Network Access to Parts of Archived Document Image Files

Martin Boliek and Michael J. Gormish, Ricoh Innovations, Inc., Menlo Park, California, USA

Abstract

Effective archive of large documents requires the ability to retrieve and browse, process, distribute, and display and print at high quality. To do so efficiently, it is desirable to access only portions of a document file that pertain to any given task. For example, retrieval might be facilitated by quick access to keywords, text, and metadata. Browsing only requires portions of a document data for viewing. High quality display and printing require different resolutions and bit-rates. Such access is key for documents whether they are stored in image form or in a page description language.

This paper presents the philosophy of progressive network access for document images embodied in the JPEG 2000 family of standards. A discussion of the partial document image data needs for certain archive and retrieval tasks is discussed. Performance measures compare JPEG against JPEG 2000 continuous-tone compression with the JPM file format and the JPIP interactive protocol for various client-sever document imaging tasks.

Progressive Access to Document Images

Even as the cost of storage falls, compression remains critical for document image files. Network access and bandwidth costs are still relatively high. Also, the analysis of a document image, performed during compression, can yield valuable information about that document.

To achieve reduced network bandwidth, the compressed image data must be presented in such a way that the correct subset of the coded data can be extracted, processed, transmitted, and decoded. To send all of the document image data when it is not needed wastes bandwidth. Furthermore, to reduce complexity at the document archive server it is necessary to extract the correct compressed document image data without complex transcoding or decoding of the codestream.

JPEG 2000 [1] is a progressive continuous-tone compression system designed to enable access to different resolutions (thumbnails, display, or printing), scalable bit-rates all the way to lossless (for appropriate pixel accuracy), and regions-of-interest (for panning and partial access). The companion JPIP interactive protocol standard [2] provides a language for requesting and sending parts of a JPEG 2000 codestream. Together, these two international standards offer access flexibility for continuous-tone imagery.

For document images, greater advantage for compression and access is achieved with the JPM mixed raster content file format [3]. JPM uses a layered object composite image model that enables sharp edge image areas, such as text, to be separately compressed with an efficient binary compressor, such as JBIG [4]. Objects and portions of objects can be accessed separately using JPIP.

Document Images Versus Rendering Formats

Using images is the natural and obvious modality for documents that have been scanned. Although much effort has been expended in analyzing the content and context of scanned documents, the imagery remains best for visualization and display.

Documents that are created electronically and archived are usually stored in a page description format such as PDF or PostScript. However, there are advantages to converting these to images before archiving as well. In many cases, the document images can be compressed to a smaller size than the original PDF (as reported in [10]). Document images are easier to prepare for display (decode versus render) and do not suffer from missing font information.

Progressive Image and Document Coding History

In the mid 1980s the ISO and ITU (then CCITT) collaborated in a unique standardization effort called the Joint Photographic Experts Group, or JPEG [5]. Their goal was to make a color image compression standard and the result was one of the most successful data interchange standards in modern history. The ever popular JPEG Baseline mode compression is fast and relatively efficient. However, in the interest of better compression, less memory usage, and lower computation, the data for reconstructing different resolutions, bit-rates, and regions is hopelessly commingled.

The original JPEG committee did anticipate the need for sophisticated access into the coded data. A sequential ordering for progressive bit-rate, a hierarchical ordering for progressive resolution, and a (completely incompatible) lossless mode are included. For a number of different reasons, these modes are not popular. The complexity of these orderings is high, the memory use is large, and they are difficult to use together. Although a latter standard, JPEG Part 3 [6], offered regional access with tiling, it only compounded the complexity. JPEG Part 3 also included a file format, but did not support pages, objects, layers, or offer a protocol for client-server interaction. In all of these orderings, there was only one way to access the data -- in the original order the data was encoded.

In the mid 1990s, Kodak, Microsoft, Hewlett Packard, and others collaborated on a file format called Flashpix [7]. Flashpix allows representing an image at different resolutions and regions. It uses JPEG-like independently compressed streams for a redundant array of image tiles at different resolutions. The redundancy, and the lack of a progressive bit-rate mode, make this format less efficient for compression than JPEG. However, paired with the Internet Imaging Protocol [8] it enables network access to the right regions and resolutions for different applications. Once again, there is no support for pages, objects, or layers.

