REMOTE BROWSING OF JPEG2000 IMAGES

David Taubman

The University of New South Wales, Sydney, Australia

ABSTRACT

JPEG2000 is a highly scalable compression standard, allowing access to image representations with a reduced resolution, a reduced quality or confined to a spatial region of interest. As such, it is well placed to play an important role in interactive imaging applications. However, the standard itself stops short of providing guidance or specific mechanisms for exploiting its scalability in such applications. In this paper we describe the JPIK protocol for interactive imaging with JPEG2000. Our results suggest, somewhat surprisingly, that image tiling (dividing into independently compressed sub-images), can be detrimental to effective browsing of large compressed images over low bandwidth connections.

1. INTRODUCTION

JPEG2000 [1] is the most recent image compression standard to emerge from the Joint Photographic Experts Group (JPEG) within the International Standards Organization (ISO). This new standard places a strong emphasis on scalability. A single JPEG2000 data stream typically contains numerous embedded subsets, which may be extracted to recover an efficient compressed representation of the original image at any of a large number of different spatial resolutions, image quality levels, or in selected spatial regions. The fact that these compressed data subsets are embedded within one another allows a low quality or low resolution image, or one whose details cover only a small spatial region, to be incrementally improved by judiciously adding missing elements from the compressed data stream.

In this paper, we consider the problem of browsing large images remotely over low bandwidth connections. The idea is to exploit the scalability of JPEG2000 data streams to deliver the most appropriate subsets from a server to an interactive client, incrementally improving the image quality and/or resolution in a manner which is most consistent with the client's interests at any given time. Although JPEG2000 has been developed with such applications in mind, appropriate mechanisms for delivering JPEG2000 data to an interactive client were not considered as part of the standardization process. As a result, a number of important questions have remained unanswered.

JPEG2000 provides multiple mechanisms for endowing the compressed data stream with spatial accessibility. Included amongst these is the most obvious strategy of partitioning the image into smaller tiles and compressing each independently. As we shall show, however, tiling can actually be a hindrance to effective remote browsing of large images. By contrast, careful exploitation of the EBCOT[2] paradigm at the heart of the JPEG2000 algorithm can provide a more convincing browsing experience, while avoiding the harsh boundary artefacts created by tiling.

The paper is organized as follows. We begin, in Section 2, with a review of the key elements in the JPEG2000 algorithm. Sections 3 and 4 then describe the structural properties of a JPEG2000 data stream which permit access to information relevant to specific spatial regions, spatial resolutions and image quality layers. In Section 5, we introduce the JPIK interactive imaging protocol, which enables an arbitrary JPEG2000 data stream to be interactively served to a remotely located client. In this context, we give experimental results providing evidence for the fact that tiling can hinder efficient browsing of large images.

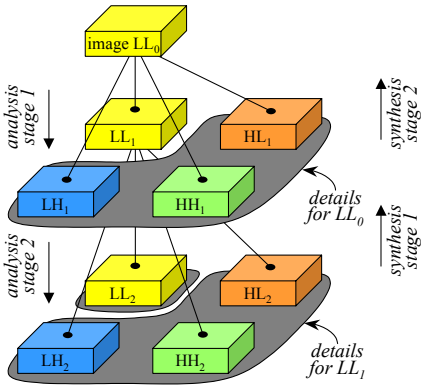

Fig. 1. *DWT with $D = 2$ levels.*

2. SUBBANDS, CODE-BLOCKS AND LAYERS

JPEG2000 is based on the Discrete Wavelet Transform (DWT) and Embedded Block Coding with Optimized Truncation (EBCOT), as illustrated in Figures 1 and 2. The DWT decomposes the image into a collection of spatial frequency subbands, having the same total number of samples as the original image. Specifically, the DWT is best understood in terms of a succession of D "analysis" stages, labeled $d = 1, 2, \ldots, D$, each of which decomposes an intermediate image, LL_{d-1}, into four spatial frequency subbands, LL_d, LH_d, HL_d and HH_d, whose total number of samples is iden-

