UNIVERSIDAD POLITÉCNICA DE VALENCIA

ESCUELA POLITÉCNICA SUPERIOR DE GANDÍA

MÁSTER EN POSTPRODUCCIÓN DIGITAL

"Estudio de emisión de vídeo sobre HTML5"

TRABAJO FINAL DE MÁSTER

Autor: **Andrés López Herreros**

Director: **Jaime Lloret Mauri**

Gandía, septiembre de 2014

"La lucha es lo más importante;
no su resultado, sino la rebelión en sí misma"

FRITZ LANG

Resumen / Abstract

El siguiente proyecto se presenta como un estudio de las diferentes posibilidades de emisión de vídeo digital a través del lenguaje de etiquetas HTML5 considerando el soporte de los navegadores del mercado frente a los diferentes formatos de contenedor y *códecs*. Igualmente, se analizan los distintos parámetros de compresión de software de codificación de vídeo para comprobar los ratios que soporten una mejor relación de tamaño/*bitrate* para su emisión en Internet bajo el estándar de HTML5.

The next project is presented as a study of the different possibilities of digital video broadcast through HTML5 markup considering the market browsers support compared to the different codecs and container formats. Similarly, the various parameters of software compression video coding are analyzed for the ratios that support a better size/bitrate for broadcast on the Internet under the standard HTML5.

Palabras Clave / Key Words

- **ESPAÑOL**: HTML5, códec, compresión, vídeo, API.
- **ENGLISH**: HTML5, codec, codification, video, API.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

1.1 INTRODUCCIÓN

El tema del siguiente proyecto es el estudio de los diferentes *códecs* de vídeo que trabajan dentro del estándar de HTML5 dentro de los límites que permiten sus APIs así como del lenguaje JavaScript. También se realizan comparativas de usabilidad y eficiencia de los tres principales algoritmos de compresión de vídeo (.mp4, .ogg, .webm) dentro de los principales navegadores del mercado y los ratios de codificación para ajustar una reproducción de vídeo online dentro de los márgenes que establece el ancho de banda utilizado.

El primer paso fue la creación de una página web donde se pudieran hacer las pruebas de almacenamiento y emisión de los archivos de vídeo codificados previamente. La idea futura es poder utilizar esta plataforma web para realizar una compresión paralela en tiempo real (*transcoding*) a través de librerías de codificación dependiendo de factores como el ancho de banda disponible del usuario así como el tipo de dispositivo que realiza la petición o la resolución del mismo. La página web http://www.html5video.es se creó adquiriendo un *hosting* facilitado por la Universidad Politécnica de Valencia (UPV) y un dominio .es disponible en Internet y contratado para 5 años. Dentro del *hosting* o espacio de almacenamiento vinculado a las DNS del nuevo dominio creé el archivo raíz o *index* que conduciría a la página principal del proyecto web y abriría las diferentes subpáginas con toda la información, tablas de mediciones y emisiones de vídeo dentro de un área reservada, por lo que utilicé también bases de datos en SQL y librerías JQuery para la implementación.

El segundo paso fue la investigación y medición de resultados a través de un archivo original en formato .mov con una profundidad de color de 8 bits y un *subsampling* de

chroma 4:4:2. La idea original era la utilización de un *source* de vídeo *lossless* sin ningún tipo de compresión con un formato .raw, una profundidad de color de 12 bits y un *subsampling* de *chroma* 4:4:4 pero un archivo de tales dimensiones ralentizaba mucho el proceso de codificación y requería de un procesador, memoria RAM y tarjeta gráfica con unas características muy altas. Con el archivo original .mov realicé las diferentes mediciones según aspectos como codificador software (VLC y SUPER), *frames* por segundo (PAL 25 y NTSC 23,976) y *bitrates* entre 64 y 7950[1] kbps para conseguir los resultados de tamaño y tiempo de codificación así como posteriormente los ratios de compresión y las gráficas de usabilidad para la emisión web de los mismos. También se realizan mediciones gráficas de la emisión de vídeo sobre HTML5 a través de un software de inyección de tráfico similar al WireShark (Testtool), en él se muestra la gráfica de paquetes recibidos por la tarjeta de red que detallo en la página 77, el retardo de la emisión (*delay*) en milisegundos, la desviación de la periodicidad de la señal (*jitter*) en milisegundos y el Ancho de Banda consumido en bits/s. Las pruebas se realizan con ejemplos de codificación de las tablas, concretamente con la compresión MPEG-4 (H.264) AVC tanto en PAL 25 como en NTSC 23,976.

El paso final es la elaboración de las conclusiones que obtengan un estudio detallado acerca de los beneficios y los inconvenientes de la codificación en los tres principales *códecs* utilizados en la web (.mp4, .ogg y .webm) con su utilización en diferentes parámetros y atendiendo a los ratios de compresión y la calidad-tamaño obtenidos. Posteriormente, se incluyen en el *hosting* de la página web (www.html5video.es) los resultados óptimos para poder reproducir de la manera más eficiente los archivos en diferentes navegadores según las exigencias de cada uno haciendo una demostración de cuál de todos los ejemplos incluidos en las gráficas de análisis son los óptimos para cada situación, siempre utilizando la API de HTML5 para los distintos navegadores web.

[1] Estándares de VLC a SUPER, desde la mínima codificación soportada hasta UHD 8K HEVC

1.2 OBJETIVOS

Por tanto, el presente proyecto profundiza en muchos aspectos de los modelos de transmisión y compresión de vídeo centrando su atención en lo que considero el principal objetivo del trabajo:

- Conocer las posibilidades de emisión de vídeo en la web y la codificación de los archivos frente a diferentes formatos y *códecs*.

Es necesario precisar los límites del trabajo ya que un ejercicio de análisis de todo el proceso electrónico y digital de conversión, emisión y codificación de vídeo puede resultar excesivamente complejo por lo que el índice creado hace hincapié en una introducción a cada uno de los campos necesarios de la telemática, informática, tecnología y codificación multimedia que se precisan para comprender todos los pasos desde la captación digital de la imagen hasta la reproducción vía web de los archivos comprimidos según parámetros como contenedores, *códecs*, ancho de banda, etc.

Por ello, la elaboración de un listado de objetivos secundarios a alcanzar con el proyecto también puede ayudar al proceso de comprensión general de la idea u objetivo general de este proyecto:

- Realizar un estudio de los principales *códecs* de vídeo y audio así como formatos para elaborar tablas que permitan un análisis de uso en los distintos navegadores del mercado.
- Codificar vídeos para obtener datos sobre diferentes parámetros y elaborar conclusiones que faciliten al usuario un ratio de compresión adecuado.
- Confeccionar un portal web con HTML5 y CSS para utilizarlo como plataforma.

- Programar código JavaScript para crear acciones de JQuery dentro de las etiquetas HTML con APIs que faciliten la reproducción de los vídeos.
- Crear un servidor Web con Apache para poner en práctica las emisiones.

1.3 PRECEDENTES DEL PROYECTO

El presente trabajo de investigación sigue la línea de varios trabajos realizados previamente que sirven como continuación o extensión de aquellos. Hasta la fecha de ejecución de este proyecto (Estudio de emisión de vídeo sobre HTML5), en la Universidad Politécnica de Valencia, sólo habían estos proyectos relacionados:

"Compresión de vídeo en TV de Alta Definición[2]" realizado por Alberto Rodríguez Zaurín en el año 2006 y tutorizado por Jaime Lloret Mauri y José Miguel Jiménez Herranz. En él se desarrolla la creación de un video clip no comprimido en HDTV así como la elaboración de la secuencia de imágenes tomando un número muy elevado de fotografías, para luego poder montar la secuencia mediante un editor de video.

"Sistemas de *streaming* de vídeo. Recopilación y comparativa[3]" realizado por Antonio Gómez Bermúdez en el año 2013 y tutorizado por Fernando Boronat Seguí. En él se desarrolla la puesta en funcionamiento de una plataforma hardware incluyendo el software asociado, que hace realidad el objetivo principal del proyecto CanalGV que consiste en la transmisión de contenidos en directo y bajo demanda sobre Internet o sobre la Intranet corporativa de la Generalitat utilizando el sistema de *streaming*.

[2] http://riunet.upv.es/handle/10251/32847
[3] http://riunet.upv.es/handle/10251/32807

"*SVD para la transmisión progresiva de imágenes y la codificación de vídeo digital*[4]" realizado por José Antonio Verdoy González en el año 20109y tutorizado por Francisco Javier Villanueva Oller y Rafael Jacinto Villanueva Micó. En él se desarrolla un algoritmo de codificación adaptativa con pérdida para imágenes digitales 2D y 3D, basado en la descomposición de valores singulares (SVD) estudiando su eficacia y razonando sobre los resultados obtenidos. La codificación propuesta resulta útil para la transmisión progresiva de imágenes.

1.4 ESTRUCTURA DEL PROYECTO

En el siguiente trabajo, el lector se va a encontrar información variada obtenida de la investigación y la programación de diferentes códigos para poner en práctica la eficiencia de los distintos algoritmos de compresión dentro del código HTML de Internet, se encuentra dividido en cinco capítulos con sus correspondientes subcapítulos donde se detalla la información necesaria para la compresión de la idea general del proyecto así como de sus objetivos secundarios:

2. El estándar HTML5

Este apartado incluye un estudio detallado de la confección de la página web creada como plataforma para la emisión de archivos multimedia. Para ello, se centra en todos los elementos necesarios para unificar vídeo y HTML, principalmente a través del *scripting* de JavaScript y las APIs que proporciona el nuevo modelo HTML5. También se valora la importancia de este nuevo estándar frente a los anteriores de HTML y XML así como la necesidad de crear un *códec* único que permita regularizar el uso de vídeo codificado en la red ya que los navegadores más importantes solamente tienen soporte para algunos de ellos. También se pretende arreglar el problema del pasado

[4] http://riunet.upv.es/handle/10251/8324

en el que el uso de vídeo en la red dependía de licencias y usos de plugins de terceros como es el caso de Adobe Flash, un problema muy importante de implementar en el nuevo código ya que más de 60% de los usuarios de la población mundial todavía usan versiones anteriores a Internet Explorer 6.0.

Además, se muestra el código del *header* del *index,* el HTML para la elección automática del formato de vídeo adecuado según el tipo de navegador así como algunas funciones de JavaScript creadas para averiguar el tipo de navegador o para ejecutar la API WebWorkers de HTML5.

3. Formatos, *códecs* y soporte en navegadores

En este apartado se explica la diferencia entre los conceptos de formato y *códec* con una larga investigación de los tres principales *códecs* del estándar HTML5 en Internet (.mp4, .ogg y webm). Estos tres formatos son los elegidos para realizar las mediciones con los codificadores de vídeo cuyos resultados se muestran posteriormente.

En el análisis de soporte en navegadores web se detalla un estudio acerca de la compatibilidad y las características de los diferentes navegadores del mercado con los formatos y *códecs* expuestos previamente. Tal y como se detalla anteriormente, existe una guerra de marketing y de imposición de *códecs* por parte de las principales empresas con la excepción de .ogg que parte de un software libre y que no cobra licencias ni a corporaciones ni a usuarios. En este apartado se podrán apreciar las diferencias en el soporte de formato por parte de una serie de navegadores que también se estudiaran en profundidad (Chrome, Internet Explorer, Firefox, Safari y Opera).

4. Estudio y análisis de la codificación y emisión de vídeo sobre HTML5

Finalmente, este es el apartado más cercano a la investigación y que servirá como trampolín para una futura Tesis Doctoral sobre las opciones de emisión de vídeo digital en *transcoding* a través de HTML con lectura de cabeceras MIME. Se ofrecen los datos, ratios y parámetros de codificación de un vídeo original sin compresión en diferentes *bitrates* y dependiendo también de otros factores como el número de *frames* por segundo (NTSC, PAL) así como el tiempo empleado para la codificación, el tamaño final y la calidad obtenida. A partir de esta información se obtienen ratios para decidir que combinación de elementos es la más óptima según las necesidades del usuario (ancho de banda, navegador utilizado, versión, procesador, resolución, dispositivo, etc.) y poder ofrecer el vídeo original con la compresión óptima desde la red dependiendo de todos estos factores.

La diferencia con otros portales como Youtube o Vimeo es que estos ofrecen diferentes opciones de calidad y resolución de un mismo vídeo pero sin ningún tipo de análisis ni interacción con la cabecera MIME, solamente guardan una copia de cada una de las versiones del vídeo y se ofrecen según la petición del usuario. En este caso, a través de la página web creada www.html5video.es se ofrecen vídeos en la zona registrada que solamente conservan el modelo original sin pérdidas en el servidor; cuando el usuario solicita uno de ellos se tienen en cuenta todos los parámetros del estudio de este apartado y el tipo de dispositivo utilizado para realizar un *transcoding* en paralelo del original y presentárselo en las condiciones óptimas personalizadas.

Finalmente, en este punto se realiza una discusión de resultados basada en el análisis de información obtenida, ratios y gráficas anteriores. En ella se deciden finalmente aspectos como cuando es más efectivo cada códec según *frame rate*, qué *códec* aporta más ratio de compresión o qué tamaños obtienen son los mejores para codificar. Para

concluir, se ofrecen las gráficas de emisión de vídeo sobre HTML5 a través de un software de inyección de tráfico para multimedia (Testtool) en el que se comparan los resultados de emisión (cliente – servidor) a través de internet mediante un ejemplo de codificación en MP4 (H.264) tanto en PAL 25 como en NTSC 23,976.

2. EL ESTÁNDAR HTML5

2.1 CONFECCIÓN DE LA PÁGINA WEB Y ENLACE CON VÍDEO

La página web es la plataforma para poner en marcha todo el estudio teórico realizado en el presente estudio. Se compone de una página principal (*index*) que referencia mediante hipervínculos los diferentes temas de la Tesina para obtener más información respecto a cada capítulo. Ha sido diseñada desde cero a través de lenguaje de etiquetas HTML para su compilación en el estándar de HTML5 lo que ha permitido trabajar junto a otro tipo de lenguajes y aplicaciones para crear una plataforma de emisión de vídeo así como de archivos multimedia [1]. Aquí se muestra un ejemplo del uso de la etiqueta API <video> para la emisión sobre HTML5.

```
<video id="movie" width="640" height="480">

    <source src="video.mp4" type='video/mp4;
    codecs="avc1.42E01E, mp4a.40.2"' />
    <source src="video.webm" type='video/webm;
    codecs="vp8, vorbis"' />
    <p>The video is available as <a href="video.mp4">H.264 in an MP4
    container</a>, or as <a href="video.webm">VP8 in a WebM container</a>.</p>

</video>
```

JavaScript ha sido utilizado para crear código que actúe desde el servidor y que permita un análisis del navegador utilizado por el usuario para facilitarle el vídeo en su codificación soportado que se encuentra en el servidor, este *script* analiza la petición del usuario y determina en la comunicación de extremo a extremo que navegador utiliza y que versión del mismo (necesario para generar, si es necesario, otro *script* que de soporte a Adobe Flash) [5]. Una vez resuelto esto se accede al servidor y se envía a

través de la API de vídeo de HTML5 el vídeo en el formato y *códec* más adecuado[5]. Otra API de HTML5 generada también con JavaScript permite que las APIs implicadas (Canvas, Vídeo, etc.) se carguen desde el servidor desde un primer acceso y de manera paralela para que cuando sean solicitadas por el usuario se encuentren disponibles en *buffer*[6] [4]. Además, se simplifica la programación del código de JavaScript a través de las librerías de JQuery y Ajax que son muy útiles para obtener la página web RIA creada junto a los estilos CSS.

2.2 CÓDIGO HTML PARA EL *HEADER* DEL *INDEX*

```
<!DOCTYPE html>

<html lang="en"><head><meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
        <meta charset="utf-8">
        <title>Andrés López Herreros</title>

        <link rel="stylesheet" media="screen"
        <link rel="alternate" type="application/rss+xml" title="RSS"
href="http://www.csszengarden.com/zengarden.xml">
  <link rel="shortcut icon"
href="http://anlpeher.alumnos.upv.es/resources/images/html5.png">
  <script src="js/modernizr.custom.63321.js"></script>

        <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

[5] Observar tabla de compatibilidad en página 21
[6] Código creado para la API WebWorkers en página 20

```
        <meta name="author" content="Andrés López Herreros">
        <meta name="description" content="Andrés López Herreros. HTML5 Video
Broadcast">
        <meta name="robots" content="all">
```

2.3 CÓDIGO HTML PARA LA ELECCIÓN AUTOMÁTICA DEL FORMATO DE VÍDEO ADECUADO SEGÚN EL TIPO DE NAVEGADOR

```
<body>

<video width="720" height="576" poster="../../resources/images/igic_logo.png"
controls>
         <source src="../../videos/mpeg-ts_dirac_mp3_presentador.ts"
type="video/ts;">
    <source src="../../videos/ogg_theora_flac_presentador.ogg" type="video/ogg;">
    <source src="../../videos/mp4_h.264_aac_presentador.mp4" type"video/mp4;">

    <!-- Respaldo para Flash (Versiones anteriores de navegadores)-->

        <object width="720" height="576" type="application/x-shockwavw-flash"
   data="mediaplayer-5.5/player.swf">
        <param name="movie" value="mediaplayer-5.5(player.swf">
      <param name="allowFullScreen" value="true">
      <param name="wmode" value="transparent">
```

```
    <param name="flashvars"
value="controlbar=over&amp;amp;image=../../resources/images/igic_logo.png&amp;amp;file=..
/../videos/mp4_h.264_aac_presentador.mp4">
    </object>

</video>
</body>
```

2.4 CÓDIGO JAVASCRIPT PARA AVERIGUAR CUÁL ES EL NAVEGADOR

```
<p>
    <script type="text/javascript">
        document.write("Estás accediendo con " + navigator.appName + " " +
navigator.appVersion);
        </script>
  </p>
```

2.5 LLAMADA A UNA FUNCIÓN JAVASCRIPT PARA EJECUTAR LA API WEB WORKERS DE HTML5

