Architectural Model for Visualization of High Definition Images on Mobile Devices

Germán Corredor[1], Daniel Martínez[1], Eduardo Romero[1] and Marcela Iregui[2*]
[1]*Universidad Nacional de Colombia, Carrera 30 # 45-03, Bogotá D.C., Colombia*
[2]*Universidad Militar Nueva Granada, Carrera 11 # 101-80, Bogotá D.C., Colombia*

Keywords: Architecture, Decoding, Images, Interaction, JPEG2000, Mobile Devices, Protocol, Multimedia, Streaming, Visualization.

Abstract: In recent years, the mobile device demand has largely increased because of the accessibility, ubiquity and portability of such devices, which are being used not only for personal purposes but also in several applications like education, science, entertainment, commerce and industry, among others. Visualization and interaction with high definition multimedia content, like large images and videos, using mobile devices, represents a challenge because of their very limited machine resources and bandwidth. For such application, this content requires special treatment so that users can properly access and interact. In this article, it is proposed an architectural model for efficient streaming and visualization of very large images on mobile devices using the JPEG2000 standard and an adapted image transfer protocol. Results show that the introduced architecture is effective for visualizing regions of large images and presents good performance, both for transmission and decoding processes, allowing a simple and dynamic interaction between user and images.

1 INTRODUCTION

Real-time access to information has become an important task in different daily activities; this is why the use of portable devices and mobile services has increased in recent years. Portability and last hardware advances of mobile devices have induced people to use several mobile applications for managing personal information, remote information access, multimedia services, etc. (Buchinger et al., 2011). Furthermore, currently, the importance of mobile devices is beyond the personal use, because they are starting to be extensively used in fields like education, science and research, entertainment, commerce, industry, among others (Qiao et al., 2008).

The increasing demand of mobile devices has promoted development of more sophisticated devices, with larger capabilities, which have broaden the scope and possibilities. In despite of the latter technological advances in such devices, they still have several external limitations such as communication networks, geographic access and internal limitations such as memory, CPU power and display size (Agu et al., 2005). All those limitations make difficult to load, decode and visualize multimedia content such as large images and videos. In some fields, like microscopy or cartography, it is necessary to work with high definition images, characterized by its large size, high resolution and quality. These images are represented by a very large volume of data, so, for visualizing them on mobile devices it is necessary to give them a special treatment to allow the users to properly access and interact.

To ease large image streaming and visualization, different transmission and coding algorithms have been developed, which facilitate storage, transmission and display of graphic content. In particular, the JPEG standard has been used for a long for coding images; however, current necessities require using more flexible and efficient standards (Sarraf and Wakim, 2007). In 2000, the Joint Photographic Expert Group (JPEG) published the JPEG2000 (J2K) standard, which, based on the concept of regions of interest and scalability, represents advances for the image compression technology, for which the image coding system has been optimized not only for being efficient, but also

*Corresponding author

Corredor G., Martínez D., Romero E. and Iregui M..
Architectural Model for Visualization of High Definition Images on Mobile Devices.
DOI: 10.5220/0004052601630170
In *Proceedings of the International Conference on Signal Processing and Multimedia Applications and Wireless Information Networks and Systems* (SIGMAP-2012), pages 163-170
ISBN: 978-989-8565-25-9

for being scalable and interoperable in network and mobile environments (Sarraf and Wakim, 2007; Rosenbaum and H. Schumann, 2006). Levels of detail, regions of interest and progressive transmission are popular concepts for handling graphical data in resource-limited environments (Rauschenbach and Schumann, 1999). Thererfore, the J2K standard is presented as a suitable tool to address a proper streaming and visualization on mobile devices.

Several works have explored the possibilities to access to J2K content by optimizing the interactive navigation, by proposing distribution protocols, management of client/server platforms, cache and prefetching strategies, among others (Taubman and Rosenbaum, 2003; Descampe, et al., 2007); (Iregui et al., 2007); (Deshpande and Zeng, 2001); (Iregui et al., 2002); (Iregui et al., 2002); (Meessen et al., 2003); (Moshfeghi and Ta, 2004). However these works do not consider the problem of minimal available capabilities of mobile devices.