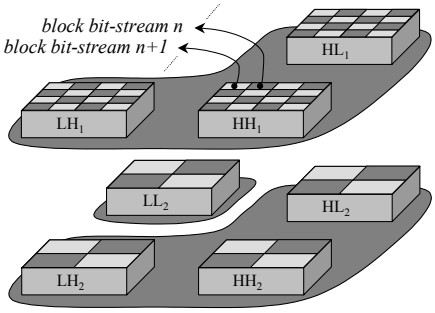

Fig. 2. *Division of subbbands from Fig. 1 into code-blocks.*

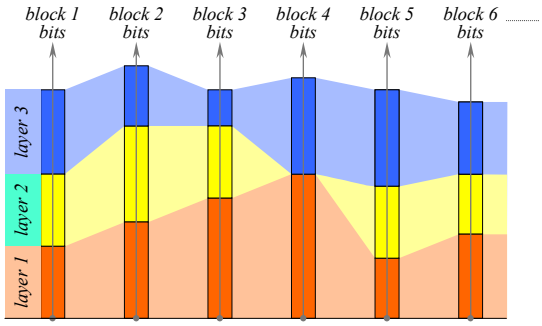

Fig. 3. *Division of embedded block bit-streams into quality layers.*

tical to that in LL_{d-1}. Denoting the original image LL_0 for convenience, each LL_d image is a reduced resolution version of the original, having width and height reduced by a factor of 2^d. The image is completely represented by its lowest resolution version, LL_D, together with the $3D$ detail subbands, LH_d, HL_d and HH_d.

The samples describing each subband are partitioned into rectangular blocks, known as "code-blocks," each of which is independently coded into a finely embedded bit-stream. Truncating the embedded bit-stream associated with any given code-block has the effect of quantizing the samples in that block more coarsely. Each block of each subband may be independently truncated to any desired length, after the compression is complete.

Resolution scalability in JPEG2000 is a direct consequence of the multi-resolution properties of the DWT. By discarding the code-blocks corresponding to the highest resolution detail subbands, and omitting the final stage of DWT synthesis, a half resolution image is reconstructed from the remaining subbands. Dropping the next lower resolution subbands leaves a quarter resolution image, and so forth.

Distortion scalability is introduced through a "quality layer" abstraction. As illustrated in Fig. 3, each quality layer represents an incremental contribution (possibly empty) from the embedded bit-stream associated with each code-block in the image, regardless of the subband to which the block belongs. The sizes of these incremental layer contributions are determined during compression, in a manner which ensures that any leading set of quality layers corresponds to an efficient compressed representation of the original image [2]. Distortion scalability is then realized by discarding one or more final quality layers.

In addition to resolution and distortion scalability, the JPEG2000 compressed data stream also admits a degree of spatial random access. This is because each code-block is associated with a limited spatial region and is coded independently. Typical code-block dimensions are 32×32 or 64×64 subband samples. Code-blocks do not form a partition of the image itself, but rather its subbands. Nevertheless each code-block has a limited region of influence within the reconstructed image, or any of the reduced resolution images LL_d, because the DWT synthesis stages involve operators with finite spatial support. Thus, given any spatial region of interest within any particular image resolution, it is possible to determine the code-blocks which contribute to the reconstruction of the target region and resolution.

3. PRECINCTS AND PACKETS

Although the subband samples in each code-block are coded independently from those in any other code-block, their bit-streams are not explicitly identified within a JPEG2000 data stream. Instead, code-blocks are collected into larger groupings known as "precincts." Each precinct is represented as a collection of "packets," with one packet for each quality layer. This organization allows for the efficient coding of information identifying the portion of each code-block's embedded bit-stream which may be found in each quality layer [2, 3].

