Improved JPIP-compatible architecture for video streaming of JPEG 2000 image sequences

J.J. Sánchez-Hernández, J.P. García-Ortiz, C. Martin, Carmelo Maturana-Espinosa and Vicente González-Ruiz
Computer Architecture and Electronics Department
University of Almería, Spain

D. Müller
European Space Agency, ESTEC
Noordwijk, Netherlands

Abstract— **This paper[1] presents a new architecture for video streaming of remote sequences of JPEG 2000 images, fully compatible with the JPIP protocol. This architecture has been developed as an improvement for the JHelioviewer project during the first Summer of Code organized by the European Space Agency. It requires only a slightly modification in the way the client/server communication is carried out, but within the standard scope. Its main feature is to allow an efficient video streaming of image sequences, with the possibility of panning/zooming movements during the reproduction. The experimental results show that the developed improvement achieves better quality and responsiveness values with respect to the original approach.**

Keywords— **JPIP, JPEG2000, video streaming.**

I. Introduction

THE powerful features offered by the novel JPEG 2000 multi-part standard [1], like lossless/lossy compression, random access to the compressed streams, incremental decoding or high degree of spatial and quality scalability, have led it to obtain the recognition as a state-of-the-art solution among applications for remote browsing of high-resolution images.

JPEG 2000, in combination with the JPIP protocol [2] defined in its Part 9 [3], has already been successfully used in many scientific areas (e.g. telemicroscopy [4] or tele-medicine [5]), and it has a significant potential in any other one where large volumes of image data need to be streamed, like for example Google Earth/Maps.

Specially in the scientific context, the images used by the remote browsing applications are not usually isolated, but they belong to a time sequence. For example, in the case of the tele-pathology area, an image is related to a tissue slide, and also belongs to a sequence of images representing the evolution of that tissue for a period of time.

In the astronomy context, one of the most representative examples is the JHelioviewer project [6], developed by the European Space Agency (ESA) in collaboration with the National Aeronautics and Space Administration (NASA). The main data served by this application is obtained from the Solar Dynamics Observatory [7], which provides, among other data products, full-disk images of the Sun taken every 10 seconds in eight different ultraviolet spectral bands with a resolution of 4096×4096 pixels.

In this kind of applications the user is commonly interested in observing not only an image alone but also the associated sequence as a video stream. A possible approach for developing this functionality is to generate a video file on demand, according to the sequence selected by the user, streaming then its content. The main drawback of this solution is that the main features offered by JPEG 2000 for remote browsing (zooming and panning of the content, efficient transmission of windows of interest, etc.) are lost. Moreover, the scalability on the server side is negatively affected by the overhead of this procedure.

When a user selects a sequence of images to browse in JHelioviewer, a kind of video file is created, where each frame is actually a hyper-link to an existing JPEG 2000 image file. This video file is encapsulated under a standard multi-image file format, so any JPIP server can work with it. Therefore, the user can either browse each frame independently, or reproduce the sequence as a common video with a defined frame-rate. In both cases the powerful features of the standard can be exploited by the user, being possible, for instance, to make a zoom during the video reproduction; the transmission is also adapted according to the window of interest requested by the client. A minimal data overhead is generated in the server side due to the creation of the hyperlinked video file, and the scalability is not reduced at all.

The approach of JHelioviewer for video streaming of image sequences is fully JPIP-compatible, but not completely efficient, specially because of the protocol itself. JPIP offers a basic functionality for video streaming, but mainly oriented for Motion JPEG 2000 files. It does not provide enough flexibility for dealing with this type of developments, as well as its definition is far from complete in these cases.

This paper presents the work carried out during the first Summer of Code of the European Space Agency. Its goal has been to modify the JPIP architecture of JHelioviewer in order to improve the video streaming, maintaining a fully compatibility with the standard, although giving an alternative meaning to some of its definitions. The results show a significant gain in responsiveness, allowing the user to start playing the video almost immediately, without interruptions or initial preloading. The average video

[1]This work has been funded by grants from the Spanish Ministry of Science and Innovation (TIN2008-01117 and TEC2010-11776-E) and Junta de Andalucía (P08-TIC-3518 and P10-TIC-6548), in part financed by the European Regional Development Fund (ERDF).

quality, during its reproduction, is also noticeably improved.