```
var miworker = new Worker ("procesoWorker.js");
```

3. FORMATOS, CÓDECS Y SOPORTE EN NAVEGADORES

3.1 CONCEPTO CONTENEDOR

Un contenedor alberga datos (pista vídeo, pista audio, marcadores, bits de metadatos, sincronización vídeo-audio, etc.), los formatos más relevantes para HTML5 y que se han convertido en estándar para internet son MPEG-4, Ogg y WebM [6]. Son soportados por el 99% de los usuarios pero presentan condiciones ya que solamente existe soporte para cada uno de ellos según el navegador utilizado tal y como se muestra en la Tabla 1 [8]:

MP4/H.264/AAC → iPad, IE9, Sarari y Chromes anteriores (quiere abandonarlo)
Ogg/Theora/Vorbis → Firefox 3.5+ y Opera 10.5+
WebM/VP8/Vorbis → Chromes posteriores, Firefox 4+ y Opera 10.6+

Para usuarios de versiones anteriores a Internet Explorer 9 (que representa cerca del 60% de los usuarios a nivel mundial) sin soporte nativo HTML5 hay que declararlo con Flash de respaldo [15]. Los formatos más conocidos son .mov de QuickTime, .mp4 de MPEG, .wmv de Microsoft, .flv de Adobe, .mkv de Matroska (base del formato WebM), .avi y .ogg de Xiph. Cada uno puede encapsular cualquier códec (excepto algunos formatos no mencionados antes que no son compatibles con códecs con velocidades de transferencia variables) [4]. WebM ha sido definido para albergar solamente VP8 y Vorbis, Ogg con Theoria, Speex o Flac y mapeos con VP8 y Dirac mientras que MP4 suele contener Mp3, AAC y H.264 [4]. RF (*Royalty-Free*) es un estándar de derechos de autor de W3C que organiza los estándares de HTML con implementaciones sin derechos de autor. Esta es una tabla con los soportes de formatos en los diferentes navegadores.

	WebM VP8	**Ogg Theora**	**MP4 H.264**
Mozilla Firefox	SI	SI	NO
Safari	NO	NO	SI
Opera	SI	SI	NO
Chrome	SI	SI	SI
Internet Explorer	NO	NO	SI

Tabla 1. Soporte de los tres formatos del estándar HTML5 en los principales navegadores

3.2 CÓDEC

Un códec (*coder – decoder)* de video define un algoritmo para codificar y decodificar un flujo de datos con o sin pérdida de calidad, se divide en dos partes: codificación que incluye transmisión, almacenamiento y cifrado; decodificación que incluye reproducción (parte más importante en el análisis y en HTML5) y edición [6]. Los *códecs* más utilizados en el estándar actual de Internet son H.264, Theora y VP8. A continuación otro análisis con la implementación de estos tres *códecs* en los navegadores más importantes que también guarda relación con la encapsulación del formato que se ofrece en la Tabla 1 [8]:

	Firefox	Chrome	IE	Opera	Safari	iOS	Android
Ogg-Theora	3.5+	3+	-	10.5+	-	-	-
MP4-H.264	-	3-11	9+	-	4+	4+	2.1+[7]
WebM-VP8	4+	6+	9+[8]	10.6+	-	-	2.3

Tabla 2. Soporte de los tres *códecs* de vídeo del estándar HTML5 en los principales navegadores y la primera versión en la que se implementa.

Hay que señalar que los *códecs* de vídeo más conocidos son MPEG-4, AVC/H.264, VC-1, MPEG-2, H.263, VP8, Dirac y Theora. Este último es el códec de código abierto que deriva de VP3 de On2 mientras que Ogg Theora/Vorbis se convirtió en el primer formato libre, Google se hizo con On2 en 2010. Por otro lado, Vorbis es prometedor ya que tiene una calidad de codificación superior a Mp3 y similar a AAC; Dirac de BBC se busca en tecnología de vídeo más moderna (óndula) pero con menor eficiencia que Theora en la web, en 2007 se pensó en Theora / Vorbis como especificación HTML5 pero Apple criticó su inferior calidad (respecto a H.264) así como su incompatibilidad con móviles. Nokia y Microsoft se unieron a la crítica a Apple imponiendo el estándar H.264 con pago de derechos que después (2010) se negó por MPEG, además, al utilizar el mismo códec que Flash facilitaría la migración[9] [9]. Pero en 2010 Google anuncia el proyecto WebM (VP8 – Vorbis) para satisfacer a Apple, Microsoft y Nokia, es libre y creado para Android. WebM es un contenedor basado en el de Matroska con el códec VP8 de igual calidad a H.264 [16]. Actualmente, Apple solo funciona con MP4 H.264/AAC, Mozilla y Opera no son compatibles con este y sí con Ogg y WebM, Google Chrome es compatible con los tres. Microsoft admitirá VP8 cuando el usuario tenga

[7] Versiones anteriores requieren de JavaScript
[8] Si el usuario instala el códec VP8 en Windows
[9] Las diferencias por marketing se reflejan en el soporte de los códecs (vídeo – audio) de los principales navegadores del mercado en la tabla 2

que instalarlo en Windows pero Google sigue animando a sitios web como Wikipedia, Youtube o Vimeo a que utilizen WebM con VP8 [12].

A continuación una tabla con los *códecs* de audio compatibles con los principales navegadores en HTML5:

	WAV	**Vorbis**	**MP3**
Firefox	SI	SI	NO
Safari	SI	NO	SI
Opera	SI	SI	NO
Chrome	SI	SI	SI
Internet Explorer	NO	NO	SI

Tabla 2.1. Soporte de los tres *códecs* de audio del estándar HTML5 en los principales navegadores.

3.3 PRINCIPALES CÓDECS DEL ESTÁNDAR

3.3.1 MP4 (H.264)

Desarrollado por MPEG (*Moving Picture Experts Group*) y el VCEG (*Video Coding Experts Group*) de manera conjunta y basado en el formato de archivo QuickTime. Su idea inicial era la de crear un estándar con una buena calidad de imagen con tasas binarias de *bitrate* inferiores a los estándares anteriores [9] [11]. La versión de MP4 de solo audio (.m4a) es frecuentemente comprimida con ACC (compresión con pérdidas), muchos tipos de *códecs* de vídeo y audio pueden ser incrustados en el contenedor de MP4 a través de emisiones privadas, otros tipos de formato de compresión son MPEG-2 y MPEG-1. También puede contener metadatos en el

estándar del formato a través de una extensión XMP (Extensible Metadata Platform) [7].

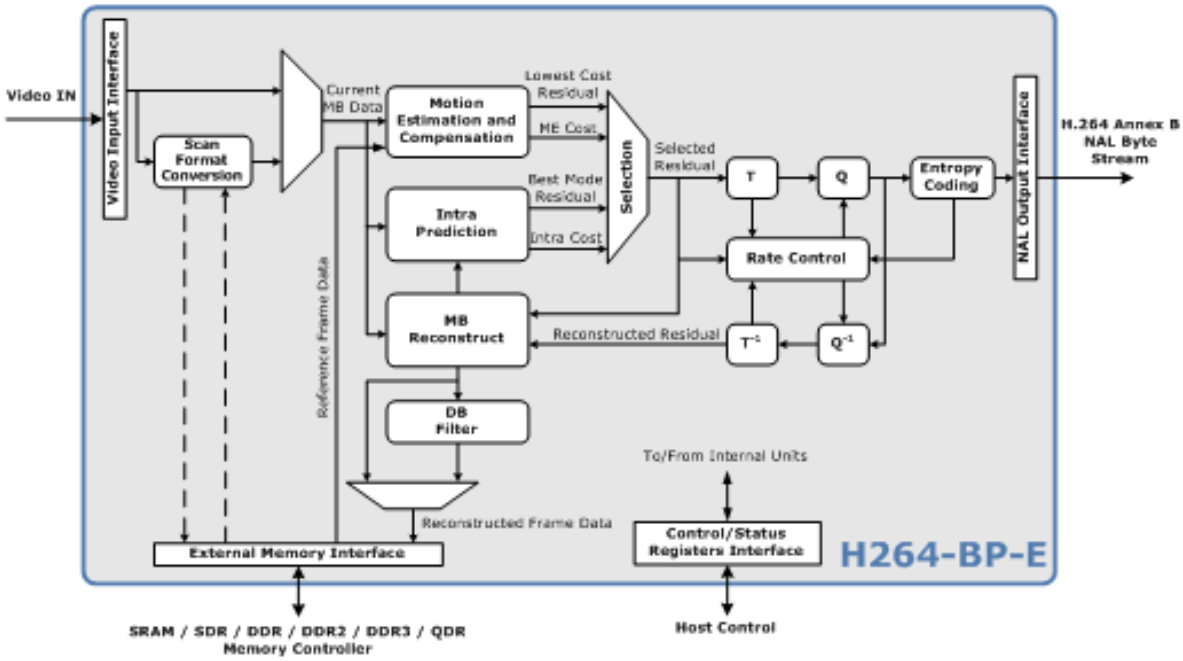

Figura 1. Funcionamiento del algoritmo de compresión H.264

El códec de vídeo del estándar de MP4 es H.264/MPEG-4 AVC (2003) y es uno de los más usados para grabación, codificación y distribución de vídeo. Presenta pérdida de información en la compresión aunque también es posible matemáticamente conseguir un algoritmo sin pérdidas (*lossless*) y *subsamplings* de *Chroma* de hasta YUV 4:2:2 o Y'CbCr 4:4:4, es uno de los aspirantes a convertirse en el códec único del estándar de internet aunque mantiene su guerra de marketing frente a OGG (Theora) y WebM (VP8) [11]. En el ST Microelectronics de Greater Noia (India) se ha investigado acerca de las características del algoritmo de codificación H.264 (Figura 1):

"The use of algorithms of H.264 HW, such as, motion estimation, intra prediction and rate control in developing VP8 HW encoder. This enables in reducing HW cost of VP8

Encoder and also development time/effort. The result shows that the proposed technique achieves the comparable quality and it is very simple and effective" [23].

No todos los navegadores estaban dispuestos a aceptar un códec único como planteaban los creadores de HTML5 tal y como se muestra en la Tabla 2. Apple, Microsoft y Nokia decidieron apostar por H.264, criticando a Theora por su inferior calidad y por la falta de compatibilidad con dispositivos móviles [6]. Sin embargo, su utilización requería el uso de derechos por lo que dejaba de ser una opción para el códec base libre de derechos de HTML5. H.264 ofrece una calidad mejor de codificación que Ogg Theora y cuenta con su nueva versión de H.265 que cuenta con una relación de compresión de datos que duplica a la de H.264 [16]. En otro apartado del artículo de Microelectronics de Greater Noia (India) se habla acerca de las implementaciones de movimiento en H.264 en comparación con VP8:

"After implementation of the same algorithms of H.264 for motion estimation, intra prediction & rate control in the VP8 reference encoder and then utilizing the proposed scheme for mapping of QP indexes between H.264 & VP8, the results are generated and compared with Google's reference encoder. In the experiments, several test sequences are encoded with different configurations by using a fixed QP to find out the actual bit-consumption. The graphs are generated between PSNR & Bitrate (average over 300 frames) with proposed scheme and VP8 reference encoder.

Four different sequences of different resolutions ate taken for comparison. The results that with the re-use of intra and inter prediction, the PSNR close to VP8 reference encoder could be achieved. These results can be further improved with slight modification in the intra/inter prediction" [23]

3.3.2 OGG (THEORA)

Desarrollado por la fundación Xiph.org[10] como código abierto convirtiéndose en el primer formato de códec de vídeo libre, es en realidad el códec de video Theora y el de audio Vorbis bajo un único contenedor OGG [8]. Ha sido propuesto en varias ocasiones como el códec base de unificación, Theora cuenta con la ventaja de que es menos complejo que sus rivales y con una necesidad de compatibilidad de hardware dedicado, esto lo convierte en una opción muy útil para dispositivos móviles [10]. El formato contenedor de OGG puede multiplexar un número independiente de emisiones de vídeo, audio, subtítulos y metadatos contando con un códec de vídeo (Theora) con pérdidas. Se ha incorporado desde 2007 a un gran número de librerías de software, sin embargo, existen otros que todavía no como es el caso de Sorenson Squeeze.

El *framework* del algoritmo OGG empieza con una cabecera llamada *Ogg Page* (Observar Figura 2) con la información y metadatos del archivo multimedia (versión, tipo de cabecera, posición, número de serie, número de secuencia de página o segmento de tramas de mapeo).

[10] http://downloads.xiph.org/releases/ogg

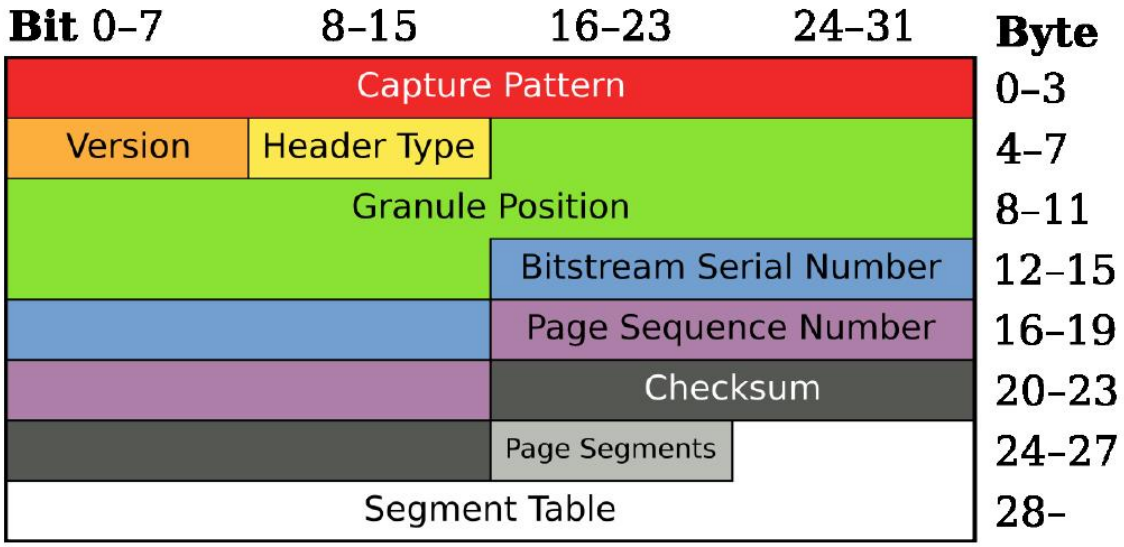

Figura 2. Capa del campo de la cabecera de metadatos OGG en la multiplexión del algoritmo (*Ogg Page*).

Los segmentos son un grupo de paquetes que crean un camino de datos unidos hacia el decodificador del algoritmo, cuando el contador del segmento llega a 255 (2^8) indica que el siguiente segmento es concatenado como una parte del mismo paquete, sin embargo, si el contador del segmento se encuentra entre 0-254 significa que ese segmento es el último del paquete tal y como muestra la Figura 3. A diferencia de MPEG-4, no existe un estándar para para introducir metadatos en un contenedor OGG, se realiza con implementaciones XML y RDF dentro del códec [9]. OGG también puede cambiar su códec de vídeo con un mapeo de tramas basado en otros algoritmos (Tarkin o Dirac).

Bit	Value	Flag	Page type
0	0x01	Continuation	The first packet on this page is a continuation of the previous packet in the logical bitstream.
1	0x02	BOS	Beginning Of Stream. This page is the first page in the logical bitstream. The BOS flag must be set on the first page of every logical bitstream, and must not be set on any other page.
2	0x04	EOS	End Of Stream. This page is the last page in the logical bitstream. The EOS flag must be set on the final page of every logical bitstream, and must not be set on any other page.

Figura 3. Fase del multiplexado OGG que indica en un campo de 8 bits el tipo de información que contiene.

El códec de vídeo Theora[11] (2008) está programado en C y cuenta con una licencia BSD por lo que Xiph ha intentado convertirlo en el códec estándar de Internet al no necesitar de pago de licencias a empresas o usuarios, Apple y Microsoft lo criticaron porque calificaban que su calidad es inferior a la de MP4 (H.264) [6]. Theora es un códec de compresión con pérdidas que se basa en la librería libtheora como implementación al formato contenedor OGG[12] y derivado del códec VP3 (On2 Technologies), tiene un *bitrate* variable basado en un esquema de compresión de vídeo DTC, también utiliza *chroma subsampling*, *intraframes*, frames predictivos y compensación de movimiento 8 a 8 de bloques DTC en su algoritmo. Sin embargo, no soporta *frames* bipredictivos (a diferencia de H.264)[13], entrelazado (i) o profundidades de bit mayores de 8 por componente (Figura 3).

Estas versiones de navegadores soportan OGG o Matroska (Theora): Mozilla Firefox 3.5+ y su versión móvil Fennec, Google Chrome 3.0.182.2+ (2009), Tizen, SeaMonkey 2.0+, Konqueror 4.4.2+, Opera 10.5+[14] y Midori. También existen *frameworks* que soportan el multimedia del códec: DirectShow con OpenCodecs, GStreamer[15], Phonon, QuickTime con Xiph QuickTime Components y Silverlight. Podemos observar una lista más detallada en las figuras 4 y 5:

[11] http://www.theora.org/
[12] También se puede implementar en el formato contenedor Matroska
[13] H.264 and Theora codecs comparison (http://compression.ru/video/codec_comparison/h264_2010)
[14] La versión 9.5 de Opera soporta el códec Theora con una construcción de vídeo experimental
[15] Soportado en módulo de Ffmpeg y basado en aplicaciones como Totem o Songbird

Name	Description	Operating Systems Supported		
		Unix-like	OS X	Windows
Firefogg [11]	A Firefox browser extension implementation of ffmpeg2theora	Yes	Yes	Yes
ffmpeg2theora [12]	A command-line program that transcodes video by decoding with FFmpeg and reencoding with libtheora to encode it	Yes	Yes	Yes
VLC	Can transcode to single-pass Theora 1.0 and optionally stream it	Yes	Yes	Yes
