

Storage system for document imaging applications

Eugene I. Ageenko* and Pasi Fränti**

*Department of Mechanics and Mathematics,
State University of Uljanovsk, Russian Federation

**Department of Computer Science,
University of Joensuu, Finland

ABSTRACT: Document images can be stored compactly using JBIG. The main drawback of the method is the lack of direct access to the compressed image file (spatial access). Here we propose a storage system based on JBIG so that spatial access is also supported. For facsimile images, the increase in the file size is only about 10 %. With this cost we achieve also a fast preview to the image using only about 2 % of the compressing file size to build the preview. The method is also 2.5 times faster in decompression than JBIG.

1. INTRODUCTION

In a typical electronic document management system paper documents are digitized and archived in compressed digital form in order to reduce the costs of archiving, updating, and reproducing of the documents. It increases the productivity of designers because the images can be easily browsed, accessed and retrieved for viewing, printing and even further processing.

The scanning process can be efficiently done using current, relatively inexpensive technology. The questions of a proper encoding algorithm and file format, however, are much more problematic. Document images can be stored compactly using JBIG, the latest binary image compression standard [3]. Unfortunately JBIG does not support spatial access (direct access to the image fragments) as the entire image prior to the accessed part must be decompressed.

The decompression of the entire image can be a major source of inefficiency. High-speed channels are not always available and the transmission of the entire document from server to client might cause inconvenient delays so that the system loses its interactivity. At the same time, only a small part of the image is often needed, or the image is processed and/or viewed fragment by fragment. In some applications, it is not even possible to decompress the entire image because the uncompressed raster image size exceeds the available memory resources (cf. GIS images [4]).

Here we propose a method to represent the images in a compressed raster form so that the spatial access and other requirements of document management systems are

met. Besides the spatial access, the method should also allow small storage size and quick preview of the image. The proposed compression method is based on the baseline JBIG with the following modifications:

- Image is segmented into separate clusters of 128×128 pixels.
- Pointers (indices) to the clusters are stored.
- Separate block level codes are included for quick preview.
- Semi-adaptive context modeling is applied instead of dynamic modeling.

For facsimile images the compression performance of the proposed method is only about 10 % worse than that of JBIG, and the difference becomes smaller when the method is applied for higher resolution images. This is a reasonable price to implement spatial access to the image. The method is also 2.5 times faster in decompression than JBIG. A fast preview is also supported; and only about 2 % of the compressing file size is needed to build the preview image.

2. COMPRESSION SYSTEM FOR DOCUMENT IMAGES

The proposed file data structure (shown in Fig. 1) is composed of two versions of the image: reduced resolution version (*preview data*), and the original image (*pixel level data*). The preview data consists of *block codes* where each pixel in the preview image represents a block of 16×16 pixels in the original image. The pixel level data is segmented into fixed sized clusters which are compressed separately using modified JBIG. The text header consists only of an identification string, image size, and the chosen parameter setup. Implementation details are discussed in the following subsections.

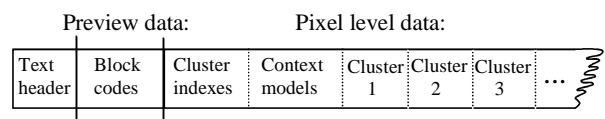


Figure 1. Data organization for document images.

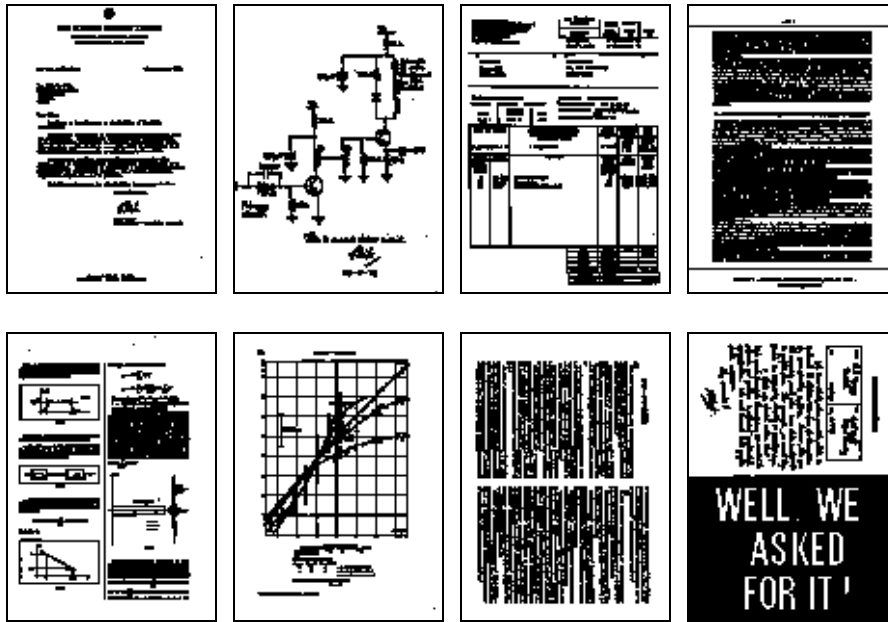


Figure 2. Preview of the CCITT images using 16×16 block size.

