# JPEG2000 and Dissemination of Cultural Heritage Over the Internet

Eugenia A. Politou, George P. Pavlidis, *Student Member, IEEE*, and Christodoulos Chamzas, *Senior Member, IEEE*

***Abstract*—By applying the latest technologies in image compression for managing the storage of massive image data within cultural heritage databases and by exploiting the universality of the Internet we are now able not only to effectively digitize, record and preserve, but also to promote the dissemination of cultural heritage. In this work we present an application of the latest image compression standard JPEG2000 in managing and browsing image databases, focusing on the image transmission aspect rather than database management and indexing. We combine the technologies of JPEG2000 image compression with client-server socket connections and client browser plug-in, as to provide with an all-in-one package for remote browsing of JPEG2000 compressed image databases, suitable for the effective dissemination of cultural heritage.**

***Index Terms*—Client–server, cultural image databases, image compression, JPEG2000, plug-ins, progressive coding, progressive transmission, sockets.**

## I. Introduction

PRESERVATION of cultural heritage has already become one of the main concerns of all developed nations. The belief that "the humanity must have strong memories of its past in order to make successful steps into the future" is strong not only to people from the cultural community but to scientists too. The involvement of the latter in the preservation and dissemination of cultural heritage consists mainly in the design and implementation of pioneering, manageable and functional Internet databases, filled with prototype cultural material of all media types, known as cultural heritage databases.

The content of a cultural heritage database consists virtually of all digital media types: text, image, sound, video, etc. Still, the most important and widespread of all media is the image. The image provides users with the necessary information concerning the shape, pattern, color and other details of objects, properties that can not be fully described or easily perceived by using text. At the same time, image acquisition or digitization has become a very simple task.

One common problem in cultural heritage databases arises out of the difficulty in managing and publishing large image information. In our Institute, a lot of work has been done on this subject during the development of such databases. One of the most important databases we have coped with was the "Ark of Refugee Heirloom" [1], a cultural heritage database that records the cultural identity of a particular part of the Hellenic population.

The objective of the work presented in this paper is to propose a new way of accessing the image data of cultural heritage databases. The idea behind our implementation is based upon the progressive transmission of images through the Internet. Our goal was to achieve a more efficient way of navigation. There are other works on this subject, but they are either not based on browsing through a common interface such as a widespread browser [2], or not based on common protocols such as HyperText Transfer Protocol (HTTP) [3]. Particularly, in [3], the proposed system consists mainly of an application specifically developed for browsing JPEG2000 images over the Internet using JPIK (JPeg2000 Interactive with Kakadu) protocol [24] rather than the World Wide Web standard HTTP. On the other hand, in [2], the system is based on the implementation of a helper application, which is a third-party piece of software. The helper application has to be distributed and is executed on the client side whenever a request from a browser activates it. In this case, the actual image data are displayed in a separate window and they cannot be manipulated from within the browser. To overcome these disadvantages, we decided to develop a browser plug-in, which is an approach frequently used for displaying content within today's internet browsers. Plug-ins are executed within the browser and are more naturally integrated into a web application. At the time this work was completed we were not aware of any other similar implementations, which is why we could not provide any comparisons on the performance aspects.

One of the major technical problems that the Internet experiences today is its data transmission rate capability. The prohibitive transmission rates for large images combined with the need for enormous storage spaces lead to the development of new coding schemes to compress image information. An example of such a compression scheme with wide acceptance on the Internet is JPEG [4]. As shown in Fig. 1, by using JPEG (compressor "cjpeg" [5], compression quality set to 15) on an image $10 \times 15$ cm at 600 dpi, we reduce the transmission time through the modem line from 1 h to about 30 s, while preserving a moderate image quality (about 32 dB PSNR).

In Dec. 2000 a new image compression method was standardized as a result of about 4 years combined research of major industry and academia: JPEG2000 [6]. It comes to fulfill tomorrow's requirements for storage, data manipulation and handling, rights management, progressive and error resilient transmission with support for region of interest coding and more, while providing with better compression and image quality. JPEG2000 employs wavelet transform to achieve energy compaction into a small number of transform coefficients

Manuscript received February 15, 2003; revised October 7, 2003. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Ioannis Pitas.

The authors are with the Cultural & Educational Technology Institute, Xanthi 67100, Greece (e-mail: politou@ceti.gr; gpavlid@ceti.gr; chamzas@ceti.gr).

Digital Object Identifier 10.1109/TIP.2003.821348


Fig. 1. JPEG compression example, (a) original image, 24 bpp, transmission time about 1 h and (b) compressed at 0.19 bpp, quality 32 dB PSNR and transmission time about 30 s.

and a highly sophisticated embedded coding, combined with a binary arithmetic encoder and rate control mechanisms, packing data into a meaningful, flexible and easily parseable codestream. This new standard shows great potential in aiding the management and transmission of images over the Internet and thus is one of the most useful tools to adapt to a cultural heritage database.

After extensive testing of various image compression technologies and studying of the way people use and search into multimedia databases over the Internet, we ended up with the following 4-step user interaction scenario:

Step 1) The user gets a thumbnail view (preview version) of actual images kept in the database.

Step 2) The user chooses one of the images for further inspection and a medium view (display version) is shown.

Step 3) The user gets a full-sized image with reduced quality.

Step 4) The user gets the full-sized full-quality original image, as kept in the database and is suitable for printing.

The transmission of data is done in a differential way, so that the user accepts only the information needed to move on to a next step in the scenario. The interaction between the user and the database server is controlled by a socket connection, while JPEG2000 decoding and display is accomplished by a client browser plug-in.

A key point of this scenario, based on the embedded and progressive nature of JPEG2000, is that each time a user requests data from the server in order to increase image resolution, the corresponding information is extracted from one and only one image file in the database. This means that the original image codestream has to be structured in such a way as to include many different resolutions and quality levels of itself. At each request for one of these resolutions or quality levels, the server is responsible for extracting the appropriate data from the original file and for transmitting it to the client where the final merging with the data already existing from previous requests would be accomplished.

