Streaming Interactivo de Secuencias de Imágenes JPEG2000 de Alta Resolución

J.J. Sánchez-Hernández, J.P. García-Ortiz,
Carmelo Maturana-Espinosa y Vicente González-Ruiz
Departamento de Informática. Universidad de Almería, Spain
Campus de Excelencia Internacional Agroalimentario, ceiA3

D. Müller
European Space Agency, ESTEC
Noordwijk, Netherlands

Resumen— **El estándar internacional JPEG2000 pretende ser un sucesor del estándar JPEG en la mayoría de sus áreas de aplicación, y especialmente en el campo de la transmisión interactiva de imágenes. Para esta tarea, JPEG2000 se basa en el uso del protocolo JPIP que permite realizar un acceso espacial de forma aleatoria en tiempo real (panning, tilting y zooming) sobre la imagen que se está visualizando de forma progresiva (streaming). En este contexto, este trabajo[1] presenta una solución compatible con el estándar JPIP para el streaming interactivo de archivos JPX que son transmitidos sobre enlaces de comunicación con un ancho de banda variable. La variación impredecible del ancho de banda durante la transmisión obliga a los clientes JPIP a tener un buffer con una mínima cantidad de datos y a implementar un mecanismo de feedback que permita al servidor JPIP decidir, en tiempo real, la cantidad de code-stream que se va a transmitir de cada imagen. Nuestra propuesta garantiza que se puede mantener el frame-rate de la visualización incluso en situaciones donde se produzcan grandes variaciones de ancho de banda y se dispongan de reducidos tamaños de buffer. Como una última ventaja, queremos remarcar que nuestra solución también puede ser utilizada para realizar streaming de secuencias Motion JPEG2000.**

Palabras clave— **Streaming media, high-resolution imaging.**

I. Introducción

En muchos campos de la ciencia y la tecnología (tales como la telemedicina, telemicroscopía y astronomía) existe una necesidad de crear grandes repositorios de imágenes. Debido a razones como economizar el coste de este proceso, la maximización de la calidad de los resultados o la imposibilidad de repetir la adquisición de las imágenes, estas imágenes necesitan ser almacenadas con la mayor calidad posible. Por otro lado, como consecuencia de la alta resolución de las imágenes y el gran número de ellas, la demanda de memoria suele ser muy alta. Con el objetivo de minimizar esto, las imágenes pueden ser comprimidas, pero debido a que la información de las imágenes es importante, se utilizan codecs sin pérdida, o muy próximos a serlo.

Otro aspecto importante a considerar es que en muchos casos, los repositorios de las imágenes se encuentran físicamente alejados de la mayoría de los usuarios que necesitan trabajar con dichas imágenes. Por ejemplo, en el proyecto JHelioviewer [1], un servidor central almacena un repositorio de imágenes de alta resolución del Sol que está en constante crecimiento, y los usuarios de forma interactiva pueden obtener y visualizar colecciones de estas imágenes usando Internet como medio de transmisión. Debido al tamaño (4096×4096 píxeles) de estas imágenes, la mayoría de los usuarios no son capaces de visualizar las imágenes completas en sus navegadores y necesitan de funcionalidades como panning, tilting y zooming. Además, estas acciones necesitan ser realizadas en tiempo real mientras las imágenes están siendo descargadas (streaming). Podemos ver un ejemplo de este tipo de interacción en la Figura 1.

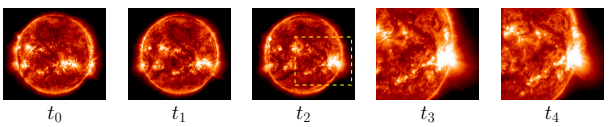

Fig. 1. Ejemplo de diferentes instantes de tiempo de una sesión de exploración de imágenes remotas con JHelioviewer. Al principio, el usuario indica la secuencia de imágenes y el frame-rate. La primera imagen se visualiza en el instante de tiempo t_0. En el instante de tiempo t_1 la imagen mostrada es ligeramente diferente de la primera, debido principalmente a la rotación del Sol. En el instante de tiempo t_2 el usuario especifica una WOI (Window Of Interest). t_3 y t_4 son los instantes de tiempo restantes, donde solamente se ha transmitido y visualizado la WOI especificada previamente.