The Open Prepress Interface (OPI) [9], developed by Aldus (now Adobe Systems, Inc.) allows for representing an image within a document in two ways. For viewing and editing, the included image can be a low resolution, low quality version. For printing, the image is replaced with a print resolution version stored somewhere else, like a central server. The actual imagery is usually in the TIFF format and can be used by any document layout or word processing format. This system is designed for high quality print production

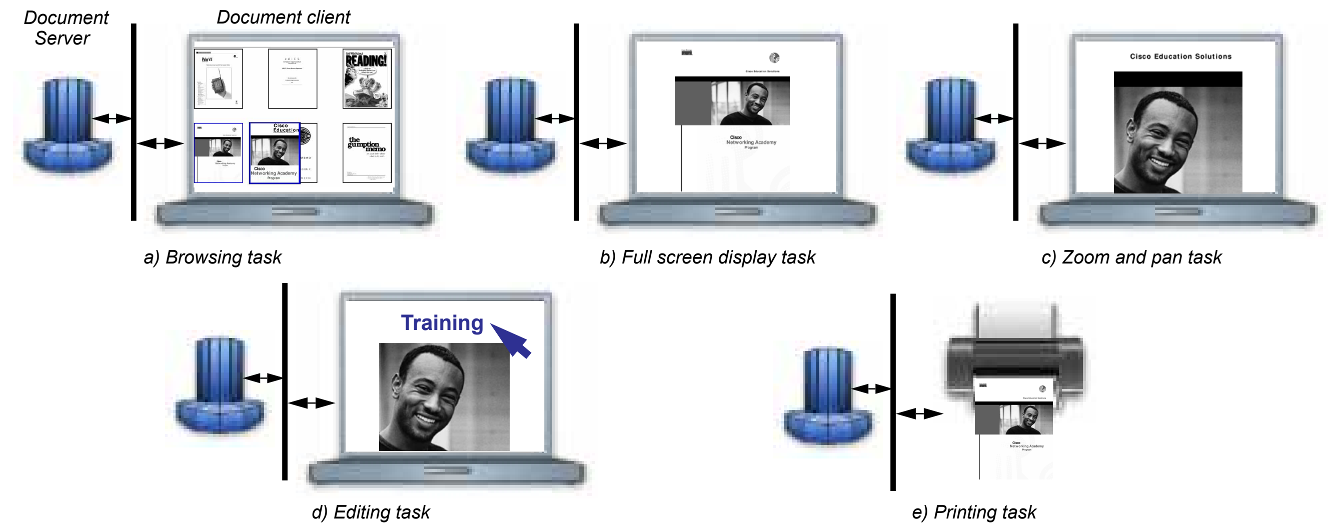


Figure 1. Example client-server task interactions

and is rather complicated for archive applications.

DjVu [10] is a compression system and file format similar to JPM. Developed by AT&T, DjVu enables layered images to be compressed by different coders. While DjVu supports layers and resolutions, it does not support objects, it is proprietary, and does not have a communication protocol.

Linearized PDF [11] is a method of ordering PDF files so that access to individual pages can be performed before all of the data has been transmitted. The data is ordered according to usage and there are hint tables that provide the composite instructions at the beginning of the file rather than the end. This approach can certainly be used on image-based PDF files, as well as rendered files. However, the access is only on a page level. There is no support for editing. Any change to the file alters the linearization characteristic.

Archive and Retrieval Tasks

Several different archive-related tasks require partial data from document images. Retrieval requires attribute indices and text from documents. Browsing requires partial imagery from the document for a reduced resolution or SmartNail presentation [12] (a technique that automatically displays the most prominent parts of a page at recognizable scale). Display on a computer requires modest resolution but high pixel precision. For reading, high resolution portions of a document may be displayed. Printing requires the highest resolution, especially for text regions, but possibly not as high pixel precision.

JPEG 2000, JBIG, JPIP, and JPM allow organization and progressive extraction and communication of document images in many different dimensions simultaneously, including by resolution, bit-rate, tile, component, layout object (including mask and image), page, and page collection. With this array of ways to describe an image, many, if not most, of the document image archive tasks can be easily supported.

New Metrics for Image Compression

The classic rate-distortion metric for image compression only measures the performance of a single preset task. The same compression system that may perform adequately for one task, such as compression for storage, may be unacceptable for another task, such as progressive rendering to a limited display client.