The rest of the paper is organized as follows: in Section II related work is analyzed. Section III is dedicated to explaining the proposed solution, which is later evaluated in Section IV. The paper ends with some conclusions and future work (Section V).

II. Related work

There do not exist works in the literature related to the context of the problem explained in the introduction section (video-streaming using JPEG 2000 standard in JHelioviewer). Anyway, we have found several works focused on video-streaming using the same standard which are described in the following lines.

Naman and Taubman [8] show a JPEG 2000-Based Scalable Interactive Video (JSIV) system. It relies on the following concepts: video sequences are stored as independent JPEG 2000 frames to provide quality and spatial resolution scalability, prediction and conditional replenishment of JPEG 2000 code-blocks to exploit temporal redundancy and loosely coupled server and client policies. The server optimally selects the best number of quality layers for each code-block transmitted and the client attempts to produce the best possible reconstructed frames from the data.

In addition, an extension of the JSIV system was presented based on the use of motion compensation to improve prediction. Experimental results confirmed the efficacy of JSIV when motion compensation is employed. In general, prediction is improved whenever the actual underlying motion can be modeled reasonably well.

JSIV, with or without motion compensation, provides meaningfully better interactivity compared to existing streaming schemes. This improvement is due to the flexible prediction policy.

In [9], Lee and Qiao propose an efficient rate control algorithm based on TCP-Friendly Rate Control (TFRC) and the stream scaler for streaming Motion-JPEG 2000 video over the Internet. TFRC is a protocol to solve the network congestion problem based on the computation of the next sending rate by means of the throughput of TCP connection. The stream scaler algorithm presents a useful feature of quality scalability that divide the video into multiple layers with a certain range of bit-rate, so each layer can be adapted to a required network bandwith for the transmission. According to the sending rate, the rate controller determines which is the most appropriate layer to be sent.

The results of the experiments show that this approach adaptively controls the rates of Motion-JPEG 2000 video, consequently network congestion results only in a graceful degradation of video quality.

Vetro et al. describe in [10] a scalable video streaming system based on JPEG 2000 standard over limited bandwidth networks oriented towards surveillance systems. It offers several streaming methods and an adaptive rate control algorithm for JPEG 2000 transcoding.

Respecting to the streaming methods, we can find the following ones: *frame-by-frame* where the spatial quality of regions of interest (ROIs) and background are controlled in a frame-by-frame manner, *background refresh* where the ROIs are transmitted successively with an occasional background refresh, therefore ROIs are superimposed on the background, and *mosaic streaming*, which is similar to *background refresh* mode, but it superimposes successive ROI images on a background in a mosaic style. It is really useful for behavior analysis and scene browsing.

The rate control algorithm allocates rate to each frame in a sequence based on target rate, buffer occupancy and ROI information. The key components include *variable rate allocation* introducing a buffer to absorb the variations in allocated rate to each frame, *frame skipping* where periodic frames with no ROI defined are skipped driving the buffer level towards its lower margin for future frames with ROI information and *quality stabilization* to establish a period in which the quality layers will be held stable. The experimental results show that the algorithm outperforms the reference uniform rate control method. The complexity of the transcoding technique is very low.

Lastly, Itakura et al. [11] present a JPEG 2000 based real-time scalable video communication system developed in Sony, named as "BEAM" video system. The objectives of this system are: scalable video communication and real-time communication. Internet Engineering Task Force (IETF) streaming protocols, with their methods based on Real-time Transport Protocol (RTP) and Real Time Streaming Protocol (RTSP) for the delivery of bit-streams with real-time requirements, are extended to handle scalable delivery of video data from a single layered coding data to heterogeneous devices and different resolution display such as HDTV, standard TV, PDA and mobile phone.

Real-time ARQ (RT-ARQ) is proposed to solve the QoS issue for real-time applications because of the low coding delay JPEG 2000 codec. Moreover, various network adaptive control techniques, rate control, Forward Error Correction (FEC) and RT-ARQ error controls are included in the system to achieve high transmission quality.

