en una zona de la imagen o tiene la misma información durante un periodo de tiempo. Cuanto más alta sea la predicción, menor será el residuo entre los valores actuales y los predichos dando origen a un archivo más pequeño.

Existen dos tipos de predicción utilizados en video:

Predicción Intra Cuadro

La predicción Intra cuadro surgió con la creación del estándar *H.264* partiendo de la necesidad que se tenía para la reducción de bits en la codificación inter, ya que trabaja con los valores originales de cada pixel.

Esta predicción se fundamenta en *averiguar* qué contenido tendrá un bloque en un frame basado en los pixeles vecinos al bloque que se va a predecir y que ya han sido codificados.

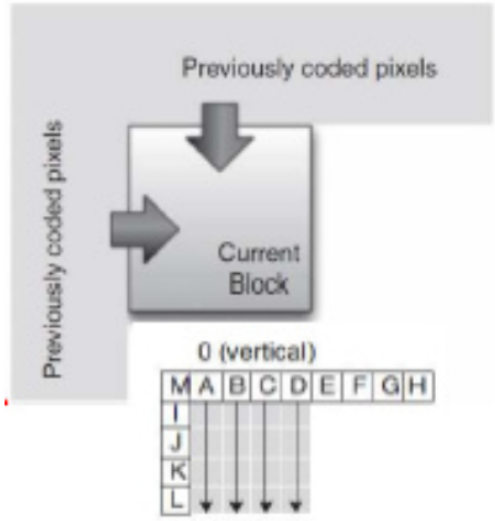

Figura 2.4: Bloque bajo predicción Intra

Dependiendo de cada codificador de video se tiene un número determinado de predicciones, y se escoge el que tenga menor diferencia respecto a la imagen original. Para finalizar se establece la diferencia entre la imagen original y la predicha para la reducción de la redundancia espacial entre los macrobloques, y se codifica dicha diferencia, también llamada residuo.

Predicción Inter Cuadro

A diferencia de la predicción Intra Cuadro, esta ya existía en los primeros codificadores de video aunque de formas mas sencillas. Su función esencial es averiguar lo que contendrá un bloque partiendo de bloques anteriores o futuros disminuyendo así la redundancia temporal entre los bloques.

Los frames *Predicted* P usan únicamente predicciones pasadas y son utilizados en los codificadores más antiguos, mientras que los frames *Bi-Predicted* B pueden usar frames de referencia pasados o futuros para reducir ań más la diferencia entre la predicción realizada

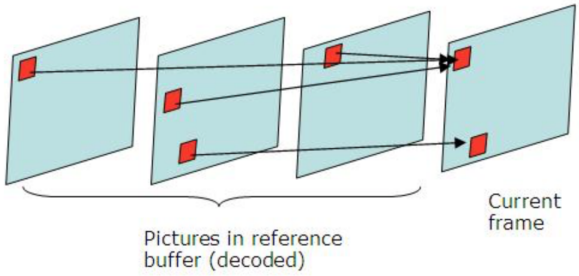

Figura 2.5: Bloque bajo predicción Inter

2.3 Decodificación básica de video

La decodificación de video se centra en la serie de técnicas utilizadas en el proceso de codificación pero realizadas de manera inversa. El tiempo de decodificación de video es mucho menor debido a que no es necesario hacer estimaciones de movimiento entre los frames, predicciones y demás técnicas de codificación. La imagen 2.6 muestra el esquema seccionado en bloques de un decodificador básico de video.

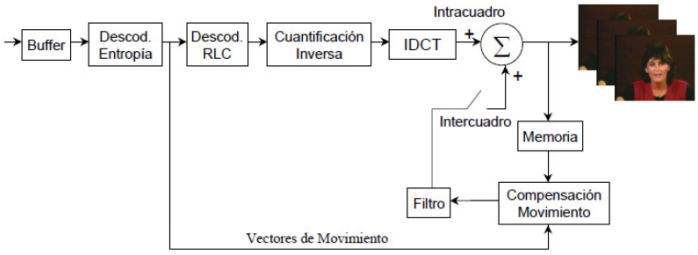

Figura 2.6: Esquema de un decodificador de video

En el caso de un video codificado como inter, la imagen original se obtiene mediante la aplicación de los vectores de movimiento a los frames, dando origen a un frame de referencia, obteniendo entonces la imagen predicha, si a lo anterior se le suma el residuo de la imagen se obtendrá la imagen original. En el caso que el video sea codificado como intra se suma la predicción realizada a partir de los pixeles vecinos en el caso de utilizar predicción intra, si no se usa este tipo de predicción, ya se tienen los valores originales.

Capítulo 3

Los codificadores del presente

3.1 Introducción

Esta sección tiene como objetivo el acercamiento de algunos de los estándares que hoy en día se posicionan como líderes en el mercado de la compresión de video, por los beneficios que trajeron tanto a la distribución y difusión de contenidos en plataformas de red, como en reducción de tasa de bits con respecto a sus codificadores antecesores.

3.2 H.264

3.2.1 Introducción

El H.264 o MPEG-4 [5] es un estándar de alta compresión de video desarrollado por el conjunto de expertos de las organizaciones ITU-T Video Coding Expert Group (VCEG/ y el ISO/IEC Moving Picture Experts Group (MPEG).

Desde el principio estas organizaciones se plantearon el proyecto de diseño y creación de un estándar de compresión que redujera notablemente la tasa de bits de sus antecesores, H.263 y MPEG-2 evitando el fuerte aumento de la complejidad de diseño con una calidad de imagen muy buena. Para eso en 2001 se unen estas organizaciones bajo e nombre de Joint Video Team (JVT) y en 2003 sacan a la luz el H.264 por parte de ITU-T y MPEG-4 por parte de ISO/IEC. El estándar se denominó H.264/MPEG-4 AVC, o simplemente H.264, que en un principio estuvo enfocado hacia el video en baja calidad principalmente para aplicaciones por internet y video conferencias, basadas en 8 bits por muestra.

Desde 2003, H.264 ha tenido diferentes extensiones, debido a que se han implementado mejoras a raíz de la evolución que han tenido las plataformas de comunicaciones, distribución de videos en red y grandes mejoras en la calidad de video obtenido con las cámaras de video actuales. Se realizaron entonces notables modificaciones del estándar para implementarlo en un mundo un poco mas profesional con altas calidades y bitrates.

La primera versión base fue la H.264/AVC, posteriormente se extendió a H.264/SVC, que a raíz de la veloz evolución de dispositivos que permiten reproducción de contenidos multimedia se lograba adaptar a la capacidad de computo y resolución de pantalla de cada dispositivo usado. Luego de esto se crea el H.264/MVC, que puede realizar codificación para multiples vistas, en este caso la reproducción de contenidos con tecnología 3D fue la que dio origen a la extensión del estándar.

3.2.2 Funcionamiento H.264

Para el proceso de codificación de una trama de video bajo el estandar de H.264, se debe tener en cuenta principalmente que este se divide principalmente en dos capas: Una primera capa denominada capa de codificación de video (VCL), en donde se realiza el proceso de codificación del video mostrado en la Figura 3.1. Posterior a la codificación existe una segunda capa llamada Capa de abstracción de red (NAL), la cual se encarga de realizar una adaptación de la información que la VCL entrega, hacia un diverso número de tecnologías para el almacenamiento y transporte de los contenidos comprimidos.

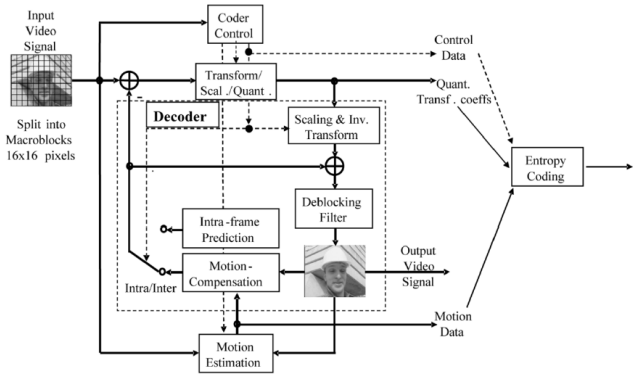

Figura 3.1: Esquema de codificador H.264

El video luego de la capa VCL se organiza en una "NAL Unit", que se compone por una cabecera, en donde se indica el tipo de datos de la capa de abstracción de

red, y otra parte más grande de la carga útil de secuencia de bytes que no se habían procesado (RBSP). Las unidades NAL pueden clasificarse según su información, puede hablase de VCL NAL units si se refieren a los datos de los frames de video y non-VCL Units cuando son solo para control, que podría a su vez ser un conjunto que pueden aplicarse a varias VCL NAL.

Al ser desarrollado por la misma organización, el proceso de codificación de H.264 es similar que se explicará en el capitulo 4 con *HEVC* ya que contiene métodos de predicción, diferentes tipos de Slices, cuantificación, filtros, entre otros.

3.3 VP8

3.3.1 Introducción

VP8 [6] Es un compresor de código abierto, desarrollado por On2 Technologies; actualmente de propiedad de Google. Es el sucesor del estándar *VP7* y se rige bajo la norma RFC6386 de 2011. Funciona basado en el sistema de bloques tradicionales como la mayoría de esquemas de codificación, hace uso principalmente de descomposición de frames en sub bloques cuadrados de pixeles, la predicción de algunos subbloques se realizan usando bloques construidos anteriormente mediante métodos de predicción al igual que *H.264* usando la transformada del coseno, además de esta utiliza igualmente la transformada de Walsgh Hadamard.

3.3.2 Historia

En el mes de Septiembre de 2008 On2 Truemotion desarrolló una versión de *VP8*. En 2009 es comprada por Google por aproximadamente 124.6 millones de dólares, y un año después deja a *VP8* como un software de código libre. Una vez Google hizo esto, la Free Software Foundations solicita que reemplace al codificador *H.264* en Youtube, y mezclar HTML5 y *VP8* para la reproducción y codificación de sus videos.

WebM es un formato multimedia desarrollado por Google que tuvo como finalidad la integración de HTML5 con los principales navegadores para hacer más eficiente la reproducción de contenidos en internet. Tuvo apoyo de empresas como Mozilla, Opera y otros 40 fabricantes de software y hardware. Microsoft por su parte también hizo parte de esta iniciativa en su novena versión de Internet Explorer.

3.3.3 Funcionamiento

VP8 [7] funciona exclusivamente con imágenes YUV 4:2:0 de 8 bits, en este formato cada pixel de 8 bits en los planos *U V* corresponden a un bloque de luma de 2x2 pixeles en el plano *Y*. Internamente *VP8* descompone cada cuadro de salida en Macrobloques de 16x16 y cuyas dimensiones *U y V* son de 8x8. Los datos a nivel de macrobloque en una trama comprimida se producen en un orden similar a la de los pixeles que comprenden cada frame, en *VP8* los Macrobloquyes se componen adicionalmente en subbloques de 4x4, cada macrobloque tiene 16 subloques Y, 4 subbloques U y 4 Subbloques V.

En *VP8* solo existen dos tipos de tramas, Intraframes e Interframes (I y P para MPEG) que, de la misma forma que en MPEG se codifican sin referencia a ninguna otra secuencia y los P se codifican en base a otros marcos anteriores más recientes. *VP8* no permite un ambiente de fotogramas perdidos, debido a que los frames dependen de sus anteriores o de sus estados naturales, esto en ocasiones genera errores en la codificación y posterior decodificación. A diferencia de MPEG no se usa predicción bidireccional ya que únicamente predice frames en base a los anteriores.

VP8 usa tres tipos de frames para sus predicciones: ultima imagen, frame de referencia alternativo y frame de oro.

El frame de oro es una especie de frame buffer que almacena un fotograma de video de un punto pasado usando flags que notifican cuando se actualiza el buffer, mejorando así la predicción que puede partir desde ese frame de oro. Por su parte el frame alternativo de referencia puede ser usado o no como referencia en la predicción, o asimismo puede ocultarse sin que esto afecte el proceso.

3.3.4 VP8 vs H.264

En aspectos comparativos a *H.264* se pueden abarcar:

- Predicción: Ambos utilizan bloques de 4x4 8x8 y 16x16, sus técnicas de predicción varian muy poco.
- Predicción Intra: *H.264* utiliza frames de tipo I B P, *VP8* utiliza únicamente frames I y P y adicionalmente los frames alternativos y los frames de oro. Sin embargo al no utilizarse frames de tipo bidireccional no se aprovecha tanto la redundancia temporal que esta ofrece, su equivalente serian los frames alternativos aunque no funcionan de la misma manera.
- En temas de coste computacional, se tiene que cuanto más nivel de detalle tenga una imagen, *VP8* tardará mas tiempo en codificarla que *H.264*. Principalmente se debe a que *VP8* tiene falencias en los métodos de predicción

respecto a su competencia y se refleja en mayores de tiempos del proceso de codificación.

3.4 VP9

3.4.1 Historia

En Junio de 2013, una nueva versión de codificadores de Google aparece. Esta vez *VP9* [8] es la siguiente y actual generación de codificación en plataformas como Youtube, es igualmente software de código abierto, siendo esto un punto muy importante frente a sus competidores. Este tema se detallará más adelante.

3.4.2 Funcionamiento

VP9 utiliza nuevas técnicas de codificación a diferencia de *VP8*, haciéndolo muy superior.

Creación de superbloques: Nuevos tamaños de bloques de 32x32 y 64x64, agregan parámetros de codificación, explota mejor la redundancia temporal principalmente en los frames Inter mediante una codificación de dos pasos: Un primer paso donde se codifican todos los bloques que son mejores de forma Inter y luego se realiza un llenado de los macrobloques con estos, mejora la calidad de bloques intra pero se tiene más computo para su decodificación.

Un punto importante para los videos en alta definición es que *VP9* tiene una DCT mas extensa, de 8x8 o 16x16 y es reajustable dependiendo del video. Esta transformada es aplicada a la luminancia y crominancia para convertir la señal de residuo luego de la codificación por *transformada de residuo* en valores transformados.

3.4.3 VP9 vs HEVC

En comparativa con *HEVC*, *VP9* tiene un gran punto a favor y son sus patrocinadores, debido a que se desarrolla por Google y su expansión se ha dado a los principales navegadores y plataformas de reproducción de contenidos tanto en ordenadores, como en dispositivos móviles y televisores. Algunos expertos de la industria de Streaming opinaban que de nada servía un codificador que fuera superior a otro si a la hora de transmitirlo nade lo podía ver por la incapacidad de reproducción de los navegadores. Respecto a *HEVC*, sin embargo cabe aclarar que Microsoft en su navegador Edge tiene incluido el soporte para este tipo de video, y recientemente en la conferencia de desarrollo de Apple 2017 se anunció que se

incluirán, tanto en la nueva versión del sistema operativo OSx High Sierra como en el iOS 11, compatibilidad de reproducción de videos con codificación *HEVC*.

Si bien las modificaciones realizadas a *VP9* con respecto a su antecesor le brindan la posibilidad de reproducir contenidos con mayor calidad, el coste computacional no deja de ser muy elevado frente a su competidor *HEVC*. Martin Rarebek y Touradj Ebrahimi concluyen en [9] que *HEVC* es mucho más potente que *VP9* (35.6 % más), en otras palabras, *HEVC* produce un contenido de igual o mejor calidad y cantidad de bits, con menor tiempo y coste de computo. En términos generales, si bien *VP9* está mas expandido debido a su gran patrocinio y *HEVC* posee mejores características de compresión, este ultimo al no ser libre y no haber podido ser implementado aún en los navegadores y reproductores, se queda estancado en un ámbito de experimentación y es una latente opción para su inclusión en el mundo de la multimedia. Sin embargo empresas productoras de contenido multimedia bajo demanda como Netflix[21], ha venido estudiando HEVC y ha probado la superioridad de éste contra sus competidores, habilitando entonces los contenidos en 4K para la reproducción de sus contenidos

Capítulo 4

High Efficiency Video Coding (*HEVC*)

4.1 Introducción

Desde hace aproximadamente 30 años el mundo multimedia en las redes de comunicaciones y dispositivos móviles ha crecido a pasos agigantados dando origen a videos de HD *(High Definition)* y UHD *(Ultra High Definition)*, entre otros formatos de grandes tamaños. Internet es la plataforma más popular encargada de la transmisión de estos contenidos, pero con el crecimiento en tasas binarias de dichos contenidos se requieren de nuevos métodos de codificación para aumentar la eficiencia de su almacenamiento o transmisión por la web, haciendo mejor uso del ancho de banda que es consumido.

En el pasado, las organizaciones ITU-T, VCEG y ISO/IEC MPEG trabajaban de manera autónoma para la creación de los diferentes estándares de compresión de video. Con la evolución de estos y la alta demanda que se tenia en materia de innovación, estas organizaciones aliaron a sus mejores expertos bajo el nombre de *Joint Collaborative Team on Video Coding - ITU (JCT-VC)* y en los últimos años han logrado popularizar en la red un formato de compresión llamado *H.264*, el cual al ser generalizado en el mundo posee ciertas ventajas; como por ejemplo la interoperabilidad que tienen los codificadores y decodificadores de video que los diferentes fabricantes crean para diversas plataformas.