To measure the performance in a system that serves several devices and performs many tasks, an ensemble of performance measures should be used. Consider the types of interaction and interchange needed for a given application. What types of visualization are needed, icons or thumbnails for browsing, full screen for identification, zoom and pan for reading, high resolution printing? What is the environment, stand-alone PC, client-sever network, low bandwidth network (e.g. cell phone)? What are the time, bandwidth, display, computation, and memory considerations? Armed with these application-specific parameters and constraints, it is possible to come up with an array of performance metrics. (This paper describes only a few of the results that might be used.)

Performance Example Experiment

To illustrate the differences in performance between JPEG 2000, JPM, and JPIP versus the original JPEG standard, consider specific tasks such as browsing, full screen display, zoom and panning, editing, and printing. Figure 1 shows these tasks which are described below.

Browsing

Visualization of representative document thumbnails or icons is an important aspect in browsing. The most common visualizations are simple thumbnails. Many image archive systems redundantly store thumbnails for this purpose. Often, however, the size of the thumbnail is user selectable. In this case it is necessary to scale down the original (or have a larger redundant thumbnail).

The proposed system (JPEG 2000, JBIG, JPM, JPIP) has the ability to extract thumbnail size versions efficiently without redundant storage. More interesting is the ability to extract parts of the document image, possibly at different resolutions. This can support the creation of SmartNails with readable tiles and text for a given size display. Since the SmartNail adaptively selects the resolution based on the user selected display size, it is a great advantage to have access into portions of the document imagery.

Full Screen Display

Most LCD and CRT displays for computers have a pitch between 72 and 100 dpi. The aspect ratio differs but most are landscape and most are not 11 inches high. Thus, to display a full document image page, resolution less than the pitch of the display is required. Once again, because of the variability in pitch, screen size, and application, the amount of data required to display the full page varies, but is almost always much less than that available in the source document image.

Zoom and Pan

A full page display of a document on most computer screens is not readable. In order to read the document on the computer screen it is necessary to zoom into the document image, showing only a portion of the screen. To continue reading, the user will want to pan around the image. Because the pattern of zooming and panning is user selectable, the data needed and the order in which it is needed varies.

Editing

Often the user will want to perform various editing, annotation, and data entry functions on a document. Organization into objects can offer logical locations for this editing. Objects can be altered, obscured, and/or replaced independently. Handling only objects at screen resolution not only reduces the editing computations, but significantly reduces the bandwidth for the download and upload operations.

Printing

For high quality, the highest resolution is required for printing. A connection directly between the server and the printer can reduced the number of times the total document is transmitted. It is transmitted directly to the printer rather than to the client computer and then the printer. This savings is possible if the entire image has not already been sent to the client computer.

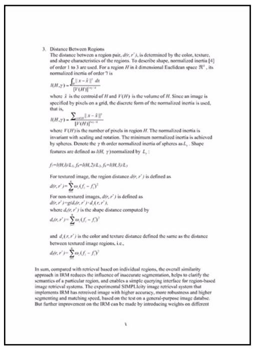
Experimental Setup

Four different encodings are compared:

1. A scanned version of the document page is separated into foreground, background, and mask images that span the entire page. The foreground and background are compressed with JPEG 2000 and the mask is compressed with JBIG.
2. A scanned version of the document page is compressed with JPEG baseline at a quantization that provides a similar quality (measured in SNR) to (*1*).
3. A rendered (noiseless source document image) version of the document page encoded as in (*1*).
4. A rendered version of the document page is compressed with JPEG baseline at a quantization that provides a similar quality (measured in SNR) to (*3*).

Five measures are reported for each encoding type and image.

a. Total size, 300dpi, letter size. This is the size of the file that is stored on the server and provides the best printed output.
b. Signal to Noise Ratio, comparing the full resolution compressed version to the original (scanned or rendered) version.
c. Number of bytes used needed to render the image at 75 dpi.
For the JPM files this includes all of the header information about the layout of the page, the type of compression used, and the location of the codestreams in the file. It also includes the complete codestreams for the JBIG compressed data, however, JPEG 2000 objects are sent at a reduced resolution appropriate for the screen.
For the JPEG encoding it is necessary to send the entire file, or decode the entire file, and recompress a low resolution version and send it.
d. The user is assumed to zoom in on the upper right quarter of the page and request full resolution data (300 dpi).
For the JPM files, the user already has the high resolution masks (JBIG codestreams), but the high resolution data for the JPEG 2000 codestreams is now sent. However, because JPEG 2000 allows regions access only data that intersects the upper right hand corned is needed.
For the JPEG encoding the user is assumed to° have received all the data in the previous step.
e. Adding an object (e.g. annotation marks) is an example of an editing operation.
For the JPM image it is possible to create an entirely new file, that references the codestreams on the server. Thus the number of bytes required is just the size of the header of this new file, plus the size of the annotation image.
For the JPEG case the annotation is added, and the image is recompressed with JPEG and sent back to the server.