In the mobile field, some works have focused on areas such as reliability and error resilience over noisy channels (Ho and Kahn, 1997) and content delivery security (Díaz et al., 2006). Liu et al (2003) presented a model for browsing of images on small displays, however they work with images of range of 1600x1200 pixels; this model is not applicable to very large images, as satellite imaging, which may have sizes of more than 15000x15000 pixels. Rosenbaum and Schumann (2006) proposed a model for viewer guidance for mobile devices by exploiting the J2K features; their solution requires to filling the omitted code-stream positions with predefined data to keep it compliant and can be decoded, however, this solution is not applicable to very large images because if the area to be processed and decoded is very large, the required processing time and memory may exceed the capacities of the device. Otherwise, Google Earth is a well-known application for accessing to large satellite images that runs on desktop computers and mobile devices. It uses a very large set of hierarchy prerendered tiles of different spatial resolutions, then, each time a user increases the level detail of the image, a new set of images is loaded (MicroImages, Inc., 2010). In this way, the application requires a complex organization and hierarchy of files and directories and a big amount of space to store the whole image. Furthermore, this application only works with the provided images and does not allow using different ones.

The present investigation introduces anarchitectural model to allow optimal interaction with very large images of general purpose from mobile devices. The proposed model facilitates a mobile client to browse very large remote images by displaying regions in a flexible and scalable way, offering a nearly seamless navigation, adapted to the restricted capabilities of the devices and the channel bandwidth. This represents an advantage over available applications that do not allow accessing and navigation in very large images or regions in full quality and resolution. Moreover, the proposed architecture is modular and decoder independent, permitting thereby easy adaptation to new models and specialized applications.

This paper is organized as follows. In section 2, a brief overview of JPEG2000 standard is presented. Section 3 introduces the proposed model for streaming and visualization of high definition images on mobile devices. In Section 4, an implementation of the proposed model is presented. In section 5, experimental results are reported, providing evidence of the performance and efficacy of the model. Finally, section 6 presents brief conclusions.

2 JPEG2000 OVERVIEW

JPEG2000 is an image compression standard designed by the Joint Photographic Expert Group, based on the Discrete Wavelet Transform and the EBCOT encoder (ISO/IEC 15444-1, 2000). This standard provides several advantages such as improved compression efficiency, lossy and lossless compression, multiple resolution representation, random code-stream access and processing and quality refinement (Rabbani and Rajan, 2002).

A single J2K data stream typically contains numerous embedded subsets, which may be extracted to recover a portion of the original image at any of a large number of different spatial resolutions, image quality layers, or in selected spatial regions. A J2K image is split into rectangular regions that are encoded independently, called tiles, but also defines collections of spatially adjacent code-blocks, known as precincts. Each precinct is represented as a collection of packets, with one packet for each quality layer, resolution level and component. These embedded compressed data subsets allow a low quality or low resolution image, or one whose details cover only a small spatial region, to be incrementally improved by adding the missing elements from the compressed data stream (Taubman, 2002).

3 SYSTEM ARCHITECTURE

The proposed application aims to be as simple as possible in order to reach acceptable response times, provided the highly limited memory, processing power and low bandwidth of mobile devices, among other critical constraints. The main idea is to allow the user to navigate in a seamless way by requesting partial content of an image, according to each region of interest, instead of processing the whole image; in such scenario, only data contributing to the requested representation are transmitted and decoded.

Through the exploitation of the granularity and scalability of J2K data streams, the most appropriate subsets are delivered from a server to an interactive client, incrementally improving the image quality and/or resolution in a consistent manner with the client's interests at any given time (Taubman, 2002). As mentioned in the previous section, J2K codestream is conformed by packets, one for each quality layer, resolution level, component and position. In this way, only those packets belonging to a region at a specific resolution level and quality percent of the image could be retrieved. Because tiles compress less efficiently, introduce unpleasant boundary artifacts at low bit-rates and provide poor resolution scaling properties (Taubman, 2002) and because of packets flexibility, in this work, packets are adopted as the fundamental unit of data exchange between server and client.

For this purpose, the server, which stores the set of still encoded images, must be capable of extracting packets from images according to a given request. To enable random access to specific portions of the codestream in a fast way an index file for each image has been off-line generated based on the JPIP specification (ISO/IEC 15444-9, 2000), facilitating a minimal memory use and reduced disk access. The index is a simple text file containing the information organized into two boxes: image information box and packet information box. The former contains information such as width and height, progression order id, number of components, number of quality layers, number of decomposition levels, etc. The latter contains the index, the features (i.e. tile, layer, resolution, component and precinct), and the starting and ending positions of each image packet. The server is also responsible for generating a sequence of J2K packets and delivering it to the client, which is located on a mobile device.