OggConvert	Transcodes supported media to Vorbis, Theora, or Dirac	Yes	?	Yes
FreeJ	"Video DJing" software that can encode to and stream Theora	Yes	Yes	?
Kdenlive	The video editor supplied with KDE	Yes	?	?
Pitivi	The video editor supplied with GNOME	Yes	?	?
LiVES	Video editing software for Linux. Can edit, encode and stream theora.	Yes	Yes	?
Thoggen	A GTK+ and GStreamer based DVD backup utility	Yes	?	?
HandBrake	Can output to Theora only with the Matroska container	Yes	Yes	Yes
Recordmydesktop	Records the screen to Ogg Theora with optional Vorbis audio	Yes	?	?

Figura 4. Ejemplos de sofware de terceros que soportan la codificación a través de la librería libtheora

Name	Description	Operating Systems Supported		
		Unix-like	OS X	Windows
VLC		Yes	Yes	Yes
Icecast		Yes	?	Yes
FreeCast	Peer-to-peer streaming. Written in Java	Yes	?	Yes
LiVES	Can stream ogg/theora/vorbis in realtime to a file or fifo.	Yes	Yes	?
Flumotion	Streaming media server.	Yes	?	?

Figura 5. Ejemplos de servidores de multimedia que pueden realizar *streaming* con Theora

Joe Crop, Alex Erwig y Unek Salvaraj valoran en su artículo "*Ogg Video Coding*" la importancia de este algoritmo para el futuro de las comunicaciones multimedia:

"Open source codecs such as the Ogg family are clearly important in todays world of ever-present multimedia systems and communications. Ogg Theora provides a

competitive opposition against the expensive, lawsuit-bound commercial codecs currently stealing the limelight such as H.264. This work has shown that Ogg Theora, even though it is technically poorer quality in comparison, can be an excelent alternative to H.264, especially with it's upcoming improvements. Ogg Theora is designed for future scalability with features such as the Ogg container format and integration with other open source projects such as Mozilla Firefox" [22].

3.3.3 WEBM (VP8)

Desarrollado por Google en 2010[16] para superar los problemas que planteaban Nokia, Microsoft y Apple respecto a Ogg Theora. Está basado en código abierto y se coloca en un formato contenedor derivado de Matroska con el códec de vídeo VP8, recientemente actualizado a VP9[17] que busca una compresión del 50% respecto al anterior pero manteniendo la misma calidad de vídeo, en audio cuenta con los *códecs* Vorbis y Opus [19]. VP8 tiene prácticamente la misma calidad de video que H.264 pero con un considerable descenso en el espacio por lo que es un serio candidato al códec base [6]. WebM es soportado por[18] Mozilla Firefox, Opera y Google Chrome; Internet Explorer 9+, Safari, Mac OS X o QuickTime exigen la instalación de un *plugin* de terceros para su uso aunque Google anunció en 2011 de que el proyecto WebM lanzaría *plugins* para ellos a través de la API <video> del estándar HTML5.

La librería ffvp8 ya se encuentra implementada para reproducción o codificación en Miro, Winamp, VLC, MPlayer, K-Multimedia Player, Ffmpeg[19], Haali Media Splitter o

[16] http://www.webmproject.org/

[17] Será utilizado a partir de 2015 como estándar en la plataforma de emisión de vídeo (http://www.youtube.com) como sustituto del formato FLV de Flash y H.264 de MPEG-4 (http://news.cnet.com/8301-1023_3-57584018-93/ googles-vp9-video-codec-nearly-done-youtube-will-use-it/)

[18] Ver Tabla 2 (Soporte de los *códecs* del estándar HTML5 en los principales navegadores)

[19] Puede codificar VP8 con el soporte de la librería libvpx pero ha sido sustituida por ffvp8

LiVES. Android tiene soporte para WebM desde la versión 2.3 (Gingerbread) y *streaming* desde la 4.0[20]. La licencia VP8 cuenta con aceleradores de hardware (RTL IP) para codificar y decodificar con un semiconductor para 1080p (hasta 60 *frames* por segundo), las compañías que ya han anunciado el soporte para esta tecnología son AMD, ARM, Broadcom, Intel, Qualcomm, Texas Instruments, Chip&Media, Nvidia, Rockchip, ZiiLABS, Ericsson y Huawei (consultar Figura 6).

Este fragmento es de un artículo de Jim Bankoski, Paul Wilkins, Yaowu Xu (Google Inc.) valorando la efectividad del códec VP8 y realizando una comparativa con MPEG-4 (H.264) en las dos librerías (libvpx y x264):

"*VP8 can make the best use of computation power in modern hardware for improving compression efficiency while maintaining fast decoding speed on majority devices connected to the web. The WebM Project libvpx reference encoder and x264 software encoder were used to produce the VP8 and H.264 files, respectively. For decoding, the latest FFmpeg decoder software (version SVN-r264000) with libavcodec (version 52.108.0) was used for both VP8 and H.264 in the test. By using the same software decoder, one can expect the level of optimization for performance to be similar for both VP8 and H.264; therefore the decoding speeds may reflect the intrinsic decoding complexity. The decoding speeds of VP8 encoded files are consistently faster, average around 30%, than those of H.264 encoded files at a similar bitrate across the two different hardware platforms."*[24]

[20] http://developer.android.com/guide/appendix/media-formats.html

El presente contempla una profunda división de los principales navegadores en su elección de un códec para sus diferentes plataformas[21]. Mozilla y Opera han mostrado su desinterés con MP4 H.264 ya que los derechos anuales son excesivos para ellos y para los usuarios y, además, porque rompe la idea de una Web con código abierto [4]. Ambos han implementado Ogg Theora y WebM en sus navegadores, Safari de Apple e Internet Explorer solo funcionan con MP4 H.264 y Google Chrome es compatible con los tres [8].

El códec VP8 es una extensión del VP7 y cuenta con el estándar RFC6386[22]; creado por On2 Technologies fue patentado por Google y lanzada la especificación a través de Creative Commons Attribution 3.0 License[23]. En principio, libvpx era la única librería de software capaz de codificar vídeo VP8 aunque un codificador basado en el *framework* x264 llamado xvp8 ha sido desarrollado por el equipo x264 con soporte no nativo para VP8, en 2010 se lanzó el codificador ffvp8 que era mucho más rápido que libvpx. Google también cuenta con un algoritmo de codificación de imagen, con las opciones con pérdidas o sin pérdidas, llamado WebP basado en la codificación *intraframe* de VP8 y con un formato contenedor basado en RIFF (*Resource Interchange File Format*).

[21] Observar el estudio de comparativa en la Tabla 2 de la página 22
[22] http://tool.ietf.org/html/rfc6386
[23] VP8 Bitstream Specification License (http://www.webmproject.org/license/bitstream/)

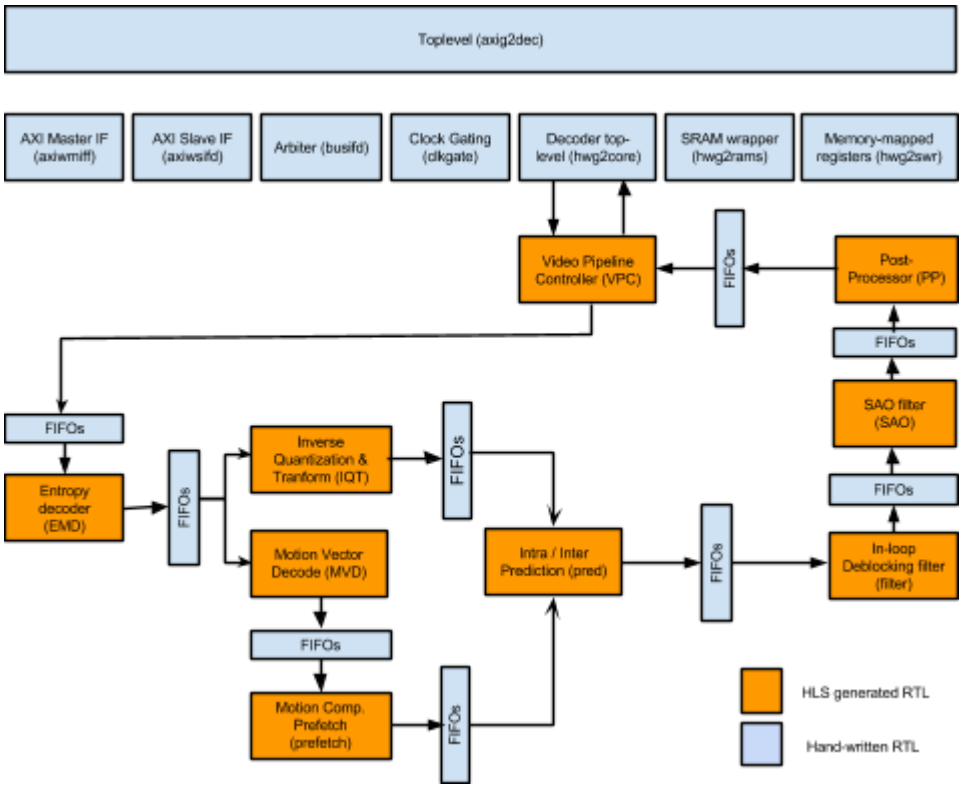

Figura 6. Funcionamiento del algoritmo de compresión VP8 junto a su aceleración de hardware RTL

Mientras H.264/MPEG-4 AVC requiere de licencias para patentes, Google ha lanzado todas las suyas de VP8 bajo una *Royalty-Free Public License* [16]. Como podremos apreciar en los resultados de las Figuras 8, 9, 13 y 14, VP8 ofrece la calidad más alta y su librería permite un modo donde los máximos recursos de la CPU mantienen por lo menos la velocidad exacta de codificación a la del tiempo real (*real time*), a diferencia de H.264, sin embargo, el código del algoritmo de VP8 ha sido criticado como incompleto al copiar y pegar del código de C.

4. ESTUDIO Y ANÁLISIS DE LA CODIFICACIÓN Y EMISIÓN DE VÍDEO SOBRE HTML5

4.1 MATERIAL Y SOFTWARE UTILIZADO

La investigación de codificación para emisión en HTML5 consiste en utilizar un fragmento de vídeo sin pérdidas por compresión para aplicar codificaciones con diferentes software y a distintos *bitrates* calculando sus tamaños finales así como sus tiempos de compresión para proceder a analizar los datos y elaborar conclusiones respecto a los tres *códecs* empleados [10] [17]. La secuencia de imágenes ha sido adquirida con la Blackmagic Cinema Camera que dispone de las siguientes características técnicas:

- *Shooting Resolution*: ProRes and DNxHD at 1920 x 1080
- *Active Sensor Size*: 15,81 mm x 8,88 mm
- *Dynamic Range*: 13 stops
- *Storage Type*: Removable 2.5” SSD
- SDI *Video Outputs*: 1 x 10 bit HD-SDI 4:2:2 *with choice of Film or Video Dynamic Range*. *When recording is set to 25p or 29.97p with SDI Overlays switched off, the SDI output format is 1080i50 and 1080i59.94 respectively.*

El equipo utilizado para el procesado de datos, mediciones y codificación es un AMD Phenom QuadCore, 2000 Mhz, de modelo HP Pavilion dv6 Notebook PC. Arquitectura ACPI X64-based PC (Mobile) AMD M785, AMD K10 con 2GB DDR3-1333 DDR3 SDRAM.

El Sistema Operativo utilizado es Microsoft Windows 7 Professional® SP2 con DirectX 11.0 además de las siguientes prestaciones:

- *Graphics*: ATI Mobility Radeon HD 5650 1024 MB
- *Storage*: Standard AHCI 1.0 Serial ATA 500 GB
- *Network*: Atheros AR9285 802.11b/g Wifi Adapter and Realtek RTL8168D/8111D PCI-E Gigabit Ethernet
- *Cache Memory*: L1 (512 Kb) L2 (2048 Kb)

El software utilizado para la codificación de los archivos es el VLC Media Player 2.1.3 RinceWind y SUPER v2014.build.60 descartando Sorenson Squeeze Premium 9.0.2.81 ya que todavía no ofrece soporte para codificación en .ogg. Realmente SUPER no es más que una interfaz gráfica para los *frameworks* Ffmpeg, MEncoder, MPlayer y otros, que normalmente se utilizan mediante consola de comandos, pero que de esta forma se pueden usar a través de una cómoda interfaz gráfica. Además, SUPER no necesita *códecs* para codificar, ya que vienen incluidos internamente.

El video utilizado para su codificación en los distintos parámetros que se explican en el capítulo 4 cuenta con las siguientes características:

Format: MPEG-4
Format Profile: Quick Time
Codec IQ: qt
File Size: 597 MB
Duration: 6s 298ms
Bitrate: 795 Mbps
Color Space: YUV
Width: 1920 *pixels*

Height: 1080 *pixels*

Aspect: 16:9

Chroma Subsampling: 4:2:2

Compression Mode: *Lossless*

Bits / (pixel * *frame*): 16.000

La idea inicial era trabajar con RAW y un *Chroma Subsampling* de 4:4:4 así como un perfil de *bitrate* superior a los 7950 Mbps empleados en este análisis para poder extender el estudio a las emergentes tecnologías de resolución superiores a 4K, pero este tipo de información conlleva problemas de procesamiento tanto en la tarjeta gráfica como en el microprocesador además de que la opción empleada también implica un modo de compresión sin pérdidas, requisito principal para la elección del vídeo original [17].

4.2 RESULTADOS EXPERIMENTALES

4.2.1 IDENTIFICACIÓN DE LOS PARÁMETROS A ANALIZAR

El primer paso para la comparación de los diferentes *códecs* de compresión es el de establecer parámetros con distintos codificadores, *frames* por segundo así como *bitrates* para obtener datos de tamaño y tiempo de codificación finales [14].

Por tanto, la comparativa de análisis estará dividida en tres bloques principales que corresponden a los diferentes softwares de codificación empleados para realizar la comparativa: VLC Media Player 2.1.3 y SUPER v2014.build.60. Dentro de estos tres bloques, el contenido del análisis se separa en cuatro subloques: tamaño en PAL 25, tamaño en NTSC 23,976, tiempo de codificación en PAL 25 y tiempo de codificación en NTSC 23,976. En estos subloques se asignan diferentes *bitrates* de codificación del

mismo vídeo original: 64, 240, 576, 1152, 1536, 2256, 3024, 4224, 5808 y 7950[24] kbps basados en los estándares de VLC y SUPER [7] [11].

4.2.2 MEDICIÓN DE DATOS

La principal característica de esta parte del análisis es una correcta sincronización para ajustar en milisegundos el tiempo empleado en la codificación, ya que no todos los software utilizan un contador de tiempo para sus trabajos. Por tanto, el contador externo sincroniza de manera muy precisa la ejecución de inicio y fin de la codificación.

La medición del tiempo se ha hecho en segundos y milisegundos mientras que la de tamaño se ha realizado en megabytes (MB), con doble decimal si es mayor que uno o con triple decimal si es menor.

4.2.3 CREACIÓN DE RATIOS

Esta parte resulta más importante de lo que parece ya que éste valor es fundamental para realizar valoraciones y estimar eficiencia de compresión de los códecs empleados. Es decir, si solamente se presentaran los datos empíricos del tamaño y el tiempo de codificación en los diferentes bitrates serían, como única opción, para el vídeo original que se ha empleado en las distintas mediciones, pero solamente para ese vídeo con sus características exclusivas.

Si queremos aplicar los valores de estas mediciones a la compresión de cualquier vídeo debemos crear un ratio de compresión que generalize el uso de esa medición a

[24] Estas tasas de transferencia tan altas ya marcan un presente en el sector de las Telecomunicaciones. Cadenas como RTVE ya apuestan por estos *bitrates* para sus emisiones regulares (Ultra HD 4K) como podemos observar en http://videopopular.es/not/2083/rtve-prueba-la-tdt-en-ultra-alta-definicion/.

cualquier vídeo. De esta manera podremos decidir que tipo de compresión es más favorable para cualquier caso, dependidendo si la prioridad es el tiempo de codificación o el tamaño final del archivo, como puede resultar el caso de un broadcast a través de Internet. La tasa de compresión para el cálculo del ratio se define como el cociente entre el tamaño del archivo de video no comprimido y el comprimido.