In the simplest case, each precinct spans an entire resolution level, d. The lowest resolution level consists of the single subband LL_D, while all others consist of the detail subbands, LH_{d+1}, HL_{d+1} and HH_{d+1}. More usefully, the spatial region occupied by the target resolution LL_d, may be partitioned into multiple precincts. In the extreme case, each precinct might contain only one code-block from each of its constituent subbands. In this case, the lowest resolution precincts would contain exactly one LL_D code-block each, while the higher resolution precincts would contain three code-blocks, one from each of the three detail subbands associated with the resolution level in question.

Apart from efficient coding of layer contributions, the collection of code-blocks into precincts and their layer contributions into packets also has advantages in structuring JPEG2000 files for random access. In particular, JPEG2000 files may include indexing tables, for use in rapidly accessing packets of interest [1, 3]. For large images, these tables could become unwieldly if individual code-block contributions, rather than packets, were identified explicitly within the file.

As discussed in Section 5, we adopt packets as the fundamental unit of data exchange between server and client components in the JPIK interactive imaging protocol. For this purpose, the precincts are best kept to the minimum possible size, with one code-block (typically with 32×32 samples) from each of the constituent subbands. On the other hand, large source files are best represented on the server using a larger precinct size (typically 4 to 16 code-blocks from each member subband), so as to improve disk access efficiency and control the size of indexing tables. Even with minimal precinct dimensions, the precinct grouping structure remains useful for interactive applications, since there is little point in transferring a single code-block without also transferring code-blocks from the other subbands in the same resolution level.

Although the JPEG2000 compression algorithm and its precinct and packet structure, provide excellent tools on which to base an interactive imaging application, the data syntax described in the standard does not permit the interactive construction of a valid data stream from arbitrarily ordered packets. Indeed there are a number of important signalling modifications required to realize efficient interactive image browsing. These extensions are the subject of a newly initiated work item, known as JPIP (JPEG2000 Internet Protocol) within the JPEG working group.

4. TILES

JPEG2000 also allows an image to be divided into smaller subimages, known as "tiles," each compressed independently of the others. Tiling represents an alternate mechanism for enabling spatial access to the compressed data. Unlike code-blocks, each tile's region of influence is completely disjoint from that of any other tile. This would appear to be a beneficial property for interactive browsing of large images. In particular, tiling tends to reduce the number of code-block samples for which information must be transmitted to service any particular spatial region of interest. This is because code-blocks have overlapping regions of influence only within the tile to which they belong.

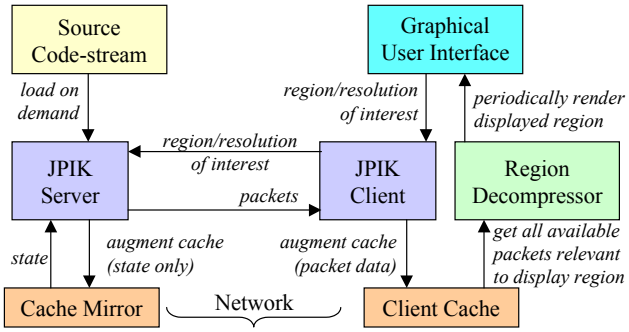

Fig. 4. *JPIK Client-server architecture.*

In addition to reducing redundancy, tiling provides a conceptually simpler framework for region-based access to the compressed image. Tiling has already found its place in various image file formats, including the Flashpix format, which was developed by the Digital Imaging Group (DIG) in part to support interactive image browsing applications. On the other hand, tiles compress less efficiently, introduce unpleasant boundary artefacts at low bit-rates, and do not have good resolution scaling properties. For example, if an 8192×8192 image is partitioned into 128×128 tiles and subsequently accessed at a resolution of 512×512, the 4 highest resolution levels of each tile's DWT must be discarded, so that the 512×512 resolution version effectively involves 8×8 tiles, for which compression efficiency is very poor indeed. The problem becomes even more substantial when working with large geospatial images which can run to many giga-bytes in size.