La aplicación JHelioviewer ha sido diseñada para visualizar secuencias de imágenes de una forma continuada, como si se tratase de un vídeo. En el modo imagen, el refinamiento progresivo característico de JPEG2000 se utiliza para mostrar la WOI[2] definida por el usuario. Sin embargo, en el modo vídeo, con una cadencia que depende del frame-rate de la reproducción, se muestra la reconstrucción de una WOI de cada imagen, con una calidad que depende de la cantidad de datos recibidos para la WOI hasta el instante de la reproducción. Cuando la WOI de la última imagen de la secuencia ha sido visualizada, la reproducción vuelve a empezar desde el inicio de la secuencia. En esta segunda pasada, la calidad de la WOI de cada imagen suele ser mejor que en la

[1]Este trabajo ha sido financiado por becas del Ministerio de Ciencia e Innovación de España (TIN2008-01117 y TEC2010-11776-E), Junta de Andalucía (P08-TIC-3518 y P10-TIC-6548) y el Fondo Europeo de Desarrollo Regional (FEDER).

[2]Una WOI es una región rectangular de píxeles definida por $(s,(x,y),(w,h))$, donde s es el nivel de resolución espacial, (x,y) es la esquina superior izquierda y (w,h) es el ancho y alto de la WOI, en píxeles. s define la resolución de la imagen mostrada y estas coordenadas hacen referencia a esta imagen, que tiene una resolución de $X/2^s \times Y/2^s$, donde $X \times Y$ es la resolución original.

primera pasada porque se han recibido más datos. Este proceso de refinamiento finaliza cuando se ha transmitido todo el code-stream de la WOI o cuando el usuario selecciona una nueva WOI.

En el marco de trabajo descrito anteriormente, este trabajo presenta y evalúa una solución compatible con el estándar JPEG2000 para la tarea de obtener de forma interactiva secuencias de imágenes de alta resolución sobre enlaces de transmisión con ancho de banda variable. La sección II hace una introducción al estándar JPEG2000 y muestra algunos de los inconvenientes que presentan las técnicas más comunes utilizadas para resolver esta tarea. En la sección III, se realiza una comparativa entre nuestra propuesta y otras soluciones similares que se han realizado sobre este campo. La sección IV presenta nuestra propuesta, que es evaluada en la sección V. Este trabajo finaliza con algunas conclusiones y líneas de trabajo futuro, en la sección VI.

II. JPEG2000

El estándar JPEG2000 [2] es un codec de imágenes de dos etapas (transformada + codificación de la entropía). Primero, MCT (Multi-Component Transform) elimina la redundancia intercomponente y la transformada 2D DWT (Discrete Wavelet Transform) elimina la redundancia intracomponente. El resultado de esta transformada es un vector de matrices de conjuntos de coeficientes wavelet espacialmente no correlacionados que están organizados en un conjunto de subbandas que permiten una representación multiresolución de la imagen. En el siguiente paso, se aplica un codec de planos de bits basado en bloques llamado EBCOT (Embedded Block Coding with Optimal Truncation). El code-stream que se obtiene como resultado puede ser accedido de forma aleatoria con el objetivo de recuperar un determinado número de componentes (escalabilidad en componentes), WOI/resoluciones (escalabilidad espacial) y calidades (escalabilidad en calidad). Finalmente, si se necesita realizar un control del bit-rate, se puede utilizar una técnica como el algoritmo PCRD-opt (Post-Compression Rate-Distortion optimization) que selecciona aquellos paquetes que minimizan la distorisión de la codificación para un bit-rate específico. En la recuperación de imágenes remotas donde el enlace de transmisión suele ser el cuello de botella del sistema, los paquetes se envían siguiendo una progresión LRCP (Layer Resolution Precinct Component) porque es la progresión que ofrece menor distorsión durante la recepción del code-stream.

También se ha estandarizado [3] un protocolo de transmisión para la recuperación interactiva de imágenes remotas, llamado JPIP (JPEG2000 Interactive Protocol). Es importante resaltar que JPIP es un protocolo stateless y que los clientes no solicitan paquetes o data-bins[3], solicitan coordenadas de una WOI. Utilizando el protocolo JPIP, un cliente puede solicitar una imagen y controlar el flujo de datos (desde el servidor hacia el cliente) por medio de diferentes técnicas. Una de las más utilizadas (por ejemplo, en la implementación del estándar que ofrece Kakadu Software [4]) es una técnica stop-and-wait basada en la recuperación incremental del code-stream donde el cliente restringe el tamaño de las respuestas que quiere recibir del servidor utilizando el parámetro `len`. La Figura 2 muestra un ejemplo de uso de esta técnica, donde el cliente especifica que cada respuesta debe ser menor o igual que 2000 bytes. Cuando el cliente recibe una respuesta desde el servidor, éste vuelve a repetir el mismo proceso hasta que se hayan recibido todos los datos de la WOI seleccionada, ajustando los valores del parámetro `len` si fuese necesario.