Tasa compresión = (Tamaño sin comprimir / Tamaño comprimido). Si el archivo original es de 597Mb y se pasa a 8Mb tiene una tasa de compresión 597/8 = 74,625. También se suele indicar de forma explícita como 597:8 (que se lee como una compresión de 597 a 8) [4] [10] [15].

4.3. RESULTADOS DEL ANÁLISIS

4.3.1 TAMAÑOS FINALES DE CODIFICACIÓN EN PAL 25 Y NTSC 23,976 SEGÚN DIFERENTES *BITRATES* UTILIZANDO EL SOFTWARE VLC

SIZE. PAL 25	**MP4 (H.264)**
64 Kbps	-
240	0,381 MB (1194:1)
576	0,613 MB (597:1)
1152	1,18 MB (597:1)
1536	1,56 MB (597:2)
2256	1,57 MB (597:2)
3024	2,26 MB (597:2)
4224	4,21 MB (597:4)
5808	5,79 MB (597:6)
7950	7,98 MB (597:8)

Tabla 3. Tamaños finales de codificación (MB) en PAL 25 según diferentes *bitrates*, utilizando el software VLC Media Player 2.1.3 RinceWind y la compresión con el formato .mp4 y el códec H.264.

SIZE. PAL 25	OGG (Theora)
64 Kbps	-
240	1,55 MB (597:2)
576	1,55 MB (597:2)
1152	1,66 MB (597:2)
1536	1,87 MB (597:2)
2256	2,47 MB (597:2)
3024	3,15 MB (597:3)
4224	4,31 MB (597:4)
5808	5,55 MB (597:5)
7950	7,88 MB (597:8)

Tabla 4. Tamaños finales de codificación (MB) en PAL 25 según diferentes *bitrates*, utilizando el software VLC Media Player 2.1.3 RinceWind y la compresión con el formato .ogg y el códec Theora.

SIZE. PAL 25	WebM (VP8)
64 Kbps	-
240	1,47 MB (597:1)
576	1,52 MB (597:2)
1152	1,55 MB (597:2)
1536	1,62 MB (597:2)
2256	1,78 MB (597:2)
3024	2,02 MB (597:2)
4224	2,60 MB (597:3)
5808	3,36 MB (597:3)
7950	4,83 MB (597:5)

Tabla 5. Tamaños finales de codificación (MB) en PAL 25 según diferentes *bitrates*, utilizando el software VLC Media Player 2.1.3 RinceWind y la compresión con el formato .webm y el códec VP8.

La evolución de la Tabla 3, tal y como se puede observar en la Figura 8, es peculiar por varias razones. Ofrece la media aritmética más baja de los tres *códecs* en la compresión con .mp4 (H.264) dentro del sistema PAL 25 pero llama la atención que obtenga los mejores resultados de tamaño en los *bitrates* más bajos y los tamaños más

altos en los *bitrates* superiores. Son características del algoritmo que define el comportamiento del códec H.264 así como el tipo de librería utilizada por el software para la compresión ya que VLC utiliza una genérica mientras que SUPER la x264 propia de ffmpeg. VLC no permite, a diferencia de SUPER, las compresiones con *bitrates* bajos como 64 Kbps por lo que se trabaja a partir de 240. La calidad final es muy baja, con un pixelado evidente en la imagen, por ello es recomendable para dispositivos de poca resolución y para vídeos de corta duración que no requieran de calidad o que se cuente con un Ancho de Banda muy bajo en la conexión.

Si obsevamos la Figura 9 que muestra la evolución de la Tabla 4, la línea roja muestra el comportamiento del códec Theora en PAL-VLC y, a diferencia del software SUPER, sustituye a .webm como el formato más equilibrado en cuanto a tamaño final y también en tiempo de codificación como veremos posteriormente (ya que este está mucho más igualado). Podemos afirmar que en archivos de corta duración es más eficiente utilizar el formato .ogg con VLC que presenta librería de *coding* genérica, a diferencia de SUPER que utiliza la x264 y donde .webm se impone como la mejor opción. Aun así, los tiempos son muy cercanos por la reducida duración aunque cuando se realice con vídeos de mayor duración sí que puede resultar decisivo, por lo que pienso que .ogg es una opción mucho más eficiente en calidad y tamaño en estos parámetros que la planteada por .mp4 (H.264).

Respecto a .webm (VP8) en la Tabla 5 es curioso observar el comportamiento de este códec en este tipo de compresión concreta y con PAL 25ya que ofrece los tamaños más altos en PAL 25 mientras que en NTSC 23,976 y en todas las mediciones con SUPER encuentra equilibrio entre tamaño y *coding* time unido a una calidad final muy alta. La excepción la presenta este número de *frames* por segundo con VLC e incluso aumenta su porcentaje de tamaño a medida que van subiendo los *bitrates*, eso sí, como observaremos a continuación se podrá comprobar como los tiempos de codificación

son prácticamente iguales que sus dos rivales. Por tanto, no aconsejo el uso de este códec en PAL 25 con VLC para vídeos de corta duración sino que, en su lugar, es mucho más acertado el uso del software SUPER con librerías ffmpeg y con un equilibrio-calidad altísimo.

SIZE. NTSC 23,976	**MP4 (H.264)**
64 Kbps	-
240	0,342 MB (1194:1)
576	0,473 MB (1194:1)
1152	0,922 MB (597:1)
1536	1,18 MB (597:1)
2256	1,70 MB (597:2)
3024	2,26 MB (597:2)
4224	3,14 MB (597:3)
5808	4,11 MB (597:4)
7950	6,01 MB (597:6)

Tabla 6. Tamaños finales de codificación (MB) en NTSC 23,976 según diferentes *bitrates*, utilizando el software VLC Media Player 2.1.3 RinceWind y la compresión con el formato .mp4 y el códec H.264.

SIZE. NTSC 23,976	**OGG (Theora)**
64 Kbps	-
240	1,54 MB (597:2)
576	1,54 MB (597:2)
1152	1,56 MB (597:2)
1536	1,59 MB (597:2)
2256	1,70 MB (597:2)
3024	2,28 MB (597:2)
4224	3,19 MB (597:3)
5808	4,15 MB (597:4)
7950	6,00 MB (597:6)

Tabla 7. Tamaños finales de codificación (MB) en NTSC 23,976 según diferentes bitrates, utilizando el software VLC Media Player 2.1.3 RinceWind y la compresión con el formato .ogg y el *códec* Theora.

SIZE. NTSC 23,976	**WebM (VP8)**
64 Kbps	-
240	1,46 MB (597:1)
576	1,52 MB (597:2)
1152	1,55 MB (597:2)
1536	1,60 MB (597:2)
2256	1,78 MB (597:2)
3024	2,02 MB (597:2)
4224	2,61 MB (597:3)
5808	3,36 MB (597:3)
7950	4,85 MB (597:5)

Tabla 8. Tamaños finales de codificación (MB) en NTSC 23,976 según diferentes *bitrates*, utilizando el software VLC Media Player 2.1.3 RinceWind y la compresión con el formato .webm y el códec WebM.

Respecto a los *framerates* de NTSC 23,976 se obtienen resultados más cercanos en los tres *códecs*. A diferencia de la codificación con SUPER, el formato .mp4 no ofrece unos tamaños tan reducidos pero mejora sensiblemente la calidad de la imagen, a través de un pixelado más suave con una compresión temporal GOP del archivo de vídeo. Un archivo MP4, OGG o WebM está compuesto de unas secuencias cíclicas llamadas GOP (*Group of Pictures*) que engloban cierto número de fotogramas, normalmente 15.

Aunque no es necesario que estos grupos estén formados siempre por el mismo número de fotogramas, se suele asignar el mismo patrón GOP en todo el archivo. En los casos en que la secuencia GOP no varía en todo el archivo es frecuente indicarla tan sólo una vez al principio pero he llegado a la conclusión en el análisis que para que un archivo de vídeo MP4 sea 100% compatible con DVD deberemos seleccionar en el programa de codificación la opción de "Cerrar todas las secuencias GOP" para que se incluya un encabezado de secuencia antes de cada GOP. Muchos reproductores no tendrán problema para reproducir un vídeo con encabezado GOP (GOP *Header*) tan

sólo al comienzo del vídeo, pero lo recomendable es indicar al compresor que añada un encabezado GOP antes de cualquier secuencia.

Los GOP están formados por tres grupos distintos de fotogramas:

- **I-*picture*** *(Intra frames*) (imagen-I, cuadros internos): Son los únicos estrictamente necesarios, cada cuadro es comprimido con un tipo de compresión llamada "*Intra frame* DCT *Coding*" (codificación interna de cuadros DCT por transformación discreta del coseno) dividiendo la imagen en grupos de 8x8 píxeles. Si nuestro GOP tan sólo contara con cuadros-I tendríamos una secuencia de JPGs, el llamado *Motion* JPEG o MJPG.

- **P-*picture* y B-*picture*** (*Predictive frames and Bidirectionally Predictive Frames*) (Imagen-P e Imagen-B - Cuadros de predicción y cuadros de predicción bidireccionales): En el caso del vídeo de análisis, con una resolución de 1920x1080p (FullHD), necesitamos representarlo con un total de 2.073.600 píxeles por cada cuadro. Si tuviésemos grabada una puesta de sol, por ejemplo, lo único que cambiaría entre cuadro y cuadro es el sol y además mínimamente. Puesto que el sol se mueve a velocidad constante (pongamos un cambio hipotético es de 73.600 píxeles entre un cuadro y otro) no es necesario almacenar la información de los 2.000.000 de píxeles restantes. Esa es la función de las imágenes-P y las imágenes-B, en lugar de estar compuestas por los 2.073.600 píxeles lo están tan sólo por los 73.600 cambiantes. Los cuadros-P analizan los cambios con respecto a cuadros I u otros cuadros P anteriores, mientras que los cuadros-B pueden analizar los cambios de cuadros-P anteriores y posteriores ("B" de bidireccionales) alcanzando los mayores grados de compresión. En el caso del vídeo de estudio es diferente ya que existe movimiento de la POV HD *Camera* y los píxeles suelen cambiar a una velocidad muy rápida, por lo que la codificación con secuencia GOP no sería tan útil como en otros casos.

Para MP4, OGG y WebM la secuencia GOP contiene un máximo de 15 cuadros tal y como muestra la Figura 7 (también recomendada para formatos de vídeo en DVD y CD) y normalmente suele ser 1 cuadro I, 4 cuadros P y 2 cuadros B entre cuadros P con el siguiente orden:

I BB P BB P BB P BB P BB

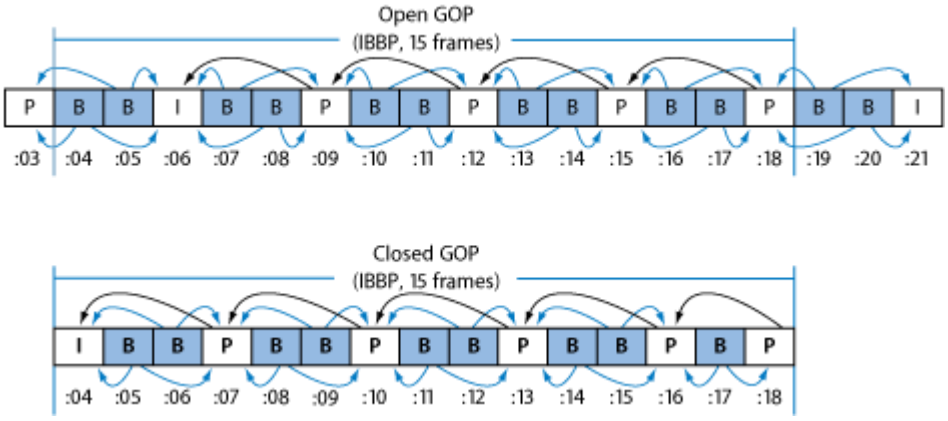

Figura 7. Ejemplo de secuencia GOP de 15 cuadros para MPEG-4, OGG y WebM

De vuelta a las mediciones con NTSC 23,976 - VLC hay que destacar que MP4, pese a la idéntica igualdad en la mayoría de los *bitrates*, empieza a conseguir tamaños menores en las últimas muestras de Kbps (mirar Tabla 6) por lo que, lógicamente, a mayores duraciones los tamaños crecerán respecto a .ogg y .webm. Por esa razón, se aconsejan estos dos últimos para la codificación con estos parámetros concretos y, especialmente, para vídeos de duraciones más altas. Dentro de los seis segundos de duración del vídeo origen hay que destacar que la calidad ofrecida por .mp4 y .ogg es inferior al algoritmo de Google.

El equilibrio alcanzado por .ogg (Theora) en las mediciones para PAL 25 se encuentra muy cercano a los otros dos *códecs* en NTSC con valores prácticamente idénticos por lo que sería difícil decidir cuál es más adecuado para estos parámetros concretos, sin embargo, hay un factor final que es calidad y que, como en casi todas las mediciones de la tabla 7 beneficia al códec de Google que ofrece la mejor relación de los tres factores (tamaño – tiempo de codificación – calidad final). Respecto a VP8, y tal como he dicho observamos en la Tabla 8, presenta el curioso inconveniente de su codificación en VLC con PAL 25 donde obtiene los tamaños finales más altos pero, en NTSC, se iguala absolutamente con los otros códecs con la importante diferencia de que su calidad final no es igualada en ningún momento por .ogg y .webm. Por estas razones, el formato .webm es el más adecuado tanto para vídeos de corta como de larga duración (debido a la tendencia de sus gráficas) como podemos observar en la Figura 9 con la excepción de VLC-PAL 25 donde .ogg se convierte en la opción más favorable.

4.3.2 TIEMPOS DE CODIFICACIÓN EN PAL 25 Y NTSC 23,976 SEGÚN DIFERENTES BITRATES UTILIZANDO EL SOFTWARE VLC

CODING TIME. PAL 25	MP4 (H.264)
64 Kbps	-
240	13s 080ms
576	13s 950ms
1152	17s 020ms
1536	18s 790ms
2256	22s 400ms
3024	28s 960ms
4224	31s 890ms
5808	33s 220ms
7950	36s 200ms

Tabla 9. Tiempos finales de codificación (ms) en PAL 25 según diferentes *bitrates*, utilizando el software VLC Media Player 2.1.3 RinceWind y la compresión con el formato .mp4 y el códec H.264.

CODING TIME. PAL 25	OGG (Theora)
64 Kbps	-
240	20s 150ms
576	20s 770ms
1152	21s 440ms
1536	21s 480ms
2256	21s 830ms
3024	23s 680ms
4224	25s 280ms
5808	28s 670ms
7950	31s 260 ms

Tabla 10. Tiempos finales de codificación (ms) en PAL 25 según diferentes *bitrates*, utilizando el software VLC Media Player 2.1.3 RinceWind y la compresión con el formato .ogg y el códec Theora.

CODING TIME. PAL 25	WebM (VP8)
64 Kbps	-
240	36s 100ms
576	36s 820ms
1152	37s 570ms
1536	38s 620ms
2256	40s 310ms
3024	42S 310ms
4224	47s 190ms
5808	50s 060ms
7950	54s 940ms

Tabla 11. Tiempos finales de codificación (ms) en PAL 25 según diferentes *bitrates*, utilizando el software VLC Media Player 2.1.3 RinceWind y la compresión con el formato .webm y el códec VP8.

Respecto a los tiempos de codificación (*Coding Time*) en PAL 25 con el software VLC, el formato .mp4, tal y como observamos en la Figura 10 y la Tabla 9, empieza con los tiempos de codificación más bajos (también con la calidad más baja) para dar paso a una continua y lógica subida donde aumenta ligeramente su calidad y también sus *Time* Coding, sin embargo, muy lejos de los obtenidos por el códec .webm. Podemos

destacar a .ogg como el formato más equilibrado de todos los analizados con estos parámetros tanto en tamaño como en tiempo de compresión. Sin embargo, el usuario debe decidir según sus intereses aquellas características que más le convengan en PAL 25 con VLC: si se trata de velocidad de codificación el más acertado es .mp4, si es por tamaño mejor .webm mientras que si se busca un equilibrio de todo sería .ogg el elegido. Aun así, dentro de las pequeñas diferencias entre ellos en tiempos tan bajos .webm es el elegido nuevamente por su calidad final.