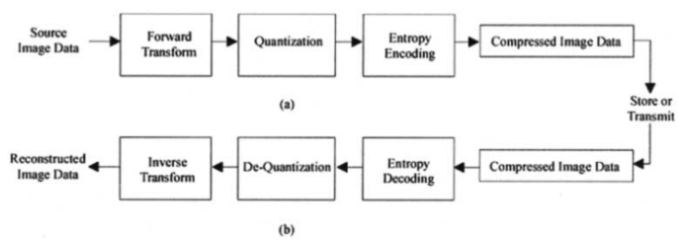


Fig. 2. Basic structure of the JPEG2000 coding scheme.

## A. JPEG2000 Algorithm

The key part for achieving progressive image transmission in our work is the use of JPEG2000, which provides with a wide set of tools for image compression. The main features of JPEG2000, in which our implementation is based upon, is progressiveness and embedded bitstream syntax, which implies the ability to extract from one compressed file many different resolutions or quality levels of an image, depending on the user's request. The algorithm structure is extremely complicated, but has already been presented and analyzed in many other works [6]–[18]. In Fig. 2, we show the basic coding steps in JPEG2000. The structure is similar to any transform coding scheme: a transform coding step, followed by quantization and entropy coding (Fig. 2(a) corresponds to the encoder, while Fig. 2(b) corresponds to the decoder). The basic building blocks of the encoder are shown in Fig. 3.

## B. Syntax of JPEG2000 Codestream

The codestream syntax specifies such fundamental quantities as image size, tile size, number of components, and their associated sub-sampling factors. It also specifies all parameters related to quantization and coding such as step sizes, precinct sizes, as well as the transform kernel employed. All features present in the codestream are signaled via the syntax, including the number of quality layers, the number of resolutions, progression orders, region of interest information, and whether or not error resilient encoding has been performed. Most coding parameters can be chosen on a tile-by-tile basis [6], [7].

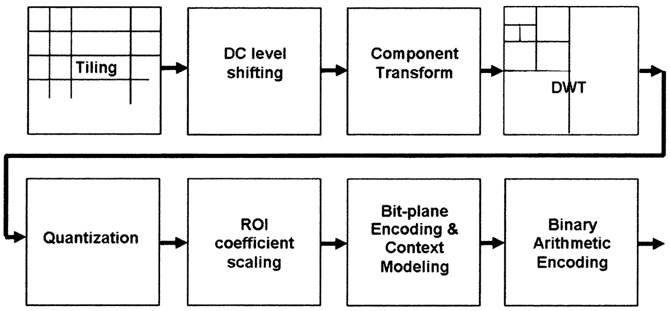


Fig. 3. Basic building blocks of the JPEG2000 encoder.

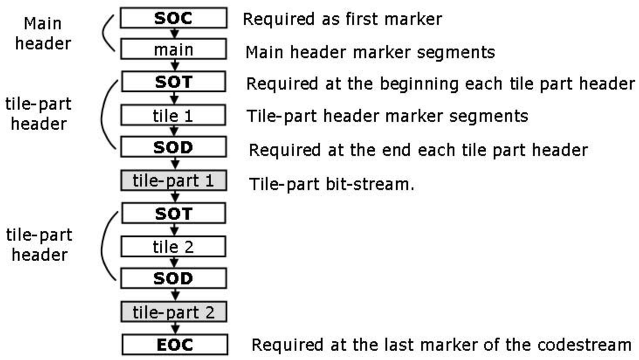


Fig. 4. Basic structure of the JPEG2000 codestream.

In the simplest case, a JPEG2000 codestream is structured as a main header followed by a sequence of tile-streams (Fig. 4). The extension of such a JPEG2000 file, as proposed by the JPEG2000 standard [6], is noted as "jpc" (JPEG Codestream).

The main header contains global information necessary for the decompression of the entire codestream. Each tile-stream consists of a tile header followed by the compressed pack-stream data for a single tile. Each tile header contains the information necessary for decompressing the pack-stream of its associated tile. Finally, the pack-stream of a tile consists of a sequence of packets [6], [7].

### *C. Progression in JPEG2000*

Progression enables increasing the quality, resolution, spatial extent and color components, as more bytes are decoded sequentially from the beginning of a compressed codestream. The type of progression in a JPEG2000 codestream is governed by the order in which packets appear within tile-streams. As such, progression can be defined independently on tile-by-tile basis. Actually, tile-streams can be broken at any packet boundary to form multiple tile-parts. Each tile-part has its own header and the tile-parts from different tiles can be interleaved within the codestream [6], [7]. There are five different progression orders supported in JPEG2000. In Fig. 5, represented graphically, are two of the progression orders with high importance for Internet image database applications: by quality and by resolution.

### *D. Client-Server Technology*

Client-server technology is based on creating sockets to achieve the communication between two or more remote computers. A socket works much like a telephone. It's the endpoint of a two-way communication channel. By connecting two sockets we can pass data between processes, even processes running on different computers, just as we can talk over the telephone.


Fig. 5. Quality (a) and resolution (b) progression in JPEG2000.