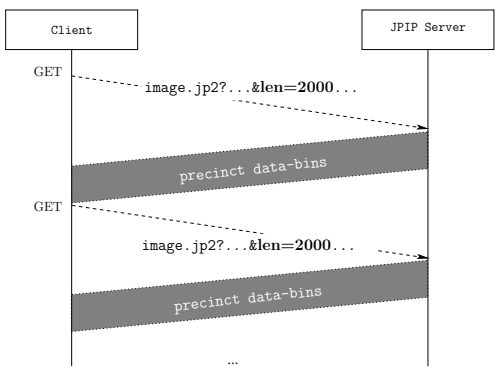

Fig. 2. Ejemplo de cómo recuperar una imagen utilizando la técnica de recuperación stop-and-wait.

Una extensión sencilla del procedimiento anterior se puede utilizar para recuperar una secuencia de imágenes si el cliente especifica en su petición un rango de imágenes. Siendo k_i el número de imágenes que se solicitan en cada i-th petición. La Figura 3 muestra un ejemplo de cómo recuperar una secuencia de 160 imágenes utilizando esta técnica. En este ejemplo, el cliente utiliza $k_i = 15$ imágenes en todas sus peticiones. Hay que tener en cuenta que dependiendo del tamaño del code-stream de las imágenes y del valor del parámetro `len`, pueden ser necesarias varias pasadas para poder transmitir completamente todas las imágenes que haya solicitado el cliente.

Claramente, k_i y $\texttt{len}_i$ tienen un alto impacto en la QoE (Quality of the Experience) del usuario quien básicamente lo que desea es poder visualizar la secuencia de imágenes con la mayor calidad posible a una velocidad de frame-rate determinada r. Tal y como se puede ver en la Figura 3, donde t_i es el tiempo de recepción del grupo de imágenes y RTT_{i+1} el Round Trip Time en la iteración $i+1$, si

$$\frac{k_i}{r} < t_i + \mathrm{RTT}_{i+1}, \tag{1}$$

el usuario sufrirá una pausa en la reproducción de la secuencia en el instante de tiempo $\sum_{x=0}^{i-1} t_x + k_i/r$.

[3]Una estructura de datos que contiene aquellos paquetes relacionados con la WOI solicitada y que son independientes, de modo que el servidor puede terminar la transmisión de la secuencia de paquetes en cualquier punto.

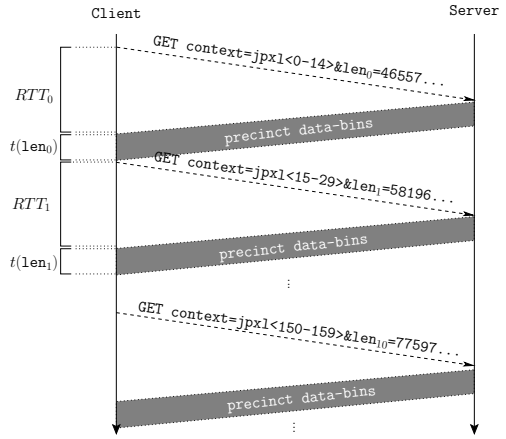

Fig. 3. Ejemplo de cómo recuperar una secuencia de imágenes utilizando la técnica de recuperación stop-and-wait.

III. Trabajos relacionados

La mayor parte de los primeros sistemas de exploración de imágenes remotas están basados en representaciones de pirámides de Gauss y Laplace y/o en el tiling en el dominio de la imagen para proporcionar operaciones de zooming, panning y tilting sobre las imágenes. Un ejemplo de esto puede ser la propuesta de Grunheit et al. [5] para realizar streaming interactivo de vistas panorámicas de alta resolución, donde las imágenes panorámicas son divididas en pequeños patches de 512×512 píxeles y son comprimidos de forma independiente con JPEG. Sin embargo, esta solución sólo tiene la capacidad de transmitir imágenes independientes (modo imagen), y no permite la transmisión de secuencias de imágenes (modo vídeo).