De todas maneras, Theora es una gran opción (comparar Tabla 10) para usuarios comprometidos con licencias libres y que busquen un equilibrio de todos estos factores dentro de los límites que imponen las librerías de *coding* de VLC y el sistema PAL. Sin duda alguna, el algoritmo de vídeo de VP8 supuso una revolución que todavía puede agrandar su nueva versión VP9. Como vemos en la Tabla 11, VP8 ofrece equilibrio en tiempos de codificación (más alto que los demás en su media) así como de tamaños finales como sus dos rivales pero, especialmente, destaca por su alta calidad final lo que decanta su uso respecto al de los demás. Vuelvo a repetir que VLC no permite codificaciones por debajo de los 240 Kbps pero en *bitrates* menores las mediciones suelen igualarse.

CODING TIME. NTSC 23,976	**MP4 (H.264)**
64 Kbps	-
240	11s 030ms
576	14s 100ms
1152	14s 510ms
1536	15s 310ms
2256	19s 270ms
3024	19s 700ms
4224	22s 980ms
5808	25s 280ms
7950	29s 130ms

Tabla 12. Tiempos finales de codificación (ms) en NTSC 23,976 según diferentes *bitrates*, utilizando el software VLC Media Player 2.1.3 RinceWind y la compresión con el formato .mp4 y el códec H.264.

CODING TIME. NTSC 23,976	**OGG (Theora)**
64 Kbps	-
240	17s 010ms
576	17s 120ms
1152	17s 440ms
1536	17s 510ms
2256	17s 840ms
3024	17s 960ms
4224	20s 150ms
5808	21s 040ms
7950	24s 290ms

Tabla 13. Tiempos finales de codificación (ms) en NTSC 23,976 según diferentes *bitrates*, utilizando el software VLC Media Player 2.1.3 RinceWind y la compresión con el formato .ogg y el códec Theora.

CODING TIME. NTSC 23,976	**WebM (VP8)**
64 Kbps	-
240	36s 030ms
576	36s 650ms
1152	38s 640ms
1536	39s 000ms
2256	42s 610ms
3024	42s 960ms
4224	46s 890ms
5808	49s 850ms
7950	53s 520ms

Tabla 14. Tiempos finales de codificación (ms) en NTSC 23,976 según diferentes *bitrates*, utilizando el software VLC Media Player 2.1.3 RinceWind y la compresión con el formato .webm y el códec VP8.

Respecto a los tiempos de codificación de VLC en NTSC 23,976 con el códec H.264 en la Tabla 12, prácticamente se mantiene igual que la codificación en PAL 25, reducido ligeramente por la pérdida de un *frame* para decodificar en cada segundo y con una tendencia regular en su subida. Exactamente en la Tabla 13 que en la 10, .ogg mantiene su tendencia regular en los tiempos de codificación con la diferencia de que

sus tamaños finales se imponen como los más altos en los *bitrates* superiores, es su único problema al igual que una pequeña diferencia de calidad con el códec de Google. Además, .ogg presenta el problema de la falta de soporte de su algoritmo en navegadores (es soportado en Opera, Firefox y Chrome solamente). El formato .webm eleva el número de segundos en el tiempo de codificación respecto a los otros dos, sin embargo, su calidad le permite ser la mejor opción para compresiones de vídeos de menor duración (Tabla 14). En duraciones mayores puede seguir la tendencia de incremento del tiempo en su codificación y, pese a la excelente calidad, puede resultar aconsejable usar la segunda mejor opción que es la de Theora. Mucho mejor que esto, para mantener equilibrio de tamaño, tiempo de codificación y calidad final excelente es usar el códec .webm (VP8) en PAL o NTSC con el software SUPER y sus librerías x264 de ffmpeg antes que la opción genérica de VLC.

4.3.3 TAMAÑOS FINALES DE CODIFICACIÓN EN PAL 25 Y NTSC 23,976 SEGÚN DIFERENTES *BITRATES* UTILIZANDO EL SOFTWARE SUPER

SIZE. PAL 25	**MP4 (H.264)**
64 Kbps	0,158 MB (1194:1)
240	0,214 MB (1194:1)
576	0,465 MB (1194:1)
1152	0,855 MB (597:1)
1536	1,09 MB (597:1)
2256	1,57 MB (597:2)
3024	2,05 MB (597:2)
4224	2,78 MB (597:3)
5808	3,54 MB (597:4)
7950	5,03 MB (597:5)

Tabla 15. Tamaños finales de codificación (MB) en PAL 25 según diferentes *bitrates*, utilizando el software SUPER v2014.build.60 y la compresión con el formato .mp4 y el códec H.264.

SIZE. PAL 25	OGG (Theora)
64 Kbps	0,064 MB (1194:1)
240	0,243 MB (1194:1)
576	0,579 MB (597:1)
1152	0,970 MB (597:1)
1536	1,20 MB (597:1)
2256	1,76 MB (597:2)
3024	2,38 MB (597:2)
4224	3,32 MB (597:3)
5808	4,30 MB (597:4)
7950	6,15 MB (597:6)

Tabla 16. Tamaños finales de codificación (MB) en PAL 25 según diferentes *bitrates*, utilizando el software SUPER v2014.build.60 y la compresión con el formato .ogg y el códec Theora.

SIZE. PAL 25	WebM (VP8)
64 Kbps	0,575 MB (597:1)
240	0,619 MB (597:1)
576	0,745 MB (597:1)
1152	1,23 MB (597:1)
1536	1,63 MB (597:2)
2256	2,40 MB (597:2)
3024	3,20 MB (597:3)
4224	4,46 MB (597:4)
5808	5,95 MB (597:6)
7950	8,43 MB (597:8)

Tabla 17. Tamaños finales de codificación (MB) en PAL 25 según diferentes *bitrates*, utilizando el software SUPER v2014.build.60 y la compresión con el formato .webm y el códec VP8.

Por otra parte, se encuentran las mismas mediciones realizadas con el software SUPER, tal y como se puede observar en la Tabla 15 y la Figura 8, donde se utiliza el códec H.264 en PAL 25, el aumento del tamaño es progresivo y los ratios casi exponenciales (1-2-4-6-8) con la peculiaridad de la codificación en 240 Kbps donde el ahorro de espacio es considerable (uno de los ratios más pronunciados con 1194:1) pero la calidad es muy baja. Este tipo de codificación es adecuada para sistemas de visionado

europeos excepto zonas SECAM como Francia con un número de *frames* por segundo diferente y enfocado a emisiones en Internet con disponibilidad muy baja de Ancho de Banda. En el mismo *Frame Rate* de SUPER encontramos el formato .ogg en la Tabla 16 que, al igual que en .mp4, no existe soporte de codificación para 64 Kbps o menos, los ratios de compresión no son tan pronunciados como en .mp4 pero la calidad final del vídeo es mejor que el anterior por lo que es adecuado para emisiones *streaming* de Internet pero con Ancho de Banda muy bajo o bajo. La codificación a 7950 Kbps ofrece una calidad muy aceptable pero .ogg solamente se podrá reproducir en los navegadores Opera, Mozilla Firefox o Google Chrome. VP8 es, sin duda, la opción de compresión más equilibrada y que ofrece una mayor calidad para los reducidos ratios de los tres *códecs*. Tampoco recibe soporte en SUPER de *bitrates* iguales o inferiores a 64 Kbps pero destaca la homogeneidad y evolución de sus ratios (1-2-3-5) de la Tabla 17 con una importante reducción de espacio en los *bitrates* más altos y con una calidad superior a sus dos competidores por lo que es perfecto para su utilización en zonas de corriente eléctrica y visionado PAL con cualquier tipo de Ancho de Banda y una oferta de *streaming* para vídeos de larga duración como pueden ser películas o series.

SIZE. NTSC 23,976	**MP4 (H.264)**
64	0,156 MB (1194:1)
240	0,215 MB (1194:1)
576	0,467 MB (1194:1)
1152	0,858 MB (597:1)
1536	1,09 MB (597:1)
2256	1,59 MB (597:2)
3024	2,09 MB (597:2)
4224	2,80 MB (597:3)
5808	3,56 MB (597:4)
7950	5,12 MB (597:5)

Tabla 18. Tamaños finales de codificación (MB) en NTSC 23,976 según diferentes *bitrates*, utilizando el software SUPER v2014.build.60 y la compresión con el formato .mp4 y el códec H.264.

SIZE. NTSC 23,976	OGG (Theora)
64	0,060 MB (1194:1)
240	0,246 MB (1194:1)
576	0,588 MB (597:1)
1152	0,918 MB (597:1)
1536	1,16 MB (597:1)
2256	1,69 MB (597:2)
3024	2,27 MB (597:2)
4224	3,16 MB (597:3)
5808	4,13 MB (597:4)
7950	5,94 MB (597:6)

Tabla 19. Tamaños finales de codificación (MB) en NTSC 23,976 según diferentes *bitrates*, utilizando el software SUPER v2014.build.60 y la compresión con el formato .ogg y el códec Theora.

SIZE. NTSC 23,976	WebM (VP8)
64	0,618 MB (597:1)
240	0,618 MB (597:1)
576	0,710 MB (597:1)
1152	0,984 MB (597:1)
1536	1,17 MB (597:1)
2256	1,72 MB (597:2)
3024	2,29 MB (597:2)
4224	3,17 MB (597:3)
5808	4,47 MB (597:4)
7950	6,07 MB (597:6)

Tabla 20. Tamaños finales de codificación (MB) en NTSC 23,976 según diferentes *bitrates*, utilizando el software SUPER v2014.build.60 y la compresión con el formato .webm y el códec VP8.

Tal y como se puede observar en la Tabla 18 y la Figura 12 los valores en tamaño final de .mp4 en NTSC 23,976 de SUPER son muy parecidos a los niveles de compresión en PAL pero reduce ligeramente sus ratios superiores al contar con un menor número de *frames* por segundo. Al igual que su análisis en PAL, este tipo de codificación es óptima para la emisión de vídeos de corta duración con Anchos de Banda muy bajos. El códec Theora, al igual que en .mp4, guarda una relación muy estrecha respecto a la

codificación en PAL con una consecuente reducción de espacio en los *bitrates* más altos debido también al menor número de fotogramas por segundo, aunque, es de destacar que a partir de los *bitrates* de más de 3000 Kbps de la Tabla 19 la proporción línea de espacio aumenta en un 28%. Por tanto, esta codificación es adecuada para *streaming* de vídeo con Ancho de Banda muy bajo o bajo, también para comprimir mucha información en poco espacio al igual que .mp4 pero la calidad es pobre. . En VP8, los resultados finales de tamaño son prácticamente idénticos a la compresión de este algoritmo en PAL manteniendo íntegramente una calidad igual a él (Tabla 20). Por tanto, es perfecto para la compresión y reproducción (doméstico u online) de vídeos de larga duración con un ratio más efectivo de codificación para zonas de visionado en NTSC o en dispositivos digitales. Al observar la Figura 13 apreciamos el incremento de tamaño en *bitrates* más bajos y el considerable descenso en los más altos, lo que garantiza siempre un tamaño final bajo y con calidad media-alta.

4.3.4 TIEMPOS DE CODIFICACIÓN EN PAL 25 Y NTSC 23,976 SEGÚN DIFERENTES BITRATES, UTILIZANDO EL SOFTWARE SUPER

CODING TIME. PAL 25	MP4 (H.264)
64 Kbps	47s 590ms
240	57s 500ms
576	41s 140ms
1152	41s 400ms
1536	42s 960ms
2256	50s 410ms
3024	45s 070ms
4224	46s 180ms
5808	53s 570ms
7950	54s 340ms

Tabla 21. Tiempos finales de codificación (ms) en PAL 25 según diferentes *bitrates*, utilizando el software SUPER v2014.build.60 y la compresión con el formato .mp4 y el códec H.264.

CODING TIME. PAL 25	OGG (Theora)
64 Kbps	1m 18s 160ms
240	1m 10s 440ms
576	1m 23s 840ms
1152	1m 19s 860ms
1536	1m 19s 880ms
2256	1m 16s 300ms
3024	1m 25s 170ms
4224	1m 24s 220ms
5808	1m 34s 890ms
7950	1m 36s 250ms

Tabla 22. Tiempos finales de codificación (ms) en PAL 25 según diferentes *bitrates*, utilizando el software SUPER v2014.build.60 y la compresión con el formato .ogg y el códec Theora.

CODING TIME. PAL 25	WebM (VP8)
64 Kbps	51s 870ms
240	47s 040ms
576	46s 720ms
1152	57s 930ms
1536	1m 01s 560ms
2256	1m 00s 270ms
3024	1m 06s 110ms
4224	1m 02s 800ms
5808	1m 11s 520ms
7950	1m 15s 310ms

Tabla 23. Tiempos finales de codificación (ms) en PAL 25 según diferentes *bitrates*, utilizando el software SUPER v2014.build.60 y la compresión con el formato .webm y el códec VP8.

En relación a la Figura 8 y observando en este caso la Tabla 21, podemos entender que los tiempos de compresión en SUPER del códec .mp4 son menores y con tamaños también menores en comparación con los otros dos *códecs*. El gran problema es la falta de calidad en pro del ahorro en el almacenamiento y el tiempo invertido en la codificación, datos que personalmente no creo que sean tan relevantes dado que las diferencias de espacio y tiempo es de escasos Kbs y ms en un momento en el que estos

dos parámetros no son tan importantes debido al aumento tecnológico de las capacidades de almacenamiento y procesamiento. Es decir, la cantidad de tiempo invertido y la reducción del tamaño finales son insignificantes comparado a algo si e importante como es la calidad final, una calidad que con este códec y con estos *bitrates* resulta muy baja.

Sin duda, y comparando la Figura 13 la codificación PAL con el formato libre .ogg es el que más tiempo implica. La calidad es sensiblemente mayor a .mp4 con las mismas características pero no a .webm que aparte de ofrecer más calidad de vídeo en pixelado proporciona un equilibrado tiempo de codificación. Al igual que habíamos afirmado en la Tabla 19, este tipo de compresión es perfecta para vídeos de corta duración con Anchos de Banda muy bajos o bajos. Sin duda, y comparando la Figura 13 la codificación PAL en SUPER con el formato libre .ogg es el que más tiempo implica (Tabla 22). La calidad es sensiblemente mayor a .mp4 con las mismas características pero no a .webm que aparte de ofrecer más calidad de vídeo en pixelado proporciona un equilibrado tiempo de codificación. Al igual que habíamos afirmado en la Tabla 19, este tipo de compresión es perfecta para vídeos de corta duración con Anchos de Banda muy bajos o bajos. Personalmente, la mejor opción de compresión para vídeos de corta y larga duración que buscan su difusión en la red con el consecuente consumo de Ancho de Banda multiplexado, dejando aparte las nuevas aportaciones de MPEG y Google (H.265 y VP9). Al igual que en los tamaños finales de PAL, VP8 ofrece un equilibrio entre tamaño final y tiempo de codificación respecto a sus otros dos rivales. Aparte de esta peculiaridad, proporciona una calidad bastante mayor, fácilmente comparable en la Tabla 23 frente a la 21 y la 22, que los otros dos por lo que es perfecto para convertirse es un estándar de emisión de vídeo en Internet, con el gran problema de la guerra de marketing entre las diferentes empresas.