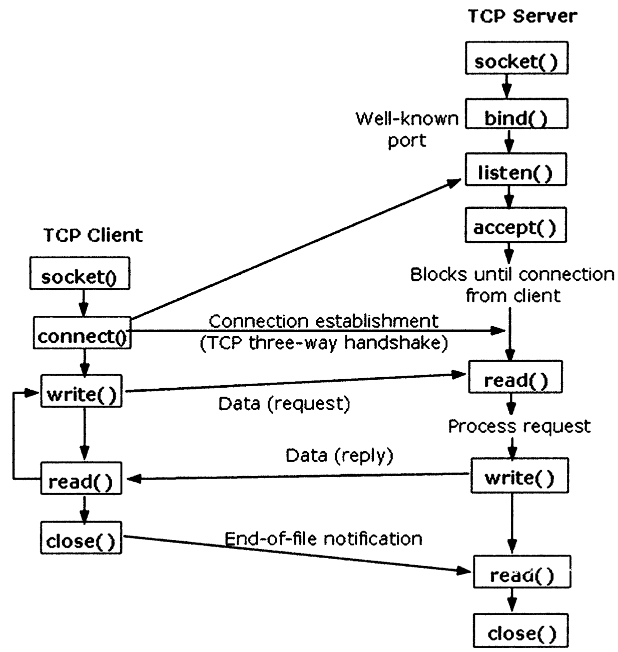


Fig. 6. Connection-oriented client-server model.

Programs written to use the TCP protocol are developed using the client-server model. When two programs use TCP to exchange data, one of the programs must assume the role of the client and the other the role of the server. The client application initiates what is called an *active open*. It creates a socket and actively attempts to connect to a server. On the other hand, the server application creates a socket and listens for incoming connections from clients, performing what is called a *passive open* (Fig. 6). When the client initiates a connection, the server is notified that some process is attempting to connect with it. By accepting the connection, the server completes what is called a virtual circuit, a logical communication pathway between the two programs. It is important to note that the act of accepting a connection creates a new socket; the original socket remains unchanged so that it can continue being used to listen for additional connections. When the server no longer wishes to listen for connections, it closes the original passive socket [19], [20].

### *E. Plug-Ins*

A plug-in is a separate code module that behaves as if it were a part of a greater integrated application. Specifically, an Internet browser plug-in is a code library that runs only from within the browser. The way the browser interacts with the plug-in is described by a programming tool set, called plug-in Application Programming Interface (API) [21].

With the Plug-in API we can create dynamic loading plug-ins able to

- register one or more MIME types;
- draw into a part of a browser window;
- receive and handle keyboard and mouse events;
- obtain data from the network using URLs;
- post data to URLs;
- add hyperlinks or hotspots to link to new URLs;
- draw into sections on an HTML page.

When the user opens an HTML page that contains embedded data of a media type that invokes a plug-in, the browser reacts with the following sequence of actions:

1) searches for a plug-in with a matching MIME type;
2) loads the plug-in dynamically into memory;
3) initializes the plug-in;
4) creates a new instance of the plug-in;
5) removes a plug-in instance when it is no longer needed;
6) unloads the plug-in from memory when the last instance of a plug-in is removed.

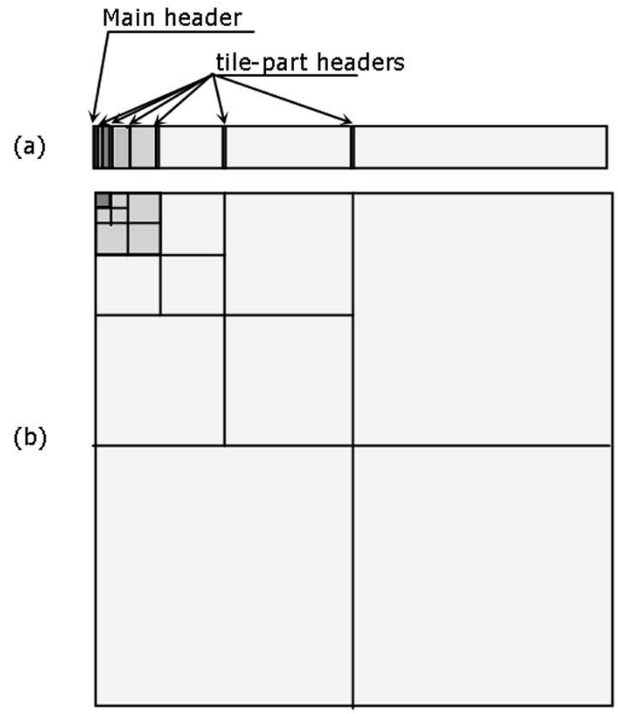


Fig. 7. Illustration of codestream creation by coding an image with six resolution levels. (a) Codestream segmentation in tile-parts and (b) 5-stage wavelet transform for six resolution levels image encoding.

## II. Implementation

For the purposes of our application all of the aforementioned technologies were combined in order to achieve a progressive and dynamic way of accessing the images in a database. Our implementation includes a server and a client application. It must be noted here that this implementation focuses mainly on efficient image transmission rather than on context-based multimedia information data-mining.

The server is, actually, a service that runs on a web server and has access to the original images, which are already losslessly compressed by JPEG2000. These images had to be properly encoded so that their progressive transmission is possible and according to a specific user interaction scenario. To this end, an independent external application for image preparation had also been developed.

The client, on the other hand, is a browser plug-in, installed on the user's computer in order to provide with the ability to process user requests, decode and display requested images.

Next, we will discuss the database image preparation along with server and client in more detail.

### A. Preparation of Database Images

To be able to extract from a single image codestream multiple segments that correspond to user requests, a specific way of encoding should be followed. Codestream data must be arranged in resolution level blocks, so that each sequential resolution level added to the previous would produce a greater resolution and specifically of double width and height (Fig. 7), due to the nature of the wavelet transform in JPEG2000.

In this work, to fulfill the application requirements, specific resolutions of images have been chosen, guided by the need for better database navigation through the Internet and better network bandwidth exploitation. These resolutions are

- $64 \times 64$ pixels for thumbnail resolution;
- $256 \times 256$ pixels for medium resolution;
- maximum resolution corresponding to the full-sized original image.

Selection of these dimensions was based upon experimental optimum values from prior work on publishing image databases through the Internet [22], [23].