En [6], Naman and Taubman desarrollan un sistema interactivo escalable de vídeo basado en JPEG2000 (JSIV). Las secuencias de vídeo se almacenan como frames independientes comprimidos con JPEG2000 para proporcionar escalabilidad en calidad, espacial y en resolución, y se hace uso de predicciones y refresco condicional de code-blocks JPEG2000 para explotar la redundancia temporal. El servidor selecciona de forma óptima el mejor número de capas de calidad de cada code-block transmitido y el cliente intenta reconstruir los frames con la mayor calidad posible. Además, se presentó una extensión de este sistema en [7] basada en el uso de la compensación de movimiento para mejorar las predicciones en el lado del cliente.

Vetro et al. describen en [8] un sistema de streaming de vídeo escalable JPEG2000 sobre redes con un ancho de banda limitado, claramente orientado a sistemas de videovigilancia. Este sistema ofrece varios métodos de streaming y un algoritmo adaptativo de control del rate para realizar el transcoding JPEG2000 que adapta la calidad de la resolución de la escena en función del ancho de banda disponible.

Recientemente, en [9], Jiménez-Rodríguez et al. proponen un método de control del rate para la transmisión de secuencias de vídeo pre-codificadas con JPEG2000 sobre canales en los que puede variar su capacidad durante la transmisión del vídeo debido a congestiones de la red, fallos hardware o saturaciones en los routers.

Ninguna de las propuestas descritas anteriormente puede ser aplicada directamente sobre el contexto de JHelioviewer debido a que ninguna proporciona la flexibilidad que se requiere, porque estas técnicas están basadas en un escenario donde la secuencia de imágenes debe ser conocida previamente y no en un escenario donde el conjunto de imágenes es seleccionado de forma interactiva desde un repositorio. Otro aspecto importante a considerar en la transmisión de secuencias de imágenes (modo vídeo) es que el envío de datos sobre una red puede introducir retardos variables (jitter) que dificulten la sincronización entre el cliente y el servidor. Y. Hui et al. [10] consideran el uso de grandes buffers y un mecanismo de feedback para mantener el cliente y el servidor sincronizados. En su propuesta, el stream multimedia es modelado como un flujo de datos controlado por buffers con tres estados (high, medium, low) que determinan el rate de la transmisión que se ha solicitado y si es el rate apropiado.

En [11] se especifica un protocolo de sincronización para garantizar una adecuada visualización de un stream multimedia en el cliente. En esta propuesta, el ciente está basado en un esquema de control que utiliza un umbral superior (BOH) y un umbral inferior (BOL) para controlar el nivel de ocupación del buffer.

Ninguno de los esquemas de sincronización que se han mencionado anteriormente son compatibles con el estándar JPIP y por lo tanto no pueden ser aplicados fácilmente al escenario que estamos considerando en nuestra propuesta. Nuestra propuesta está basada en un esquema de sincronización sencillo pero eficiente totalmente compatible con el estándar JPIP donde el cliente es el único que controla el estado del buffer y envía mensajes de feedback al servidor utilizando parámetros del estándar JPIP.

IV. Propuesta

Hemos modificado la técnica stop-and-wait por otra técnica más eficiente basada en un pipeline. Por lo tanto, en el lado del cliente, cuando el usuario especifica una WOI, el cliente envía solamente una petición. Por ejemplo, para reproducir un vídeo de 160 frames a 20 fps con un ancho de banda de 10 Mbits/segundo, el cliente solamente tendría que enviar al servidor la petición que se muestra en la Figura 4.

En el lado del servidor hemos considerado que las peticiones de una WOI son acumulativas, de modo que el cliente no tiene que volver a reenviar algunos de los parámetros que ya han sido enviados en cada una de las peticiones, debido a que estos parámetros se mantienen a lo largo de toda la sesión.