En el mes de enero de 2013 se llevó a cabo la primera versión de un estándar que, según estudios, lograba reducir hasta en un 50 % el regimen binario de *H.264* para contenidos en muy altas resoluciones y lograba mantener la calidad que el

compresor *H.264* ofrecía. Bajo el nombre de H.265 o *High Efficiency Video Coding* (*HEVC*) [10][11][12], se desarrolló este codificador que según estudios actuales se dice que es de 1.5 a 2 veces más complejo en procesos de codificación y decodificación con respecto a su antecesor, y es además especial para tráfico de videos con resoluciones de HD, UDH(4K y 8K) que generan exceso de bits en las redes actuales. Posteriormente en abril de 2015 se crea la ultima versión de *HEVC* especialmente diseñado y ejecutado en entornos que permiten procesamiento y arquitecturas en paralelo. Además de este tipo de procesamiento, el estándar *HEVC* tiene como objetivo mejorar los sistemas de recuperación de errores, mayor efectividad en la codificación, soportar resoluciones mayores, mejorar los métodos de predicción, entre otros.

4.2 Esquema del codificador/decodificador *HEVC*

El proceso de codificación básico de *HEVC* posee características típicas de cualquier codificador de video, como predicciones inter o intra; tal y como el primer codificador que se creó en el mercado, H.261. Los avances que se han producido en el diseño de *HEVC* se han enfatizado en el interior de sus modelos y no en la estructura de este.

En la figura 4.1 se muestra el diseño del codificador *HEVC*. Se observa un *frame* que hace parte de una secuencia de video, esta imagen esta dividida en diversas regiones de igual tamaño y será codificada con predicción *intra*, es decir que no dependerá de ningún otro *frame* anterior ni posterior, los *frames* a continuación se predecirán de manera *inter* haciendo uso de los vectores de movimiento y el residuo generado entre los *frames* anteriores al que se predecirán.

El codificador usa además, bloques que hacen parte del esquema de decodificación, debido a que en su funcionamiento se debe añadir la reconstrucción de las señales que codifica para de esta forma usar los *frames* reconstruidos en vez de los originales.

Además de lo anterior, el decodificador *HEVC* usa estimaciones de movimiento, codificación de entropía, dominio espacial de las imágenes y filtros que reducen ruido en la secuencia. La secuencia codificada viaja en unidades *Network Abstraction Layer (NAL)*, en su interior se tiene una zona denominada cabecera y una segunda llamada *Raw Byte Sequence Payload (RBSP)*. Además de ello existen dos tipos de unidades NAL; las que llevan el video en sí, denominados *Video Coding Layer (VCL)* y otras que poseen información necesaria para la decodificación del video, como por ejemplo la resolución de los *frames*.

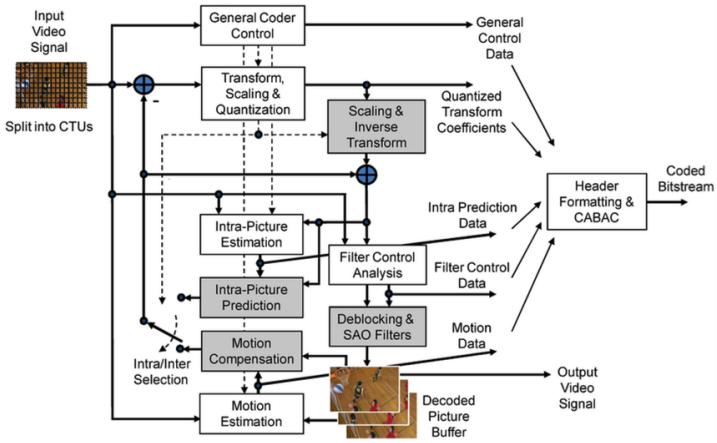

Figura 4.1: Diagrama de Bloques de Codificador *HEVC*

Ejemplo de decodificación HEVC

En este apartado se describirá un ejemplo rápido de como se realiza la decodificación de una secuencia de video *NAL*. La Figura 4.2 precisa el diagrama de bloques del decodificador *HEVC*.

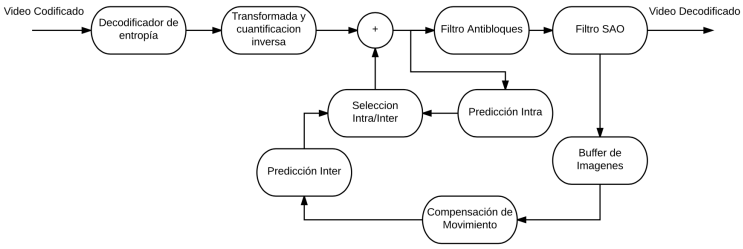

Figura 4.2: Diagrama de Bloques de Decodificador *HEVC*

- El primer paso es la *decodificación de la entropía*, la cual generará un aumento en el numero de bits de la información que proporcionan parámetros como tipo de macrobloque, vector de movimiento, predicción, entre otros.
- Estos bits son transportados luego a un *bloque de cuantificación y transformación DCT inversa*, los cuales recuperan los niveles de amplitud de los coeficientes y los transforman del dominio de frecuencia al dominio espacial.

- Dependiendo de la salida que se tenga del multiplexor selector del tipo de predicción se tiene la imagen decodificada, en el caso que se tenga predicción intra los coeficientes se añadirán a la predicción intra, mientras que si son inter, se añadirán de nuevo a la predicción inter.
- Para finalizar la decodificación se pasan las imágenes a los filtrados antibloques y SAO, esto eliminará ciertas distorsiones generadas en la codificación, dando origen a la imagen decodificada, que será almacenada en una memoria, o buffer de almacenamiento de imágenes.

4.3 Partes de un codificador *HEVC*

A continuación se realizará un análisis profundo de los bloques que componen la codificación *HEVC*.

4.3.1 División de frame

En estándares anteriores a *HEVC*, la división de la imagen se realizaba por medio de *slices* que podían ser divididos también en macrobloques. En el caso de este codificador las imágenes se dividen en bloques llamados *Coding Tree Units (CTU)*, de diversos tamaños a saber: 16x16, 32x32 y 64x64 pixeles. Los paquetes de 32x32 y 64x64 son tamaños más grandes que los de los anteriores estandares, mejorando de esta forma la compresión, debido a que a menor numero de bloques en la imagen menos información habrá que enviar de cada uno y mejor se aprovecha la redundancia del contenido del bloque. En la figura 4.3 se puede observar gráficamente lo mencionado, a la izquierda se tiene una imagen dividida en macrobloques en el estándar *H.264* con tamaños pequeños, mientras que a la derecha *HEVC* muestra su beneficio al dividir en CTUs de diversos tamaños la imagen, según su redundancia espacial.

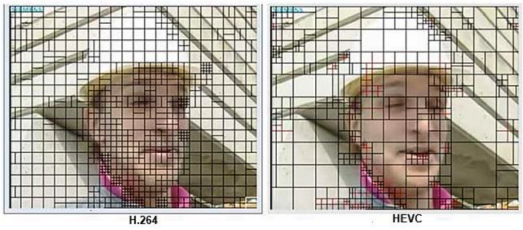

Figura 4.3: División en *frames* de H.264 vs *HEVC*

Coding Units

La subdivisión de una unidad *CTU* se hace a través de tres bloques denominados *Coding Tree Block (CTB)*, dos de los cuales representan información de crominancia y el otro de luminancia, este ultimo usualmente posee el doble de tamaño que los de crominancia. Cabe resaltar que los CTUs son de tamaños variables y se adaptan a las necesidades que tenga que suplir el codec. Sumado a lo anterior, los *CTBs* se pueden dividir aun más, en unidades denominadas *Coding Blocks (CBs)*, y cuya división se realiza gracias a una estructura denominada *quadtree*, la cual usa el contenido del *CTB* para la división en *CBs*. Por otro lado la unión de dos crominancias *CB* y una luminancia *CB* da origen a una *Coding Unit (CU)*.

En la figura 4.4 se muestra la división de un *CTU* en sus respectivos *CTBs*, así como un *CTB* dividido en *CBs* y un *CB* dividido para dar origen a un *CU*. Además se muestra gráficamente estas divisiones en estructura quadtree en la Figura 4.5.

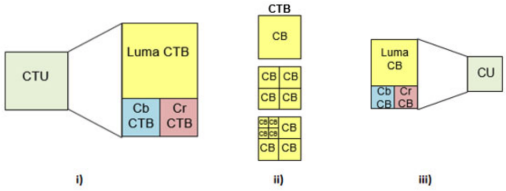

Figura 4.4: División de CTU en CTBs, CTB en CBs y CU en CBs

Con respecto a *H.264*, *HEVC* da la posibilidad de aumentar el tamaño de los mabrobloques, debido a que con los videos en HD y UHD, la probabilidad que existan partes de la imagen con zonas planas, es decir con la misma o similar información, es más elevada. A diferencia del estándar pasado en donde los tamaños más grandes eran de 16x16 pixeles, *HEVC* usa bloques de mayores tamaños para la codificación de estas zonas.

Prediction Units

Al codificar una imagen en una área específica se hace a nivel de *CUs*, pero aun estas pueden ser bastante grandes para determinar el tipo de predicción que se va a seleccionar, de modo que, siguiendo la estructura de quadtree, cada *CU* puede ser dividido aun más. Estas divisiones son denominadas *Prediction Unit (PU)*, que se define como la unidad elemental de predicción.

Si se va a realizar una predicción intra, el tamaño del *PU* es el mismo que el de *CU* que lo contiene; es decir división 2Nx2N. El único caso diferente a este se presenta

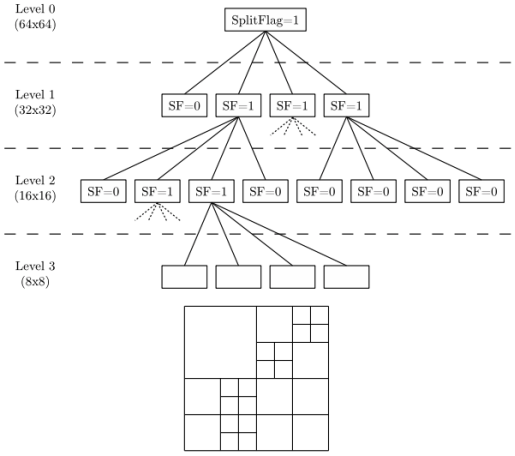

Figura 4.5: Estructura quadtree de división de *CTU* hasta *CU*

cuando el *CU* es el mínimo que se permite, una bandera de codificación permite escoger esa división o puede subdividir en cuatro PU de la forma NxN, alcanzando un tamaño de 4x4 pixeles.

Si se realiza una predicción inter, el área del *CU* puede dividirse en 1 2 o 4 *PUs*. Para que se diera un particionamiento de 4 *PUs* se debería tener un *CU* de tamaño mínimo y sus *PUs* son de tamaño mayor a 4x4. A diferencia de la codificación intra, en este caso los *PUs* no pueden lograr ese tamaño. *HEVC* crea entonces un estándar de división llamado *Asymmetric Motion Partitioning AMP* que logra particionar un *CU* en dos bloques de diferente tamaño, teniendo en cuenta que deben ser siempre mayores de 16x16.

La figura 4.6 muestra algunos de los posibles particionamientos de los *PUs*. Las *CUs* que se predicen de modo *inter* se pueden aplicar cualquiera de los modos, mientras que *intra* solo permite las dos primeras formas de división.

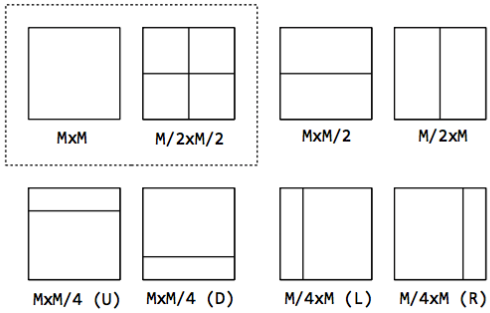

Figura 4.6: Particionamiento de *PUs*

Transform Units

Las *Transform Units (TU)* Son las unidades básicas formadas a partir del residuo de la predicción, siempre tienen forma cuadrada 32x32, 16x16, 8x8 (luminancia) y para cada uno de estos tamaños se define una función basada en la DCT y además se aplica el proceso de cuantificación.

La división en unidades cada vez más pequeñas para los procesos descritos anteriormente (CTu, Cus, PUs, TUs) se adaptan en el proceso de predicción a cada imagen, sus composiciones y posibles cambios. En la figura 4.7 se muestra un ejemplo de división de la imagen *quadtree*, en donde las lineas azules representan *CUs*, los verdes PUs y los rojos son codificaciones *intra*.

Las secciones de la imagen que tienen mayor detalle se dividen en bloques más pequeños para poder predecir con mayor certeza el siguiente movimiento del *frame* obteniendo residuos menores. Aquellos de mayores tamaños son partes del *frame* que se mantienen muy similares en el video, y los rojos son parte de aquellos *frames* predichos como *intra*, debido a que no se tienen referencias anteriores de bloques similares, posiblemente por objetos nuevos que aparecen en la secuencia.

Con respecto a *H.264*, *HEVC* usa transform units que pueden contener varios PBs en el modo de predicción inter.

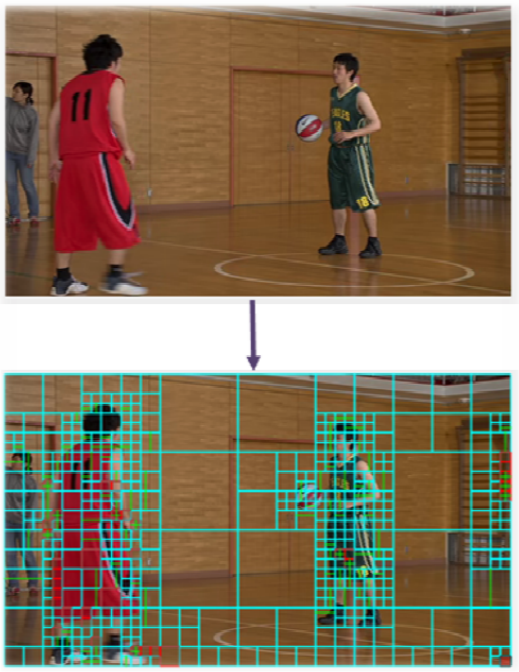

Figura 4.7: Ejemplo de división de imagen

Slices

Anteriormente se analizó el concepto de división de imágenes en *CTUs*, sin embargo hacer estas divisiones en un objeto tan grande como por ejemplo un *frame* de alta calidad conlleva algunos problemas, el principal inconveniente es la alta tasa de probabilidad que se presenten fallos o perdidas en los datos que conforman el *frame*, para este problema desde el estándar *H.264* se planteó una solución mediante el uso de *slices*.

Un *slice* se define como una region o secuencia de *CTUs* que están contenidos en un *frame* y son decodificados de manera independiente de otros *slices* que hagan parte del mismo *frame*, esto quiere decir que no necesita información de ningún otro *slice* para mostrar la información que contiene. Dependiendo también de su tamaño, un *slice* puede comprender un *frame* completo o una region. En la figura 4.8 se muestra gráficamente el proceso de división por *slices* de un *frame*.

El objetivo principal de un slice es abrir la posibilidad de una resincronización en una perdida de datos. Para esto se debe restringir el número de bits por cada *slice*, limitando así el número de CTUs.

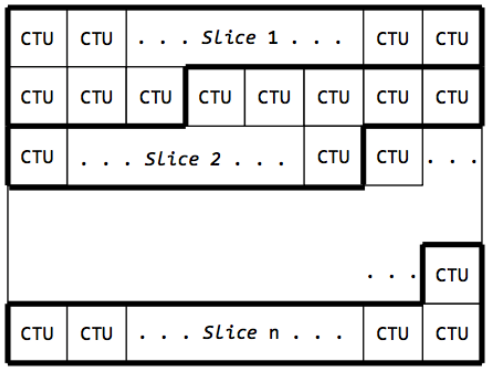

Figura 4.8: División de un *frame* por *slices*

Con respecto a *H.264*, los *slices* no disponen de la herramienta *Flexible Macroblock Ordening* **FMO**, la cual sitúa los macrobloques en los *slices* de forma flexible. en *HEVC* no se usa debido a su complejidad y elevado costo.

Tipos de Slices

- *Slice tipo I*: En estos *slices* se usa únicamente la predicción *intra* para su proceso de codificación.
- *Slice tipo P*: En estos *slices* los *CUs* usan tanto la predicción *intra* como la predicción *inter*, pero se debe hacer uso necesariamente de un único vector de movimiento compensado para la predicción *inter frames P*.
- *Slice tipo B*: En estos *slices* se añade a la capacidad de un *slice* tipo *P* y la posibilidad de usar un máximo de dos vectores de movimiento.

Tiles

Los *tiles* son regiones de forma rectangular; de hasta 20 columnas y 22 filas autocontenidas e independientes entre si y de los *CTUs*. Un *slice* puede contener multiples *tiles* y viceversa.

El propósito principal de los *tiles* es ser usados dentro de las técnicas de procesamiento paralelo y aumentar sus capacidades de codificación y decodificación.