3. Distance Between Regions
The distance between a region pair, $d(r, r')$, is determined by the color, texture, and shape characteristics of the regions. To describe shape, normalized inertia [4] of order 1 to 3 are used. For a region H in k dimensional Euclidean space $\Re^k$, its normalized inertia of order γ is

$$I(H,\gamma)=\frac{\int_H \| x-\hat{x} \|^\gamma dx}{[V(H)]^{1+\gamma/k}}$$

where $\hat{x}$ is the centroid of H and $V(H)$ is the volume of H. Since an image is specified by pixels on a grid, the discrete form of the normalized inertia is used, that is,

$$I(H,\gamma)=\frac{\sum_{x\in H} \| x-\hat{x} \|^\gamma}{[V(H)]^{1+\gamma/k}}$$

where $V(H)$ is the number of pixels in region H. The normalized inertia is invariant with scaling and rotation. The minimum normalized inertia is achieved by spheres. Denote the γ th order normalized inertia of spheres as L_γ. Shape features are defined as $I(H, \gamma)$ normalized by L_γ:

$f_7=I(H,1)/L_1$, $f_8=I(H,2)/L_2$, $f_9=I(H,3)/L_3$

For textured image, the region distance $d(r, r')$ is defined as

$$d(r,r')=\sum_{i=1}^{6}\omega_i(f_i-f_i')^2$$

For non-textured images, $d(r, r')$ is defined as
$d(r,r')=g(d_s(r,r'))\cdot d_t(r,r')$,
where $d_s(r, r')$ is the shape distance computed by

$$d_s(r,r')=\sum_{i=7}^{9}\omega_i(f_i-f_i')^2$$

and $d_t(r,r')$ is the color and texture distance defined the same as the distance between textured image regions, i.e.,

$$d_t(r,r')=\sum_{i=1}^{6}\omega_i(f_i-f_i')^2$$

In sum, compared with retrieval based on individual regions, the overall similarity approach in IRM reduces the influence of inaccurate segmentation, helps to clarify the semantics of a particular region, and enables a simple querying interface for region-based image retrieval systems. The experimental SIMPLIcity image retrieval system that implements IRM has retreived image with higher accuracy, more robustness and higher segmenting and matching speed, based on the test on a general-purpose image databse. But further improvement on the IRM can be made by introducing weights on different

3

Figure 2. Image used for testing

Two images are used. These are shown in Figure 2. Uncompressed, these images have 2550 x 3296 = 8,404,800 pixels.

* Compound image with text, graphics, and imagery shown in Table 1.
* Text image with different fonts and mathematical equations shown in Table 2.

Results and Observations

In all cases, the JPM compression performs better than the JPEG file (*rows a* and *b* in Table 1 and Table 2). The savings is more significant on the text image because the JBIG compressor is far more efficient on binary data. Notice that, for the rendered (noiseless source) image, JPEG requires a lot more data to approach the JPM file distortion. This distortion is really close to lossless.

On the compound image scan comparison (*row b, col 1, 2*) the SNR does not tell the whole story. The bit allocation between the sharp text and imagery is excellent and the JPM and the image is quite acceptable. The JPEG image, however, is unacceptable with color contouring, block noise, and a blotchy mosaic.

In the case of the full screen display (*row c*) between 50% and 75% of the whole page file for the JPM file. This includes all of the bytes in the file for the low resolution rendering, but ignores the fact

Table 1: Comparison of different encodings and formats for compound image

	1) Scan JPM	2) Scan JPEG	3) Render JPM	4) Render JPEG
a) Full size, 300dpi	101,533 bytes	175,584	161,402	1,948,544
b) PSNR, full size	26.4 dB	25.5	41.4	41.4
c) 75 dpi, full screen	72,644 bytes	175,584	78,373	1,948,544
d) 300 dpi, zoom and pan, extra bytes, total bytes	+ 7,222 bytes total 79,866 bytes	+ 0 175,584	+ 35,954 114,327	+ 0 1,948,544
e) Add object, bytes to server	1,755 bytes	288,888	4,661	2,160,084