Considerando los comentarios anteriores, hemos propuesto una técnica que permite controlar el flujo de datos basándonos en la estimación del ancho de

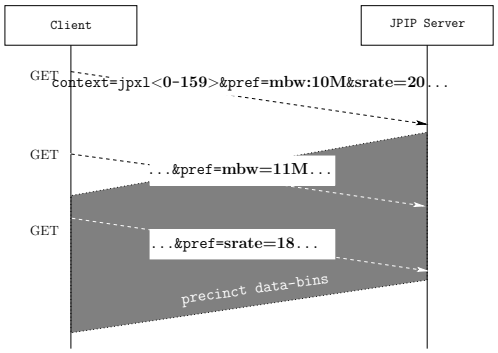

Fig. 4. Ejemplo de cómo recuperar una secuencia de imágenes utilizando una técnica basada en un pipeline.

banda disponible entre el servidor y el cliente, y la velocidad a la que se va a reproducir la secuencia en el cliente. Los resultados experimentales demuestran que nuestra propuesta es efectiva y totalmente compatible con el estándar JPIP, debido a que está basada en el uso de los parámetros `mbw` y `srate` para transmitir vídeo con el protocolo JPIP. El parámetro `mbw` (Maximum Bandwidth), indica el máximo rate con el que el cliente espera recibir los datos, expresado en bits/segundo y `srate` (Sampling Rate) se utiliza para indicar que la media de la tasa de muestreo por segundo no puede ser mayor que este valor. Estos parámetros se han definido para controlar flujo de datos de una transmisión de un vídeo Motion JPEG2000 (muy similar a la transmisión de una secuencia de imágenes JPEG2000). El parámetro `len` no se utiliza en esta técnica.

Con el objetivo de entender cómo se utilizan estos parámetros, vamos a suponer que el servidor recibe una petición para tranmitir una secuencia de imágenes que van a ser visualizados con un frame-rate de 20 imágenes/segundo (`srate`=20) y la estimación del ancho de banda disponible es de 10 Mbits/segundo (`mbw`=10). En esta situación el servidor debería enviar:

$$\frac{10\frac{\text{Mbits}}{\text{second}}}{20\frac{\text{pictures}}{\text{second}}} = 0.5 \text{ Mbits/picture.}$$

En la implementación descrita anteriormente, el cliente le comunica al servidor el ancho de banda disponible cada segundo y también le comunicará cualquier modificación que se produzca en el valor del frame-rate con el que se va a reproducir la secuencia de imágenes.

Se han establecido dos modos de funcionamiento en el cliente y en el servidor, dependiendo de los parámetros que se envían en las peticiones del cliente.

Modo Imagen. Este es el modo de funcionamiento por defecto y se activa cuando se abre el vídeo por primera vez o cuando el vídeo es pausado por el usuario. En este modo el servidor envía las imágenes completas, de manera que no envía ningún paquete de la siguiente imagen hasta que no ha enviado todos los paquetes de la imagen actual.

Modo Vídeo. Este modo se activa cuando el usuario inicia la reproducción del vídeo y necesita que que las peticiones que se envían al servidor contengan los parámetros `mbw` y `srate`. Estos parámetros sólo se pueden utilizar en el modo vídeo. En este caso, el servidor calcula el número de bytes que se deben enviar para cada imagen y envía aproximadamente la misma cantidad de bytes de cada una de ellas.

A. El Cliente

Uno de los principales problemas en el streaming interactivo es la sincronización entre el cliente y el servidor de modo que el cliente sea capaz de tener suficientes datos de las imágenes en su buffer para mantener la velocidad de reproducción de la secuencia a lo largo de toda la transmisión, debido a los retardos de la red provocados por el jitter. Para resolver este problema se ha propuesto un algoritmo de sincronización implementado en el lado del cliente basado en la estimación del ancho de banda, teniendo en cuenta las estimaciones previas y los errores cometidos. Todas las operaciones de este proceso se realizan en el lado del cliente con el objetivo de aliviar la carga del servidor liberándolo de esta tarea y por lo tanto mejorar la escalabilidad del servidor.

A continuación se presenta una descripción del algoritmo utilizado por el cliente para realizar la estimación del ancho de banda.