Normalmente la imagen se divide en regiones rectangulares como las mostradas en la siguiente figura, sin embargo surge un problema con los *tiles* que contienen varios *slices*, para ello se propone que los *slices* no deben contener ningún *CTB* fuera del *tile* y de la misma forma si un *slice* tiene varios *tiles* el *slice* debe comenzar desde el primer CTB del primer *tile* y terminar con el ultimo CTB del ultimo *tile*. La figura 4.9 muestra la división rectangular de la imagen en *tiles* de la forma tradicional y la forma en que se realiza el procesado de los *tiles* al lado derecho de la misma.

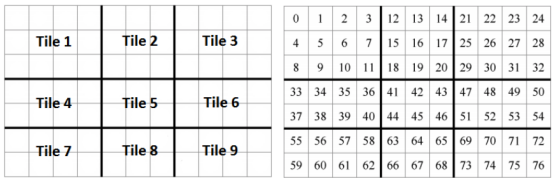

Figura 4.9: Ejemplo de división de imagen en *tiles*

Con respecto a *H.264*, los *tiles* son más flexibles en *HEVC* debido a que las técnicas de procesamiento paralelo son consideradas menos complejas de FMO

4.3.2 Proceso Paralelo Wavefront (WPP)

El proceso paralelo wavefront se usa para dividir en filas de *CTUs* a un *slice* determinado. La decodificación de cada fila se pone en marcha en el instante en que se tengan los datos de entropía de la fila anterior, en otras palabras la codificación de entropía se deduce a partir de la fila inmediatamente anterior.

El proceso de funcionamiento de WPP comienza con un hilo que se encarga del procesamiento de la primera fila, luego que los dos primeros *CTUs* sean procesados un segundo hilo puede comenzar a procesar la segunda fila y en cuanto el segundo *CTU* de esta segunda fila sea procesado un tercer hilo de procesamiento comenzara el proceso en la tercera fila, y así sucesivamente. La figura 4.10 muestra el funcionamiento de WPP.

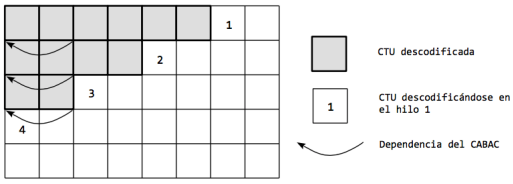

Figura 4.10: Funcionamiento de WPP

La condición de dos CTUs de la fila anterior debe cumplirse debido a que dependiendo de la información que se tenga, la predicción intra puede ejecutarse y el cálculo de los vectores de movimiento también, añadiendo mejoras de velocidad de decodificación ya que se accede más rápido a los datos de los *tiles* o WPP, evitando la decodificación de todo el *slice*.

4.3.3 Predicción Intra

Dentro de los *frames* que hacen parte de un video existen enormes similitudes; por ejemplo en una escena de playa, existirán bloques prácticamente iguales del mar o la arena, es así que los codificadores de video aprovechan este tipo de similutudes; llamado redundancia espacial, para usar bloques codificados anteriormente y usarlos en la imagen actual, reduciendo la tasa de bits utilizada en el video.

La predicción intra se hace a nivel de *PU*, mediante matrices cuadradas de tamaño NxN. Puede contener cinco conjuntos de pixeles representados por las letras A, B, C, D, E, de forma que la disponibilidad de los pixeles se determina por la posición de la PU. La figura 4.11 muestra gráficamente estos cinco conjuntos de pixeles.

Para *HEVC* se crearon más modos de predicción intra que en *H.264* en total se definen 35 modos para luminancia; *Planar (modo 0), DC (modo 1), 33 modos angulares (2 al 34)* y para la crominancia el codificador elige uno de los modos de predicción *intra planar, intra angular e intra dc*.

Modos principales de predicción intra

- *Predicción Intra DC*: Este se considera el modo más simple, incluido en estándares anteriores como el *H.264*, usa todos los pixeles de la *PU*, de los conjuntos B y D. Se usa el valor medio de las muestras de las muestras vecinas al bloque a codificar. Como se aprecia en la Figura 4.12.

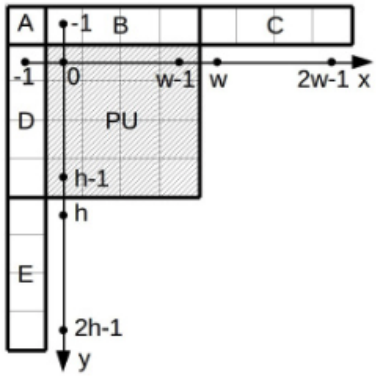

Figura 4.11: Conjunto de pixeles A,B,C,D,E

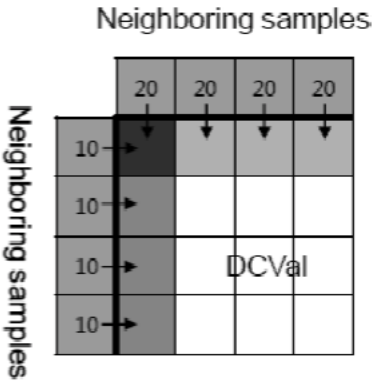

Figura 4.12: Predicción intra DC

- *Predicción Intra Planar*: Es considerado como el modo que más recursos computacionales consume para codificar. Se basa en el promedio de interpolaciones lineales; es decir una aproximación matemática para dar un valor a un pixel desconocido usando 4 muestras. Esta predicción se implementó principalmente para mejorar la calidad de los paisajes con gradientes pequeños en su variación. *H.264* solo era compatible con tamaños de PB de 16x16, mientras que en *HEVC* se logra compatibilidad con todos los tamaños de PB.

- *Predicción Intra Angular*: En este modo en cada *TB* se hace una predicción sobre muestras cercanas en una dirección determinada. En total *HEVC* establece 33 direcciones utilizables para este propósito. Los ángulos se han diseñado para cubrir con gran profundidad las zonas próximas a todas dirección horizontal, vertical y diagonal, debido a que la mayoría de *TBs* son similares a sus próximos en dichas direcciones. En la figura 4.13 se observan gráficamente los modos pertenecientes a la predicción intra angular.

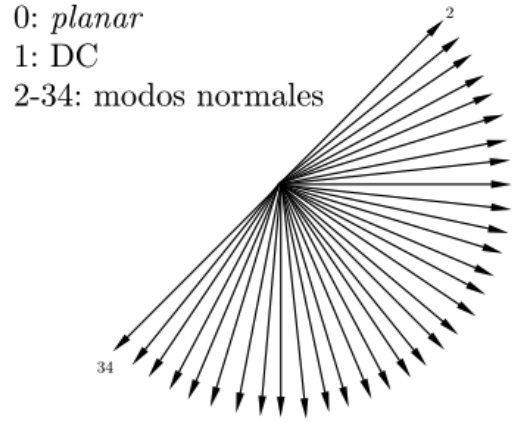

Figura 4.13: 33 ángulos definidos para intra angular

4.3.4 Predicción Inter

La predicción inter es aquella basada en imágenes anteriores para su funcionamiento, explota similitudes de los *frames* de referencia vecinos en el dominio del tiempo, lo que la vuelve más compleja que la predicción intra.

Esta codificación se desarrolla a nivel de *PU* y puede tener forma rectangular o cuadrada, esto permite a las *PUs* adaptarse mejor a las formas de los objetos evitando dividirlas mucho. Para codificar un bloque, se busca uno en imágenes anteriores similares al que se pretende predecir y el resultado de esta búsqueda se conoce como *vector de movimiento*, que apunta hacia la zona del *frame* de referencia donde el residuo es mínimo respecto a la otra imagen.

Modos de Predicción de movimiento

En *HEVC* se puede predecir el vector o vectores de movimiento mediante un proceso llamado *Advance Motion Vector Prediction (AMVP)*, mejorando la codificación de la imagen obteniendo el vector movimiento de un bloque vecino y seleccionando el vector de una serie de candidatos que luego calculará la diferencia entre el vector de movimiento predichos y el real. El resultado se transmite al decodificador.

Otra forma para realizar la predicción de movimiento es la llamada *Merge mode*, en la cual se obtiene la información espacial o temporal de los bloques vecinos y selecciona los vectores de movimiento candidatos. La diferencia con el modo anterior es que la lista de candidatos se calcula a través de los vectores vecinos, transmitiendo el indice del mejor candidato.

La figura 4.14 muestra la posición de los 5 candidatos de modo espacial, se comprueban en orden a1 b1, b0, a0, b2. La selección se da si el candidato en la posición se codifico como inter y se encuentra dentro de los límites del *frame*. En el caso del candidato temporal se toma el *PU* de la posición inferior derecha del *frame* de referencia, si no existe este se toma el que se encuentra en el centro.

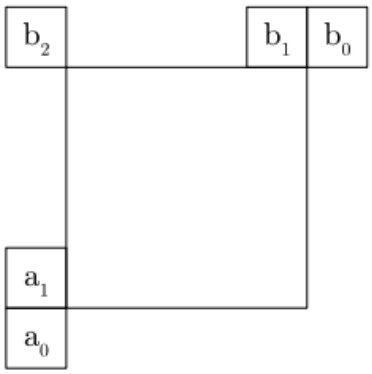

Figura 4.14: Candidatos espaciales

4.3.5 Transformada Y Cuantificación

Al igual que otros estándares de codificación, en *HEVC* se aplica la transformada discreta de coseno *DCT* en la codificación de los bloques *Transform Blocks*. De esta forma se logra hacer el paso de dominio espacial al dominio de la frecuencia. Esta transformada divide el bloque en *TBs* de tamaños 4x4, 8x8, 16x16 y 32x32 agrupando así la energía de cada bloque que luego se usará para la compresión de la imagen. En la figura 4.15 se observa el resultado de una matriz 8x8 con niveles de luminancia, y luego la misma con los coeficientes resultantes de la DCT. Se observa además que la energía esta acumulada en la esquina superior izquierda (bajas frecuencias).

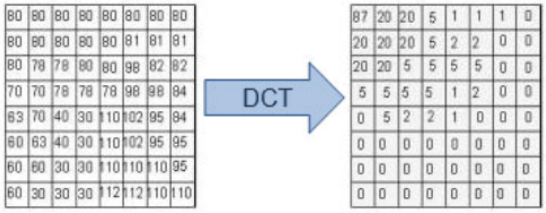

Figura 4.15: Matriz de coeficientes luego de la DCT

La transformada DCT o transformada *core* aplicada a *HEVC* se considera bidimensional, obtenida de transformadas unidimensionales horizontales y verticales, el resultado muestra NxN coeficientes en donde en la coordenada (0,0) se encuentra el coeficiente DC. Otro caso se presenta con los bloques 4x4 de luminancia, ya que el tipo de transformada usada será la transformada discreta del seno (DST) y su beneficio se centra en este tamaño de bloque debido a que los coeficientes con mayores amplitudes se ubican en posiciones cercanas a las esquinas del bloque y son poco variables, en cambio con tamaños mayores la energía se concentra en la esquina izquierda del bloque, por lo que se usa la DCT.

Luego de la aplicación de la transformada, el proceso de cuantificación toma lugar. Usado en *H.264* también, se basa en la aplicación de la cuantificación escalar, de forma tal que a todos los coeficientes que se transformaron en el paso anterior se les aplique el mismo factor de cuantificación para los *TUs*.

4.3.6 Codificación Entrópica

En *HEVC* la codificación de entropía se encarga de hacer una asociación de menor longitud de bits a los elementos más comunes. El estándar antecesor, *H.264* usaba 2 codificadores de entropía, el *Context Adaptative Variable Lenght Coding* (CAVLC) y el denominado *Context-Adaptative Binary Arithmetic Coding* (CABAC). Por su parte *HEVC* adoptó y mejoró este ultimo diferenciándolo en la mejora de su velocidad mediante la paralelización. En otras palabras el algoritmo transforma la información de elementos como las cabeceras de las *slices* o algunas unidades NAL a lenguaje binario, selecciona el modelo de probabilidad y analiza la información de elementos vecinos para mejorar la estimación, y por último a esos datos se les aplica el proceso de codificación aritmética.

4.3.7 Filtro Antibloques

Debido a que la codificación y decodificación se se realiza a nivel de bloques, en la imagen se verán uniones de bloques si no se aplica ningún post proceso. En codificación de grandes bloques como *HEVC* este efecto se visualiza más fácilmente y lo que se hace en este caso es aplicar un filtro antibloque que suavizará notablemente los bordes de los bloques para obtener un efecto más continuo de la imagen y de mejor calidad. Es un filtro pasa bajo de tipo FIR que actúa filtrando de manera horizontal los bloques verticales y viceversa. Concluido esto en los bloques PU se obtiene una imagen más suave. La figura 4.16 muestra gráficamente la comparación entre un *frame* sin el filtro antibloques y a la derecha se observa la mejora de la imagen post filtrado.

Figura 4.16: Imagen sin y con filtro Antibloques

4.3.8 Filtro SAO

El filtro *Sample Adaptive Offset* (SAO) se agregó en *HEVC* debido a que este maneja bloques de tamaños considerablemente grandes y se pueden generar interferencias o distorsiones no deseadas en las imágenes conocidas como *ringing* y *banding*. El filtro SAO es un proceso no lineal que cuando se habilita realiza una clasificación de los pixeles de una region basándose en su intensidad o en su contorno y de esta forma se encarga de la reconstrucción de la señal decodificada suavizando y eliminando estas interferencias en la imagen. La figura 4.17 muestra un ejemplo de una imagen sin filtro SAO y a la derecha el resultado de su aplicación.

Figura 4.17: Imagen sin filtro SAO y luego de la aplicación del filtro.

Modos principales de Filtro SAO:

- *Band Offset Mode*: Este modo se basa en la amplitud de cada muestra, y el rango de las amplitudes es dividido en 32 segmentos llamados bandas, estas son agrupadas a su vez en conjuntos de 4 y se añade un valor positivo o negativo, suavizando de esta forma las áreas en las que las bandas puedan generar interferencias.
- *Edge Offset Mode*: Este modo compara el valor de una muestra con 2 de los 8 vecinos que tiene usando patrones horizontales, verticales y diagonales.

Capítulo 5

Streaming de video

5.1 HTTP

HTTP [13]. Hypertext Transfer Protocol o HTTP (en español protocolo de transferencia de hipertexto) es un protocolo de comunicación desarrollado por el World Wide Web Consortium y la Internet Engineering Task Force. Forma parte de la capa de aplicación de la pila de protocolo OSI y es el protocolo más usado en la actualidad para la transferencia de elementos que hacen parte de una arquitectura de dispositivos en la web.

HTTP es un protocolo que usa el método de petición y respuesta entre un cliente y un servidor que hacen uso de una serie de mensajes para realizar peticiones de los recursos que un servidor puede ofrecer a sus clientes. En el panorama actual, un navegador es un cliente que accede a un servidor web con dirección IP fija y que da servicio a solicitudes procedentes de múltiples navegadores distintos.

HTTP usa TCP como protocolo de transporte, en vez de ejecutarse sobre UDP. Lo primero que se realiza en HTTP es iniciar una conexión TCP con el servidor, a continuación, la petición es enviada mediante su interfaz de socket y accede a los recursos del servidor mediante sus sockets. El servidor http recibe mensajes a través de sus sockets y responde a través de los mismos.

TCP posee dos características esenciales: es un protocolo orientado a la conexión y con servicio de transferencia de datos fiable, esto se traduce en que es un protocolo que intercambia información de control (negociación) antes que se realice el intercambio de información mediante un canal full duplex de intercambio de información, cuando esta comunicación ha terminado, la conexión se debe cerrar. Y es de transferencia fiable ya que TCP está diseñado para que los mensajes lleguen

en el orden adecuado y sin error en los datos transmitidos, además de un uso de control de congestión de la red por la que transita y de los recursos que poseen el emisor y receptor para una mejor transferencia que, en este documento, será de mucha importancia.

Debido a que HTTP no tiene en cuenta la información del estado del cliente, si un cliente solicita dos veces un mismo recurso, el servidor reenviará el objeto las veces que se soliciten. Se define entonces como un protocolo sin memoria.

5.1.1 Conexiones persistentes y no persistentes

Respecto a las conexiones HTTP persistentes y no persistentes se tiene que, en las aplicaciones de internet, se crean sesiones durante tiempos prolongados en donde el cliente hace una serie de solicitudes y asimismo el servidor responde a estas. Cuando se desarrolla una aplicación se debe tener en cuenta si se debe enviar solicitudes para cada petición o si deben enviarse de forma separada. A esto se conoce como conexiones persistentes y no persistentes.

HTTP por defecto utiliza conexiones persistentes ya que este tipo tiene ventajas sobre las no persistentes. Una conexión no persistente se basa en una serie de peticiones y respuestas de cliente a servidor, por ejemplo, si se desean acceder a cierto número de recursos alojados en un cliente, y el cliente desea tomarlas una a una, debe realizar peticiones al servidor por cada uno de estos, eso genera un problema de tiempo, conocido como el tiempo RTT, o Round trip time (Tiempo de ida y vuelta) que genera que la comunicación entre estos sea un poco más lenta que las conexiones persistentes. En la figura 5.1 se presenta un ejemplo gráfico del problema generado en las conexiones no persistentes

Como se observa el RTT de las conexiones persistentes aumenta el tiempo de comunicación de HTTP no persistente.