In order to have images encoded in a way that the first resolution level has maximum dimensions $64 \times 64$ pixels, we have to estimate the number of decomposition levels needed for each image in order to achieve the thumbnail resolution of $64 \times 64$ pixels (1), and to compress each image using lossless JPEG2000 coding

$$\mathrm{n_w} \leq \log_2 \frac{\mathrm{w}}{\mathrm{w'}}, \mathrm{n_h} \leq \log_2 \frac{\mathrm{h}}{\mathrm{h'}} \tag{1}$$

$$\mathrm{n} = \lceil \max\{\mathrm{n_w}, \mathrm{n_h}\} \rceil \tag{2}$$

where $w$ and $h$ are the width and height of the original image, $w'$ and $h'$ are the desired width and height, and $n_w$ and $n_h$ stand for how many times the image must be divided by 2 across the width and height resolutions respectively, to produce a desired resolution. In our case the values of $w'$ and $h'$ are preset to 64 pixels. Thus, the number of wavelet decomposition levels $n$ must be the immediate greater integer of the maximum value between $n_w$ and $n_h$ (2) (the ceiling operator $\lceil\ \rceil$ denotes the smallest integer greater than or equal to the value it operators on).

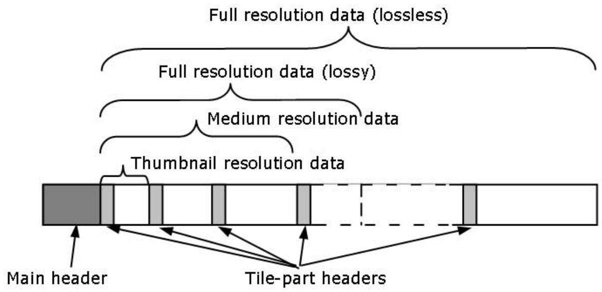


Fig. 8. Codestream structure of a database image.

This way images will be encoded with $n+1$ resolution levels (there will be $n+1$ SOT[1] markers in the codestream), ensuring that the image thumbnail will be of $64 \times 64$ pixels maximum resolution. A representation of the codestream structure of such an image is illustrated in Fig. 8.

The JPEG2000 encoding is done by the source code of the kakadu software [24], which we embedded in our application. The equivalent command line for compressing an image as guided by the needs in our work using the kakadu encoder would be

```
kdu_compress -rate - Corder=RLCP ORGt-
 parts=R ORGgen_plt=yes Clayers=3 Cre-
 versible=yes Clevels=n
 -i input_image -o output_codestream
```

Hereafter, the database compressed images are ready to be used by the server for codestream processing and progressive transmission to any remote user through the Internet.

### B. The Server

The server must be working in multithreaded mode, so that it can process many client connections at the same time, something mandatory for our application. The server has the ability to send an image of any resolution by parsing the image codestream and selecting the required segments. Additionally, it can automatically create the respective HTML files through which the clients can view the desired images in desired resolutions.

The way that the server interacts with the client is depicted in Fig. 9. Each time a client requests the thumbnail resolution of an image, the server sends the respective codestream segment, including the main header. When the client has a specific resolution already available and requests for a greater one, the server sends the appropriate differential information: those tile-parts that correspond to the requested resolution, not including the main header and the already sent tile-parts. The client should be able to merge them into one single meaningful codestream.

For automation of the process, the HTML pages in which the images of each resolution are being displayed are created on the server side, always a step ahead of the next greater resolution.

[1]SOT stands for Start-Of-Tile. SOT is a marker in a JPEG2000 codestream denoting the beginning of a tile or a tile-part. For more information refer to [6, pp. 22–23] and [7, pp. 541–542].

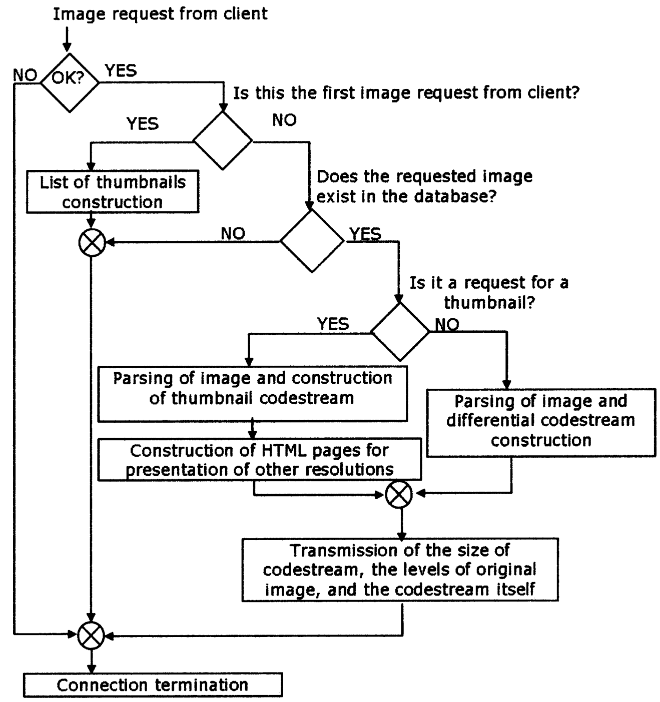


Fig. 9. Logical diagram of server response to client's requests.

The low-resolution image codestream generation at server side is based on file pointer operations, which parse all or part of the original image codestream.

- Whenever read bytes belong to the main header, they are copied to a temporary buffer, with simultaneous replacement of the marker segments XRsiz, YRsiz of marker SIZ (0xFF51) and the wavelet decomposition marker segment of marker COD (0xFF52), with new values appropriate for the correct decoding and representation of the image.
- Whenever read bytes belong to tile-part data or tile-part headers, which are necessary for the requested resolution, they are copied to a new file.
- At the end of the process, all data from the temporary buffer are written to the beginning of the new file and, finally, the file is sent to the client.