Entrada:

$s(R_i)$	Tamaño de la última respuesta R_i
$t(R_i)$	Tiempo necesario para recibir R_i
$w = 0.5$	Factor de corrección
λ	Tiempo de buffering
$\alpha = 0.1$	Peso para la medida del ancho de banda actual
$\texttt{mbw}_0$	Ancho de banda durante la transmisión de los metadatos
P	Velocidad de reproducción
N	Número de imágenes en el buffer

Salida:

$\texttt{mbw}_i$	Ancho de banda estimado

Algoritmo:

```
# Última medida del ancho de banda
(1) B = s(R)/t(R)
# Media ponderada del ancho de banda
(2) mbw_i = αB + (1 − α)mbw_{i−1}
# Media del error del ancho de banda
(3) ε_i = (ε_{i−1} + |B − mbw_i| / max(B, mbw_i)) / 2
# Tamaño actual del buffer en segundos
(4) λ' = N/P
# La estimación del ancho de banda ha sido muy alta
(5) if(λ' < λ) then
(6)     mbw_i = mbw_i(1 − w(λ'/λ + 1))
# La estimación del ancho de banda ha sido muy baja
(7) else if(λ' > λ) then
(8)     mbw_i = mbw_i(1 + ε_j)
(9) end if
```

En nuestra implementación, este algoritmo se ejecuta de forma periódica cada segundo, aunque este periodo puede ser ligeramente superior dependiendo de $t(R_i)$. Este tiempo depende del ancho de banda real que se ha usado en la respuesta R_i y de $s(R_i)$. El primer factor es incontrolable (es una consecuencia directa de la carga de la red) y el segundo depende del tamaño de los últimos data-bins recibidos.

B. El Servidor

Para cada petición, el servidor envía un conjunto de data-bins de un conjunto de imágenes. Estas imágenes son especificadas por el cliente haciendo uso del parámetro `context` como se puede ver en la Figura 4. El número de data-bins que el servidor envía de cada imagen depende del valor de los parámetros `srate` y `mbw`. Tal y como muestra la siguiente expresión, el servidor envía aproximadamente (dependiendo del tamaño de los data-bins) el siguiente número de bits de cada imagen: $\frac{\texttt{mbw}}{\texttt{srate}}$.

Aunque este procedimiento de control del bit-rate no es el más óptimo, debido a que con poca frecuencia los code-streams de las imágenes son truncados exactamente al final de una capa de calidad y debido a que no existe una priorización de los paquetes durante la transmisión para ofrecer la mínima distorsión en la reconstrucción de las imágenes, se ha seleccionado este método por dos razones: (1) los requisitos computacionales del servidor son muy bajos y (2) en la práctica este método funciona bastante bien porque en la mayoría de los casos el punto de truncado está muy próximo a un punto de truncado óptimo de una capa de calidad, si existen suficientes capas de calidad.

V. Evaluación

Para llevar a cabo los experimentos, se han utilizado dos escenarios de red reales con diferentes capacidades de ancho de banda. El escenario S_1 representa una conexión con alto ancho de banda entre cliente y servidor, 1.5 MB/s, y el escenario S_2 representa un escenario con poco ancho de banda, 120 KB/s. Las secuencias de vídeo utilizadas en los experimentos han sido dos archivos JPX, *Sun* y *Stockholm*, formados por un conjunto de imágenes diferentes. *Sun* es una secuencia de imágenes del Sol formada por 1000 frames de 4096×4096 píxeles con 8 bits/componente y comprimidas con JPEG2000 utilizando una compresión con path irreversible, con 8 capas de calidad y 9 niveles de resolución. *Stockholm* es una secuencia que ofrece una vista panorámica de la ciudad de Estocolmo [12]. La secuencia está compuesta por 604 imágenes de 1280×720 con 8 bits/componente, y comprimidas con JPEG2000 utilizando una compresión con path irreversible, con 8 capas de calidad y 6 niveles de resolución.

En el experimento 1 hemos estudiado el impacto del uso de precintos en la secuencia *Sun* cuando es enviada a través de los escenarios S_1 y S_2 utilizando la técnica basada en pipeline. Las imágenes han sido codificadas con precintos de 128×128 y sin precintos. En el lado del cliente se ha establecido un frame-rate de 10 frames/segundo y no se ha utilizado buffering.

La Figura 5 muestra los resultados de la exploración del impacto el uso de precintos. Se puede observar que el uso de precintos mejora el valor medio del PSNR en ambos escenarios.