Por su parte, HTTP persistente funciona mediante una conexión que se mantiene abierta por un tiempo, luego de una primera respuesta de parte del receptor, este tiempo puede ser determinado o ilimitado. En este caso el problema de los tiempos de RTT se eliminan ya que el cliente solicitara al servidor los recursos que desee y solo se deberá tener en cuenta los tiempos de envío.

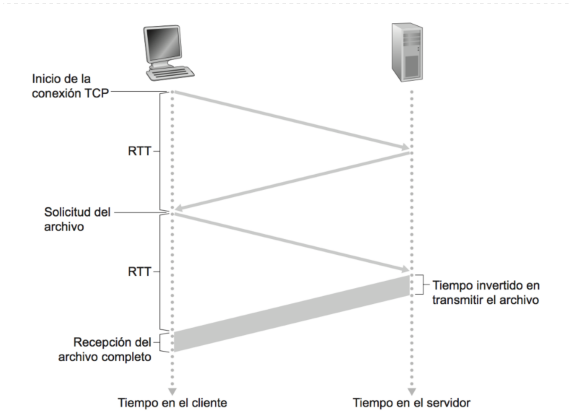

Figura 5.1: Conexión HTTP no persistente

5.1.2 Métodos de petición en HTTP

HTTP define unos métodos de petición [14] que se utilizan en los mensajes que se intercambian entre cliente y servidor. Cada uno de estos indica la acción que se desea llevar a cabo en cada petición o respuesta. A continuación se introducen los más utilizados.

- GET: Es el método mediante el cual se pide recurso especifico. Por ejemplo GET /volumen/hola.txt obtendrá el archivo hola de formato texto ubicado en volumen.
- HEAD: Solicita una respuesta a la que hace el método GET, la diferencia es que en esta petición no se obtiene el cuerpo de la petición, usualmente usado para recuperación de encabezados sin contenido.
- POST: Es el método por el cual dentro de una petición se incluyen datos determinados hacia un recurso determinado, por ejemplo al solicitarse una página, dependerá de lo que el usuario haya introducido en los campos del formulario de este método. Usado comúnmente para actualización de recursos existentes o nuevos.
- PUT: Este método permite al usuario o aplicación cargar o subir un recurso determinado en una ruta específica del directorio del servidor, por ejemplo PUT /volumen/archivo/hola.txt HTTP/1.1 subirá el archivo hola en formato texto plano a la ruta volumen.

- DELETE: Es el método mediante el cual un usuario o aplicación puede eliminar un recurso determinado de un servidor.
- TRACE: Método que solicita a un servidor que en el mensaje que se reciba luego de una petición, adjunte todos los mensajes que se añaden en cada paso por algún servidor intermedio.

5.2 Streaming sobre HTTP

El streaming adaptativo [15] es una técnica usada en streaming multimedia, para mejorar la distribución de contenidos que se realiza sobre HTTP (aunque también puede usarse en RTP y RTSP).

El modo de funcionamiento del streaming adaptativo se basa en detectar la capacidad de CPU, ancho de banda y en general los recursos disponibles por el sistema que tenga disponible el cliente para posteriormente con esta información se adapta la calidad del video que se va a transmitir. Para ello se hace indispensable el uso de codificadores de video que permitan modificar los bit rates de cada contenido para su adaptación. De esta forma se elimina uno de los problemas más importantes que se tienen con la transmisión en streaming actual; el almacenamiento en buffer (buffering), como resultado del streaming adaptativo se tiene un buffering muy pequeño adaptado a redes de bajas y altas capacidades.

HTTP Adaptative Streaming (HAS) es un método de video que se transmite sobre HTTP, en donde el archivo origen es codificado con multiples bit rates. Luego cada uno de esos videos es codificado en partes secundarias. Una vez se comienza el streaming, el cliente solicita la codificación con menor bit rate, si el cliente encuentra que su capacidad de descarga es mayor, solicitará una calidad superior y así sucesivamente. Cuando se encuentra una calidad de video que sobrepasa la capacidad de la red, se solicita un segmento con menor bit rate y será ese el seleccionado para su transmisión. Además de lo anterior mencionado, HAS tiene beneficios sobre los servidores de streaming tradicionales.

En primer lugar se tiene una tecnología construida sobre HTTP, contrario a tecnologías como RTP, los paquetes no tienen dificultades atravesando firewalls o dispositivos NAT. Segundo, desde que HAS es implementado en clientes, la adaptación lógica reside únicamente en él. Por ello se reducen las conexiones persistentes entre el servidor y el cliente. No mantiene tampoco los estados de información de sesión en cada cliente. Como tercer beneficio se tiene una infraestructura basada en HTTP con servidores y caches que pueden ser de igual forma adaptadas a condiciones óptimas de funcionamiento de este protocolo.

5.3 HTTP 2.0

Es la nueva versión del protocolo de transferencia de hipertexto [16][37], en este no se modifica la estructura de la versión 1, sino que por su parte se centra en las mejoras de conexión, compresión de las cabeceras, entre otras, generando mejoras en un 60 % en cuestiones de velocidad de carga de páginas con respecto a la primera versión.

A principios del 2012 se genera el proyecto SPDY, que generó un protocolo con el mismo nombre para implementar en HTTP, mejorando latencias y tiempos en general, siendo este protocolo la base para la creación de HTTP2 en el año 2012, para que posteriormente, en el año 2015, se realizaran sus primeras implementaciones en navegadores.

5.3.1 HTTP2 vs HTTP1.1

Para el desarrollo de la segunda versión de HTTP, no se requirieron cambios en el funcionamiento de las aplicaciones que se basan en este protocolo, sin embargo la principal diferencia entre las versiones es la velocidad a la que trabajan.

Dentro de las características principales de esta nueva versión de HTTP se encuentra la carga de contenidos mediante una única conexión con el servidor para atender múltiples solicitudes de un cliente de manera paralela, descargando contenidos al mismo tiempo y no mediante múltiples conexiones TCP como se hace en HTTP1. La multiplexación de HTTP2 soluciona el problema anterior debido a que, si bien las páginas actualmente poseen cientos de recursos, mediante este proceso se reduce el número de mensajes enviados al mismo tiempo y no se debe esperar a hacer petición tras petición y que el servidor las responda. Además de esto, la reducción de la latencia ocurre entre otras, por la eliminación de información duplicada en una conexión.

Otra importante característica de HTTP2 es que es un protocolo binario, lo que supone que la facilidad de interpretación es mucho mayor, es más compacto en el transporte y es menos propenso a errores respecto a protocolos como HTTP1 "basado en texto"por su gran cantidad de espacios, finales de línea, entre otros. Además estos protocolos tienen varias formas de interpretar los mensajes, por su parte la versión 2 de HTTP posee solo una forma.

HTTP2 realiza compresión de las cabeceras que normalmente se conocían con la primera versión. Comúnmente HTTP primera versión debía solicitar múltiples conexiones a través de múltiples peticiones que, en esta versión, tanto la cabecera de los mensajes se comprimen disminuyendo tiempos, como los mensajes con información redundante se descartan haciendo uso únicamente de lo importante de

los cambios que puedan generarse entre una solicitud a otra mediante el uso del algoritmo HPACK.

Otra innovación en HTTTP2 es el concepto *server push* que permite al servidor enviar información al cliente antes que este lo solicite, mediante predicciones realizadas en los recursos que se estén solicitando haciendo estimaciones de lo que puede requerirse. Para esto se envían en una misma respuesta de solicitud del cliente esta y otras respuestas, para que en el caso que se soliciten se tengan de manera inmediata. Sin embargo esta característica es configurable para los usuarios.

5.4 HTML

5.4.1 Introducción

Sus siglas en ingles son HyperText Markup Language [17], es decir, lenguaje de marcas de hipertexto. Es un lenguaje que permite realizar páginas web y se considera el lenguaje web más importante ya que gracias a su invención se dio paso al desarrollo y expansión del consorcio WWW (World Wide Web Consortium). Este además permite a una página web tener imágenes, juegos, video, texto entre otros. El estándar está a cargo del consorcio WWW puesto que la organización se encarga de estandarizar la mayoría de las tecnologías que están vinculadas con la web.

La función principal del lenguaje HTML es la descripción de los contenidos y también la definición de otros lenguajes web. Los *tags* o *etiquetas* que se forman en el texto tienen la capacidad de conectar entre si varios formatos y conceptos. El lenguaje tiene una estructura que se compone de etiquetas como ya se ha mencionado, las etiquetas se especifican por medio de paréntesis angulares o corchetes: <..>, todos los componentes se encargan de dar forma a la estructura general del lenguaje web.

El HTML [18] da paso a algunos códigos que se conocen como scripts. Estos brindan instrucciones a los navegadores que tienen como función procesar el lenguaje. Los scripts más utilizados son PHP y JavaScript.

El lenguaje fue desarrollado en el año 1985 por la CERN (Organización Europea de Investigación Nuclear), esto con el fin de hacer un sistema que almacenará información y que estuviera conectado entre sí y con otros sistemas por medio de hipervínculos. Antes de HTML, en 1945 se diseñó un dispositivo llamado *memex* el cual se consideró como un suplemento de la memoria que almacenará todo tipo de documento, aunque no llegó a ser materializado. Seguido de esto fue diseñado por Douglas Engelbart un entorno de trabajo por medio del computador que se denominó OnLine System que ofrecía una ayuda para que se hicieran más fáciles las

búsquedas en un mismo organismo. Ted Nelson en el año 1965 ideó una estructura la cual estaba conectada electrónicamente y más tarde esto dio paso a crear la World Wide Web en el año 1989, un sistema de hipertexto por el cual se compartía diferente información mediante el internet.

Actualmente existen editores Web que se encargan de permitir que los diseñadores creen páginas web por medio de herramientas gráficas sin conocer el código HTML.

5.4.2 HTML5

Por su parte, HTML 5 [19] [20] [31] es la última versión en el lenguaje que permite realizar páginas web por medio de hipervínculos. Esta versión tiene algunas características especiales y por eso solo los últimos navegadores pueden leer la información y mostrarla correctamente en la pantalla del ordenador, ya que los antiguos no pueden interpretar algunas etiquetas que la componen. Tiene la función de brindar contenidos multimedia para componer y dar un mejor sentido a la página web. El lenguaje le indica al navegador cuando un texto tiene un enlace y la dirección URL a la que se debe dirigir en caso que se haga click en el enlace. HTML5 comparte algunas características con sus antecesores, pero contiene algunos cambios que le otorgan gran importancia.

Algunas ventajas son: Una sintaxis mucho más clara, sus elementos semánticos son concretos, no depende tanto de ciertas librerías de java y por ultimo le da más velocidad al navegador a la hora de ejecutar y dibujar la página web.

En el año 2004 se fundó el grupo de trabajo WHAt con diferentes miembros tales como la Fundación Mozilla, Opera Software y Apple y así se dio inicio a HTML5. A medida que paso el tiempo muchas empresas se sumaron a hacer uso del lenguaje y así mismo aumento la cantidad de compatibilidad de HTML5.

Otra de las principales características que permite HTML5 es la inclusión de videos sin la necesidad de hacer llamado a complementos externos. Basta con utilizar la etiqueta <video>para que el navegador entienda que debe reproducir un video y <src>para seleccionar el archivo que se desea reproducir, ya sea llamado localmente mediante el uso de una url de una pagina externa o en ella misma. HTML5 es compatible con videos WebM, MP4 y Ogg, codificados por ejemplo en VP8 o VP9 para el caso de WebM y H.264 HEVC para el caso de MP4 o cualquiera de sus antecesores.

5.5 Formatos de video compatibles con HTML5

5.5.1 MP4

Es un formato contenedor [22], es decir puede contener una serie de codecs y estándares internacionales de audio, video y datos creados para la web. Constituido por algoritmos que se encargan de codificar las imágenes, los datos, los audios y videos, lo que permite una optimización a la hora de tener codificación y distribución en las redes y la calidad de almacenamiento.

Este formato multimedia sirve para grabar todo tipo de datos de video y audio digital, imágenes, etc. Se le da el nombre de MP4 gracias a la extensión de los archivos *.mp4 y la popularidad de este formato fue brindada históricamente gracias al reproductor iTunes.

5.5.2 Webm

Es un formato multimedia de software libre desarrollado por Google [23], se creó con el principal propósito de ser un formato compatible en su totalidad con HTM5 por lo que, al ser fabricado por Google, utilizó en su primera versión a *VP8* como codificador de video y lo unió con el codificador de audio *Vorbis* desarrollado por la fundación Xiph.org. Para juntarlos se hace uso del contenedor libre multimedia Matroska, mediante el cual se dio origen a Webm.

Este formato se creó en 2010 y cuenta con el apoyo de empresas reconocidas como Opera, Google, Mozilla, entro otros. Es actualmente el formato multimedia que se utiliza en plataformas como YouTube.

5.6 Bibliotecas y utilidades para codificación de video

5.6.1 x264

X264 [24] es una biblioteca que se encarga de la codificación de flujos de video H.264/MPEG-4 AVC- Está disponible como un codificador de video *video for Windows* y también como una interfaz de línea de comandos para programas como Ffmpeg. Se han hecho algunas interfaces gráficas como AutoAc y MeGUI entre otras.

Las versiones base del códec actualmente no son muy compatibles con algunos estándares como lo son: HD DVD/Bluray, esto es consecuencia de la carencia de algunos metadatos que son necesarios. Sin embargo, se han intentado algunas nue-

vas versiones para lograr la compatibilidad dando paso a la creación de contenidos adecuados en altas resoluciones.

5.6.2 x265

X265 [25] Es una biblioteca y una aplicación, contiene software libre y código abierto y se usa para la codificación de video. Se inició el desarrollo en marzo del año 2013 y el 2 de mayo del año 2014 se liberó la versión 1, las características y el rendimiento estuvieron sujetas al constante desarrollo. Varias empresas que manejaban el desarrollo y recibían licencias de uso comercial le dieron la financiación a x265. El código fuente del programa se encuentra escrito en C++.

A diferencia de su antecesor x264, esta aplicación se desarrolla principalmente para la codificación de videos de altas resoluciones haciendo uso de H.265 como estándar para codificar y tratar una señal de video, permitiendo un mejor resultado para el crecimiento exponencial de resoluciones de pantallas y videos.

5.6.3 FFmpeg

FFmpeg [26] es un conjunto de software libre que tiene la capacidad de hacer diversas funciones como convertir, grabar, codificar y hacer streaming de audio y video. Contiene una biblioteca de codificación y decodificación llamada libavcodec, y puede hacer uso de diversas más, tanto de video como de audio. Aunque FFmpeg puede ser compilado por la mayoría de los sistemas operativos, se desarrolló para Linux originalmente.

Este programa tiene una gran ventaja ya que puede ser usado tanto por usuarios con altos conocimientos en temas de tratamiento de señales de audio y video o por usuarios principiantes. El programa es capaz de elegir el tipo de códec solamente con escribir la extensión, por ejemplo al usar FFmpeg para codificación de un archivo .mp4, el programa hará una llamada a la librería x264 para completar este proceso.

5.6.4 MP4box

Mp4box [27] es una herramienta / conversor que funciona bajo línea de comandos, permite manipulación de archivos multimedia y también exportar e importar pistas de video y audio. Su función es realizar el cambio de formato de archivos, para este proyecto, por ejemplo, se utilizó para transformar el formato resultante luego de la codificación HEVC y H.264, cuyos archivos resultantes poseían extensión .264 y .265 respectivamente, haciendo el paso a mp4 para poder visualizarlo en reproductores multimedia. Aunque solo contiene una línea de comando existen

interfaces gráfica como YAMB que permite usar esta herramienta de un modo más sencillo.

5.7 Navegadores actualmente compatibles con HEVC

5.7.1 Google Chromium

Es un navegador web [28] [29] de código abierto y el objetivo que tienen para Chromium es que sea un administrador de pestañas tipo shell para la web con propósitos de desarrollo más allá de los navegadores tradicionales, dando paso a que no sea un navegador web común. El navegador tiene compatibilidad con HTML5 y CSS3, también cuenta con la facilidad de personalización lo que permite adaptar los gustos de cada usuario.

Otras características importantes del navegador Chromium son: Cuenta con extensiones como JavaScript, HTML y CSS para expandir las funcionalidades del navegador, tiene una rapidez alta para dar resolución de direcciones IP, cuenta con una arquitectura multiproceso, es decir, maneja cada complemento o consulta que se ejecute en un proceso separado lo que le brinda al usuario estabilidad y seguridad, la navegación en modo incógnito al igual que otros no mantiene ni los cookies ni el historial de navegación. Por ultimo tiene preferencia en el momento de sincronizar, mantiene los marcadores, ajustes y extensiones instalados en otros navegadores.

Aunque los navegadores comparten muchas características y parte de código, existen ciertas diferencias en sus licencias. Un dato interesante sobre el nombre del navegador es que fue tomado del elemento cromo que es el metal resultantes del cromo.