The client has a Graphic User Interface (GUI) that allows the user to facilitate the comprehension of the content through an appropriate representation and means for interaction, by specifying which regions of a particular image must be displayed with a certain degree of detail. The GUI enables to set up the Window of Interest (WoI) coordinates, the resolution level and the quality percent. For this purpose, a method that calculates the necessary packets to meet the user requirement was designed. Since most of the data may be reused and a non-redundant data transmission is desired, a cache management is performed by storing the received packets; in this way, only missing data are requested to the server.

Although the J2K compression algorithm and its packet structure provide excellent tools on which to base an interactive image application, the data syntax described in the standard excludes the interactive construction of a valid data stream from arbitrarily ordered packets (Taubman, 2002). As not every available image data might be needed, it is necessary to do some modifications to data to keep compliance; previous works have addressed this issue by filling the omitted code-stream positions with predefined data (Rosenbaum and Schumann, 2006). However, in this work it is proposed a different strategy: based on the information of the requested region, the image main header is modified as to include specifically the packets required to reconstruct the queried ROI. In other words, the image size, quality layers and resolution levels are set to the ROI. In this way, the decoding process is faster and efficient because the transcoding process requires only little computing power and memory, and because the decoder do not have to process trash or empty data. This strategy allows using the decoder with no change, to decompress a set of packets; so that any J2K decoder can be used.

3.1 Server

The server stores the J2K images and their respective index files and thumbnails. It sends packets and information of the images to the client through a TCP/IP channel.

- User interface (UI): It is an interface through which the system's administrator can manage the server. It naturally allows the addition of new images to make them available to the client.
- J2K Encoder: It is responsible for encoding the images in the J2K standard. It uses a predefined set of parameters to perform encoding. Once the image is encoded, it is stored as a file at a specific location on the server.
- Index generator: It generates an index file for each codified image. The index file is used to get

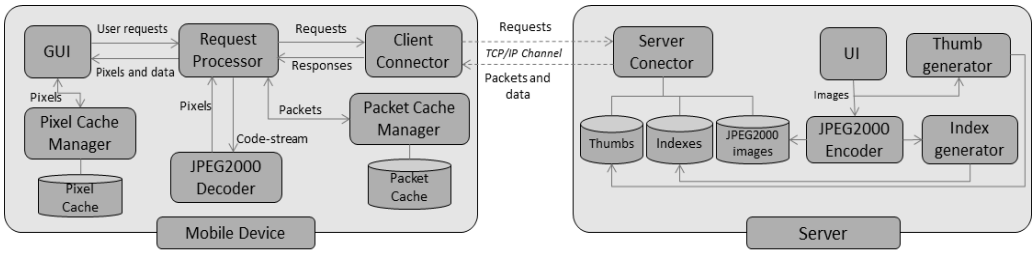

Figure 1: Top-level runtime view of the architecture.

image information and to ease random access to specific portions of the codestream.

- Thumb generator: It generates a thumbnail file for each image. The thumbnails are used to rapidly allow the client to access and view the available images.
- Server connector: It is responsible for communication with the client. It receives and processes the client requests and sends packets and image data.

3.2 Client

The client is located in a mobile device that allows the user to interact through a GUI by requesting and viewing regions of interest at certain level of resolution and quality.

- Client Connector: It is responsible for communication with the server. It sends request messages to the server and sends the received response data to the RP.
- Request Processor (RP): The RP is an interface between the GUI and the Client Connector. It is responsible for processing the requests from the GUI. It sends the corresponding request to the Client Connector, sends and retrieves packets from the Packet Cache Manager, generates a compliant J2K codestream to send it to the J2K decoder and sends the generated pixels to the GUI.
- Packet Cache manager: It stores all the received J2K packets and recover them when are requested by the RP. The cache size is parameterizable and may be set depending of the capacity of the device. The packet cache manager currently uses a Least Recently Used policy to discard packets and free memory when it is almost full.
- J2K Decoder: The J2K decoder is responsible for decoding the requested image region. It receives as input a transcoded compliant bitstream and returns the set of pixels corresponding to the image representation.
- Pixel Cache manager: It stores the decoded pixels. It currently uses a policy to discard packets based on the distance; it stores the adjacent areas of a user request and deletes them when the user is centred in a non-adjacent (distant) region when memory is almost full. The size of the pixel cache is also parameterizable.
- Graphic User Interface (GUI): It allows the user to specify which regions of an image should be transmitted, at which degree of detail.