Table 2: Comparison of different encodings and formats for text image

	1) Scan JPM	2) Scan JPEG	3) Render JPM	4) Render JPEG
a) Full size, 300dpi	83,932 bytes	158,236	29,714	1,496,431
b) PSNR, full size	27.0 dB	26.7	74.17	68.0
c) 75 dpi, full screen	37,525 bytes	158,236	29,714	1,496,431
d) 300 dpi, zoom and pan, extra bytes, total bytes	+ 11,602 bytes total 49,127 bytes	+ 0 158,236	+ 0 29,714	+ 0 1,496,431
e) Add object, bytes to server	1,755 bytes	244,582	2,843	1,766,403

that the image can be satisfactorily displayed long before all the bytes are received.° Much of the extra low resolution data is only needed for higher resolution display. For the same full screen display the entire JPEG file must be transmitted. For the text image, the entire rendered JPM file (*row c, col 3*) is transmitted because all of the data is in the compressed with JBIG. This implementation does not use the resolution reduction feature of JBIG.

To zoom into the image only requires a small amount of new data (*row d*), given the entire page has already been sent at low resolution. Note that, because of the high quality of the low resolution image sent (*row c*) the initial zoomed display is quite good even before the additional data is received.

Note that this particular steps also did not make use of JPIPs ability to send only a subset of the set of masks. Because initial transmission was the full page, requiring all the masks be sent, there is data that would not have been sent if the zoomed image were first.

To add an annotation to the file (*row e*) a small binary image is created. For JPM, this new object, and the updated header information, requires little data be transferred back to the server. The new document image contains the new image and (if no other action is taken) the undo image. The JPEG image was completely decompressed, the new object was (irrevocably) merged with the original image, and the new image recompressed with additional loss (even in the unedited regions).

Conclusions and Future Work

This paper suggests, but does not offer, measures for document image representation performance in an archival system. Such metrics can be proposed, formulated, and tested.

Progressive network access to documents stored as images can reduce the bandwidth, memory footprint, computation, and storage required for a number of different archival tasks.

References

[1] JPEG 2000, *JPEG 2000 image coding system: Core coding system*, ISO/IEC 15444-1:2004, www.iso.org.

[2] JPIP, *JPEG 2000 image coding system:* Interactivity tools, APIs and protocols, ISO/IEC 15444-9:2005, www.iso.org.

[3] JPM, *JPEG 2000 image coding system: Compound image file format*, ISO/IEC 15444-6:2003, www.iso.org.

[4] JBIG, *Progressive bi-level image compression*, ISO/IEC 11544:1993, www.iso.org.

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

[6] JPEG Part 3, *Digital compression and coding of continuous-tone still images: Extensions*, ISO/IEC 10918-3:1997, www.iso.org.

[7] International Imaging Industry Association, I3A, *Flashpix*, http://www.i3a.org/i_flashpix.html.

[8] International Imaging Industry Association, I3A, *Internet Imaging Protocol*, http://www.i3a.org/i_iip.html.

[9] Aldus (Adobe), *Open Prepress Interface*, partners.adobe.com/public/developer/en/ps/5660_OPI_2_0.pdf.

[10] L. Bottou, P. Haffner, P.G. Howard, et. al., *High Quality Document Image Compression with DjVu*, http://www.djvuzone.org/djvu/techpapers/jei/jei.ps.gz, 1998.

[11]Adobe Systems, *PDF Reference, Fifth Edition, Version 1.6*, http://partners.adobe.com/public/developer/pdf/index_reference.html#5, 2004.

[12]K. Berkner, E.L. Schwartz, C. Marle, *SmartNails - Display and Image Dependent Thumbnails*, IS&T/SPIE Electronic Imaging Conf., San Jos, January, 2004.

Martin Boliek

Martin Boliek is the leader of the Color Image Processing group at the Ricoh Innovations, Inc. in Menlo Park, California. He was the original proposer of the JPEG 2000 standardization effort and the editor of JPEG 2000 Part 1 and Part 2. He has received a BA in Physics from the University of California, Santa Cruz, an MSEE from the University of California, Davis, and an MBA from the University of San Francisco.

Michael J. Gormish

Michael Gormish works on image and document processing at Ricoh Innovations, Inc. He has BS degrees in Mathematics and EE from the University of Kansas and a MSEE and Ph.D. from Stanford University.° He worked on JPEG 2000 Parts 1, 2, 4, 6, 8, and 9 dealing with compression, compliance, compound image file format, security and interactive protocols.