Desde el año 2008 hasta la actualidad se ha ido modificado y mejorando Chromium. Es el proyecto de código abierto que se refleja en liberaciones en Chrome que es el proyecto final, por esto su historia está relacionada. Tanto Chrome como Crhomium pertenecen a Google, su diferencia radica en que chromium es de código abierto.

En temas relacionados a video y audio, este navegador soporta etiquetas de audio y video de tipo WebM, Theora, H.264, Vorbis, entre otras. En este trabajo se hace uso de la distribución distribuida por Github con soporte para videos en HEVC. Esta versión se puede encontrar en *https://github.com/henrypp/chromium/releases*

5.7.2 Edge Browser

Este navegador web [30] [31] es desarrollado por la empresa Microsoft, está diseñado para ser un navegador ligero de código abierto, basado en los estándares web. Contiene herramientas de modo de lectura y anotación. Edge se encuentra incluido en Windows 10 lo que da paso a que Internet Explorer sea reemplazado. Inicialmente se le denominaba como proyecto con el nombre de *Spartan* y tenía su respectivo nombre en código.

En el año 2014 a mediado del mes de diciembre Mary Jo Foley, escritora en tecnología, dio a conocer que Microsoft desarrollaba un navegador web llamado *Spartan*. Seguido de esto a inicios del año 2015 The Verge obtuvo detalles relevantes sobre el nuevo proyecto y días después Microsoft lazó oficialmente *Spartan* el día 25 de enero de 2015. *Spartan* estuvo disponible en las ejecuciones de Windows 10 el día 30 de marzo. El logo y la marca del navegador son similares a los de Internet Explorer, esto con el fin de permitirle al usuario relacionar a Edge como un navegador web. El día 29 de abril de 2015 se dio su nombre final Microsoft Edge.

El navegador web tiene unas características especiales que lo destacan y brindan un buen servicio a sus usuarios, Contiene integrado un lector PDF y también Adobe Flash Player. No es compatible con algunas tecnologías que se encuentran en la actualidad, así que se remplazan con un sistema de extensiones. También es capaz de integrarse con plataformas en línea de Microsoft. Una utilidad muy grande es que los usuarios pueden hacer anotaciones o comentarios en las páginas web y se pueden compartir o almacenar. El asistente Cortana brinda un control de voz y la función de búsqueda está relacionada con la búsqueda en la barra de direcciones. Actualmente Edge se encuentra predeterminado para Windows 10 tanto en computadores como en móviles.

Para este proyecto se hace énfasis en este navegador ya que Microsoft declaró que sus últimas versiones iban a soportar HEVC, lo cual, al desarrollar este trabajo se observa como cierto y se convierte en una importante herramienta para la reproducción de contenidos de última generación de video streaming.

5.8 Dynamic Adaptative Streaming over HTTP (DASH)

Conocido como MPEG-DASH [32] [33], es una norma ISO para streaming de velocidad de transmisión adaptativa, esto permite que se tenga una muy buena experiencia para el usuario en el streaming de contenidos multimedia por medio del internet desde servidores web HTTP. Esta técnica funciona rompiendo el contenido de las secuencias en segmentos de archivos que se basan en HTTP, los segmentos a su vez se componen por algunos intervalos de tiempo de reproducción de contenido. Estos segmentos pueden ser organizados como SegmentBase, SegmentTimeline,

SegmentList o SegmentTemplate, esto depende básicamente del uso que se va a dar.

Esta técnica es la principal solución de streaming basado en HTTP. El protocolo que usa MPEG-DASH es TCP, es por esto que no se debe confundir con un protocolo de transporte. La infraestructura que lo conforma es de un servidor web HTTP que se utiliza generalmente para entregar contenido sobre todo toda la WWW. Se les permite a algunos objetos que se encuentren conectados a internet como lo son los celulares, tablets, televisores, computadores entre otros, que consuman contenido multimedia. La estandarización de una solución de streaming adaptativo se destina a brindar confianza y asegurando que dicha solución se puede adoptar a una aplicación universal.

MPEG-DASH se desarrolló principalmente por MPEG. A mediados del año 2010 se empezó a trabajar con DASH en ese momento se convirtió en un proyecto, pero a mediados de abril del año 2012 se volvió un estándar publicado como ISO/IEEC 23009-1:2012.

El funcionamiento de MPEG-DASH se basa en una secuencia de video original que se copia en varios segmentos con diferentes bit rates, es decir el mismo video con multiples calidades visuales. Estos segmentos a su vez se dividen en el tiempo en representaciones que permiten tener puntos de referencia dentro de la trama para realizar cambios de definición cuando se reproduzca. Para ello se crea un archivo de texto "manifest .mpd"que como resultado permitirá que el video cambie automáticamente de definición a una menor si la red se encuentra saturada y subirá la calidad progresivamente si se va liberando el ancho de banda, evitando así que hayan buffers que detengan la reproducción continua. Normalmente los videos que utilizan DASH o cualquier método de streaming adaptativo comienzan su carga con resoluciones bajas, ya que se aprovecha el inicio del video para cargar las siguientes en la máxima resolución posible y presentar el contenido con la mejor calidad hacia el usuario final.

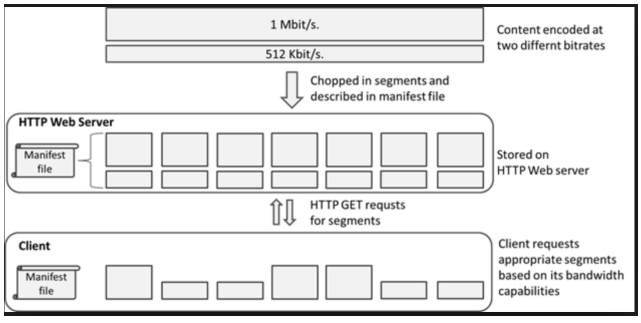

Figura 5.2: Esquema de funcionamiento de DASH.

Para la utilización de DASH en este proyecto, se hace uso del reproductor de video MPEG-DASH llamado Dash.js, un programa de código abierto escrito en Java Script y desarrollado por la organización GitHub que puede ser usado libremente en aplicaciones que requieran de reproducción de video. Es posible incorporar este reproductor a la mayoría de navegadores multimedia actuales, Chrome, Edge, Firefox, Safari, entre otros.

5.9 Evaluación objetiva de la calidad de video usando PSNR

Relación señal a ruido de pico [35] es un término que se utiliza para definir la relación entre el valor máximo que puede tomar la energía de una señal y el ruido que afecta dicha señal. En el caso de la compresión de video, el ruido es la distorsión que introduce el codificador, o sea, las pérdidas obtenidas al comprimir. Esta relación se expresa en unidades logarítmicas, con unidades de decibeles, esto se debe a que muchas señales tienen un rango dinámico alto. Los valores que toma el parámetro están en un rango de 30 y 50 dB, si la codificación es buena será mayor el valor de PSNR.

El PSNR tiene muchos usos, pero el uso más frecuente es como medida cuantitativa de la calidad a la hora de reconstruir el ámbito de la compresión de imágenes, para poder definir es necesario la formulación del error cuadrático medio.

Capítulo 6

Evaluación de desempeño de HEVC

6.1 Introducción

En este capitulo se evaluará el desempeño que posee el codificador HEVC en comparación con H.264 mediante diversas pruebas de calidad. Estas pruebas mostrarán que la eficiencia de codificación de HEVC cumple la teoría acerca de una mejora de casi el doble con respecto a su antecesor.

Para las siguientes pruebas se utilizó un ordenador MacBook Pro con sistema operativo macOS Sierra Version 10.12.3, procesador Intel Core i5 con dos núcleos, cada uno de de 2.6GHz, Memoria RAM de 8 GB MHZ DDR3.

Para la realización de las pruebas se usará el software de codificación ffmpeg que se describe en el capitulo anterior. A su vez se hace llamado a x264 y x265 para la codificación eficaz de las secuencias en algunos de los casos. Estas dos herramientas al ser desarrolladas por la misma organización, poseen características y parámetros de evaluación y configuración similares para ambos estándares.

A continuación se mostrarán las pruebas y los resultados obtenidos por cada una de estas. Además se detallarán las características, tamaños, resoluciones, frames y demás parámetros de importancia de las secuencias de video utilizadas para cada test.

6.2 Eficiencia de codificación absoluta y relativa

6.2.1 Codificación absoluta

La eficiencia de codificación absoluta hace referencia a la compresión neta que tiene cada codificador a evaluar, una compresión con mayor tamaño corresponde a una ganancia de codificación menor; en otras palabras, cuanto más grande sea el archivo final luego de su proceso de codificación y compresión, se dirá que tiene menor ganancia de codificación que uno que tenga menor tamaño. Por su parte, la eficiencia de codificación relativa hace referencia a la diferencia o cantidad de bit rate ahorrado que se obtiene sobre un mismo video codificado en HEVC y H.264, esta diferencia se obtendrá mediante una sencilla fórmula matemática que dará como resultado el ahorro obtenido.

Para estas pruebas se tomaron en cuenta 4 valores de CRF (Constant Ratio Factor) que es la calidad por defecto que se establece para la codificación. Normalmente este valor oscila entre 0-51, teniendo como recomendación de los desarrolladores de los compresores que el valor óptimo está en el rango de entre 18 y 28. Para la mayoría de pruebas de este capitulo, el valor CRF=23 será el más adecuado debido a que es el intermedio entre 18 y 28 y el más recomendado. Las secuencias utilizadas para las pruebas de este capitulo se pueden encontrar compiladas en la pagina de secuencias de video sin compresión en formato .y4m que la fundación sin animo de lucro Xiph pone a disposición. Se encuentra en la sección videos de su pagina web: $https://media.xiph.org/video/derf/$, estos videos poseen todas las normas de copyright y derechos de autor y redistribución libre para fines netamente experimentales.

Para las pruebas de codificación absoluta y relativa se utilizaron las secuencias de la Tabla 6.1.

Secuencia	Resolución (px)	FPS	Duración	Frames	Tamaño
Four People	1280 x 720	50	10s	600	830.8 MB
Square and time Lapse	4096x2160	60	10s	600	15.93 GB
Wind and Nature	4096x2160	60	20s	1199	31.82 GB

Tabla 6.1: Secuencias Prueba 1.

El propósito es verificar la eficiencia de compresión de HEVC y H.264 con videos de diferentes resoluciones y complejidad de imagen ya que se tiene que *Four People* es una secuencia sin mucho detalle, con poco movimiento y baja resolución en comparación con las otras dos, mientras que *Square and time Lapse* posee considerablemente mayor cantidad de detalle, más movimiento y mayor resolución, por

su parte *Wind and nature* aunque posee la misma resolución que *Square and time Lapse* posee el doble de frames pero el detalle y el movimiento es considerablemente menor.

Se muestran los resultados obtenidos en la tabla 6.2 en donde se consolidan los tamaños de archivo final obtenidos al realizar la codificación de los videos anteriores con los diferentes valores de CRF para ambos estándares.

HEVC	CRF=7	CRF=15	CRF=23	CRF=31
Four People	25.8 MB	4.6 MB	1.2 MB	450 KB
Square and T	309 MB	64.2 MB	17.4 MB	6 MB
Wind and N	445 MB	63.5 MB	16.5 MB	6 MB
H.264	CRF=7	CRF=15	CRF=23	CRF=31
Four People	59.6 MB	9.1 MB	1.91 MB	686 KB
Square and T	633 MB	123.5 MB	33.5 MB	12.5 MB
Wind and N	949 MB	130 MB	33.3 MB	11.9 MB

Tabla 6.2: Resultados Obtenidos prueba 1.

La figura 6.1 muestra la eficiencia de codificación absoluta de HEVC y H.264 para las secuencias *Four People* y para *Square and Time Lapse* basados en la información obtenida en la tabla 6.2, si bien se puede observar que las dos tienen un comportamiento similar gráficamente, no son comparables debido a que la primera tiene menor resolución, menor tamaño del archivo original y menor cantidad de movimiento. La figura se divide en las siguientes gráficas comparativas.

(a). Codificación absoluta Four People.
(b). Codificación absoluta Square and Time Lapse.
(c). Ampliación valores altos CRF Four People.
(d). Ampliación valores altos CRF Square and Time Lapse.
(e). Comparación 4 codificaciones CRF medios y altos.

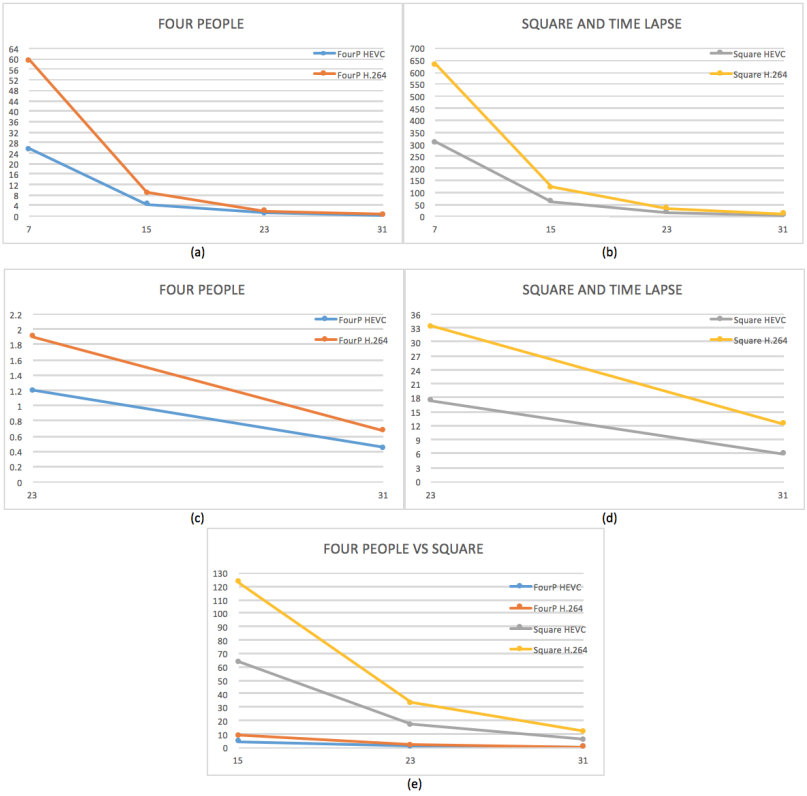

Figura 6.1: Codificación absoluta secuencias Four People y Square and Time Lapse

Para realizar una prueba con mejores resultados, se utilizó una tercera secuencia *Wind and nature* para comparar con *Square and Time Lapse* esta posee características de misma resolución, posee el doble de tamaño original (15.93 GB vs 31.82 GB), y el doble de duración (10s vs 20s), pero tiene la diferencia de tener considerablemente menor cantidad de movimiento. La figura 6.2 se divide en las siguientes gráficas comparativas.

(a). Codificación absoluta Square and Time Lapse.
(b). Codificación absoluta Wind and Nature.
(c). Ampliación valores altos CRF Square and Time Lapse.
(d). Ampliación valores altos CRF Wind and Nature.
(e). Comparación 4 codificaciones.
(f). Ampliación de comparación 4 codificaciones valores medios y altos.

Se observa que para ambos casos que la ganancia de codificación absoluta es mayor para HEVC comparada con H.264 para cualquiera de los videos. En la figura 6.2 se observa el resultado obtenido para ambas secuencias

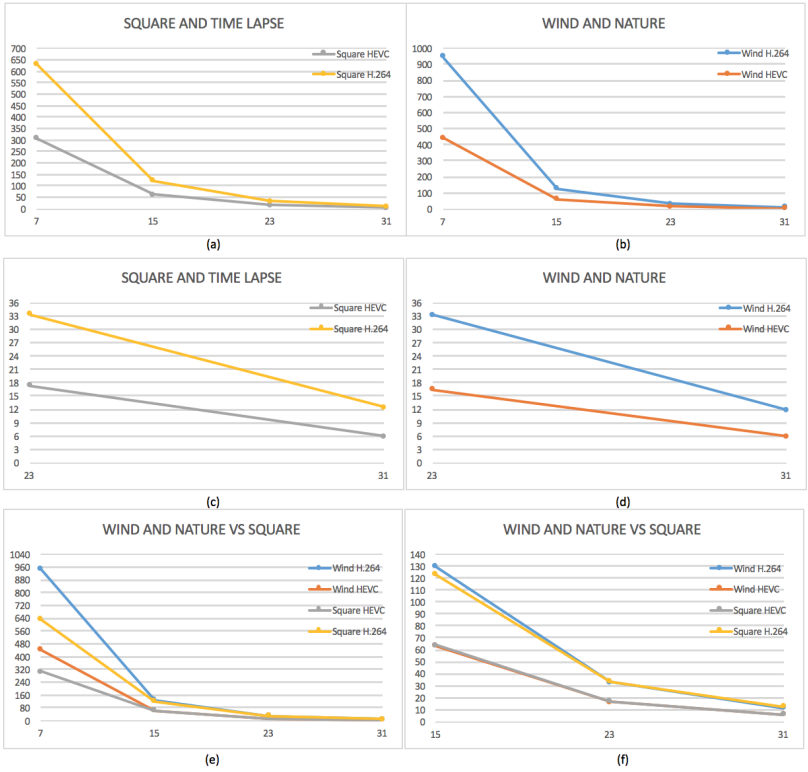

Figura 6.2: Codificación absoluta Square and Time Lapse y Wind and Nature

Partiendo de los resultados obtenidos y mostrados en las figuras 6.1 y 6.2 se concluye:

- La complejidad de *Square and Time Lapse* es mucho mayor debido a la cantidad de detalles y movimiento que tiene, esto se traduce en que el resultado de la codificación arroje un tamaño de archivo comprimido similar a los de la secuencia *Wind and Nature*, inclusive teniendo en cuenta que esta última posee el doble de tamaño y duración, en otras palabras, la ganancia de codificación absoluta es directamente proporcional a la complejidad del video, cuanto más alta sea la complejidad, mayor será el tamaño del archivo.