CODING TIME. NTSC 23,976	MP4 (H.264)
64 Kbps	56s 890ms
240	47s 220ms
576	48s 300ms
1152	43s 070ms
1536	42s 530ms
2256	43s 620ms
3024	44s 500ms
4224	52s 920ms
5808	48s 320ms
7950	50s 670ms

Tabla 24. Tiempos finales de codificación (ms) en NTSC 23,976 según diferentes *bitrates*, utilizando el software SUPER v2014.build.60 y la compresión con el formato .mp4 y el códec H.264.

CODING TIME. NTSC 23,976	OGG (Theora)
64 Kbps	1m 11s 430ms
240	1m 15s 410ms
576	1m 15s 630ms
1152	1m 08s 970ms
1536	1m 10s 350ms
2256	1m 11s 990ms
3024	1m 16s 620ms
4224	1m 20s 640ms
5808	1m 27s 870ms
7950	1m 36s 080ms

Tabla 25. Tiempos finales de codificación (ms) en NTSC 23,976 según diferentes *bitrates*, utilizando el software SUPER v2014.build.60 y la compresión con el formato .ogg y el códec Theora.

CODING TIME. NTSC 23,976	**WebM (VP8)**
64 Kbps	45s 920ms
240	48s 120ms
576	55s 460ms
1152	48s 360ms
1536	50s 890ms
2256	56s 510ms
3024	57s 960ms
4224	1m 00s 680ms
5808	1m 00s 640ms
7950	1m 05s 200ms

Tabla 26. Tiempos finales de codificación (ms) en NTSC 23,976 según diferentes *bitrates*, utilizando el software SUPER v2014.build.60 y la compresión con el formato .webm y el códec VP8.

Para finalizar, los mismos datos de tiempo de codificación de SUPER pero en NTSC 23,976 muestran la comparativa de la Figura 13 donde demuestra que las direcciones de las líneas de evolución son muy parecidas con escasos milisegundos (ms) de diferencia por el recorte de prácticamente un *frame* en cada segundo del vídeo original (Tabla 24). La opción .mp4 vuelve a convertirse en la más rápida en comparación a .ogg y .webm aunque, vuelvo a insistir, por muy poco, mientras que la calidad entre éste y, especialmente .webm, sí que es importante. Dentro de la linealidad de la figura 13 cabe destacar el punto de incremento importante del tiempo al subir el *bitrate* más de 4000 Kbps y que permite que acerque sus tiempos con *bitrates* más elevados al equilibrado códec VP8. Con estos *bitrates*, H.264 ofrece calidades muy bajas para usuarios con escasa necesidad de calidad de gráfica de imagen. En relación a la Figura 13 podemos entender que los tiempos de compresión (*time codes)* en VLC del códec .ogg (Tabla 25) son menores y con tamaños también menores en comparación con los otros dos *códecs*. El gran problema es la falta de calidad en pro del ahorro en el almacenamiento y el tiempo invertido en la codificación, datos que personalmente no creo que sean tan relevantes dado que las diferencias de espacio y tiempo es de escasos Kbs y ms en un momento en el que estos dos parámetros no son tan

importantes debido al aumento tecnológico de las capacidades de almacenamiento y procesamiento. Es decir, la cantidad de tiempo invertido y la reducción del tamaño finales son insignificantes comparado a algo si e importante como es la calidad final, una calidad que con este códec y con estos *bitrates* resulta muy baja.

Dentro de NTSC, al igual que sucedería en SECAM, acerca más las medias de tiempo de codificación de los tres *códecs*, situando a .webm en el centro de ellos al conseguir un algoritmo medio entre tamaño y time *coding* pero, con los más importante, una calidad final buena o muy buena (incluso excelente en *bitrates* bajos como sucede con el nuevo VP9, su evolución). Es perfecto para la compresión de vídeos de corta duración como los ejemplos de portales (Youtube, Vimeo) o para películas o series donde mantiene una calidad muy cercana a la original pese al gran cambio que supone la codificación en cualquiera de estos tres *códecs*, el paso de *Subsampling* de *Chroma* de 4:4:2 a 4:1:1. La única peculiaridad de la medición con este número de *frames* en SUPER-VP8 (Tabla 26) es la subida del tiempo no lineal al trabajar con 576 Kbps (con una inesperada subida del 9% respecto a la media) repitiendo varias veces la medición.

4.4 GRÁFICAS COMPARATIVAS

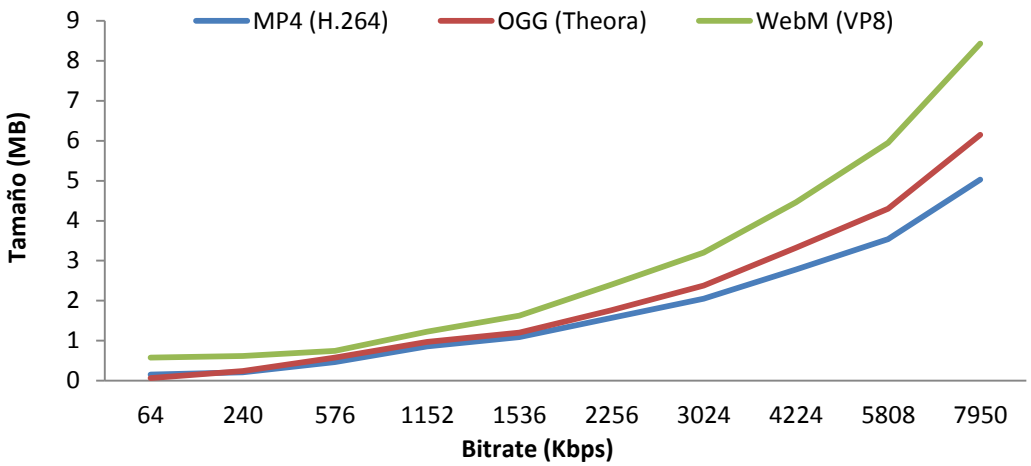

Figura 8. Tamaños finales de codificación (MB) en PAL 25 según diferentes *bitrates*, utilizando el software VLC Media Player 2.1.3 RinceWind.

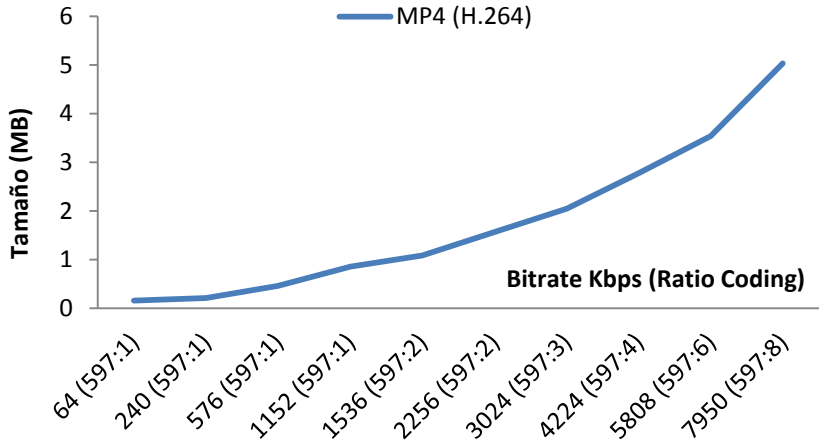

Figura 8.1 Tamaños finales de codificación (MB) en PAL 25 según diferentes *bitrates*, utilizando el software VLC Media Player 2.1.3 RinceWind para MP4 (H.264) con ratios de compresión.

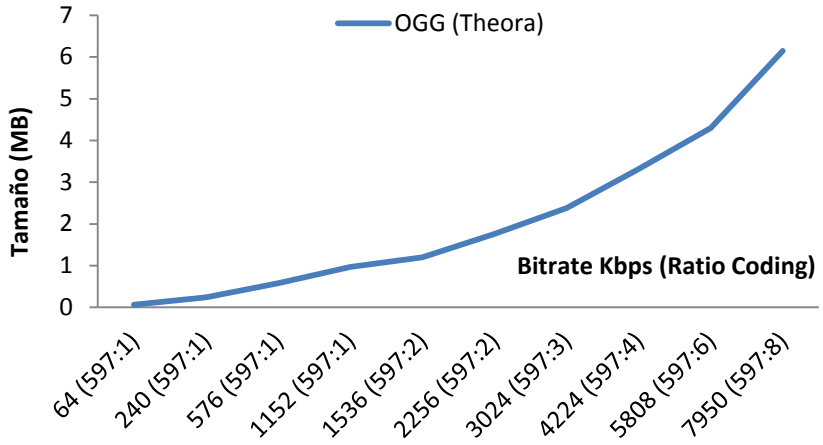

Figura 8.2 Tamaños finales de codificación (MB) en PAL 25 según diferentes *bitrates*, utilizando el software VLC Media Player 2.1.3 RinceWind para OGG (Theora) con ratios de compresión.

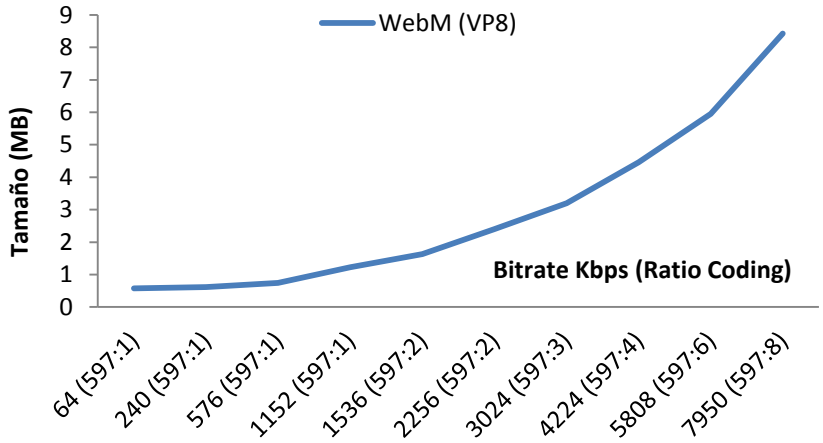

Figura 8.3 Tamaños finales de codificación (MB) en PAL 25 según diferentes *bitrates*, utilizando el software VLC Media Player 2.1.3 RinceWind para WebM (VP8) con ratios de compresión.

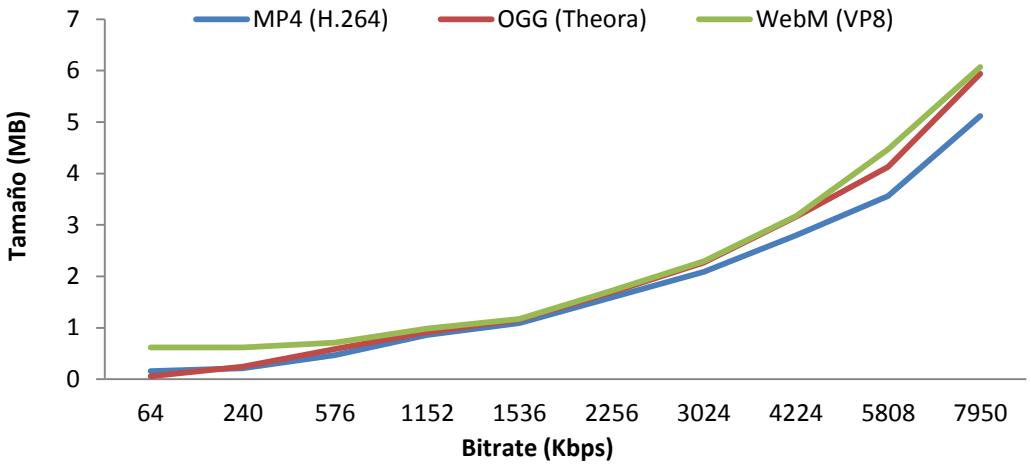

Figura 9. Tamaños finales de codificación (MB) en NTSC 23,976 según diferentes *bitrates*, utilizando el software VLC Media Player 2.1.3 RinceWind.

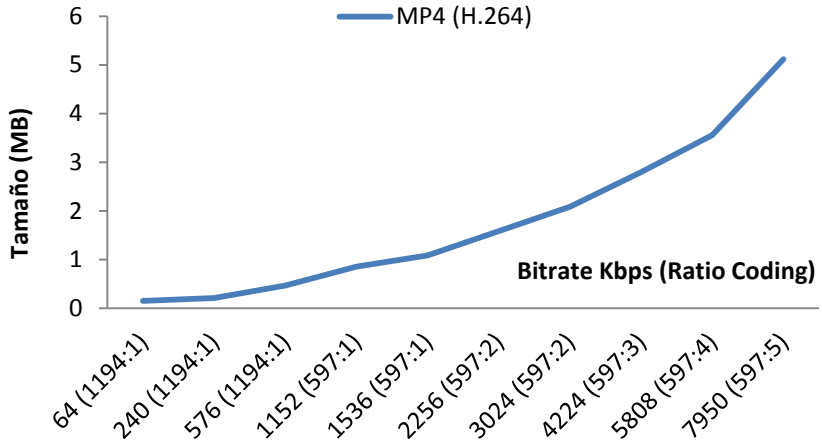

Figura 9.1 Tamaños finales de codificación (MB) en NTSC 23,976 según diferentes *bitrates*, utilizando el software VLC Media Player 2.1.3 RinceWind en MP4 (H.264) con ratios de compresión.

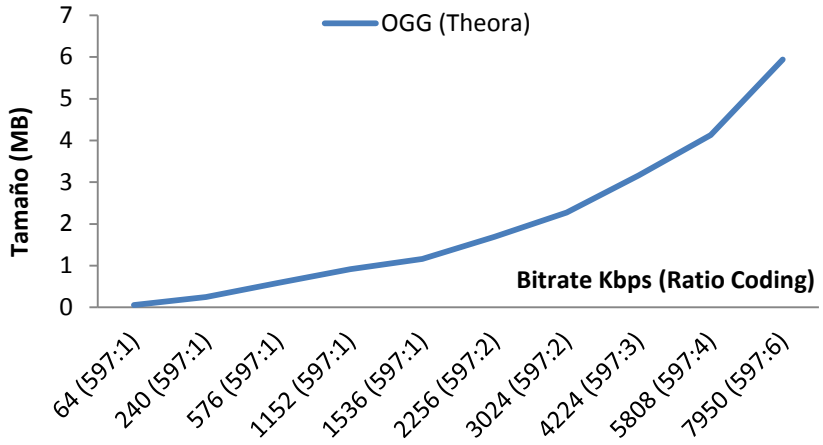

Figura 9.2 Tamaños finales de codificación (MB) en NTSC 23,976 según diferentes *bitrates*, utilizando el software VLC Media Player 2.1.3 RinceWind en OGG (Theora) con ratios de compresión.

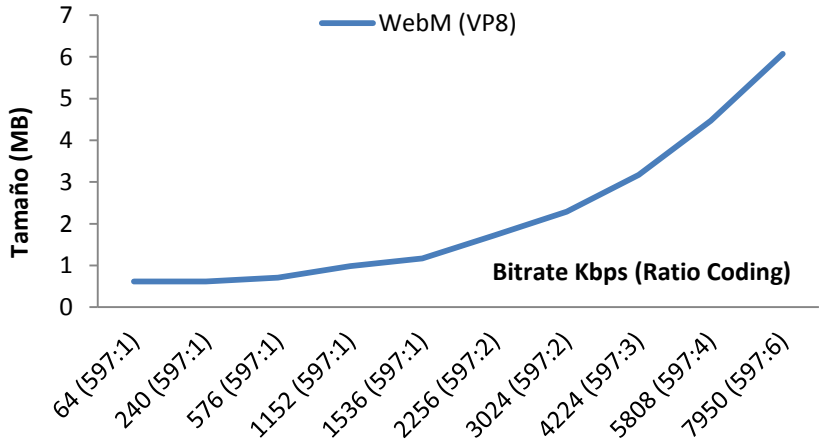

Figura 9.3 Tamaños finales de codificación (MB) en NTSC 23,976 según diferentes *bitrates*, utilizando el software VLC Media Player 2.1.3 RinceWind en WebM (VP8) con ratios de compresión.

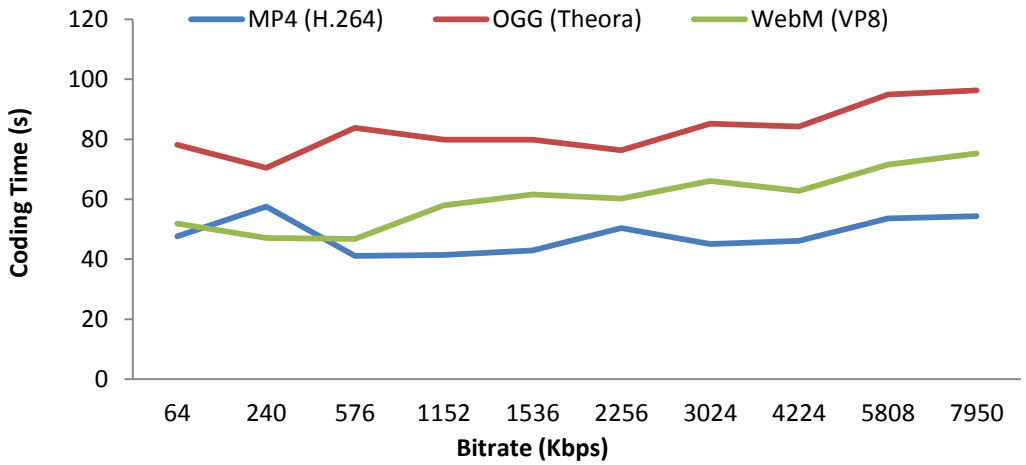

Figura 10. Tiempos finales de codificación (s) en PAL 25 según diferentes *bitrates*, utilizando el software VLC Media Player 2.1.3 RinceWin.

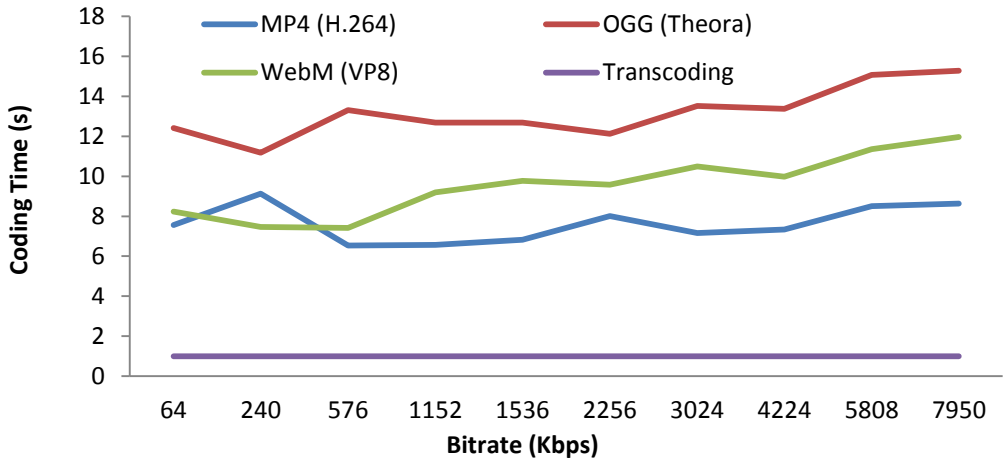

Figura 10.1 Media de codificación, por segundo de vídeo, en PAL 25 según diferentes *bitrates*, utilizando el software VLC Media Player 2.1.3 RinceWin en comparación al tiempo real de reproducción.

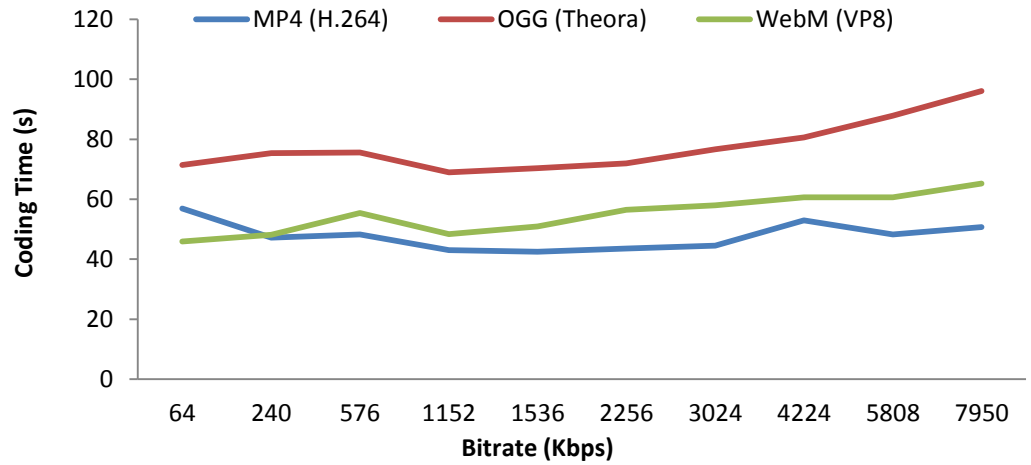

Figura 11. Tiempos finales de codificación (s) en NTSC 23,976 según diferentes *bitrates*, utilizando el software VLC Media Player 2.1.3 RinceWind.

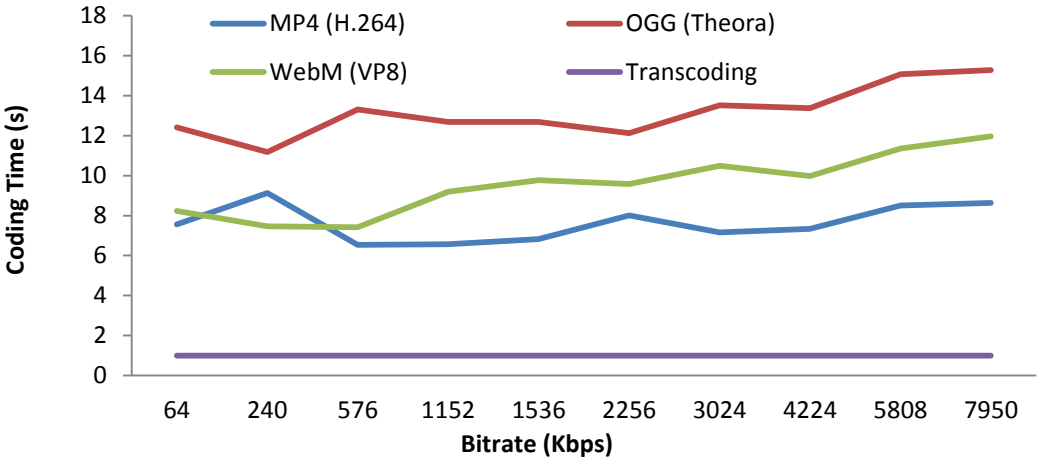

Figura 11.1 Media de codificación, por segundo de vídeo, en NTSC 23,976 según diferentes *bitrates*, utilizando el software VLC Media Player 2.1.3 RinceWin en comparación al tiempo real de reproducción.

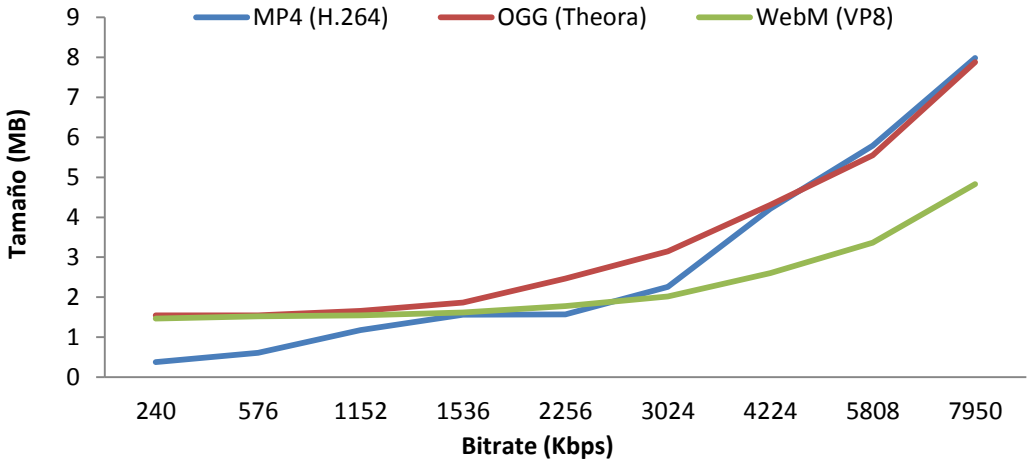

Figura 12. Tamaños finales de codificación (MB) en PAL 25 según diferentes *bitrates*, utilizando el software SUPER v2014.build.60.

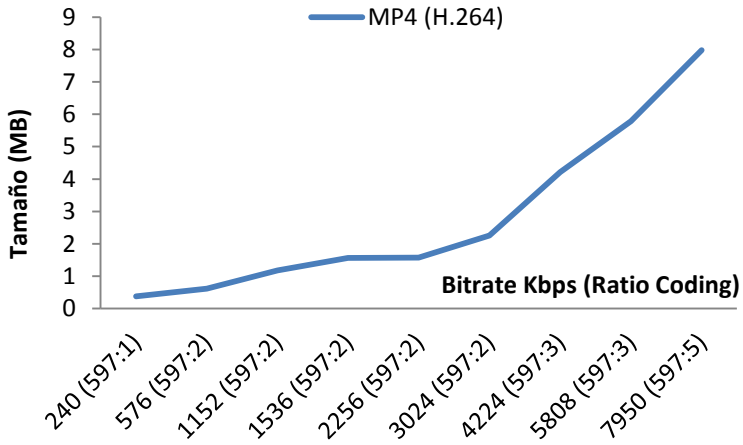

Figura 12.1 Tamaños finales de codificación (MB) en PAL 25 según diferentes *bitrates*, utilizando el software SUPER v2014.build.60 en MP4 (H.264) con ratios de compresión.

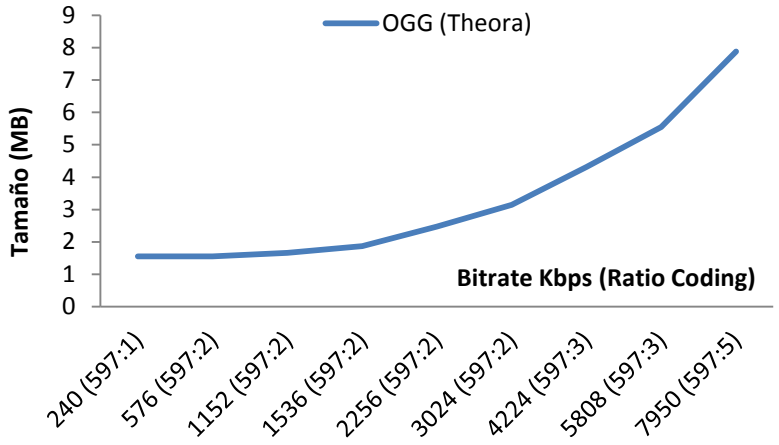

Figura 12.2 Tamaños finales de codificación (MB) en PAL 25 según diferentes *bitrates*, utilizando el software SUPER v2014.build.60 en OGG (Theora) con ratios de compresión.

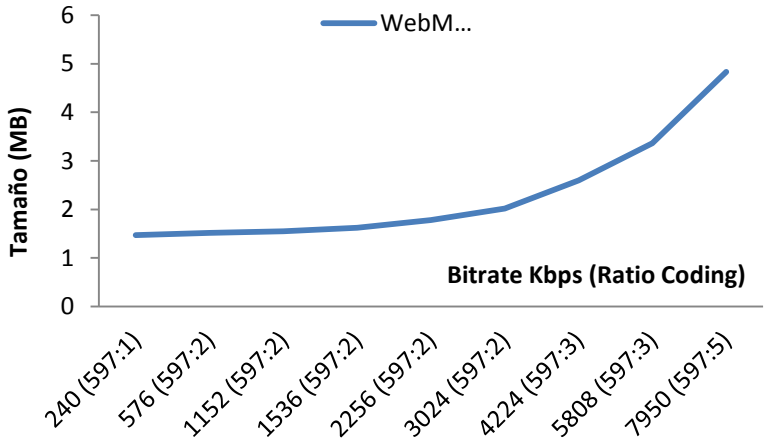

Figura 12.3 Tamaños finales de codificación (MB) en PAL 25 según diferentes *bitrates*, utilizando el software SUPER v2014.build.60 en WebM (VP8) con ratios de compresión.

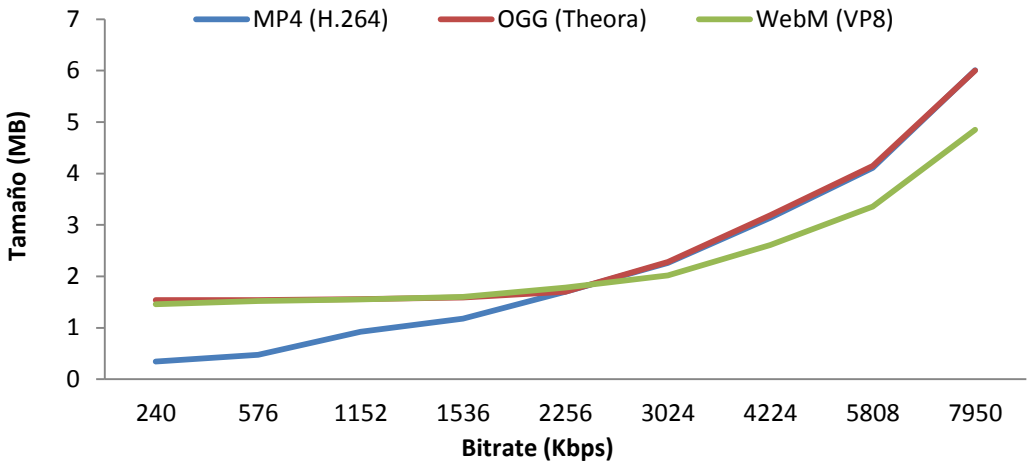

Figura 13. Tamaños finales de codificación (MB) en NTSC 23,976 según diferentes *bitrates*, utilizando el software SUPER v2014.build.60.

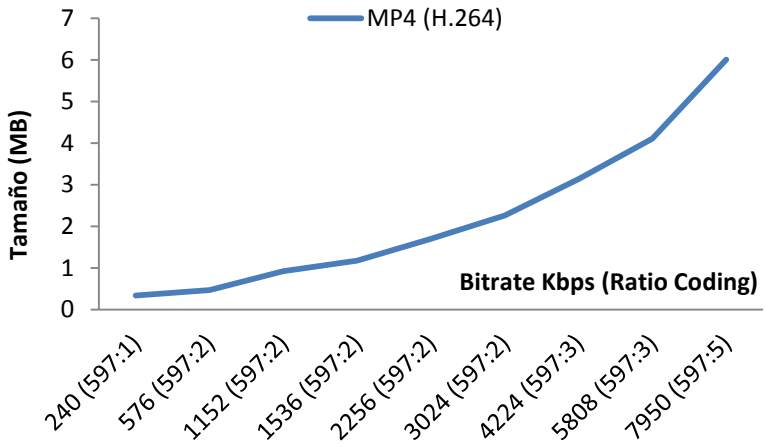

Figura 13.1 Tamaños finales de codificación (MB) en NTSC 23,976 según diferentes *bitrates*, utilizando el software SUPER v2014.build.60 en MP4 (H.264) con ratios de compresión.

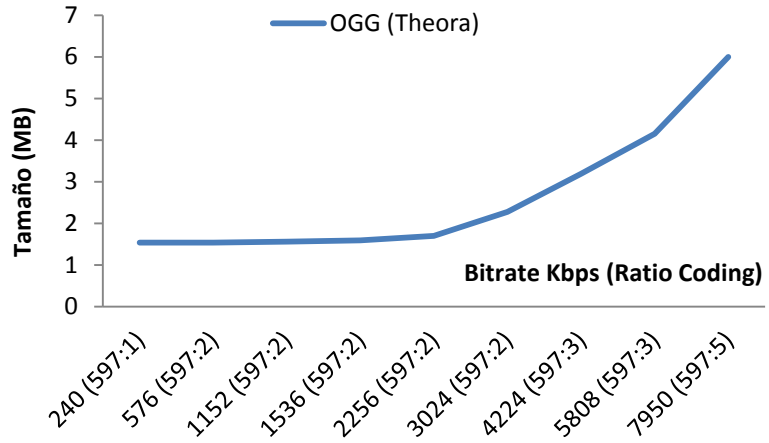

Figura 13.2 Tamaños finales de codificación (MB) en NTSC 23,976 según diferentes *bitrates*, utilizando el software SUPER v2014.build.60 en OGG (Theora) con ratios de compresión.

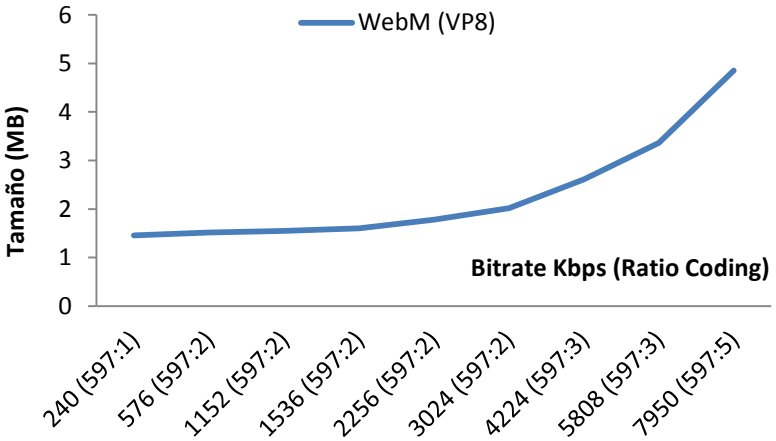

Figura 13.3 Tamaños finales de codificación (MB) en NTSC 23,976 según diferentes *bitrates*, utilizando el software SUPER v2014.build.60 en WebM (VP8) con ratios de compresión.

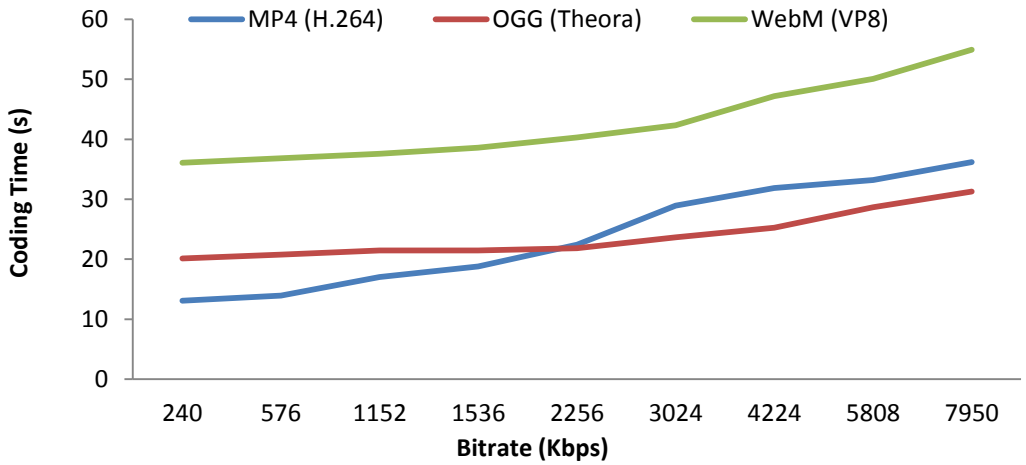

Figura 14. Tiempos de codificación (segundos) en PAL 25 según diferentes *bitrates*, utilizando el software SUPER v2014.build.60.

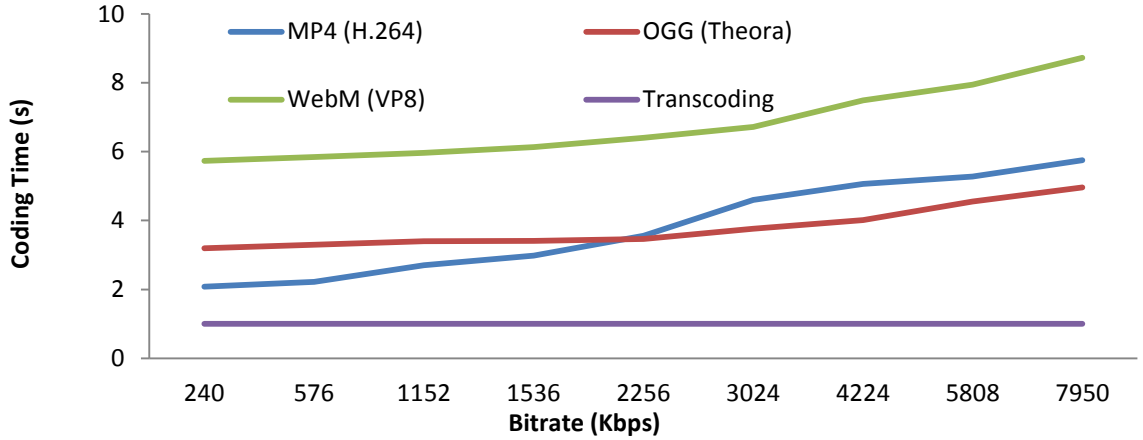

Figura 14.1 Media de codificación, por segundo de vídeo, en PAL 25 según diferentes *bitrates*, utilizando el software SUPER v2014.build.60 en comparación al tiempo real de reproducción.

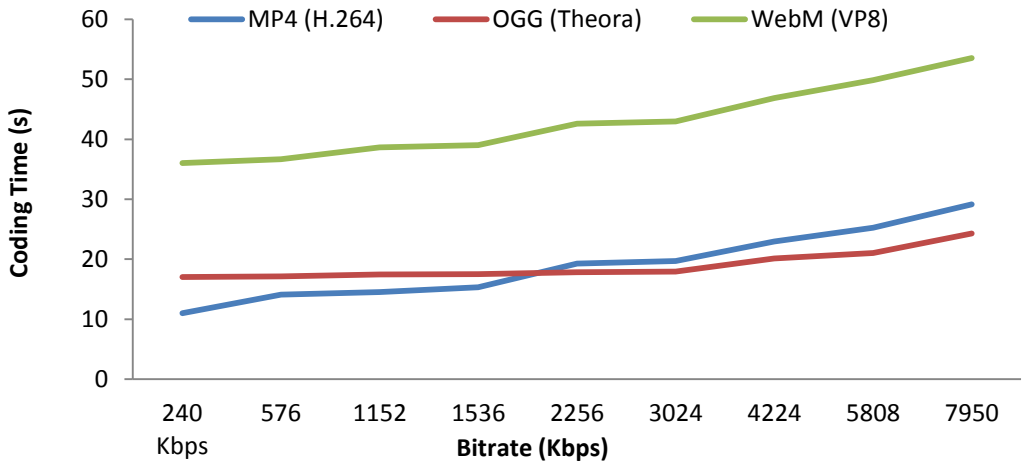

Figura 15. Tiempos de codificación (segundos) en NTSC 23,976 según diferentes *bitrates*, utilizando el software SUPER v2014.build.60.

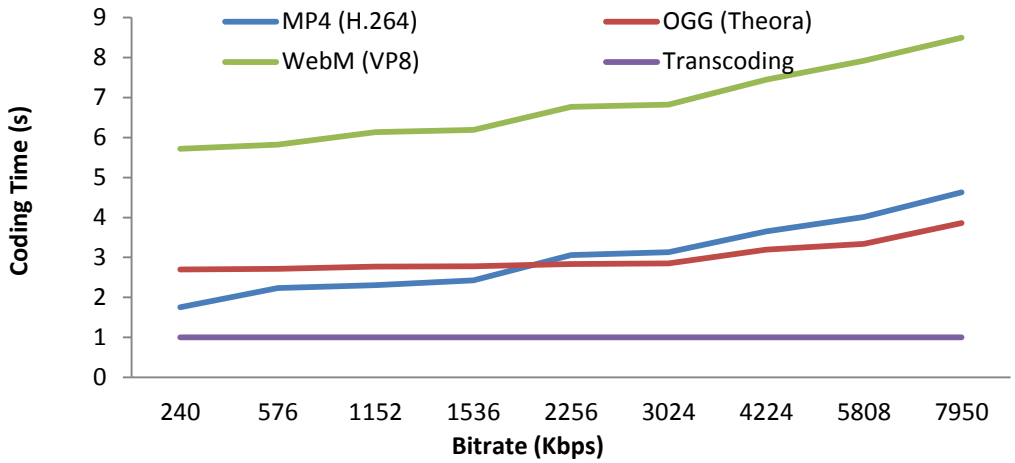

Figura 15.1 Media de codificación, por segundo de vídeo, en NTSC 23,976 según diferentes *bitrates*, utilizando el software SUPER v2014.build.60 en comparación al tiempo real de reproducción.

Figura 16. Gráfica de emisión sobre HTML5 en PAL 25 del vídeo comprimido en MP4

Figura 17. Gráfica de emisión sobre HTML5 en NTSC 23,976 del vídeo comprimido en MP

4.5 DISCUSIÓN DE RESULTADOS

Los ratios de compresión de cada una de las mediciones permiten no limitar los resultados a únicamente el vídeo utilizado sino ampliarlo a cualquier ejemplo, un futuro estudio también podría ser trabajando con diferentes duraciones ya que, en este caso, el ejemplo trabaja con una duración ligeramente superior a seis segundos [21]. De esta manera, después del análisis y estudio de un solo ejemplo se pueden crear los ratios de compresión proporcionales a cualquier ejemplo y que pueden ayudar a decidir al usuario que tipo de codificación utilizar (bitrate, formato, códec, *frame rate* y software) dependiendo de sus necesidades; es importante aclarar que otros tres parámetros muy importantes también pueden cambiar la orientación del usuario (ancho de banda, resolución y procesador).

Las Figuras 8, 9, 12 y 13 muestran los resultados en sus gráficas correspondientes de los tamaños finales de codificación utilizando PAL-VLC (Figura 8), NTSC-VLC (Figura 9), PAL-SUPER (Figura 12) y NTSC-SUPER (Figura 13) y cuyas conclusiones específicas son mostradas a continuación de cada grupo de figuras. Junto a ellas también se encuentran otras gráficas que muestran los ratios de compresión específicos de cada formato y su códec de vídeo, en ellas podemos extraer conclusiones específicas de cada algoritmo de compresión que pueden servir al usuario a la hora de codificar. Dentro de la Figura 8 se encuentran las sub figuras 8.1, 8.2 y 8.3 donde obtenemos las siguientes conclusiones, los mejores ratios de compresión en tamaño dentro de PAL y con el software VLC los ofrece el formato MP4 con una clara tendencia a reducir su tamaño cuando se superan los 2000 Kbps de *bitrate* donde OGG y WebM crecen proporcionalmente pero con ligeras subidas en su progresión mientras que MP4, dentro de su lógica progresión (ver Figura 8.1) reduce sus tamaños hasta separarse en el máximo de 7950 Kbps un 15% de OGG y más de un 30% con WebM que es el que más tamaño proporciona.

Desde los 64 Kbps iniciales hasta los 1000 Kbps los valores en la gráfica de los tres formatos es prácticamente idéntico y, a partir de este *bitrate*, MP4 se convierte en la opción de menor tamaño en su codificación con una tendencia constante respecto a los demás, también hay que tener en cuenta que los tiempos de codificación sitúan a MP4 (H.264) también como la opción más rápida, dentro de ella destacamos el momento en que se empieza a comprimir con 2256 Kbps donde MP4 empieza a distanciarse de OGG y WebM con un ratio de 597:2 frente al 597:3 de los otros, la opción más lenta es OGG con una media de tiempo un 38% más alta que MP4 y situando a WebM en una media entre ambos en *code rating*. Los ratios más altos de esta compresión PAL-VLC los ofrece WebM con un ratio de 597:8 en 7950 Kbps y los más bajos MP4 con un ratio de 1194:1 desde el inicio con 64 Kbps hasta los 576. También hay que destacar que los mejores resultados en la calidad de todos los *bitrates* son los de WebM (VP8) seguido con una gran diferencia por OGG (Theora) y, MP4 que es la mejor opción en tamaño final y tiempos de codificación en PAL-VLC, se convierte en el último respecto a calidad final por lo que el usuario debe decidir qué parámetro destaca en su compresión en 25 *fps* con VLC: tamaño (MP4), tiempo de codificación (MP4) o calidad final (WebM).