La Figura 6 muestra los resultados del experimento 2, que examina el impacto del número de capas de

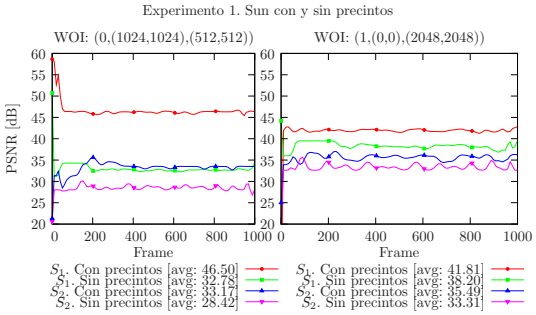

Fig. 5. Experimento 1. Estudio del impacto del uso de precintos sobre los escenarios S_1 y S_2.

calidad en la secuencia *Sun* cuando es transmitida a través de los escenarios S_1 y S_2, utilizando la técnica basada en pipeline. En el lado del cliente se ha establecido un frame-rate de 5 frames/segundo y no se ha utilizado buffering.

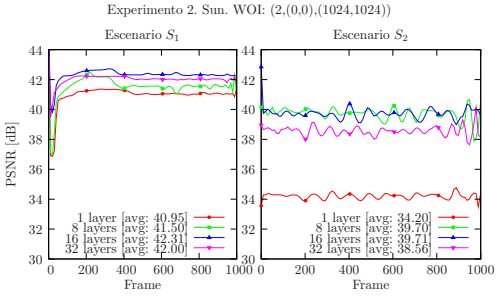

Fig. 6. Experimento 2. Estudio del impacto del número de capas de calidad sobre los escenarios S_1 y S_2.

El experimento 3 examina el comportamiento de las dos técnicas, stop-and-wait y pipeline, con el objetivo de hacer una comparación del impacto sobre la calidad de la imagen cuando el vídeo se reproduce a diferentes frame-rates y no existe buffering en el cliente. En este experimento se han transmitido ambas secuencias, a través de los escenarios S_1 y S_2. Las imágenes fueron codificadas con 8 capas de calidad y con precintos de 128×128. En el lado del cliente se ha evaluado el impacto de utilizar frame-rates de 5, 10, 15 y 20 frames/segundo.

En las Figuras 7 y 8 se puede observar que la técnica propuesta puede mejorar la calidad de las imágenes respecto a la técnica stop-and-wait, para todos los frame-rates que han sido testeados.

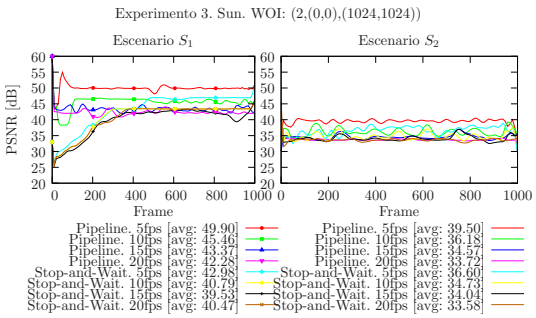

Fig. 7. Experimento 3. Una comparativa entre las dos técnicas en ambos escenarios S_1 y S_2.

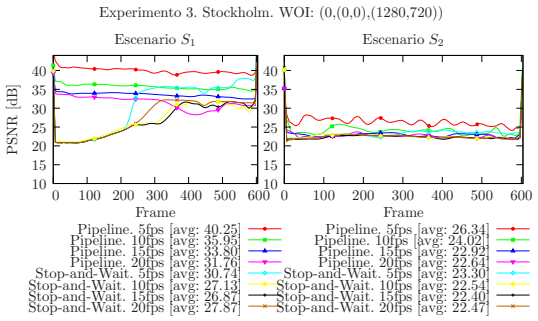

Fig. 8. Experimento 3. Una comparativa entre las dos técnicas en ambos escenarios S_1 y S_2.

El experimento 4 examina el impacto del uso de un buffer con diferentes tamaños, cuando ambas secuencias de imágenes son transmitidas sobre los escenarios S_1 y S_2, utilizando la técnica basada en pipeline. Las imágenes han sido codificadas con 8 capas de calidad y con precintos de 128×128. El frame-rate utilizado en el cliente ha sido de 5 y 10 frames/segundo. Las Figuras 9 y 10 muestran los resultados de cada uno de los experimentos realizados.