All these approaches can not be directly applied to the JHelioviewer context according to how the data is handled and transmitted. Moreover, many of these commented works do not maintain a fully compatibility with the JPIP protocol.

III. Proposal

JHelioviewer is based on a client-server architecture. The communication between the client-side browser and the JPIP server is based on request and response messaging using JPIP on top of HTTP. The target JPEG 2000 images are stored in an image repository, while metadata extracted from header in-

formation is saved in a metadata repository, so users can search for the metadata to locate data of interest.

Using JPIP, a client can request a sequence of images by means of different techniques. One of the most used (for example, in `kdu_show` and `kdu_server` [12]) it is based on the incremental retrieval of JPEG 2000 images. In this technique, the client controls the amount of data that the server sends using a parameter in the request that specifies the maximun size of the response from the server. For example, when the client sends the message:

```
GET /image.jp2?rsiz=512,512&fsiz=1024,
1024&len=2000...
```

the client is requesting a WOI of the image `image.jp2` with a size of 512×512 placed within the maximum resolution level with a size equal or smaller than 1024×1024. It is also specified that the response should be equal or less than 2000 bytes. When the client receives the whole response from the server, it will repeat the same process until the desired WOI is complete, ranging the `len` parameter if necessary for balancing the data flow.

A simple extension of the last procedure can be used to send a sequence of images. In this context the client specifies in its request a range of images (instead of a single image) and a maximum response size. JHelioviewer sends different requests of 15 frames in each one, or less if the total number of them is not multiple of 15.

For example, for an image sequence of 160 frames, the client could send the next messages:

```
GET context=jpxl<0-14>&len=46557...

GET context=jpxl<15-29>&len=58196...
...

GET context=jpxl<150-159>&len=177597...
```

In this case, the server sends approximately the same amount of data for each frame of the range specified in every request, although this is not specified in the JPIP standard.

Unfortunately, this technique, based on a stop-and-wait data-flow control algorithm, has several drawbacks when we are performing an interactive retrieval of a sequence of images. The first one is related to the overhead produced by the control because a large number of requests is generally needed to retrieve the total sequence. A second disadvantage is focused on the quality of the reconstructed video at the receiver because it is quite difficult the synchronization between the player and the server. For this reason, the quality of the reconstructed images (that is proportional to the amount of data received per image) can be very variable and therefore, unpleasant for the user due to the oscillations of the available bandwidth.

Considering the previous comments, we propose a data-flow control algorithm based on an estimation of the available bandwidth between the server and the client, and the picture rate of the sequence at the receiver. As the experimental results demonstrate, our proposal is effective and fully adapted to JPIP, as it uses the standard parameters [2]: `mbw` (maximun bandwidth) and `srate` (sampling rate), which are defined to control the data-flow of the transmission of Motion JPEG 2000 video (very similar to the transmission of JPEG 2000 image sequences). The `len` parameter is not used in our data-flow control algorithm.

In order to understand how these parameters are used, let's suppose that the server receives a request of a sequence of frames to be displayed using a frame-rate of 20 pictures/second (`srate`=20) and the estimation of the available bandwidth is 10 Mbits/second (`mbw`=10). In this situation the server should send

$$\frac{10^7 \frac{\text{bits}}{\text{second}}}{20 \frac{\text{images}}{\text{second}}} = 0.5 \text{ Mbits/image.}$$

In the described implementation, the client communicates to the server the available bandwidth and the picture-rate in the following situations:

1. When the client observes a significant difference between the estimated bandwidth and the real one.
2. When the user changes the frame-rate of the image sequence.

To solve the problem with the overhead produced by the large number of requests performed to retrieve the total sequence, we have modified the communication schema taking into account that the server response is always segmented in chunks (usually with a length of $1KB$). Therefore, on the client side, when the user specifies a WOI, the client sends only one request. For example, in order to play a 160-frames video at 20 fps with a bandwidth of 10 Mbits/second, JHelioviewer will send just one request to the server as follows:

```
GET context=jpxl<0-159>&pref=mbw:
10M&srate=20&...
```

On the server side, the server checks if a new request is received just after sending each chunk. If a new WOI request is received, it will send an empty chunk, in order to complete correctly the current response, and it will attend the new request. We have considered the WOI requests as cumulative, so the client wouldn't have to repeat some parameter values in each request, because the parameter values can be kept along the same session.