En la figura 9 aparece el mismo análisis de codificación con el software VLC pero con un *frame rate* para voltajes de 23,976 *fps* (NTSC) seguido de las sub figuras 9.1, 9.2 y 9.3 con la misma secuencia de contenedores de formato – códec. Nuevamente, MP4 se convierte en la opción con tiempos de compresión más bajos pero, a diferencia, de PAL, las curvas de los tres formatos son muy parecidas (especialmente entre OGG y WebM) y destacando MP4 (H.264) a partir de los 3000 Kbps donde se empieza a distanciar gradualmente de los otros dos *códecs*. En la gráfica correspondiente a los tiempos de codificación en NTSC-VLC se repite la opción de MP4 como el formato más rápido y de OGG como el más lento con una media de un 29% superior respecto al

primero, WebM se mantiene entre ambos. Hay que destacar el punto de compresión con un *bitrate* de 576 Kbps donde MP4 consigue un ratio de 1194:1 por un 591:1 de OGG y Web; aunque son tamaños muy pequeños el algoritmo de H.264 casi reduce a la mitad a los otros dos convirtiéndolo en el ratio más pequeño frente al mayor que es de 597:6 de OGG y WebM en 7950 Kbps (observar las Figuras 9.1, 9.2 y 9.3).

Pasando a la Figura 12 podemos apreciar la comparativa de tamaños finales de codificación en PAL 25 con el software SUPER y, a continuación, los ratios de cada uno de los formatos por separado en las tres sub figuras siguientes (12.1, 12.2 y 12.3). Aquí cambia todo al convertirse MP4 en la opción de mayor tamaño (junto a OGG) y WebM en la de menor, también destacando que el códec VP8 es la opción más lenta en los *time coding* con cerca del 50% más que MP4 y OGG que se encuentran muy cercanos en tiempo de codificación. Dentro de las gráficas de tamaño hay que destacar el importante cambio de MP4 (H.264) en su progresión de *bitrate*, en los menores se convierte en la opción de menor tamaño final hasta que sobrepasa los 3000 Kbps donde cambia bruscamente a la opción con mayores tamaños, OGG también pero lo hace de una forma continua y gradual. Respecto a los ratios en PAL-SUPER destaca el 1194:1 de MP4 como el más bajo y 597:6 de MP4 y OGG como los más altos. La mejor calidad es nuevamente del códec de Google (VP8) con una diferencia significativa tanto en calidad como en errores respecto a los otros dos.

Finalmente, en la Figura 13 aparece la evolución de las gráficas de los tres *códecs* con un *frame rate* de 23,976 *fps* y compresión con el software SUPER. Las tres sub figuras siguientes (13.1, 13.2 y 13.3) muestran los ratios de codificación para cualquier tipo de vídeo, repito que sin estos ratios los resultados solamente podrían aplicarse al vídeo analizado y no una aplicación general a cualquier archivo multimedia. Como conclusión podemos afirmar que, al igual que PAL-SUPER, el formato WebM se convierte en la opción con menores tamaños de codificación mientras que MP4 y OGG casi calcan sus

resultados a excepción de los modelos de *bitrate* de 240 a 2256 Kbps donde (al igual que en el estudio de la Figura 9 y sus sub figuras) MP4 se convierte en el formato con los tamaños menores. Ocurre exactamente igual que en PAL-SUPER donde existe un momento crítico para MP4 (H.264) que se convierte en punto de inflexión en su evolución, es el paso a 2256 Kbps con un cambio de ratio de 597:1 a 597:2 (1,18 MB a 1,70 MB) con un incremento repentino del 30%. Respecto a los tiempos de codificación observamos que MP4 y OGG se mantienen como las opciones de menor tiempo en NTSC-SUPER con una distancia media de casi el doble con WebM que llega tiempos de compresión cercanos a un minuto por los seis originales del vídeo en los *frame rates* más altos (5808 y 7950 Kbps) tanto en PAL como en NTSC. Sin embargo, este tiempo de codificación también convierte al códec VP8 en la opción final de menos errores y mayor calidad de imagen al igual en todas las opciones. Respecto a los ratios (observar Figuras 13.1, 13.2 y 13.3) son muy similares a sus análogos de PAL 25 con un mínimo de MP4 en 240 Kbps de 1194:1 (0,342 MB) y un máximo también de MP4 en 7950 con 597:6 (6,01 MB), estos cambios en los ratios de los tres *códecs* son producidos por las peculiaridades de cada uno de sus algoritmos. A continuación, se muestra una tabla con el resumen de cada una de las opciones y sus resultados en tamaño, tiempo de codificación y calidad final [21].

	PAL-VLC	**NTSC-VLC**	**PAL-SUPER**	**NTSC-SUPER**
Menor tamaño	MP4	MP4	WebM	WebM
Mayor tamaño	WebM	WebM	OGG	OGG
Menor *time coding*	MP4	MP4	MP4	MP4
Mayor *time coding*	OGG	OGG	WebM	WebM
Menor calidad	MP4	MP4	MP4	MP4
Mayor calidad	WebM	WebM	WebM	WebM

Tabla 27. Relación de parámetros de QoE según *frame rate* y software para MP4, OGG y WebM.

En referencia a los tiempos de codificación (*time coding*) existen cuatro figuras que comparan los resultados de tiempo de codificación de PAL-VLC (Figura 10.1), NTSC-VLC (Figura 10.2), PAL-SUPER (Figura 14.1) y NTSC-SUPER (Figura 14.2) con la media de un segundo de reproducción (1.0x) en tiempo real. Es decir, se comprueba que códec está más cerca de poder realizar un *transcoding* en paralelo en tiempo real sin tener que llenar un *buffer* con la consecuente pérdida de tiempo, por tanto, el resultado de la división del tiempo de codificación de cada códec entre la duración del vídeo original (6,298 s) debería ser inferior a 1 para que se realice un *transcoding* en paralelo y en tiempo real, la Figura 16 ofrece la demostración matemática:

$$\frac{coding\ time}{6{,}298\ (\mathrm{s})} < 1 \longrightarrow real\ transcoding\ (1.0\mathrm{x})$$

Figura 18. Fórmula de cálculo de un *transcoding* en tiempo real basado en la duración del vídeo original.

Generalmente, el formato (códec) que más se acerca a los valores necesarios de *transcoding* es MP4 (H.264) ya que, como podemos observar en la Tabla 27 y en las Figuras 10, 11, 14 y 15, presenta los *time code* más bajos de los tres *códecs* que conforman en la actualidad el estándar de HTML5. La tasa de *bitrate* más cercana al *transcoding* en tiempo real es la de MP4 con 240 Kbps en la compresión con SUPER y con *frame rate* NTSC 23,976 *fps,* la más lejana es la de OGG (Theora) con 7950 Kbps en la compresión con VLC y con *frame rate* PAL 25 *fps* [21]. Es muy importante la consideración de que estos tiempos se pueden acercar (y también bajar de 1) con otro procesador más potente, con un códec distinto a los del estándar o con otro software de codificación como, por ejemplo, Ffmpeg.

En las Figuras 16 y 17 se aprecian los resultados gráficos de la emisión de vídeo sobre HTML5 en el puerto 80 recogidos por la tarjeta de red que se detalla en la página 40. Para ello se utiliza el software de inyección de tráfico de datos multimedia (testtool)

que es muy similar al WireShark, se inyectan vídeos codificados de los ejemplos, concretamente con el algoritmo MP4 (H.264) tanto en PAL 25 (Figura 16) como en NTSC 23,976 (Figura 17). Podemos observar una mayor desviación de las líneas de las gráficas en PAL 25 ya que ofrece más información por segundo (cuenta con prácticamente un *frame* más), esto también se refleja en los picos de retado (*delay*) y la desviación de la periodicidad de la señal (*jitter*) con picos que no aparecen en NTSC 23,976 así como un consumo de Ancho de Banda más regular, los picos de PAL 25 en la Figura 16 corresponden a los *frames* con más información de la secuencia GOP (concretamente los *intraframe* que tienen información de referencia para los demás *frames* de la secuencia GOP). Por tanto, podemos afirmar que la emisión de vídeo sobre el estándar HTML5 es más regular en PAL 25 por varias razones: corresponde al voltaje eléctrico de nuestra zona, presenta un número de *frames* por segundo más exacto que NTSC 23,976, muestra con sus picos en los paquetes recibidos y en el consumo de Ancho de Banda los *intraframe* de referencia de la secuencia GOP y produce mejores resultados gráficos tanto en la imagen como en los parámetros de emisión que ofrece la herramienta de software testtool.

5. CONCLUSIONES

5.1 CUMPLIMIENTO DEL OBJETIVO

El objetivo principal del proyecto, así como los secundarios, han sido cumplidos al obtener todos los datos necesarios para elaborar las primeras conclusiones respecto al tema de investigación así como de los diferentes ratios y resultados del apartado práctico. Mediante la investigación teórica se ha conseguido adquirir los conocimientos necesarios respecto a *networking*, protocolos y lenguajes HTML, APIs, JavaScript y sus correspondientes librerías así como una introducción a estilos. La parte más importante es la que detalla los algoritmos de compresión y las diferencias entre los tres principales *códecs* del estándar de internet (.mp4, .ogg y .webm)

La parte más específica es el estudio de la compatibilidad de dichos *códecs* con los principales navegadores del mercado así como el cálculo de ratios y gráficas de eficiencia de la codificación de un vídeo original en diferentes *bitrates* y *frames* por segundo, así se obtienen datos de sus tamaños finales y tiempos de codificación para que el usuario o la cabecera MIME puedan decidir cuál es la mejor opción según una serie de características: tipo de dispositivo, ancho de banda disponible, navegador, etc.

Por lo tanto, se ha conseguido un estudio y análisis profundo de las posibilidades que ofrecen los principales *códecs* del mercado en su codificación en diferentes parámetros: con los software VLC y SUPER, con diferentes bitrates y tanto en PAL 25 como en NTSC 23,976; con ello se han creado matrices de estudio que permite al usuario ajustar la configuración óptima de compresión según sus necesidades: duración del archivo multimedia, ancho de banda disponible, calidad deseada, espacio de almacenamiento limitado, *streaming*, etc. Esto ayudará a un conocimiento de las posibilidades de los diferentes formatos del estándar HTML5 así de su compatibilidad

con los diferentes navegadores del mercado para realizar codificaciones con software gratuito enfocadas a las necesidades personales de cada usuario y mejorar su QoE. Asimismo, el presente TFM ha ayudado a conocer la aplicación de diferentes lenguajes de programación, etiquetado o librerías para su aplicación en la emisión de vídeo digital como son el caso de diferentes APIs como Canvas, Video, WebWorkers o los lenguajes HTML o JavaScript junto a sus librerías como JQuery.

5.2 CONCLUSIONES SOBRE EL PROYECTO

Partiendo de que VLC no permite una codificación con una tasa de bits tan pequeña como 64 Kbps ya que genera un error en la compresión y, a partir de una tasa de bits de 240 Kbps, ya se puede hacer un estudio sobre las relaciones entre los diferentes parámetros:

En PAL a 25fps se obtienen valores de tamaño ligeramente superiores al sistema NTSC a 23,976 fps debido claramente a la diferencia de cerca de un *frame* entre un sistema y otro, esta discrepancia de sistemas se debe a los distintos voltajes que existen entre diferentes regiones. Por tanto, los valores de tamaño son algo superiores en el sistema PAL tanto en MP4 (h.264) como en OGG (Theora) mientras que en WebM (VP8) son prácticamente idénticos en PAL como en NTSC. Los tamaños de codificación son muy similares utilizando VLC y SUPER en los formatos MP4 y OGG, sin embargo, se aprecia una significativa diferencia de tamaño en el formato WebM de Google, son superiores a los otros dos *códecs* cuando se utiliza el software SUPER con la librería x264 en valores que suponen casi el doble de su compresión mediante el software VLC. El tiempo de codificación en SUPER también es mayor que en VLC tal y como se puede observar en las figuras 14 y 15, este dato temporal de compresión justifica un mayor tamaño en el resultado final.

Finalmente, la opción más favorable respecto a calidad-tamaño-tiempo, es la compresión mediante VLC en el formato WebM (VP8) de Google ya que su tamaño final es el más bajo, con un tiempo de codificación menor y con una calidad cercana a *bitrates* mayores del formato MP4 (H.264). Tal y como se había detallado anteriormente, se ha descartado el uso de Sorenson Squeeze por la falta de soporte para la codificación en el formato ogg. Las mediciones con VLC Media Player 2.1.3 y SUPER v2014.build.60 se realizaron con diferentes *bitrates* entre 64 y 7950 kbps (éste último corresponde al muestreo del archivo original MPEG-4 4:2:2 *lossless*). En el apartado Vc se muestran los resultados con los ratios correspondientes para poder hacer una valoración individual de cada compresión según sistema PAL o NTSC, formato Mp4, WebM u ogg así como el diferente *bitrate* aplicado. También se muestran las gráficas correspondientes a los valores de compresión utilizados en los dos software en el apartado Vd.

Por tanto, las conclusiones finales son las siguientes: la codificación con menor tamaño siempre corresponde a MP4 en el sistema PAL y NTSC con VLC y a WebM en el sistema PAL y NTSC con SUPER, el mayor tamaño a WebM en el sistema PAL y NTSC con VLC y a OGG en el sistema PAL y NTSC con SUPER, el menor tiempo de codificación siempre corresponde a MP4 en PAL y NTSC y tanto en VLC como en SUPER, el mayor tiempo de codificación a OGG en el sistema PAL y NTSC con VLC y a WebM en el sistema PAL y NTSC con SUPER, y finalmente, la mejor calidad siempre corresponde a WebM mientras que la menor calidad a MP4 en todas las opciones. Por tanto, MP4 (H.264) ofrece tiempos de codificación y de tamaño muy bajos a cambio del sacrificio en la calidad final, justo al contrario de WebM (VP8) que aumenta sus tiempos de codificación y sus tamaños finales en favor de una excelente calidad final; OGG (Theora) se encuentra equilibrada entre ambos *códecs* en todos los sentidos, pero