### C. The Client

The client consists of a browser plug-in application. The plug-in is responsible for receiving the coded JPEG2000 codestream from the server, decoding it, and then displaying the resulting image inside user's browser.

According to EMBED tag syntax, which embeds various objects in a browser window, the following syntax for embedding our plug-in in an HTML page has been used:

```
<EMBED
 src="name.jpc"
 onclick="(javascript routines)"
 type="MIME type"
 width="width"
 height="height"
 host="Server IP"
 size="Requested resolution">
```

where *src*, *onclick*, *type*, *width* and *height* specify standard attributes of the EMBED tag, whereas *host* and *size* specify plug-in-specific attributes and were introduced for the purposes of our application. Specifically:

- The *src* parameter denotes the name of the image that is being called.
- The *onclick* parameter is a general method of HTML tags and denotes the way browser responds when user clicks the left mouse button inside the plug-in area.
- The *type* parameter denotes data MIME type. This type is defined during the plug-in development and in the present work was defined as "application/x-Netscape-jpc".
- The *width* and *height* parameters define the dimension of the area, inside browser's window, that the plug-in will occupy, and this area must be at least equal to the size of the image that is to be displayed.
- The *host* parameter defines the server's IP address, from which a client requests image data.
- The *size* parameter defines the desired resolution of the image.

After the browser locates, from the MIME type of the EMBED tag, the suitable plug-in that must be executed, it calls it and the whole control of receiving and sending data is passed to the plug-in. Particularly, when the browser starts, all available plug-ins are loaded into memory. If the browser has to handle a page that uses a plug-in to display its content, then the respective plug-in is called. In case the user did not have another browser window open with the same plug-in active, the plug-in is initialized and allocates resources for global shared data. Then a new instance of the plug-in is created into browser's memory, to which the input parameters are passed. These parameters are the attributes of the EMBED tag that called the plug-in. Then the browser draws an area where the plug-in is allowed to display its data. The dimensions of this area are defined, as mentioned before, by the input parameters *width* and *height*. The *src*, *host*, and *size* parameters are used afterwards, during the request to the server for sending a codestream. After appropriate codestream processing by the client, the codestream is decoded and the image is ready to be displayed inside the browser's window. When the user decides to close the page then the browser removes the plug-in instance from memory, but does not release the global shared data that were allocated during the process of initialization. This will happen only if the plug-in that is to be removed from memory is the last active instance, when there are no other pages using the same plug-in. The procedure is described graphically in Fig. 10.

At the beginning of the exchange process between the client and the server, the user's identity is certified, while, at the same time, a message is sent to server for the creation of the thumbnails list, and the control returns to the client browser, which waits to accept user requests. Whenever the user requests an image then one of the following can happen (Fig. 11):

- If the requested image already exists in the browser's cache, then no request is sent to the server and the client just decodes the stored codestream.
- If the requested image exists in the browser's cache in a lower resolution, the client requests the differential information that is the data needed to be appended to the existing codestream in order to produce the new resolution. Information contained in the main header of the codestream, which exist in browser's memory, have to be changed, since they do no longer correspond to the new image resolution. The client must modify the main header in a way that the final codestream can be correctly decoded.

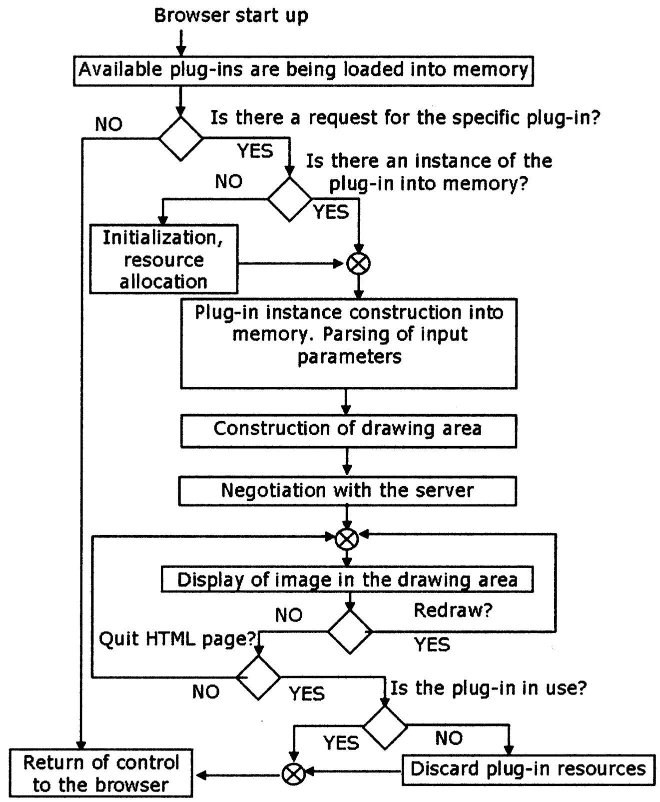


Fig. 10. Logical diagram of plug-in structure.

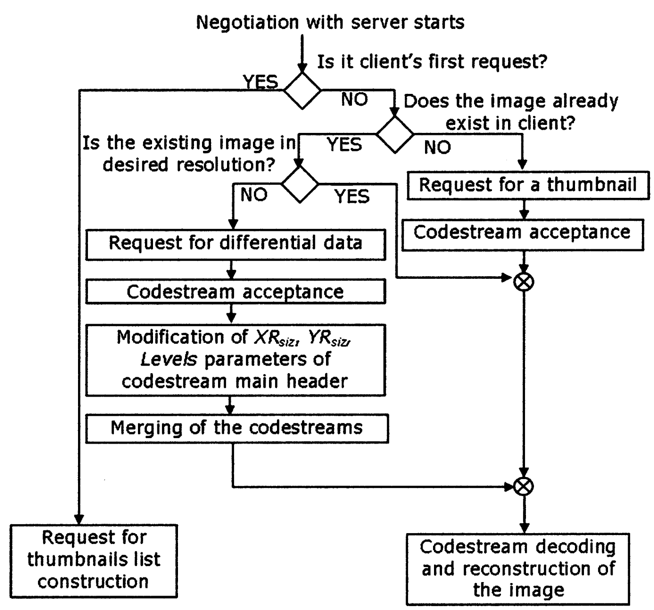


Fig. 11. Logical diagram showing server's processing of client requests.