The indices of the compositing layers associated with a codestream context can be supplied in two ways. For example, if we have a JPX file with 150 frames we can do the next queries:

```
GET context=jpxl<0-149>&...
```

1) In this case we have a range which first value is smaller than the second one. The server will send all the images of the JPX file from first (0) to last (149).

```
GET context=jpxl<20-19>&...
```

2) In this case we have a range which first value is greater than the second one. The server will send all the images of the JPX file using a circular mode.

We have established two operational modes on client and server side: image and video mode.

- **Image Mode**
  Client: This mode is enabled when an user opens a new video or the video is paused by the user (This is the default operating mode). The queries in this mode not contain the client preferences: `mbw` (max bandwidth) and `srate` (sampling rate). These client preferences only can be used in video mode.
  Server: In this case, the server not send any packets from the next frame until it has sent all the packets from the current frame.

- **Video Mode**
  Client: This mode is enabled when the video is playing. In this mode the queries must contain the client preferences: `mbw` (max bandwidth) and `srate` (sampling rate).
  The client needs to estimate the arrival time of each packet and calculate an estimation of the bandwidth. This estimated bandwidth will be notified to the server when ocurrs a variation that exceed a threshold established by the client. The client also must notify to the server if the user has changed the frame-rate.
  Server: In this case, the server calculates the number of bytes that must be sent for each frame.

IV. Evaluation

In this section a comparison between the original `esa_jpip_server` and our proposal implemented on the `esa_jpip_server` has been carried out. The experiments show the quality of the reconstructed video when the user plays a video sequence without interruptions i.e., there is no interactivity.

The experiments have been performed in a simulated scenario. Both, the original `esa_jpip_server` and the improved `esa_jpip_server` were running on a machine at the University of Almería, Spain. The client was the JHelioviewer browser [6], running on the same network. The available bandwidth between client and server has been controlled with `trickle` [13], a lightweight userspace bandwidth shaper.

The video used in the experiments is a linked JPX file created with 444 frames of the Sun of 4096×4096 pixels with 8 bits/component, and compressed with JPEG 2000 using the irreversible path, with 8 quality layers and 9 resolution levels with RPCL progression. The precinct sizes are set as 128×128. The images were captured by the SDO Observatory using the AIA Instrument, in the 17.1 nm channel, between the time 00:00:00 of the day 2011/05/24 and the time 00:00:00 of the day 2011/05/26. A time step of 3 minutes was selected.

The sequence of WOIs $(x, y, width, height, r)$ that has been used during the visualization of the video is as follows: $(0, 0, 512, 512, 5)$, $(0, 0, 512, 512, 6)$, $(127, 127, 639, 639, 6)$. We refer to r as the resolution of the image, with $r = 0$ corresponding to the lowest avilable resolution and $r = D$ corresponding to the original image resolution. D is the number of wavelet decomposition levels, or stages.

In the experiments, the user requests the defined image sequence using a WOI and waits until the complete sequence has been received (after several passes depending on the selected frame-rate and the available bandwidth). The download and upload bandwidth consumption have been set to 150 KB/s and 64 KB/s, and the video frame rate has been set to 20 frames per second.

In the context of the remote browsing systems it is very interesting to evaluate the quality of the reconstruction of the served images, measured by means of the PSNR [dB] versus the time [seconds] when the images have been displayed at the receiver. This measure is related to the user experience because the user always wants to see the best quality as soon as possible.

Figures 1, 2 and 3 show that our proposal can achieve better PSNR values in less time, which translates to a better viewing experience. The results show that the original `esa_jpip_server` can't display the video at 20 frames per second with the available bandwidth during the first seconds of play, while the improved `esa_jpip_server` gets better PSNR values along the transmission. When the PSNR value of the video stream at the client is between 10 and 20 dBs, a black frame is shown at the client because it has not yet received any data from the server.