realmente, el mejor códec para compresión y emisión de vídeo sobre el estándar de HTML5 es el algoritmo de Google WebM (VP8).

La repercusión de este tema de investigación es la futura mejora del servicio de emisión de vídeo en páginas web. Los actuales portales como Youtube o Vimeo ofrecen diferentes resoluciones de un mismo vídeo que se encuentran almacenadas en sus servidores, sin embargo, a través de un *transcoding* en paralelo desde el server y con una comunicación cliente-servidor con la cabecera MIME y el protocolo HTTP se puede obtener información del usuario (dispositivo, navegador o ancho de banda) para ofrecer el vídeo original en el códec adecuado, con el *bitrate* correspondiente y una opción PAL o NTCS personalizada. Esto ahorraría gran cantidad de almacenamiento en los servidores pero también requeriría de una gran capacidad de procesamiento; el resultado sería una oferta personalizada a cada usuario de manera transparente el cual realizaría la petición del archivo multimedia y los diferentes protocolos y scripts de HTML5/JavaScript/MIME decidan la mejor opción de *transcoding* en paralelo para dicho usuario teniendo en cuenta todos los factores analizados: PAL 25 o NTSC 23,976, Ancho de Banda disponible, duración del archivo, tipo de dispositivo, resolución soportada y compatibilidad de formato-códec con el navegador.

5.3 PROBLEMAS ENCONTRADOS Y CÓMO SE HAN SOLUCIONADO

El principal problema del *transcoding* en paralelo es la gran capacidad que necesita el servidor para hacer varios multiplexados de bit vídeo-audio simultáneamente, ya que se necesita una velocidad de procesador muy alta.

Respecto al estudio, los problemas existentes son los mismos que presentan la gran variedad de *códecs* dentro de Internet en la actualidad y la consecuente guerra de marketing de las diferentes empresas para instaurar el algoritmo de codificación que

más les interese e imponerlo en el estándar de navegación. El resultado es una guerra de formatos donde algunos navegadores sí que los reproducen (defienden) y otros no, por lo que hay que crear scripts dentro del código de la API de vídeo de HTML5 para poder detectar el tipo de navegador y si existe soporte para el códec utilizado, para ello, se puede hacer una codificación en paralelo (*transcoding*) o almacenar diferentes versiones del mismo vídeo con formatos diferentes para cualquier tipo de navegador que pueda realizar la petición. A esto hay que sumarle el inconveniente del auge de los nuevos dispositivos como *tablets* y *smartphones* que ofrecen resoluciones diferentes y que necesitan un tratamiento personalizado para ofrecer una página web y también para la reproducción por descarga o *streaming* del material seleccionado, la solución es la preparación dentro del código HTML de la página para poder ofrecer toda la información de manera personalizada a cada dispositivo. Un último problema es que más del 65% de los usuarios del planeta utilizan Internet Explorer 6 o alguna versión anterior del navegador de Microsoft por lo que presentan el problema de no poder reproducir los vídeos que se ofrecen actualmente en las páginas web con los principales formatos empleados (.mp4, .ogg y .web). Para solucionar este problema hay que añadir muchas líneas de código para que las páginas sean compatibles con flash y sus extensiones (.flv) y, por tanto, depender de *plugins* y software de terceros, algo que HTML5 ha decidido eliminar a través de su nueva API de vídeo.

5.4 APORTACIONES PERSONALES

Gracias a este proyecto he aumentado mis conocimientos en formatos digitales avanzados al conocer tanto las diferencias de los principales *códecs* del estándar de Internet así como su compatibilidad con los principales navegadores del mercado. Para llegar al análisis final de las posibilidades concretas de cada uno en su codificación también ha sido necesario aprender el funcionamiento interno de sus *coders-decoders*

en el algoritmo lo que me ha permitido aprender las características propias de cada uno de los contenedores y *códecs* de vídeo.

La plataforma práctica de prueba ha requerido un estudio y su consecuente aprendizaje y mejora de lenguajes de programación como JavaScript o C que permitan poner en marcha un inicio de emisión de vídeo personalizada a través del lenguaje de etiquetas HTML, un lenguaje que también me ha permitido ampliar conocimientos así como posibilidades en su interacción con archivos multimedia gracias a profundizar en sus diferentes APIs que, por supuesto, también se encuentran relacionadas con la programación *scripting* de JavaScript y de sus librerías JQuery y Ajax.

Personalmente, el estudio también me ha permitido desarrollar cualidades como la exigencia o profesionalidad que requiere el rigor de este tipo de trabajos de investigación. Por tanto, el respeto de las pautas académicas en referencias, bibliografía, conclusiones o datos de análisis ha mejorado mi capacidad de desarrollo, búsqueda de información y rigor en la redacción o búsqueda de información, lo que me sirve para futuras publicaciones o trabajos de divulgación académica y científica.

5.5 FUTURAS LÍNEAS DE TRABAJO

Algunas de los puntos tratados en la tesina pueden servir para la mejora o creación de otras ideas relacionadas con el uso de video en diferentes medios o aplicaciones. Por ejemplo, para medir las respuestas y los parámetros de los clientes en una conexión de vídeo HTML5 para servicios de *e-learning* o para la programación de códigos QR para interactuar con vídeos en presentaciones con Prezy, Power Point, etc [18]. Otra idea es la mejora de los sistemas de detección de tipo dispositivo visor de vídeo polimedia, además, se pueden mejorar características relacionadas con la codificación de vídeo en Sistemas de videovigilancia [3].

El trabajo de investigación presente también se puede ampliar a otros parámetros que también intervienen en el proceso de codificación de un archivo multimedia de vídeo y que pueden mejorar el tamaño final, el tiempo de compresión, la QoE o la calidad. De esta manera, se puede continuar y ampliar la línea de investigación de esta Tesina con un análisis de los mismos datos pero sumando información respecto a diferentes resoluciones, tipos de dispositivos, consumo de Ancho de Banda, colorimetría, consumo de procesador o influencia del tipo de navegador o el Sistema Operativo (kernel). En un futuro proyecto se podrían calibrar estos parámetros según su análisis para mejorar la usabilidad y optimización de la compresión hasta un *transcoding* en paralelo cuya codificación sea inferior por segundo al tiempo real, esto permitiría con una elección basada en JavaSript que compresión se ofrece al usuario dependiendo de todos los factores mencionados y codificar el vídeo en tiempo real sin que exista un *buffer* o un *delay* apreciable.

La línea de investigación de esta idea basaría su decisión en el intercambio de información entre el cliente y el servidor del protocolo HTTP, con la ayuda del código JavaScript dentro de HTML, así se podría tomar la decisión de qué códec y que tipo de compresión aplicar dependiendo del tipo de petición y las características del usuario. Para ello, es imprescindible conocer los protocolos de *networking* existentes y poder aprovechar la información que aportan las cabeceras MIME. La idea final es una oferta de vídeo con un gran ahorro de espacio en el almacenamiento del servidor y donde el protocolo de comunicación cliente-servidor pueda tener la capacidad de decidir cuál es la mejor opción de codificación para el usuario en cuestión, a diferencia de otros portales de vídeo actuales [2] [10] [13].

6. BIBLIOGRAFÍA

6.1 REFERENCIAS

[1] Jaime Lloret, Carlos Palau y Manuel Esteve, *Structuring connections between content delivery servers groups*, Future Generation Computer Systems. Vol. 24, Issue 3, Pp. 191-201, Marzo 2008

[2] Satu Jumisko-Pyykö, Jukka Häkkinen, "Evaluation of Subjective Video Quality" Department of Psychology, University of Helsinki, P.O.Box 9, 00014 University of Helsinki, Finland

[3] Ruth Herrero Doñate, Jaime Lloret Mauri, José Miguel Jiménez Herranz, "*Estudio de códecs de compresión MPEG-4 para su aplicación a la videovigilancia"* Departamento de Comunicaciones, Universidad Politécnica de Valencia, Spain, 2005

[4] Jaime Lloret, Miguel Garcia, Antonio, J. Ferrer y Maria Morant, *HDTV compression analysis for storage and transmission over Internet*, The 5th WSEAS International Conference on DATA NETWORKS, COMMUNICATIONS and COMPUTERS (DNCOCO 2006), Bucarest (Rumania), 16-18 de Octubre de 2006

[5] Jaime Lloret, Alejandro Canovas, Jesus Tomas and Marcelo Atenas, *Control de algoritmos de red y protocolos para la mejora Qoe en móbiles*, Computer Communications. Vol. 35, Issue 15, Pp. 1819-1827. Septiembre 2012.

[6] Jaime Lloret, Miguel Garcia, Marcelo Atenas, Alejandro Canovas, *A Stereoscopic Video Transmission Algorithm for an IPTV Network Based on Empirical Data*, International Journal of Communication Systems. Vol. 24, Issue 10. Pp. 1298–1329. Octubre 2011

[7] Alejandro Canovas, Miguel Garcia, Jaime Lloret, Marcelo Atenas and Rafael Rizo, *Improving IPTV QoE taking the suitable MPEG-2/MPEG-4 Quantizer based on Jitter, Delay and lost packets measurements*, International Journal On Advances in Telecommunications, Vol. 3, Issue 3&4, Pp. 129 – 139. Diciembre 2010

[8] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. Sullivan "Rate-Constrained Coder Control and Comparison of Video Coding Standards", IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 688-703, July 2003

[9] R. Koenen, "Overview of the MPEG-4 standard", ISO/IEC JTC1/SC29/WG11 N1730, Luglio 1997

[10] "Advanced video coding for generic audiovisual services", ITU-T Recommendation H.264, November 2007

[11] R. Koenen, "Overview of the MPEG-4 standard", ISO/IEC JTC1/SC29/WG11 N1730, Luglio 1997

[12] Rubén Tortosa, Jaime Lloret Mauri, Juan R. Díaz, "*Optimal Codec Selection Algorithm for Audio Streaming"* Departamento de Comunicaciones, Universidad Politécnica de Valencia, Spain

[13] Ali Safa Sadiq, Kamalrulnizam Abu Bakar, Kayhan Zrar Ghafoor, Jaime Lloret, *An Intelligent Vertical Handover Scheme for Audio and Video Streaming in Heterogeneous Vehicular Networks*

[14] Apteker, R.T., Fisher, J.A.,Kisimov, V.S., Neishlos, H. *Video Acceptability and Frame rate. IEEE Multimedia,* 3(3):32—40, 1995

[15] McCarthy, J. D., Sasse M. A. and Miras D. "*Sharp or Smooth*?: Comparing the Effect of Quantization vs. Framerate for Streamed Video", 2004, Proc. of the 2004 conference on Human factors in computing systems. Vienna: 2004. p. 535-542.

[16] Wiegand T., Sullivan G. J., Bjøntegaard G., Luthra A., *Overview of the H.264/ AVC Video Coding Standard, IEEE Transactions on Circuits and Systems for Video Technology*, Vol.13,No.7(2003). pp. 1-19

[17] Winkler, S. & Faller, C. *Audiovisual quality evaluation of low-bitrate video. Proc. SPIE/IS&T Human Vision and Electronic Imaging*, vol. 5666. San Jose, United States of America: January 2005. pp. 139-148

[18] Pedro V. Mauri, Jaime Lloret, Miguel Garcia and Maria Morant, *Proyecto para el diseño de una red inalámbrica para la transmisión de vídeo en el proyecto educativo y divulgativo "Explora El Encín": educación ambiental para todos*. X

International Congress on Project Engineering, Valencia (España), 13-15 Septiembre de 2006

[19] Miguel García, Antonio J. Ferrer, María Morant, Jaime Lloret, *Editing and Compressing HDTV videos for storage and transmission over Internet*.

[20] Jaime Lloret, Pedro V. Mauri, José M. Jiménez, Juan R. Díaz, *802.11g WLANs Design for Rural Environments Video-surveillance*

[21] Andrés López Herreros, Jaime Lloret Mauri, *Study of Video Coding with HTML5 Standard Codecs*. Universitat Politècnica de València (UPV), 2014.

[22] Joe Crop, Alex Erwig, Vivek Selvaraj, *Ogg Video Coding*, 2010

[23] Pankaj Kumar Bansal, Vijay Bansal, Mahesh Narain Shukla, Ajit Singh Motra, *VP8 Encoder – Cost Effective Implementation*, ST Microelectronics India Private Limited, DCG, Greater Noida, India

[24] Jim Bankoski, Paul Wilkins, Yaowu Xu, *Technical Overview of VP8, an Open Source Video Codec for the Web*. Google Inc. Mountain View, CA, USA. 2012

6.2 BIBLIOGRAFÍA COMPLEMENTARIA

ÁLVAREZ GARCÍA, ALONSO (2012), *HTML5*, Madrid, Grupo Anaya S.A.

BOWEN, RICH (2000), *Servidor Apache,* Madrid, Pearson Educación S.A.

GOLDSTEIN, ALEXIS (2011), *HTML5 y CSS3*, Madrid, Ediciones Anaya Multimedia.

KABIR, MOHAMMED J. (1998), *Apache Server*, California, IDG Books Worldwide.

PLEIFFER, SILVIA (2011), *Vídeo con HTML5*, Madrid, Ed. Anaya Multimedia.

POWELL, THOMAS (2001), HTML: *The Complete Reference*, New York, McGraw-Hill.

ZAKAS, NICHOLAS C. (2005), *JavaScript*, Madrid, Ed. Anaya Multimedia.