- La eficiencia de H.264 y H.265 se reduce para valores bajos de CRF en videos con alta calidad y resolución independientemente de la complejidad de imagen que tengan.

- Se observa la superioridad de HEVC de codificación absoluta para todos las pruebas estudiadas ya que reduce el tamaño del archivo resultante tanto para videos de baja resolución como para aquellos con calidades y complejidades altas.

Codificación Relativa

En este apartado se estudia la ganancia codificación relativa entre las 3 secuencias del apartado anterior, es decir el porcentaje de bits ahorrado entre el video codificado en H.264 y en HEVC. Para esto se utilizará la sencilla formula matemática con los datos de la tabla 6.2: *((H.264 - HEVC) / H.264) * 100*, como resultado obtendremos el porcentaje medio de bitrate ahorrado con la codificación HEVC vs H.264

VIDEO	CRF=7	CRF=15	CRF=23	CRF=31
Four People	56.63 %	49.45 %	37.17 %	33.82 %
Square and T	51.18 %	48.01 %	48.05 %	52 %
Wind and N	53.14 %	51.15 %	50.45 %	49.57 %

Tabla 6.3: Resultados Codificación relativa.

Se muestra en la figura 6.3 el resultado gráfico de la ganancia de codificación relativa obtenida para las 3 secuencias de video.

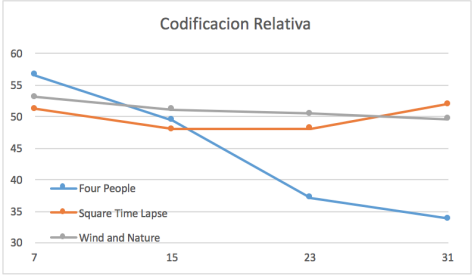

Figura 6.3: Codificación relativa

Partiendo de los resultados obtenidos en este apartado se puede concluir:

- Para videos de baja complejidad y resolución el comportamiento del codificador HEVC no es tan relevante como para los videos de alta calidad. En el caso de Four People se observa como la ganancia de codificación relativa decae conforme aumenta el CRF, es decir que cuanto menor es la calidad establecida (CRF altos) HEVC no codifica considerablemente más que H.264. Esto es debido a que los detalles y resolución del video no son suficientes

para que los métodos de predicción, filtros, tamaños de bloques actúen de manera superior a lo que hace H.264. Sin embargo ase obtiene una mejora de entre 33 % y 56 %

- Para el caso de los videos con mayores resoluciones se observa un comportamiento constante en la codificación relativa que tiene HEVC vs H.264. Independiente de la calidad definida se tienen valores de entre 48 % y 53 %, un rango muy pequeño en comparación con los videos con baja resolución y detalle. Esto demuestra que se cumple uno de los objetivos de HEVC para alto desempeño en videos de alta calidad.

6.3 Tiempo de codificación

En esta sección se realiza una comparativa del coste computacional en términos de tiempo de codificación que requieren diferentes secuencias de video con diferentes características. El ahorro de tiempo representa el porcentaje de ahorro en tiempo de los codificadores HEVC y H.264, de esta forma un valor menor de tiempo se traducirá como mayor ahorro en de tiempo.

Para las pruebas de tiempo de codificación se utilizaron las secuencias descritas en la tabla 6.4:

Secuencia	Resolución (px)	FPS	Duración	Frames	Tamaño
Four People	1280 x 720	50	10s	600	830.8 MB
Park Run	1280 x 720	50	10s	504	696.7
Dancers	4096x2160	60	19s	600	31.82 GB
Big Buck Bunny	1290 x 1080	40	9min 56s	14315	44.56 GB

Tabla 6.4: Secuencias Prueba 2.

Se presenta a continuación la tabla 6.5, con los resultados obtenidos para la codificación con HEVC y H.264 de las secuencias anteriores con valores de calidad CRF 3, 13, 23, 33, 43 y 51.

CRF	3	13	23	33
Four HEVC (seg)	71.67	22.06	15.23	13.61
Four H.264 (seg)	34.9	19.84	15.77	14.25
Park R HEVC (seg)	98	72	31	19.12
Park R H.264 (seg)	37	30.15	19.8	17.49
Dancers HEVC (min)	23.15	10.17	6.51	6.51
Dancers H.264	18.02	10.12	6.54	6.56
Bunny HEVC (min)	53.12	33.53	18.12	15.3
Bunny H.264 (min)	64.1	60.26	53.32	52.1

Tabla 6.5: Resultados Comparación tiempo de codificación.

Se presenta en la figura 6.4 el resultado gráfico de los datos obtenidos para Four People, en la figura 6.5 el resultado de tiempos de Park Run, la figura 6.6 muestra los tiempos de Dancers y los resultados de Big Buck Bunny se presenta en la figura 6.7. El eje Y en las figuras 6.4 y 6.5 se da en segundos, mientras que en las figuras 6.6 y 6.7, por su alta complejidad / duración el eje Y se presentan en minutos.

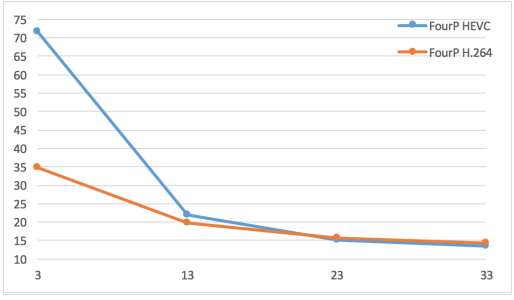

Figura 6.4: Gráfico de tiempos Four People HEVC vs H.264

Partiendo de los resultados obtenidos en este apartado se tienen las siguientes conclusiones:

- De las dos primeras secuencias con características similares de resolución se puede concluir que la complejidad del video es un factor que tiene repercusiones también en términos de complejidad computacional reflejada en tiempo que tarda en realizarse la compresión. En el video con poco movimiento se observan valores de tiempo similar entre HEVC y H.264, mientras que la diferencia de tiempo obtenida en el video con considerable mayor cantidad de movimiento es más notable para todos los valores de CRF. En este caso además se observa que H.264 tiene superioridad frente a HEVC en valores bajos de CRF y tiende a igualarse para valores de CRF medios y altos.

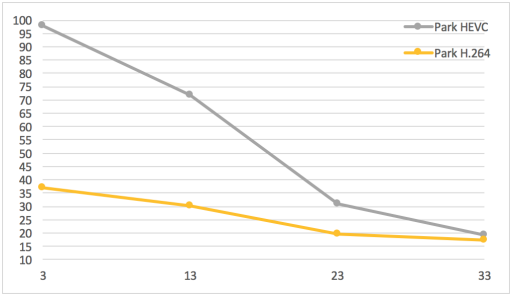

Figura 6.5: Gráfico de tiempos Park Run HEVC vs H.264

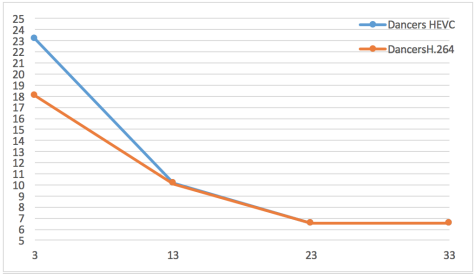

Figura 6.6: Gráfico de tiempos Dancers HEVC vs H.264

- Por su parte, el video *Dancers* que posee mayor resolución que los dos anteriores, presenta comportamientos de casi igualdad de tiempo para los dos codificadores para todos los valores de CRF. Este es un resultado un tanto interesante, debido a que este resultado puede obtenerse debido al tipo de video que es *Dancers*, ya que tiene la particularidad de tener movimientos muy suaves y una gran cantidad de color negro uniforme en el fondo, por esto se observa que los tiempos son bastante similares para valores bajos de CRF y se vuelve casi imperceptible para valores medios. En términos de compresión se tuvo que HEVC fue superior a H.264 con 26.4MB y para HEVC, ambos con CRF=23.

- Se observa una notable superioridad de HEVC para el video *Big Buck Bunny*, con alta complejidad y larga duración. Se concluye entonces que HEVC logra en la práctica lo que se plantea en la teoría acerca que sus métodos de codificación están pensados para videos de alta resolución, en un video como este con larga duración es cuando el codificador HEVC puede utilizar en su totalidad los métodos de predicción mejorados para los frames que lo componen, adicional a esto se observó una notable mejoría en términos de compresión, por ejemplo 236.4 MB en H.264 y 133.9 para HEVC, ambos con CRF=23. Se observa una diferencia en la forma de la gráfica resultante, para

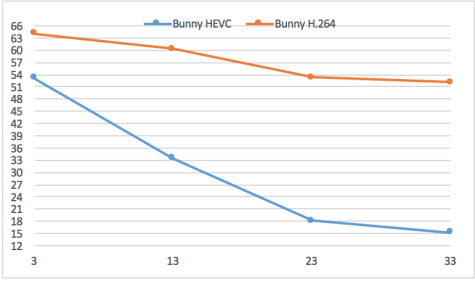

Figura 6.7: Gráfico de tiempos Bunny HEVC vs H.264

valores bajos de CRF la diferencia no es tan grande pero conforme aumenta el valor, HEVC muestra bastante superioridad.

- Se observó en términos generales que HEVC logra tiempos de compresión similares para cualquier video, independiente de su complejidad y calidad respecto a su antecesor H.264. Esto es un factor de suma importancia ya que si bien en algunas ocasiones H.264 fue superior, esta mejora no es demasiado alta y se diluyó con valores altos de CRF. Se observó en las figuras 6.4 y 6.5 una superioridad de entre el 50 % y el 30 % y exclusivamente en valores bajos de CRF. Se añade también que si bien en términos de tiempos la diferencia no es muy diferente, en términos de codificación relativa la superioridad de HEVC se mantuvo en todos los casos.

6.4 Evaluación de Eficiencia

En esta sección se realizará la evaluación global de la eficiencia de los compresores para videos en HD con diferentes características mediante el análisis de la relación señal a ruido (PSNR) y el bit rate obtenido que resultan luego de su proceso de codificación. Este parámetro se describe en el capitulo anterior y se tiene que, cuanto mas alto es el resultado de PSNR, mejor es la codificación realizada.

Se presentan a continuación las secuencias utilizadas y sus características. Se seleccionan 6 secuencias, las 3 primeras en alta definición y con diversas complejidades debido a sus diferencias de movimiento, y de igual forma se seleccionan 3 secuencias en 4K con complejidades diferentes.

Secuencia	Resolución (px)	FPS	Duración	Frames	Tamaño
Mobcal	1280 x 720	50	10s	504	696.7 MB
Park Run	1280 x 720	50	10s	504	696.7MB
Shields Ter	1280 x 720	50	10s	504	696.7
Ritual Dance	4096x2160	60	10s	600	15.93 GB
Square and Time Lapse	4096x2160	60	10s	600	15.93 GB
Tunnel Flag	4096x2160	60	10s	600	15.93 GB

Tabla 6.6: Secuencias Prueba 3.

Se relaciona en la tabla 6.7 los resultados obtenidos de PSNR promedio para los dos codificadores con un CRF = 23.

Video	PSNR	Bit Rate (kbps)	Video	PSNR	Bit Rate (kbps)
Mobcal H.264	37.647	4745	Mobcal HEVC	37.821	2171
Parkrun H.264	34.355	22257	ParkrunHEVC	34.899	14793
Shield H.264	37.038	5584	Shield HEVC	36.854	2836
Ritual H.264	42.334	27286	Ritual HEVC	41.994	13954
Square H.264	41.465	26344	Square HEVC	41.009	13881
Tunnel H.264	41.398	33602	Tunnel HEVC	41.605	17403

Tabla 6.7: Resultados Prueba 3.

Se relaciona en la Figura 6.8 el resultado gráfico de la información consolidada en la tabla 6.7.

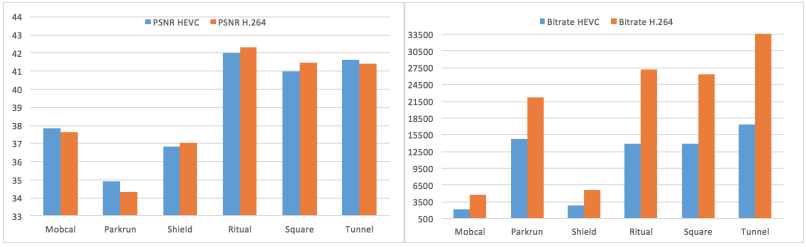

Figura 6.8: Resultados PSNR y Bit Rate de videos diferentes resoluciones

De esta sección se concluye:

- Se observa en las gráficas que el PSNR es muy similar para HEVC en comparación con H.264 en la totalidad de las secuencias, sin importar su resolución, tamaño, calidad, complejidad, etc. se tiene una calidad visual final muy similar.

- En aspectos de bit rate se observa que HEVC es superior a H.264 y esta superioridad crece conforme la calidad y complejidad del video es mayor. Por ejemplo de los videos en HD, *Park Run* posee elevado movimiento y cantidad de detalle respecto a los otros dos y se nota gráficamente una mayor diferencia de bitrate entre HEVC y H.264. En los videos 4K se observa que esta diferencia es más notoria que en los videos en HD. Se comprueba la superioridad de HEVC respecto a una menor tasa de bit utilizados para mostrar la misma calidad visual con videos cuya complejidad y resolución sea mayor.

6.5 Prueba de relación señal a ruido vs Bit rate

En este apartado se realiza la evaluación de ambos codificadores comparando las curvas que determinan la relación entre el nivel de compresión, expresado como bit rate del video una vez codificado, y el nivel de calidad, medido con el valor PSNR.

En esta prueba se evalúan dos secuencias:

Secuencia	Resolución (px)	FPS	Duración	Frames	Tamaño
Four People	1280 x 720	50	10s	600	830.8 MB
Ritual Dance	4096x2160	60	10s	600	15.93 GB

Tabla 6.8: Secuencias Prueba 4.

Estas secuencias se codificarán con un bit rate variable y se obtendrá entonces una comparación en términos de calidad de imagen; que se dará en el eje Y con el valor del PSNR, contra la que ofrezca menor bit rate. Aquella gráfica que se encuentre por encima de la otra será la que ofrece una mejor eficiencia de Bit Rate vs PSNR.

Se utilizarán valores de CRF comunes para este tipo de pruebas: 22, 27, 32 y 37.

El resultado gráfico se puede observar en las figuras 6.9 y 6.10, construidas a partir de los resultados obtenidos en las tablas 6.9 y 6.10.

Four People H.264	PSNR	Bit Rate
22	42.567	1772.97
27	40.422	873.51
32	37.905	488.97
37	35.136	286.19
Four People HEVC	PSNR	Bit Rate
22	41.855	1070.92
27	39.605	565.56
32	37.053	318.98
37	34.462	183.42

Tabla 6.9: Resultados PSNR Bit Rate Prueba 4 Secuencia 1.

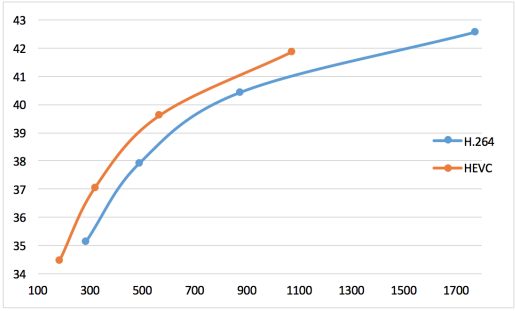

Figura 6.9: Resultados PSNR vs Bit Rate Four People

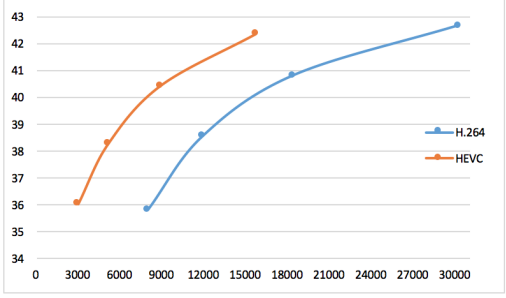

Figura 6.10: Resultados PSNR vs Bit Rate Ritual Dance

Ritual Dance H.264	PSNR	Bit Rate
22	42.668	30129.01
27	40.816	18324.46
32	38.549	11916.39
37	35.842	7947
Ritual Dance HEVC	PSNR	Bit Rate
22	42.347	15663.1
27	40.43	8828.98
32	38.289	5085.24
37	36.042	2914.57

Tabla 6.10: Resultados PSNR Bit Rate Prueba 4 Secuencia 2.