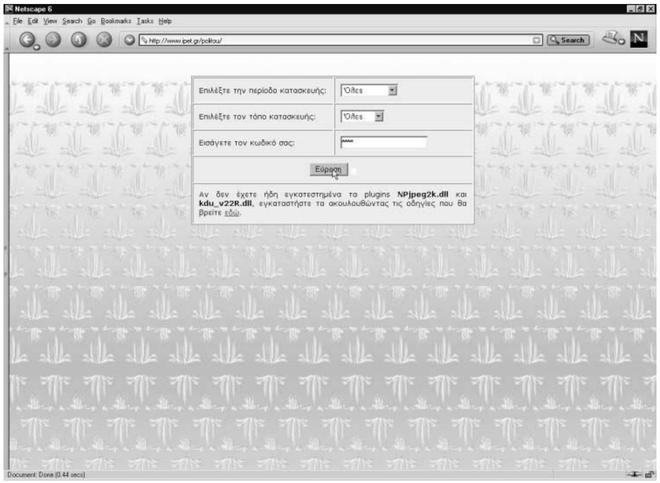

Fig. 12. User identification and main category selection page.

Fig. 13. List of thumbnails.

- If the requested image does not exist in the browser's cache, this indicates that it is the first time the user requests that image, and therefore, the requested resolution is the thumbnail. The client sends the appropriate message to the server and receives data for the 64 × 64 pixels resolution, which are data contained in the first tile-part of the original codestream, including, of course, the main header.

## III. Case Study: The "Ark of Refugee Heirlooms" Database [22]

For a case study we have tested our application using actual image data from the database developed for the purposes of another project of our Institute: "The Ark of Refugee Heirlooms". We tried to develop an Internet database system based on samples from the original database of the Ark and to imitate the Ark's usability and user interface, while employing our newly developed client-server JPEG2000 plug-in technology. The result of this attempt can be found in [25]. When a user visits the database for the first time a page for identification is displayed, as depicted in Fig. 12. At the same time the server creates the

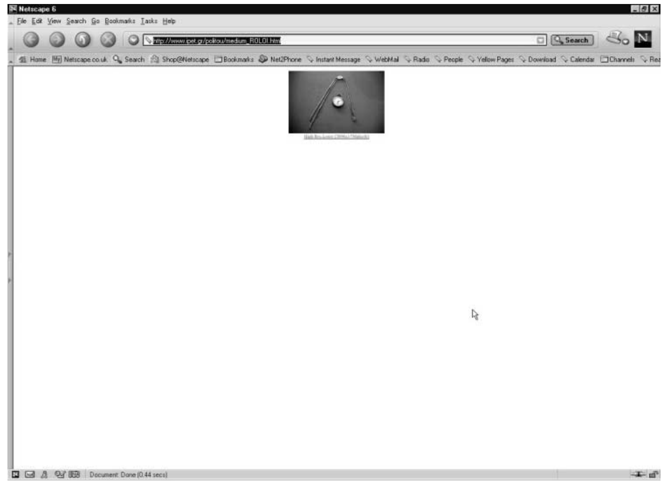

Fig. 14. Medium resolution page for image "watch."

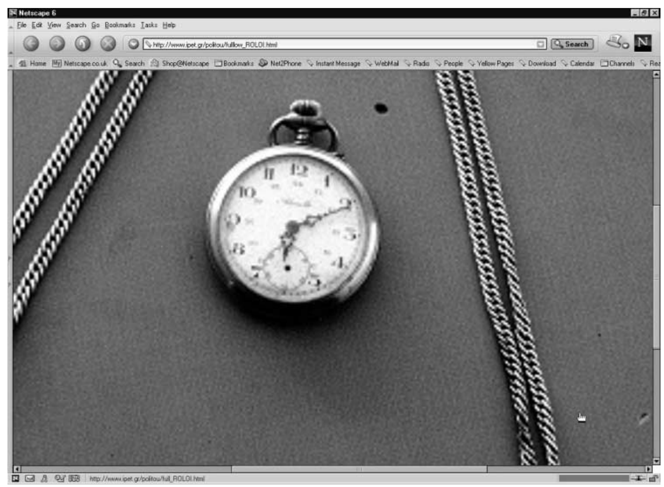

Fig. 15. Maximum resolution page for image "watch," quality 1 bpp.

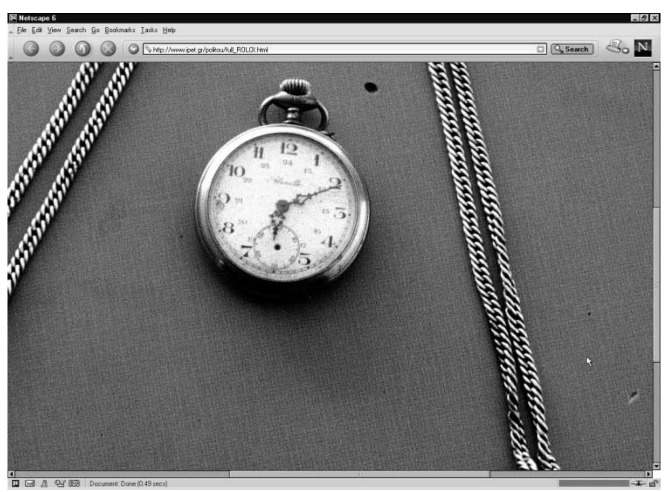

Fig. 16. Maximum resolution page for image "watch," in highest quality.