4 SYSTEM FOR STREAMING AND VISUALIZATION OF IMAGES

A simple prototype was developed in order to validate the proposed architecture. Hereafter, there is a brief description of the features of the developed application. Since the architecture consists of a client and a server, we developed a stand-alone application for each node.

4.1 Server Implementation

As mentioned in the previous section, the server stores the J2K images and their respective index files and thumbnails, and allows communication with the client. The server was developed in the Java platform, Standard Edition, version 1.6.

Uncompressed images and coding parameters are sent to the J2K encoder, which generates a coded image file; encoding was performed with the Jasper J2K implementation (Adams and Kossentini, 2007). The start of packet (SOP) marker was used to facilitate the index file generation. The index generator receives as input an encoded image; it reads the markers of the image and extracts the necessary information to generate the index file. Finally, the Server Connector is provided with a socket that receives and processes the client requests and sends the corresponding responses.

4.2 Client Implementation

The client was developed in the Android platform, version 2.2. The client runs a Graphic User Interface, a Request Processor, a J2K decoder, a Pixel Cache Manager, a Packet Cache Manager and a Client Connector. The communication with the server is done through a socket that the Client Connector manages.

The GUI allows the client to connect to the server by typing the corresponding IP address. Once the connection is established, it is requested the list of available images with their respective thumbnail, which are shown to the user through the GUI. When the user selects one image, it is requested the load of that one and the server sends its information (size, number of resolutions, quality layers, etc.) and its main header. Next, the user selects a region of interest by setting the coordinates, a level of resolution and quality percent. If the requested region is not in the pixel cache, the request is sent to the RP, which calculates the necessary packets to meet that requirement and looks at which packets are stored in packet cache memory, in this way, it only requests to the server the missing data. Once the client has the necessary packets, either from the server or from the cache, they are transcoded into a compliant codestream. As mentioned in the previous section, to let the packets to be decoded, it is necessary to modify the image main header. For the decoding process it was used the Jasper Software, so it was developed an interface function that allows to send to Jasper the codestream and to receive the corresponding pixels. Finally, the pixels are sent to the graphic interface to be presented to the user, that pixels are stored on the pixel cache.

5 EXPERIMENTAL RESULTS

Two nodes were used to perform the experiments: A server and a mobile device. The former was a desktop computer with operating system Windows 7, 4 GB RAM memory and 2.71 GHz dual-core processor. The latter was the Amazon Kindle Fire, a tablet with operating system Android 2.3, 1024×600 display size, 512 MB RAM memory and 1 GHz dual core processor. The mobile device was connected to a Wi-Fi network with a bandwidth of 25 Kbps.

The experiments were performed with an uncompressed satellite image of 786433 KB and resolution 16384x16384 pixels. The image was J2K compressed by using the following coding parameters. LRCP progression order, 3 components, 10 quality layers, 4 resolution levels (5 resolutions), lossless codification, precinct size of 64x64 for each resolution and codeblock size of 64x64. The final size of that image was 125544 KB, reaching a compression rate of 84.04%. The generated index for this image has a size of 99629 KB and the reading and parsing process is about 7 seconds. Because the index is stored in the server side, its format was not optimized for storage size.

5.1 Comparing JPEG and JPEG2000

In the first experiment, the performance of streaming and visualization by using JPEG and JPEG2000 standards was compared. As the JPEG standard does not provide multiresolution representation, a JPEG pyramid was constructed, i.e., the original image was coded with 5 different image sizes, each corresponding to a different resolution level. The total size of the 5 JPEG images was 170529 KB, reaching a compression of 78.32%.

Transmission and processing time were compared for a set of user requests. In this case, the user starts at a certain region of interest and zoom in over it, from the first resolution to the fourth, at full quality. Results are shown in Table 1.