Partiendo de los resultados obtenidos en este apartado se puede concluir:

- Esta es la forma más adecuada que se usa para evaluar a fondo la eficiencia que tienen dos o mas codificadores entre sí, ya que da un valor certero de la diferencia que tiene la calidad de imagen respecto al nivel de compresión obtenido. En las pruebas anteriores se tenía en cuenta solo uno de estos valores, pero en este caso sí se evalúan en conjunto. Las gráficas comparativas 6.9 y 6.10 muestran que HEVC está por encima de H.264 para videos con bajas y altas resoluciones, ya que si bien los valores de PSNR son similares, el bitrate sí que es diferente, dando como resultado la superioridad de HEVC frente a H.264.

Capítulo 7

Desarrollo de un entorno para streaming de video HEVC usando DASH

7.1 Introducción

Además de las pruebas realizadas en el capítulo anterior y demostrando la superioridad de HEVC, este proyecto pretende realizar un acercamiento a cómo debería realizarse el desarrollo de un escenario real de un sistema de distribución de contenidos mediante un servidor en internet usando HTTP1.1 que permita la reproducción de videos codificados con HEVC, observando de manera real el comportamiento de este codificador en un entorno de multimedia actual. Esto se realizó mediante la creación de un sitio web en internet que aloje los contenidos, junto con los resultados de las pruebas comparativas de la sección anterior y de rendimiento general de HEVC, para que esta sea un elemento de análisis del potencial de HEVC. Por otra parte este sitio web dará un veredicto de las diferentes formas para lograr la reproducción de este contenido con los navegadores actuales y determinar el escenario ideal para llevar a cabo una plataforma de distribución de contenidos multimedia basado en videos codificados en HEVC.

7.2 Streaming en HTML5 con video HEVC en HTTP1

En este proyecto se ha desarrollado una pagina web en internet bajo el dominio www.hevcstreamingcomparison.com escrita en su totalidad sobre el lenguaje de programación HTML5 que, como se describió anteriormente tiene como novedad la facilidad de incorporar videos mediante una sencilla linea de comando usando los comandos *<video>* y *<source>,* sin necesidad de la instalación de los acostumbrados complementos o plugins de reproducción para los navegadores dependiendo de las características de cada uno. Es entonces donde HTML5 establece una gran ventaja para la reproducción tanto en ordenadores como en teléfonos y otros dispositivos móviles que usan navegadores compatibles con HTML5 y permiten la reproducción de videos. Si bien como se estableció en capítulos anteriores, la incorporación de HTML5 se desarrolló en un principio para ser una herramienta para videos de Google, actualmente brinda soporte adicional para la reproducción de videos de diversos formatos y codificaciones. En este sitio se mostrará entre otras cosas, la puesta en marcha de esta incorporación de videos en HTML5 sobre el servidor HTTP1.1 con secuencias codificadas en diversos formatos y de igual forma se mostrará cuales navegadores y bajo que sistemas operativos reproducen cada uno de estos, dependiendo de sus versiones y características.

7.3 Descripción del sitio web desarrollado

En este apartado se describen las distintas secciones del sitio web y el trabajo desarrollado asociado a cada una de las mismas.

7.3.1 CODECS

Se brinda información muy general de los codificadores HEVC y H.264. Además de ello en la parte inferior haciendo click en la imágen correspondiente al codificador, se tiene acceso a una comparación subjetiva de carácter visual de un video con alta calidad y larga duración codificado en H.264 y HEVC. Se recomienda la reproducción de los mismos en pantalla completa y con una buena conexión a internet para observar el video sin interrupciones y poder detallar a fondo las diferencias visuales entre los codificadores. Por otra parte esta pestaña está creada para observar el gran potencial que tienen los codificadores del ultima generación MPEG para videos con altas resoluciones.

Se relaciona la secuencia utilizada en la tabla 7.1.

Secuencia	Resolución (px)	FPS	Duración	Frames	Tamaño Original
Big Buck Bunny	1290 x 1080	40	9min 56s	14315	44.56 GB

Tabla 7.1: Secuencia para evaluación subjetiva.

7.3.2 COMPARACIÓN VISUAL

En esta pestaña se pretende facilitar el proceso de evaluación subjetiva del usuario que desee observar en detalle secuencias codificadas en H.264, HEVC, VP8 y VP9 con características similares haciendo uso del software ffmpeg. Se recomienda dejar que el corto video llegue hasta el final y luego expandir a modo completo para observar pequeños detalles de este último frame, sin embargo, se adjuntan imágenes en alta resolución de un mismo frame codificado en los 4 estándares para y se aprecien así las diferencias.

Se utilizan las secuencias mencionadas en la tabla 7.2.

Secuencia	Resolución (px)	FPS	Duración	Frames	Tamaño
Four People	1280 x 720	50	10s	600	830.8 MB
Rush Field Cuts	1290 x 1080	30	19	560	2.36GB

Tabla 7.2: Secuencias Comparación Visual.

Para efectos de un mejor análisis de los detalles de cada codificador, se presentan las figuras 7.1 y 7.2.

En la secuencia Rush Field Cuts con considerable mayor cantidad de movimiento y complejidad, sí que puede apreciarse a simple vista las grandes diferencias entre los codificadores, esto debido a la exigencia que se hace a los codificadores en términos de predicción, filtrado, efecto bloque, etc. Se presenta en las figuras 7.3 y 7.4

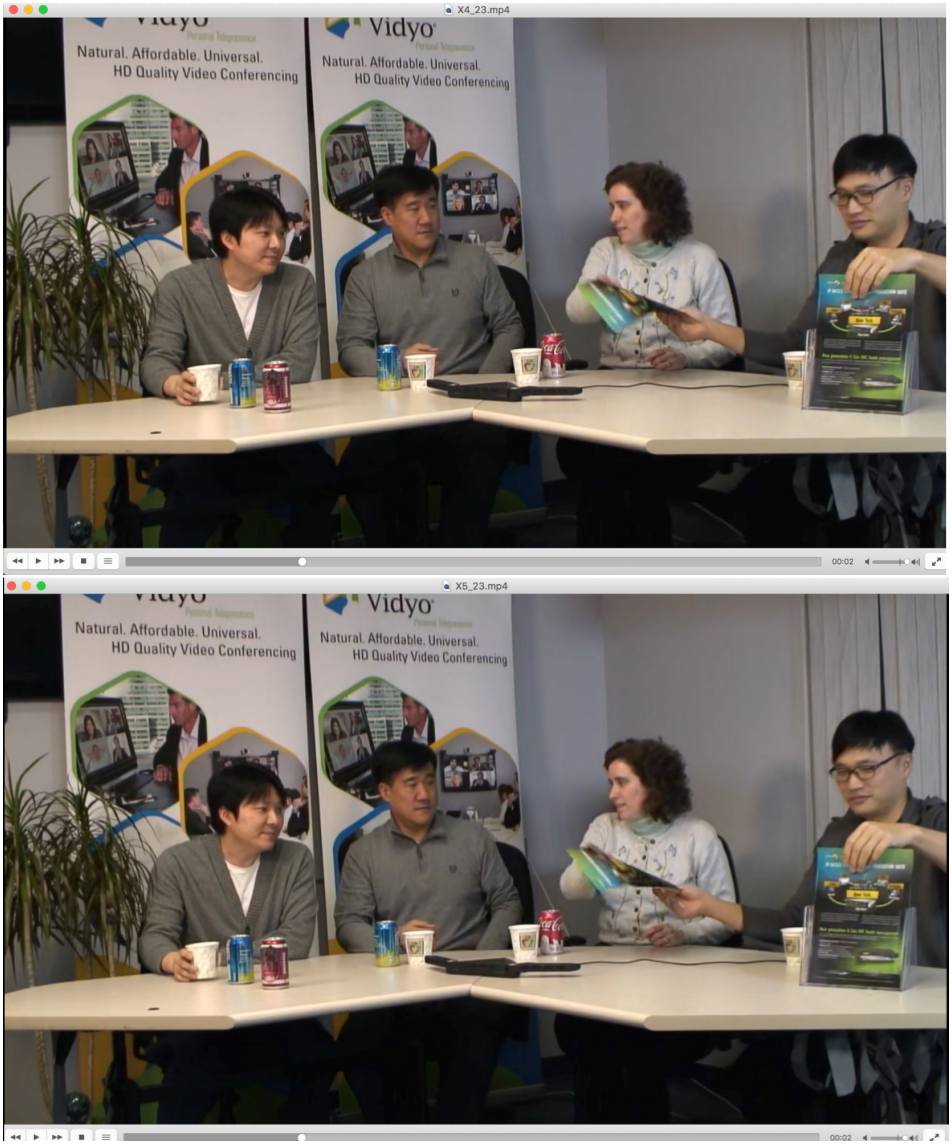

Figura 7.1: Comparación H.264 vs HEVC Four People

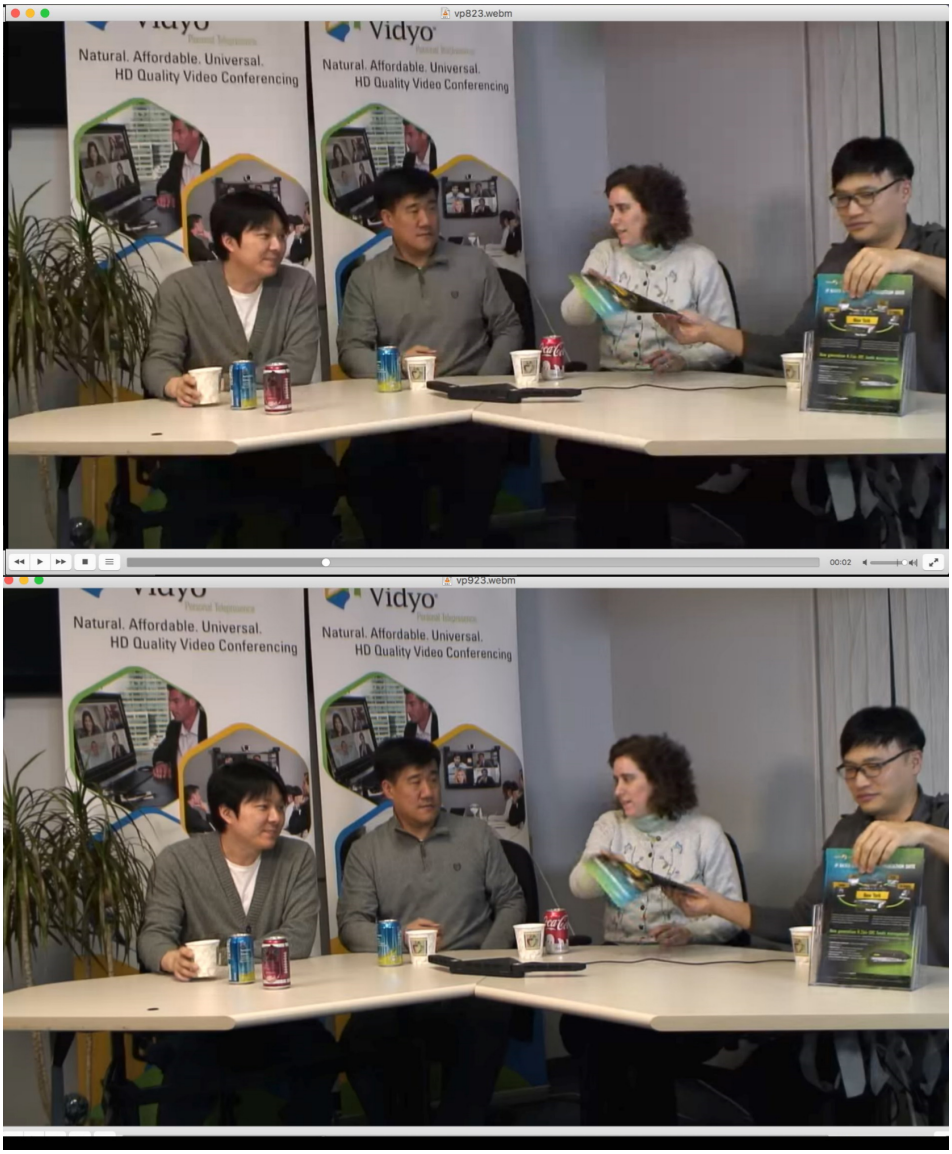

Figura 7.2: Comparación VP8 vs VP9 Four People

Figura 7.3: Comparación H.264 vs HEVC Rush Field Cuts

Figura 7.4: Comparación VP8 vs VP9 Rush Field Cuts

Se presenta en la figura 7.5 la compatibilidad que se obtuvo para la reproducción de videos streaming almacenados en diferentes navegadores de la actualidad con los sistemas operativos correspondientes.

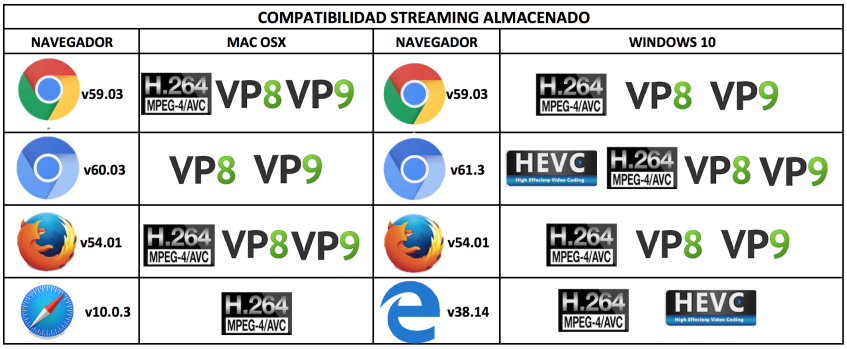

Figura 7.5: Compatibilidad Video Streaming

Se concluye entonces que:

- VP8 tiene un marcado efecto de bloque que se puede observar fácilmente en el fondo de la escena respecto a los demás.
- VP9 tiene menor detalle visual respecto a HEVC, esto se puede apreciar directamente en la parte derecha del cabello de la mujer.
- HEVC posee mejor manejo de filtro antibloques que H.264, esto se observa detallando el fondo de la escena.
- VP9 posee mayor efecto de suavizado de la imagen respecto a VP8, pero no a HEVC.
- HEVC posee menor efecto bloque que H.264, se puede observar claramente en Rush Field Cut.

7.3.3 DASH

Se ha realizado también la configuración de reproducción de streaming adaptativo haciendo uso de DASH, también definido en capítulos anteriores que, básicamente permite al usuario una calidad de experiencia superior a la tradicional, debido a que se eliminan los comunes buffers de carga que interrumpen la reproducción del video por falta de recursos de ancho de banda de la conexión a internet que se tenga. A cambio de esto, se realiza la especificación de un archivo *.mpd* que selecciona automáticamente videos con diferentes resoluciones creados mediante codificadores con soporte para DASH [37], esto se traduce en archivos con diferentes tamaños

que, dependiendo del ancho de banda que tenga cada usuario, se adaptará el video que mejor se adapte para que no se interrumpa el curso del video y por el contrario únicamente haya un cambio en la calidad de la imagen.

Mediante el uso de Ffmpeg se realiza la configuración del video que se desea convertir a DASH mediante una serie de comandos que especificarán las características de cada secuencia, por ejemplo, el bitrate máximo y mínimo del video, el codificador de audio y video, la duración en frames de las secuencias, la resolución y el formato multimedia que se desea. Una vez se realiza esta configuración se crea el archivo dash mpd mediante mp4box que, selecciona las diferentes secuencias generadas con ffmpeg y lo convierte en un archivo de texto xml con formato .mpd que será el que posteriormente el navegador web interpretará y seleccionará los videos correspondientes.