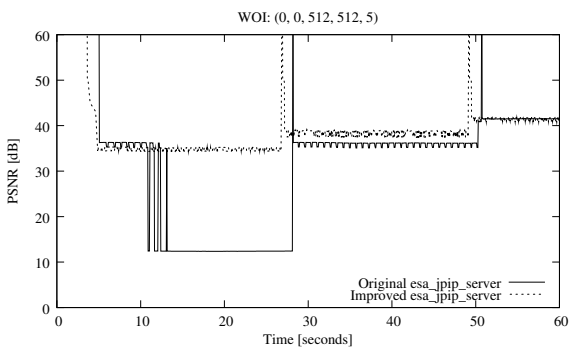

Fig. 1. Experiment 1. WOI located in the coordinates $(0, 0, 512, 512)$ with a resolution image of 512×512

V. Conclusions

The experiments results show that our proposal implemented on the ESA JPIP server achieves better quality and responsiveness values respect to the original approach. Moreover, our proposal is effective and fully compliant with the JPIP standard, because it based on the use of parameters which are defined to control the data-flow of the transmission of Motion JPG 2000 video.

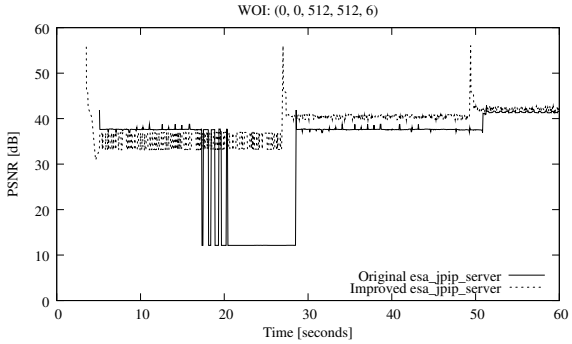

Fig. 2. Experiment 2. WOI located in the coordinates $(0, 0, 512, 512)$ with a resolution image of 1024×1024

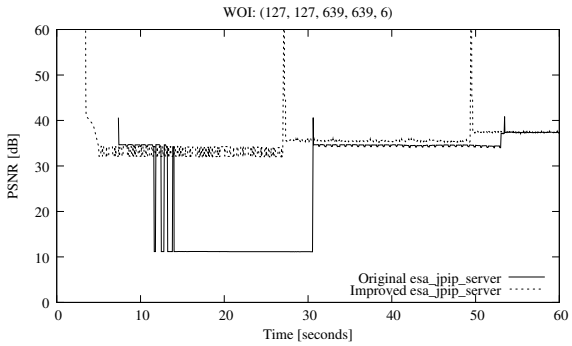

Fig. 3. Experiment 3. WOI located in the coordinates $(127, 127, 639, 639)$ with a resolution image of 1024×1024

References

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

[2] D. S. Taubman and R. Prandolim, "Architecture, Philosophy and Perfomance of JPIP: Internet Protocol Standard for JPEG2000," in *International Symposium on Visual Communications and Image Processing*, Julio 2003, vol. 5150, pp. 649–663.

[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] V. Tuominen and J. Isola, "The application of JPEG 2000 in virtual microscopy," *Journal of Digital Imaging*, 2007.

[5] K. Krishnan, M.W. Marcellin, A. Bilgin, and M.S. Nadar, "Efficient transmission of compressed data for remote volume visualization," *IEEE Transactions on Medical Imaging*, vol. 25, pp. 1189–1199, September 2006.

[6] 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.

[7] W. Pesnell, "The Solar Dynamics Observatory: Your eye on the Sun," in *37th COSPAR Scientific Assembly*, 2008, vol. 37 of *COSPAR, Plenary Meeting*, pp. 2412–+.

[8] 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.

[9] M.H. Lee and Rong-Yu Qiao, "Rate control with stream scaling for motion-jpeg2000 video over the internet," in *Broadband Multimedia Systems and Broadcasting, 2008 IEEE International Symposium on*, 31 2008-april 2 2008, pp. 1 –5.

[10] 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.

[11] E. Itakura, S. Futemma, Guijin Wang, and K. Yamane, "Jpeg2000 based real-time scalable video communication system over the internet," in *Consumer Communications and Networking Conference, 2005. CCNC. 2005 Second IEEE*, jan. 2005, pp. 539 – 543.

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

[13] "Trickle: A lightweight userspace bandwidth shaper," `http://monkey.org/~marius/trickle`.