Results show that the number of transmitted bytes for a JPEG image is greater than the JPEG2000 image; due the JPEG standard does not allow a straight access to regions of the image, it is necessary to request and to decode the whole image in order to present it or a region to the user.

This represents an issue when the transmitted image is too large, because it is required a long time for downloading it and because the device may not have enough memory to store and process it. Furthermore, processing time for JPEG is less than for JPEG2000 for small size images; however, for large images in the JPEG format, due the limitation in memory and processing power, the application generated an error and could not decode and show the image representation.

5.2 Simulating a User Browsing Protocol

For this experiment, memory usage and transmission and computing times were measured for a set of user requests by simulating a browsing protocol for which a user scrolls over adjacent and overlapping regions. The user starts viewing an image in the lowest resolution level and 50% of quality. Next, the user finds an interest area, zooms in until the fourth resolution level. Finally, as the user stops at this

Table 1: Bytes and decoding and computing times for a satellite image by using the JPEG and the JPEG2000 standards. N/D (No data) means that the request was not processed because of memory overflow.

Resolution Level	JPEG			JPEG2000		
	Transm. bytes	Transm. time (ms)	Computing time (ms)	Transm. bytes	Transm. time (ms)	Computing time (ms)
1	576288	710,4±134,42	147,1±36,82	283472	2102,6±622,36	1660±158,73
2	2334063	2332±353,16	469,7±11,21	210359	1466,8±141,65	1602±15,82
3	9310361	8355,3±1582,5	N/D	216217	1808,5±764,87	1747,3±285,53
4	35151251	N/D	N/D	199148	1446,1±21,87	1582,7±15,83

area, a quality refinement is evaluated by requesting the remaining quality layers, until the maximum quality level. The browsing protocol is shown in figure 2.

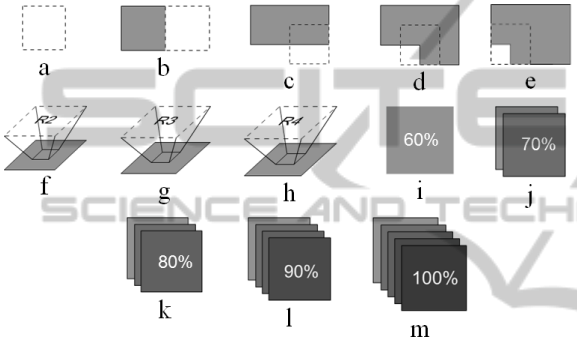

Figure 2: Representation of the requested regions. Dashed areas correspond to regions to be requested, while shaded áreas correspond to previously requested areas, stored in cache. Figure (a) shows the first request in which there are not data stored in cache. Figure (b) shows the request of an adjacent region to the first one. Figures (c) to (e) shows requests of areas that overlap previously requested areas. Figures (f) to (h) shows requests of areas simulating a zoom in, since the second resolution level to the fourth one. Figures (i) to (m) shows requests for a quality refinement.

While the original uncompressed image is 768 MB in size, at the end of the browsing protocol, only 952 KB of compressed data are transmitted. Transmission of the packets took only 4s and the image processing took only 14s for the whole navigation.

5.2.1 Measurement of Transmission and Computing times

In this experiment, the transmission and computing times were measured for the browsing protocol. Figure 3 presents the results of this experiment and includes the accumulated time of transmission and computing.

Requests (a) to (e) represent the scrolling over adjacent and overlapping areas. In this case, it was used a packet cache policy because some regions intersect with previously requested regions. Once the graphic area representation is obtained, it is added to the screen to compose the whole region. Response times of the first request are relatively high because there are not data in cache. As the area of the second region does not overlap with the first, every packet has to be transmitted and decoded. For next requests, transmission and decoding times are lower because of the intersected regions.

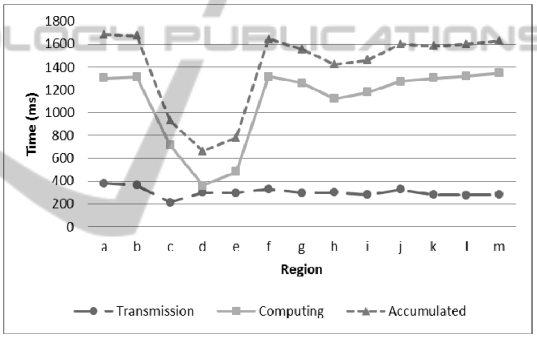

Figure 3: Variation of transmission and decoding times for a browsing protocol. Requests (c) to (e) represents regions that overlap with previously requested data, so, less amount of data must be transmitted and processed. Requests (f) to (h) represents a zoom in over a region. Requests (i) to (m) represents a quality refinement process.