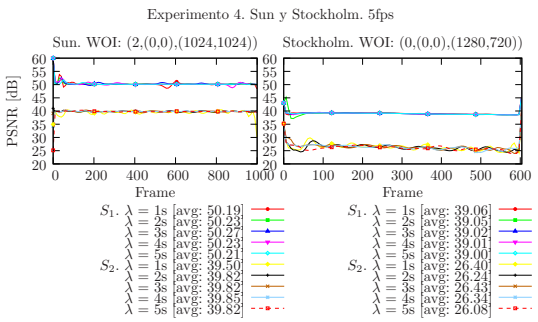

Fig. 9. Experimento 4. Estudio del impacto del uso de un buffer con diferentes tamaños.

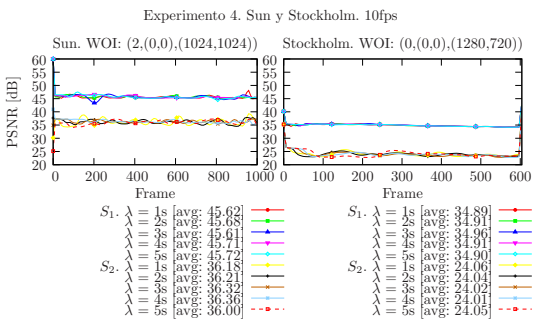

Fig. 10. Experimento 4. Estudio del impacto del uso de un buffer con diferentes tamaños.

VI. Conclusiones

Los resultados de los experimentos demuestran que nuestra propuesta garantiza el mismo frame-rate durante toda la reproducción incluso en situaciones donde se produzcan variaciones significativas del ancho de banda disponible y se disponga de un tamaño reducido del buffer. Nuestra propuesta es efectiva y totalmente compatible con el estándar JPIP.

Como línea de trabajo futuro se plantea mejorar el método que realiza el control del rate en el servidor.

Referencias

[1] D. Müeller, B. Fleck, G. Dimitoglou, B. W. Caplins, D. E. Amadigwe, J. P. Garcia Ortiz, A. Alexanderian B. Wamsler, V. Keith Hughitt, and J. Ireland, "JHelioviewer: Visualizing large sets of solar images using JPEG 2000," *Computing in Science and Engineering*, vol. 11, no. 5, pp. 38–47, September 2009.

[2] International Organization for Standardization, "Information Technology - JPEG 2000 Image Coding System - Core Coding System," ISO/IEC 15444-1:2004, September 2004.

[3] International Organization for Standardization, "Information Technology - JPEG 2000 Image Coding System - Interactivity Tools, APIs and Protocols," ISO/IEC 15444-9:2005, November 2005.

[4] "Kakadu JPEG 2000 SDK," `http://www.kakadusoftware.com`.

[5] C. Grunheit, A. Smolic, and T. Wiegand, "Efficient representation and interactive streaming of high-resolution panoramic views," in *Image Processing. 2002. Proceedings. 2002 International Conference on*, 2002, pp. III–209 – III–212.

[6] Aous Thabit Naman and David Taubman, "Jpeg2000-based scalable interactive video (jsiv)," *IEEE Transactions on Image Processing*, vol. 20, pp. 1435– 1449, May 2011.

[7] A.T. Naman and D. Taubman, "Jpeg2000-based scalable interactive video (jsiv) with motion compensation," *Image Processing, IEEE Transactions on*, vol. 20, no. 9, pp. 2650 –2663, sept. 2011.

[8] Anthony Vetro, Derek Schwenke, Toshihiko Hata, and Naoki Kuwahara, "Scalable video streaming based on jpeg2000 transcoding with adaptive rate control," *Adv. MultiMedia*, vol. 2007, pp. 7–7, January 2007.

[9] Member Leandro Jiménez-Rodríguez, Francesc AulÃ--Llinás and Michael W. Marcellin, "Fast rate allocation for jpeg2000 video transmission over time-varying channels," *IEEE Transactions on Multimedia*, vol. 15, pp. 15–26, January 2013.

[10] Jun Li Joseph Y. Hui, Ezhan Karasan and Junbiao Zhang, "Client-server synchronization and buffering for variable rate multimedia retrievals," *IEEE Journal on Selected Areas in Communications*, vol. 14, pp. 226–237, January 1996.

[11] Mourad Daami and Nicolas D. Georganas, "Client based synchronization control of coded data streams," *IEEE International Conference on Multimedia Computing and Systems*, pp. 387–394, June 1997.

[12] "Stockholm video sequence," `ftp://ftp.ldv.e-technik.tu-muenchen.de/pub/test_sequences/720p/720p5994_stockholm_ter.yuv`.