list of thumbnail images and corresponding HTML files automatically. After being certified the user could review the list of thumbnails, which in our case study consists of a limited number of images from the original database, as illustrated in Fig. 13. In case the user wishes to receive and display an image

TABLE I
COMPARATIVE RESULTS FOR THE FOUR DIFFERENT RESOLUTIONS OF A DATABASE IMAGE "WATCH"

| Resolution (pixels) | Uncompressed image (Kbytes) | Compressed image (Kbytes) | Bits per pixel (bpp) | Transmitted image data (Kbytes) | Transmitted bits per pixel (bpp) | Compression ratio | Transmission time (s) |
|---|---|---|---|---|---|---|---|
| 43x28 | 3.5 | 2.27 | 16.21 | 2.27 | 15.57 | 1.54:1 | 0.32 |
| 169x110 | 54.5 | 26.59 | 12.20 | 24.32 | 10.71 | 2.05:1 | 3.47 |
| 2696x1756 (lossy) | 13870 | 578 | 1.04 | 553.68 | 0.96 | 24.00:1 | 79.09 |
| 2696x1756 (lossless) | 13870 | 4152.29 | 7.48 | 3598.61 | 6.23 | 3.34:1 | 514.05 |

in medium resolution (which will be 256 × 256 pixels maximum), the appropriate link must be selected and the new image will be displayed in a new browser window with a link to the greater resolution, as shown in Fig. 14. Furthermore, if the user needs to display an even greater resolution (max resolution, low quality—1 bpp), a new browser window will open, and a new cycle of downloading the differential codestream, merging with previous data and displaying the final image will commence (Fig. 15). Finally, if the user wishes to view the image in the highest resolution and quality (as originally stored in the database), the same procedure is followed (Fig. 16) but the download time is expected to increase.

Table I summarizes comparative compression and transmission results for image "watch," which is a representative image from the database of the Ark, in all resolutions. Notice that sending the image in maximum resolution—low quality (1 bpp) greatly reduces the coded image size and therefore improves significantly the transmission rate of differential information through the Internet, while at the same time the user is able to sufficiently review its content. It can be noted here that measurements on the whole set of standard test images (a total of 76 images) showed that the average PSNR for JPEG2000 compression at 1 bpp is about 34.5 dB.

## IV. CONCLUSIONS

Usage of JPEG2000 leads to image databases where images are compressed and each stored only in a single file. These files consist of embedded information adequate to retrieve and present them in a variety of ways, even without stream decoding (just parsing). Some of the possibilities are progressive display by resolution and quality, zooming in a particular area, dynamic region of interest and copyright clearance and management.

In this work we developed an infrastructure for progressive by resolution and quality transmission of images that were losslessly encoded and stored in databases. To this end, the new standard in image compression, JPEG2000, has been employed, which is characterized by its feature of embedded, progressive and high-performance coding. A server application that prepares and handles the image database, parses JPEG2000 codestreams and handles client requests has been developed. Additionally, a client browser plug-in was implemented that is responsible for handling user requests, receiving and merging JPEG2000 codestreams, and, finally, decoding and displaying the database images at four preset resolutions, optimized for navigation and network utilization. For the image coding, the kakadu JPEG2000 implementation of the New Wales University at Sidney was embedded in our application. The developed infrastructure is applicable to cultural heritage databases and, thus, can aid in the dissemination of cultural heritage over the Internet.

## REFERENCES

[1] E. Politou, I. Tsevremes, A. Tsompanopoulos, G. Pavlidis, A. Kazakis, and C. Chamzas, ""Ark of Refugee Heirloom"—a cultural heritage database," in *Proc. EVA 2002: Conf. Electronic Imaging and the Visual Arts*, Florence, Italy, Mar. 25–29, 2002.

[2] S. Deshpande and W. Zeng, "Scalable streaming of JPEG2000 images using hypertext transfer protocol," in *Proc. ACM*, Oct. 2001, pp. 281–372.

[3] D. Taubman, "Remote browsing of JPEG2000 images," in *Proc. IEEE Int. Conf. Image Processing, ICIP 2002*, Rochester, NY, Sept. 22–25, 2002.

[4] W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Compression Standard*. New York: Van Nostrand Reinhold, 1993.

[5] Independent JPEG Group, [Online] Available: http://www.ijg.org.

[6] Information Technology—JPEG 2000 Image Coding System—Part 1: Core Coding System, 2000.

[7] D. S. Taubman and M. W. Marcellin, *JPEG2000, Image Compression, Fundamentals, Standards and Practice*. Norwell, MA: Kluwer, 2002.

[8] A. N. Skodras, C. Christopoulos, and T. Ebrahimi, "JPEG2000: The upcoming still image compression standard," in *Proc. 11th Portuguese Conf. Pattern Recognition (RECPAD 2000)*, Porto, Portugal, May 2000, pp. 359–366.

[9] C.Ch. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG2000 still image coding system: an overview," *IEEE Trans. Consumer Electron.*, vol. 46, pp. 1103–1127, Nov. 2000.

[10] D. Santa Cruz and T. Ebrahimi, "An analytical study of the JPEG2000 functionalities," in *IEEE Int. Conf. Image Processing*, Vancouver, BC, Canada, Sep. 2000.

[11] M. W. Marcellin, M. Gormish, A. Bilgin, and M. Boliek, "An overview of JPEG 2000," in *Proc. IEEE Data Compression Conf.*, Snowbird, UT, Mar. 2000.

[12] T. Chinen and A. Chien, Visual Evaluation of JPEG2000 Color Image Compression Performance, ISO/IEC JTC1/SC29/WG1 N1583, Mar. 2000.