Un archivo típico mpd. se presenta en la figura 7.6. En este se aprecian varias secciones, una primera con información básica del archivo mpd, posteriormente se observan varias secciones llamadas *Representation* identificadas con un id que identifica cada una de las secciones creadas con Ffmpeg, las características de audio y video que tienen y el ancho de banda que utilizarán para su reproducción.

h264dash.mpd

```
<?xml version="1.0"?>
<!-- MPD file Generated with GPAC version 0.7.2-DEV-rev67-g975c0eb-master  at 2017-06-16T20:46:45.753Z-->
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" minBufferTime="PT1.500S" type="static" mediaPresentationDuration="PT0H9M56.459S" maxSegmentDuration="PT0H0M3.000S"
profiles="urn:mpeg:dash:profile:isoff-on-demand:2011">
 <ProgramInformation moreInformationURL="http://gpac.io">
  <Title>h264dash.mpd generated by GPAC</Title>
 </ProgramInformation>

 <Period duration="PT0H9M56.459S">
  <AdaptationSet segmentAlignment="true" lang="und" subsegmentAlignment="true" subsegmentStartsWithSAP="1">
   <Representation id="1" mimeType="audio/mp4" codecs="mp4a.40.2" audioSamplingRate="403786593" startWithSAP="1" bandwidth="260676">
    <AudioChannelConfiguration schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
    <BaseURL>video720_track2_dashinit.mp4</BaseURL>
    <SegmentBase indexRangeExact="true" indexRange="854-3285">
     <Initialization range="0-853"/>
    </SegmentBase>
   </Representation>
   <Representation id="3" mimeType="audio/mp4" codecs="mp4a.40.2" audioSamplingRate="403786593" startWithSAP="1" bandwidth="131654">
    <AudioChannelConfiguration schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
    <BaseURL>video540_track2_dashinit.mp4</BaseURL>
    <SegmentBase indexRangeExact="true" indexRange="854-3285">
     <Initialization range="0-853"/>
    </SegmentBase>
   </Representation>
   <Representation id="5" mimeType="audio/mp4" codecs="mp4a.40.2" audioSamplingRate="403786593" startWithSAP="1" bandwidth="65695">
    <AudioChannelConfiguration schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
    <BaseURL>video360_track2_dashinit.mp4</BaseURL>
    <SegmentBase indexRangeExact="true" indexRange="854-3297">
     <Initialization range="0-853"/>
    </SegmentBase>
   </Representation>
  </AdaptationSet>
  <AdaptationSet segmentAlignment="true" maxWidth="1280" maxHeight="720" maxFrameRate="24" par="16:9" lang="und" subsegmentAlignment="true" subsegmentStartsWithSAP="1">
   <Representation id="2" mimeType="video/mp4" codecs="avc1.64001F" width="1280" height="720" frameRate="24" sar="1:1" startWithSAP="1" bandwidth="1324575">
    <BaseURL>video720_track1_dashinit.mp4</BaseURL>
    <SegmentBase indexRangeExact="true" indexRange="936-3355">
     <Initialization range="0-935"/>
    </SegmentBase>
   </Representation>
   <Representation id="4" mimeType="video/mp4" codecs="avc1.64001F" width="960" height="540" frameRate="24" sar="1:1" startWithSAP="1" bandwidth="708534">
    <BaseURL>video540_track1_dashinit.mp4</BaseURL>
    <SegmentBase indexRangeExact="true" indexRange="936-3355">
     <Initialization range="0-935"/>
    </SegmentBase>
   </Representation>
   <Representation id="6" mimeType="video/mp4" codecs="avc1.64001E,mp4a.40.2" width="640" height="360" frameRate="24" sar="1:1" audioSamplingRate="403786593" startWithSAP="1"
bandwidth="452527">
    <AudioChannelConfiguration schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
    <BaseURL>video360_dashinit.mp4</BaseURL>
    <SegmentBase indexRangeExact="true" indexRange="1435-3854">
     <Initialization range="0-1434"/>
    </SegmentBase>
   </Representation>
  </AdaptationSet>
 </Period>
</MPD>
```

Figura 7.6: Resultado archivo MPD

Se muestra en la figura 7.7 un sencillo código necesario para la reproducción del archivo .mpd. Como se mencionó en capítulos anteriores, se hace necesario el uso del reproductor *dash.all.min.js* el cual se encuentra alojado en la misma carpeta de la página y se le hace llamado dentro del código de reproducción HTML5.

```
<center>
       <script src="dash.all.min.js"></script>
        <video width="700" height="500" data-dashjs-player
src="video720_dash.mpd" controls="controls"> </video>
     </center>
```

Figura 7.7: Código para reproducción de DASH bajo reproductor dash.js

El resultado de las peticiones hechas por el cliente hacia el servidor de videos de H.264, y el retorno de los diferentes videos dependiendo de la conexión se muestra en la Figura 7.8, en donde se hace uso del inspector del navegador Mozilla que actúa como un monitor de red, este se puede abrir con el botón F12 o haciendo click derecho y luego dar click en inspeccionar. Una vez esto se debe buscar la pestaña RED, dentro de esta se pueden analizar entonces las peticiones GET que se realizan al servidor y los archivos que este retorna dependiendo de su ancho de banda. En el ejemplo de la figura se observa como progresivamente va subiendo la calidad del video, esto se debe a que el archivo mpd en un principio busca ganar tiempo descargando los archivos mas pequeños y aprovechar este tiempo cuando comienza la reproducción en baja calidad para descargar aquellos que tengan mejor calidad y darle al usuario una mejor experiencia. Esto se puede observar en la columna de tiempos, donde se observa que el tiempo de descarga aumenta si se descarga una sección de video con mayor calidad. Además la imagen de abajo muestra el resultado de una prueba realizada con la misma red, luego de generarle tráfico. Se observa entonces que el tiempo aumenta de manera considerable para la descarga de las secuencias.

Para efectos del desarrollo del escenario ideal en donde funcionaría el video HEVC basado en tecnología adaptativa DASH se tiene que, el único navegador que luego de las pruebas, soportó este escenario fue el navegador Microsoft Edge en su versión 38.14 correspondiente al sistema operativo Windows 10 Pro, versión 2016. Se probó también con la versión 25.1058 correspondiente a Windows 10 Pro 2015 con Microsoft Edge 25.105 pero no se obtuvo resultado positivo.

La figura 7.9 muestra el resultado de la prueba del monitoreo de red de elementos de video DASH codificados con HEVC.

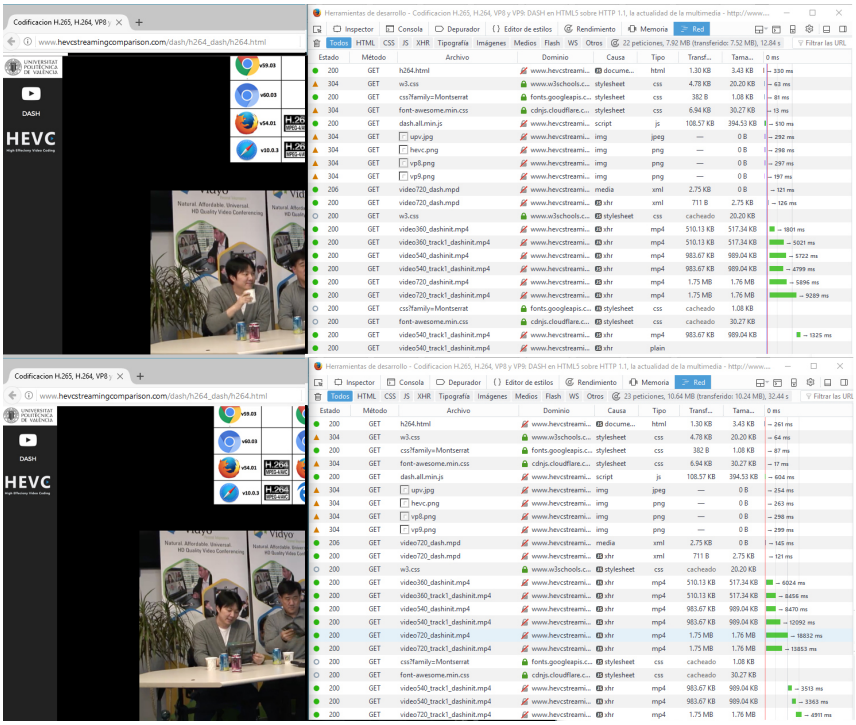

Figura 7.8: Comparación tiempos H.264 DASH con aumento de tráfico en la red

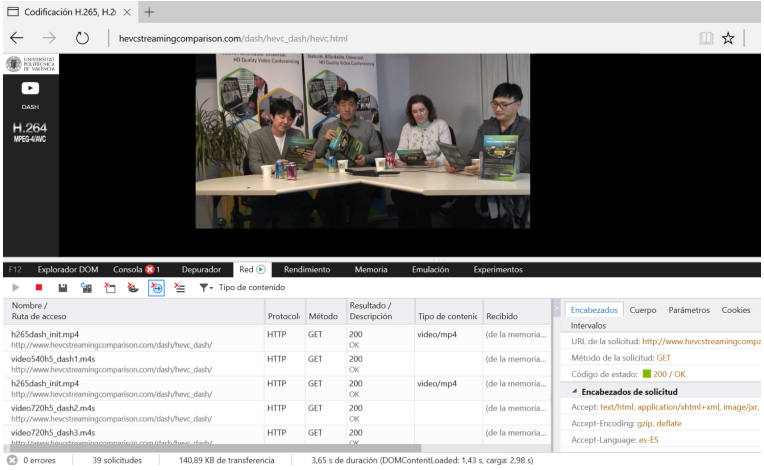

Figura 7.9: Video HEVC DASH sobre HTTP Microsoft Edge

Se presenta en la figura 7.10 la compatibilidad que se obtuvo para la reproducción de videos DASH en diferentes navegadores de la actualidad con los sistemas operativos correspondientes.

COMPATIBILIDAD DASH			
NAVEGADOR	MAC OSX	NAVEGADOR	WINDOWS 10
v59.03		v59.03	
v60.03		v61.3	
v54.01	H.264 MPEG-4/AVC	v54.01	H.264 MPEG-4/AVC
v10.0.3	H.264 MPEG-4/AVC	v38.14	H.264 MPEG-4/AVC HEVC

Figura 7.10: Compatibilidad DASH

7.3.4 VS

En esta pestaña del sitio web se muestran las pruebas y resultados obtenidos en el capítulo anterior comparando el desempeño de HEVC contra H.264.

Capítulo 8

Conclusiones

Se observó que HEVC es un estándar cuyas prestaciones son sin lugar a dudas superiores respecto a H.264, haciendo uso de diferentes secuencias de video con varias características, tamaños, resoluciones, complejidades, entre otros diferentes factores. se logró evidenciar que HEVC trabaja de forma más eficiente generando archivos con altísima calidad visual, con una relación señal a ruido mayor y con hasta un 56 % de reducción en el tamaño final del archivo. Lo anterior se convierte en un importante hito para el desarrollo de los codificadores de ultima generación que deben actuar en videos con cada vez mayor complejidad, resolución y tamaño, debido al exponencial avance en materia de generación de contenidos.

Aunque HEVC es muy llamativo debido a sus importantes mejoras respecto a los estándares líderes de la actualidad, hay que tener presente que al no ser software de libre distribución tiene una gran desventaja respecto a los desarrollos que lidera Google ya que se ha evidenciado que al transcurrir los años esta empresa está creciendo a pasos agigantados con las plataformas de video que le pertenecen y que sin lugar a dudas lideran el mercado, lo que se hace más imponente para aquellas empresas que quieran distribuir contenidos en la web y hacer uso de codificación de ultima tecnología porque sin duda preferirán aquella que tenga mas compatibilidad, mayor alcance y genere menor costo económico. Sin embargo hay grandes proveedores de contenido multimedia bajo demanda como Netflix que estudian la mejor opción para mejorar la calidad de sus contenidos, por lo que ha optado por el análisis y uso de HEVC para sus contenidos en definiciones como 4K, haciendo uso del Edge Browser.

Se logra cumplir el objetivo de este proyecto al elaborar una plataforma de distribución de contenidos de video en internet bajo HTTP que permita la reproducción de videos codificados en HEVC tanto de manera almacenada como haciendo uso de DASH. Se evidenció que este ultimo posee ciertas ventajas debido a su facilidad

de adaptación a las características de la red, ofreciendo al usuario final una mejor calidad de experiencia en su forma de visualizar contenidos.

Se ha demostrado experimentalmente que si bien HEVC posee grandes ventajas sobre su antecesor H.264. aún no es un estándar que se pueda utilizar en la época actual con la misma facilidad debido a su problema de compatibilidad para reproducción de contenidos en los navegadores actuales. El único navegador que permitió compatibilidad total y sin hacer uso de complementos adicionales fue el Microsoft Edge en su versión 38.14, ya que aunque Chromium permite reproducir video almacenado, lo hace a través del complemento de compatibilidad con HEVC distribuido por GitHub. Es entonces que mediante este trabajo se mostró que un punto de partida para el futuro del streaming almacenado y adaptativo con importantes mejoras sobre el estado actual de la distribución de contenidos puede ser posible haciendo uso de las herramientas descritas y el navegador mencionado anteriormente.

Como trabajo futuro se propone la implementación del mismo esquema de servidor de contenidos para videos en HEVC, pero montado sobre HTTP2 que, como se analizó en el capitulo 5, posee importantes mejoras respecto a la actualidad del protocolo en la web ya que, gracias a la implementación de una única conexión se podrán atender paralelamente las peticiones que el cliente haga y además debido a las estimaciones de los posibles recursos que el cliente requiera se reducen considerablemente los tiempos de carga de los videos aprovechando mejor los recursos de red disponibles.

Capítulo 9

REFERENCIAS

[1] https://www.census.gov/popclock/

[2] https://www.gsmaintelligence.com

[3] Introduction to video compression Berkeley Design Technology, Inc.

[4] Javier Joglar, Video Compression, 2009.

[5] H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia.

[6] VP8 http://aprendiendo2veces.blogspot.com.co/2013/05/vp8-vs-H.264.html.

[7]https://tools.ietf.org/html/rfc6386

[8] VP9 http://www.ietf.org/proceedings/85/slides/slides-85-videocodec-4.pdf

[9] Martin Rerabek and Touradj Ebrahimi, Comparison of compression efficiency between HEVC/H.265 and VP9 based on subjective assessments, Available. https://infoscience.epfl.ch/record/200925/files/article-vp9-submited-v2.pdf

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

[11]Enrique de la Torre Moya, Evaluación y Diseño de algoritmos eficientes para la mejora de un transcodificador de video HEVC/VP9, Trabajo fin de grado, Julio 2014.

[12] Gabriel Cebrián Márquez, Algoritmo de pre-análisis para el estándar de codificación de vídeo HEVC, TFM, UNIVERSIDAD DE CASTILLA-LA MANCHA, Julio de 2015

[13] RFC, Hypertext Transfer Protocol – HTTP/1.1 Available: http://www.ietf.org/rfc/rfc2616.txt

[14] Hypertext Transfer Protocol, Available: https://es.wikipedia.org/wiki/Hypertext_Transfer_Pr

[15] Adaptive Bitrate Streaming, Available: https://en.wikipedia.org/wiki/Adaptive_bitrate_strea

[16] RFC HTTP2.0 https://tools.ietf.org/html/rfc7540 La Televisión Digital: Fundamentos y teorías, Manuel Cubero Enrici, 2009

[17] Miguel Angel Alvarez. Qué es HTML.
Available: http://www.desarrolloweb.com/articulos/que-es-html.html.

[18]Juan Jose Pino Reyes. Qué es HTML?. Available: https://devcode.la/blog/que-es-html/.

[19] J. T. Llorens,Qué es HTML5 y para qué sirve? Available:
http://www.blogeninternet.com/2013/04/que-es-html5-y-para-que-sirve.html.

[20] https://medium.com/netflix-techblog/update-on-html5-video
-for-netflix-fbb57e7d7ca0.

[21] https://medium.com/netflix-techblog/a-large-scale-comparison-of-x264-x265-and-libvpx-a-sneak-peek-2e81e88f8b0f.

[22] Red grafica. MP4, un formato multimedia - Red Gráfica Latinoamérica Available: http://www.cad.com.mx/.

[23] WebM: an open web media project, Available: https://www.webmproject.org

[24] "x264,"2017. Available: https://es.wikipedia.org.

[25] "x265,"2016. Available: https://es.wikipedia.org.

[26] Alegsa, Qué es ffmpeg.exe? - Proceso/archivo: ffmpeg.exe
Available: http://www.alegsa.com.ar/Proceso/ffmpeg.exe.php

[27] T. MundoDivX.com - Edición de archivos MP4 con YAMB y MP4Box Available: http://www.mundodivx.com/edicionsimple/yamb.php.

[28] Open SUSE Chromium openSUSE Available: https://es.opensuse.org/Chromium.

[29] Chromium Project Chromium (navegador). Available: https://es.wikipedia.org.

[30] "Microsoft Edge", 2017. Available:
https://es.wikipedia.org/w/index.php?title=MicrosoftEdgeoldid=99744774.

[31] La Nacion. Así es Edge, el nuevo navegador web de Microsoft que reemplazará a Internet Explorer en Windows 10. Available: http://www.lanacion.com.ar/1788954-asi-es-edge-el-nuevo-navegador-web-de-microsoft-que-reemplazara-a-internet-explorer-en-windows-10.

[32]Encoding "DASH Streaming", Available: https://www.encoding.com/mpeg-dash/.

[33] "Dynamic Adaptive Streaming over HTTP,"2017.
Available: https://en.wikipedia.org/

[34]T. Innovadora, "TECNOLOGIA INNOVADORA: CUALES SON LAS APLICACIONES O USOS DE HTML5?", 2012.
Available: http://tecnologiainnovadoraunad.blogspot.com.co/2012/05/cuales-son-las-aplicaciones-o-usos.html.

[35] Academic. PSNR. Available: http://www.esacademic.com/dic.nsf/eswiki/889352.

[36] Jeroen van der Hooft, Stefano Petrangeli, Tim Wauters, Rafael Huysegems, Patrice Rondao Alface, Tom Bostoen, and Filip De Turck. HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks, IEEE COMMUNICATIONS LETTERS, VOL. 20, NO. 11, NOVEMBER 2016

[37] https://github.com/Dash-Industry-Forum/dash.js/