5. EXPERIENCE WITH THE JPIK PROTOCOL

Our recent experience with interactive image retrieval has been in the context of the JPIK protocol[1]. JPIK is a connection-oriented network communication protocol using TCP, and optionally UDP, for the underlying network transport. As suggested by Fig. 4, the client communicates changes in its current region, resolution and image components[2] of interest, which the server uses to customize its ongoing transmission of compressed data. The server maintains a mirror image of the state of the client's cache, transmitting only new data which is not already available to the client.

The JPIK server essentially delivers a sequence of JPEG2000 packets to the client. The server recovers code-blocks of interest from the source file, taking advantage of packet indexing information included in the file to minimize its memory and disk access requirements. The server assembles new JPEG2000 packets on the fly, using the smallest dimensions compatible with the code-blocks they contain. As mentioned already, the source file will often use larger precincts to improve disk retrieval efficiency, while smaller precincts are preferred for efficient network communications. An efficient variable length signalling scheme is used to identify the particular packets which are being transmitted so that the client knows how to slot them into its cache.

An important feature of the JPIK protocol is that the server does not need to wait for the client to request specific image data.

[1]The acronym stands for "JPeg2000 Interactive with Kakadu." A complete description of the JPIK protocol may be found by following the links at *http://www.kakadusoftware.com*.

[2]Separate image components may be used to represent distinct colour channels, slices from a 3D image data set, or layers from a compound document. Each component is compressed separately.

Fig. 5. 512×512 *region cropped from a* 2944×1966 *image browsed using JPIK; reconstruction based on* 59.4 *kBytes of transmitted data; only luminance component shown here.*

Instead, the server uses whatever notion the client provides concerning its region and resolution of interest, to deduce an efficient schedule for progressively improving the client's viewing experience within this region, taking advantage of the fact that the client may already have some of the relevant data in its cache. Although the protocol can effectively serve tiled or untiled images, the cache tracking strategy is particularly important for untiled images, since code-blocks (and hence precincts) have overlapping regions of support within the image and are frequently reused to service new image regions.

It is worth mentioning that the JPIK approach is not the only way to interactively serve JPEG2000 images. [4], for example, describes a strategy in which the client issues HTTP/1.1 byte range requests into the server's source file. This has the advantage that it can be deployed with existing HTTP servers. However, the client must download a separate index file to discover how it should generate the appropriate byte range requests. Demands on the server's resources are also completely at the mercy of the way in which clients make requests.

By contrast, the JPIK approach allows the server to balance the client's needs with its own limited disk and memory resources. Our JPIK server implementation attempts to minimize disk thrashing, while maintaining client responsiveness, by combining its knowledge of the client's region of interest with dynamic estimates of the channel delay and bandwidth statistics. Experiments have found that a 1 GHz Pentium III with only 384 MBytes of RAM can effectively serve more than 300 clients simultaneously, transcoding each image on the fly from the larger precinct sizes preferred for disk accesses, to the smaller precincts which are preferred for efficient network transmission.

Fig. 5 shows the image quality obtained after a brief browsing session, in which the user quickly zooms into an initial low resolution version of the image, focusing attention on the 10 digit number displayed on the banner flown by the left-most boat in the image. The original uncompressed image is 17.4 Mbytes in size, whereas only 59.4 kB of compressed data are transmitted here.

Notice that image quality decreases progressively, with distance from the region of interest. In fact, Fig. 5 shows only a 512×512 region cropped from the complete image; quality con-

Fig. 6. *Same as Fig. 5, with* 128×128 *tiles and* 62.4 *kBytes sent.*

tinues to decline toward the borders of the full image, not shown here. This behaviour provides convenient visual cues for interactive navigation within the image. It may be attributed to the overlapping regions of interest of the code-blocks, as well as the fact that the first few seconds of transmission are spent sending a low resolution version of the image, until sufficient information is available for the interactive user to select a region of interest.

Fig. 6 reveals the image quality obtained after browsing a tiled version of the same image shown in Fig. 5, with the same region of interest and tiles of size 128×128. The two images have been compressed in such a way as to yield reconstructed images with almost exactly the same MSE. In particular, the MSE of the region of interest recovered after browsing is exactly the same in both cases. To satisfy the client's region of interest, the server transmits a total of 62.4 kBytes to the client. A small amount of information is supplied for tiles which do not intersect with the region of interest. This is because the server starts transmitting low resolution image data immediately, until the interactive user receives sufficient information to identify and select a region of interest.