[13] T. Ebrahimi, D. Santa Cruz, J. Askelöf, M. Larsson, and C. Christopoulos, "JPEG 2000 still image coding versus other standards," in *Proc. SPIE Int. Symp.*, San Diego, CA, July 30–Aug. 4 2000.

[14] C. Christopoulos, J. Askelof, and M. Larsson, "Efficient encoding and reconstruction of regions of interest in JPEG2000," in *Proc. Eur. Signal Processing Conf. (EUSIPCO-2000)*, Tampere, Finland, Sept. 2000.

[15] ——, "Efficient region of interest encoding techniques in the upcoming JPEG2000 still image coding standard," in *Proc. IEEE Int. Conf. Image Processing*, Vancouver, BC, Canada, Sept. 2000.

[16] D. Santa Cruz, M. Larsson, J. Askelof, T. Ebrahimi, and C. Christopoulos, "Region of interest coding in JPEG2000 for interactive client/server applications," in *Proc. IEEE Int. Workshop Multimedia Signal Processing*, Copenhagen, Denmark, Sept. 1999, pp. 384–389.

[17] I. Moccagata, S. Sodagar, J. Liang, and H. Chen, "Error resilient coding in JPEG-2000 and MPEG-4," *IEEE J. Select. Areas Commun.*, vol. 18, pp. 899–914, June 2000.

[18] G. Pavlidis, A. Tsompanopoulos, N. Papamarkos, and C. Chamzas, "JPEG2000 over noisy communication channels—the cost analysis aspect," in *Proc. IEEE ICIP 2002*, Rochester, NY, Sept. 22–25, 2002.
[19] *Intro to TCP/IP Programming*, [Online] Available: http://www.catalyst.com/support/tutorials/tcpintro/index.html, Dec. 16, 2002.
[20] D. E.Douglas E. Comer, *Internetworking With TCP/IP*. Englewood Cliffs, NJ: Prentice-Hall, Sept. 2000.
[21] *Netscape Plug-In Guide*, [Online] Available: http://developer.netscape.com/docs/manuals/communicator/plugin/index.htm, Jan. 15, 1997.
[22] *Ark of Refugee Heirloom—A Cultural Heritage Database*, [Online] Available: http://www.ceti.gr/kivotos, 2001.
[23] *Thracian Electronic Thesaurus*, [Online] Available: http://thesaurus.duth.gr, 1999.
[24] D. Taubman, *Kakadu Software—A Comprehensive Framework for JPEG2000*, [Online] Available: http://www.kakadusoftware.com, 2002.
[25] Cultural & Educational Technology Institute, http://www.ipet.gr/kivotos_paper, Oct. 2002, currently accessible only with Netscape, password=1234.

**Eugenia A. Politou** received the diploma degree in electrical and computer engineering from Democritus University of Thrace, Xanthi, Greece, in 1998, and the MSE degree in digital image processing and Internet databases from Democritus University of Thrace, Hellas, in 2002.

During 1999 and 2002 she was working in the Cultural & Educational Technology Institute (CETI), Hellas, as an Adjunct Researcher on database design and multimedia development. In 2003, she joined the Computer Laboratory, Oxford University, Oxford, U.K., where she is a Research Assistant on the eDiaMoND research project and specifically on the database design. Her research interests include database design, image processing, and Internet programming.

Ms. Politou is a member of the Technical Chamber of Greece since 1999.

**George P. Pavlidis** (S'01) received the diploma degree in electrical and computer engineering from the Engineering School of Democritus University of Thrace, Xanthi, Greece, in 1995. He is currently pursuing the Ph.D. degree in digital image segmentation, compression, and transmission from Democritus University of Thrace.

Since 1991, he has been a member of numerous research and development projects involving informatics and multimedia in education, and the production of interactive multimedia applications. In 2002, he joined the Cultural & Educational Technology Institute to work as a Multimedia Engineer.

Mr. Pavlidis has been a member of the Technical Chamber of Greece since 1995.

**Christodoulos Chamzas** (S'75–M'79–SM'85) was born in Komotini, Greece. He received the Diploma degree in electrical and mechanical engineering from the National Technical University of Athens, Athens, Greece, in 1974 and the M.S. and Ph.D degrees in electrical engineering in 1975 and 1979 from the Polytechnic Institute of New York, Farmingdale.

From 1979 to 1982, he was an Assistant Professor with the Department of Electrical Engineering, Polytechnic Institute of New York. In September 1982, he joined AT&T Bell Laboratories, Holmdel, NJ, where he was a Member of the Visual Communications Research Department until 1990, where he worked on adaptive systems, mobile communications, multimedia image databases, and image coding. Since September 1990, he has been a member of the Faculty of the Electrical Engineering Department at Democritus University of Thrace (http://www.ee.duth.gr), where he was Department Head (2001–2003), Director of the Electric Circuits Analysis Laboratory of the Image Processing and Multimedia Unit (http://ipml.ee.duth.gr) and of the Cultural And Educational Technologies Institute (http://www.ceti.gr). He has been a major player in the definition, design, and implementation of the CCITT/ISO (JBIG, JPEG, etc.), standards for coding, storage, and retrieval of images (color and bilevel) an area where he holds six international patents. In 1985–1986, he was a Visiting Professor with the Department of Computer Science at the University of Crete, Iraklion, Greece. From 1994 to 1998, he was a Member of the Telematics for Knowledge Working Party of EE, representing Greece. He has held summer positions in Greece, England, Portugal, as well as at Bell Laboratories. His primary interests are in digital signal processing, image coding, multimedia, and communications systems. He is currently interested in the implementation of multimedia image database algorithms either with fast software or with VLSI design.

Dr. Chamzas is a member of the Technical Chamber of Greece and Sigma Xi. He was an Editor for the IEEE TRANSACTIONS ON COMMUNICATIONS (1989–1996).