Requests (f) to (h) represent a zoom in operation over a specific area. The requested regions correspond to areas of 512x512 pixels. Results show that the transmission time slightly increases because the use of packet cache. Regarding the decoding process, given that the requested area is lower than the device's display size, it is responding in proper times. Other experiments have shown that requesting areas larger than the display may have repercussions on the system performance. It is highly recommended that requested areas do not double the screen size.

Request (i) to (m) stand for a quality refinement operation over a specific area. Here, the user was starting at a specific region with 50% of quality, and

progressively the system requests and decodes the layers corresponding to higher quality percentages. On the one hand, this process takes advantage of the cache policy by using the previously requested packets so that the transmission times are reduced. On the other hand, decoding time increases for a quality layer with respect to the previous. This happens when decoding a new quality because it requires using the whole data of the previous layer and there is not a direct method to perform cache management for quality layers. Results show that response times increases when the quality percentage increases, however, the system offers the user flexibility to select the desired maximum quality for each request, according to his/her necessities; in this way, better response times can be obtained.

5.2.2 Memory Usage Evolution

In this experiment was measured the memory usage of the application for the browsing protocol previously depicted. The memory was measured each second during the execution of the browsing session, for a total of 20 seconds. Results are shown in figure 4.

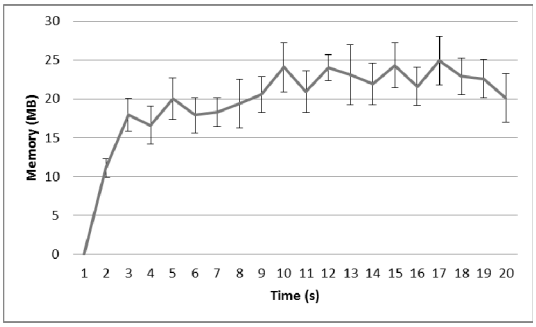

Figure 4: Evolution of memory usage of the application for a browsing protocol.

Results show that the application memory usage keeps between about 15 and 25 MB, which represents only 4.88% of the total memory of the device. That means, the application is using only a small part of the device's memory, so, the memory usage is efficient and it is applicable to most of the current devices.

General results show that the introduced architecture not only allows visualizing regions of very large images but also presents a good performance, both for transmission and decoding processes. The procedure to determine every packet contributing to a desired ROI requires low computing power and decoding time is reduced because the area of the requested image is not larger than the display size. It was defined a maximum response time to visualize a requested area of 5 seconds and the results shown the user do not have to wait more than 2 seconds. Additionally, a software-based decoder was used, however, if a hardware-based one is used, response times may improve in a meaningful way. The visualization is adaptable to user necessities; because it can display the best quality and the best resolution level adapted to the possibilities of the device (memory, computing power and screen size). Also, it was demonstrated that use of J2K is most appropriated for large images on mobile devices than JPEG. The use of J2K has important benefits; the standard inherently provides a high compression performance, but also a structure of the encoded data which allows accessing the stream only in those packets contributing to the required ROI and thereby much bandwidth can be saved because it is not necessary to transmit and process the whole image. In the server side, the use of J2K standard gives an improved compression efficiency, which allows storing several large images and their respective indexes without requiring a lot of disk space.

6 CONCLUSIONS

In this paper, an architecture model for streaming and visualization of high definition images, characterized by its high resolution, quality and size, on mobile devices was presented. The proposed model allows a user to request regions of the image at certain level of resolution and quality. The proposed solution takes advantage of the J2K granularity to allow accessing to high definition images to make an optimal content delivery to mobile devices with different capabilities and bandwidth. The client only decodes the required data to generate the representation of the image region, so the memory and processing usage is minimized and the system presents proper response times. Due to the modular design and the loosely coupled decoder, it is easy to adapt the architecture to new models and to develop specialized applications. Future work includes index generation optimization and implementing cache strategies based on user navigation.