To further investigate the impact of tiling on image browsing, we configure the JPIK client to use a 640×480 window as its region of interest into the 2944×1966 image. Initially the image is requested at quarter resolution, where the window occupies most of the image. After 5 seconds, the full image resolution is requested, with the 640×480 region of interest placed roughly at the centre of the image. This continues for another 10 seconds, after which we measure the image quality of the low resolution and full resolution regions of interest, in terms of PSNR. Untiled and 128×128 tiled versions of the image are considered at varying transmission rates from 2 kB/s to 16 kB/s, with the results shown in Table 1. Evidently, tiled and untiled images yield similar image quality within the high resolution window[3], but the tiled image performs much more poorly at low resolution.

Although it is difficult to precisely equalize the conditions associated with tiled and untiled image transmission, some useful conclusions may be drawn from our experiments. Even if tile boundary artefacts are not visible in the region of interest itself, their presence in surrounding regions is somewhat disturbing. Moreover, tiles do not appear to offer an advantage in transmission efficiency. In part, this is a consequence of the fact that the tiled image contains many more small code-blocks and hence packets than the untiled image. For example, at resolution LL_3, the 128×128 tiles measure only 16×16 so that the maximum code-block size for each of the associated subbands is 8×8. These small code-blocks do not compress efficiently; the packet signalling overhead also grows as code-blocks become smaller. Effective browsing of large images generally requires a reduced resolution version of the image to be received with sufficient quality to identify regions of interest and this process is much less efficient when the image has been tiled.

[3] In this example, the untiled image performs slightly better, but the high resolution window is not specifically aligned with tile boundaries. In rare cases where it is aligned, tiled images generally perform better in the full resolution window.

Table 1. *Image quality (PSNR) over a* 640×480 *window within quarter and full resolution versions of a* 2944×1966 *image.*

transmission rate	low resolution PSNR		high resolution PSNR	
	tiled	untiled	tiled	untiled
2 kB/s	15.6 dB	23.0 dB	26.3 dB	27.0 dB
4 kB/s	20.8 dB	26.6 dB	29.3 dB	29.6 dB
6 kB/s	23.7 dB	28.6 dB	30.5 dB	30.5 dB
8 kB/s	26.3 dB	30.3 dB	32.4 dB	32.2 dB
12 kB/s	28.5 dB	32.7 dB	35.7 dB	35.8 dB
16 kB/s	30.6 dB	34.7 dB	38.4 dB	38.7 dB

6. CONCLUSIONS

Our experience with the JPIK protocol suggests that tiling JPEG2000 images can be disadvantageous. Exploiting the fact that subbands are coded in small independent blocks, it is possible to avoid tiling altogether. Block boundaries in the subband domain are smeared out in the image domain, due to the spatially expansive properties of DWT synthesis. As a result, hard boundaries are not observed and the image quality decays more or less smoothly with distance from the region of interest. We argue that this provides a more compelling and informative browsing experience than that achieved with tiled images.

Although tiles limit the amount of information not directly connected with the region of interest which must be transmitted to a client, this potential advantage appears to be outweighed by their lower compression efficiency and poor visual properties in regions where relatively little compressed data is received. Perhaps even more importantly, tiles do not have good resolution scaling properties. If a large image is browsed at reduced resolution, the effective tile size can become very small, leading to poor compression and blocking artefacts similar to those experienced with JPEG.

7. REFERENCES

[1] ISO/IEC 15444-1, "Information technology – JPEG 2000 image coding system – Part 1: Core coding system," 2000.

[2] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Trans. Image Proc.*, vol. 9, pp. 1158–1170, July 2000.

[3] D. Taubman and M. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards and Practice*. Boston: Kluwer Academic Publishers, 2002.

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