REFERENCES

ISO/IEC 15444-1, 2000. *Information technology - JPEG2000 image coding system – Part 1: Core*

coding system.

ISO/IEC 15444-9, 2004. *Information technology - JPEG2000 image coding system – Part 9: Interactivity tools, APIs and protocols*

Rabbani and Rajan, 2002. An overview of the JPEG2000 still image compression standard. *Signal Processing: Image communication*, 17:3–48.

Taubman and Rosenbaum, 2003. Rate distortion optimized interactive browsing of JPEG2000 images. In *IEEE International Conference on Image Processing (ICIP2003)*, volume 3, pages 765–768.

Descampe, et al., 2007. Pre-fetching and caching strategies for remote and interactive browsing of JPEG2000 images. *IEEE Trans. on Image Processing*, 16(5):1339-1354.

Iregui, Gómez and Romero, 2007. Strategies for efficient virtual microscopy in pathological samples using JPEG2000, *Micron* Vol 38 pag 700-713.

Liu et al, 2003. Automatic browsing of large pictures on mobile devices. *Proceedings of the 11th ACM international conference on Multimedia*.

Qiao, Feng and Zhou, 2008. Information presentation on mobile devices: Techniques and practices. *Lecture Notes in Computer Science*, Volume 4976/2008, 395-406.

Burigat, Chittaro and Gabrielli, 2008. Navigation techniques for small-screen devices: An evaluation on maps and web pages. *International Journal of Human-Computer Studies*, Vol. 66, No. 2, pp. 78-97.

Agu et al., 2005. A Middleware architecture for mobile 3D graphics. *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops*.

Buchinger et al, 2011. A survey on user studies and technical aspects of mobile multimedia applications, *Entertainm. Comput.*, doi:10.1016/j.entcom. 2011.02.001

Rauschenbach and Schumann, 1999. Demand-driven image transmission with levels of detail and regions of interest. *Computers & Graphics* 23 857-866.

Sarraf and Wakim, 2007. Improving JPEG2000 images delivery over GPRS mobile networks. *Proceedings of the 6th WSEAS Int. Conf. on Electronics*, Hardware, Wireless and Optical Communications, Corfu Island, Greece.

Rosenbaum and Schumann, 2006. JPEG2000 based viewer guidance for mobile image browsing. *12th International Multi-Media Modelling Conference Proceedings.*

Taubman, 2002. Remote browsing of JPEG2000 images. *Proceedings IEEE ICIP*.

Deshpande and Zeng, 2001. Scalable streaming of JPEG2000 images using Hypertext Transfer Protocol. *Proceedings ACM Multimedia*.

Iregui, Chevalier and Macq, 2002. Optimal caching mechanisms for JPEG 2000 communications. *EUSIPCO 2002 - European Signal Processing Conference*, Toulouse, France, Sept. 2002, Proceedings Vol. 3, pp. 201-204

Iregui et al., 2002. Flexible access to JPEG2000 codestreams. *23rd Symposium on Information Theory in the Benelux*, May 29-31, 2002, Louvain-la-Neuve, Belgium

Meessen et al, 2003. Layered architecture for navigation in JPEG2000 Mega-Images. *WIAMIS 2003 - 4th European Workshop in Image Analysis for Multimedia Interactive Services*, April 9-11, London, UK, Proc., pp. 92-95

Moshfeghi and Ta, 2004. Eficient image browsing with jpeg2000 internet protocol. In *SPIE: Medical Imaging 2004: PACS and Imaging Informatics*.

Díaz et al., 2006. JPEG2000 images protection sent to mobile devices. *Proceedings of the I International Conference on Ubiquitous Computing: Applications*, Technology and Social Issues. Spain.

Ho and Kahn, 1997. Image transmission over noisy channels using multicarrier modulation. *Signal Processing: Image Communication*, Volume 9, Number 2, January 1997 , pp. 159-169

Hasan and Kim, 2009. An automatic image browsing technique for small display users. *11th International Conference on Advanced Communication Technology*.

Adams and Kossentini, 2007. *Jasper: A software-based jpeg-2000 codec implementation*. Available at <http://www.ece.uvic.ca/~frodo/jasper/>

MicroImages, Inc., 2010. *Google Earth Structure*. Available at <http://www.microimages.com/documentation/TechGuides/76googleEarthStruc.pdf>