2.1 Preview data

The preview data is not actually a binary (reduced resolution) version of the original image but each block is classified either as *all-white*, *all-black*, or *mixed* block. The classifications are coded by two binary decisions: all-white blocks are represented by a single 0-bit, all-black blocks by a bit sequence of 10, and mixed blocks by 11, as proposed in [1]. The actual coding is performed by the same binary arithmetic coder that is used for the pixel level data.

The preview image is constructed from the block codes using a simple resolution reduction technique. Each pixel in the preview image represents a 16×16 block in the original image. The color of a pixel is white if the corresponding block type is all-white; otherwise it is black. This kind of preview is usually sufficient to identify the image, see Fig. 2. Slightly better preview quality can be obtained if the mixed blocks are represented as a color of gray.

The block coding method has several advantages. (1) It is much simpler to implement than the progressive mode of JBIG. (2) It does not increase the bit rate because the pixels in uniform (all-white and all-black) blocks can be omitted without compression. Only the pixels of the mixed blocks need to be compressed. This fact compensates the overhead due to the block codes completely. (3) The decrease in the pixel level data means also a speed-up in decompression times by a factor of 2.5, on an average.

2.2 Pixel-level data

Spatial access is supported by dividing the image into fixed size clusters of 128×128 pixels. A *cluster index table* is constructed from the pointers indicating where the data of each cluster is located in the compressed file. The index table is stored at the beginning of the pixel level data. To restore any part of the image, only the clusters consisting of the desired pixels need to be decompressed. The cluster size is a compromise between compression efficiency and decoding delay. The smaller the cluster, the shorter is the decoding delay but the greater the overhead of the indices.

To permit the clustering, each cluster must be compressed/decompressed independently from the other clusters. In principle, the clusters could be treated as independent images and JBIG applied to them separately. In JBIG the image is compressed pixel by pixel in scan raster order using dynamic context-based probability modeling and QM-coder, the arithmetic coding component in JBIG [5, 6]. Dynamic modeling has the advantages that only one pass over the image is needed and no overhead (models or code tables) need to be stored in the compressed file. The learning cost, however, becomes relatively high when coding smaller sizes of data. Semi-adaptive modeling is therefore used.

In semi-adaptive modeling two-passes over the image are needed. In the first pass (*analysis phase*) statistical models are constructed for each context and stored in compressed file. The block codes are constructed during

	STORAGE SIZE (kB)			DECOMPR. SPEED (s)		
	JBIG	EDM <i>Total</i>	EDM <i>Preview</i>	JBIG	EDM <i>Total</i>	EDM <i>Preview</i>
CCITT 1	14708	16819	297	32	8	0.19
CCITT 2	8491	9834	372	32	5	0.30
CCITT 3	21990	24553	510	32	13	0.37
CCITT 4	54291	61766	514	33	26	0.29
CCITT 5	25823	28272	515	33	13	0.33
CCITT 6	12552	14084	494	32	8	0.42
CCITT 7	56305	58453	801	33	23	0.43
CCITT 8	14229	16149	770	32	10	0.72
TOTAL:	208389	229930	4273	32.4	13.3	0.38

Table 1: Compression performance of the proposed storage system.

the same analysis phase. The models are obtained by calculating the frequencies of white and black pixels for each context separately. The resulting probabilities of each context are then mapped to the nearest state in the probability estimation machine of the QM-coder and stored in the compressed file using one byte per context. The total overhead for storing the model is 2^k bytes for a k -pixel context.

In the second pass (*compression phase*) the pixels of the mixed blocks are compressed by semi-adaptive JBIG. The same models are used for every cluster. The coder is reinitialized and the models are restored each time when the compression of a new cluster starts. After the cluster has been coded, the data buffer is filled by dummy bits and flushed to the code stream. For the details of the semi-adaptive modeling scheme, see [2].

3. RESULTS

The proposed method for electronic document management systems (denoted here *EDM*) was tested by compressing and decompressing the standard CCITT test images at 200 dpi. The optimal context size for the semi-adaptive model was found to be 9. Larger contexts are impractical because the overhead of the models increases exponentially as a function of context size. Testing results are summarized in the table 1. All test runs were performed using a Pentium-90 computer (80 MIPS).

The compression performance of the proposed method is only about **10 %** worse than that of JBIG. This is quite reasonable cost for supporting spatial access for the compressed file. The compression deficiency originates from the overhead of the index table (0.5 kB) and the context models (0.5 kB); and from the prediction

inaccuracy near the cluster boundaries. It is expected that the difference becomes smaller when the method is applied for larger and/or higher resolution images. Note that the overhead of the block codes is fully compensated by the fact that the pixels in uniform blocks can be omitted without compression. The pixel level data is reduced approximately by the same amount than is the overhead of the block codes.

The preview data (block codes) takes only about **2 %** of the compressed file size allowing instant preview. Once the preview data, context models and the cluster indices are read into memory (about 6 % of the file size in total), any cluster can then be accessed by sequentially decompressing its pixel level code starting from the position given by the index of the cluster. Each cluster takes about 0.4 % of the pixel level data. The method is also 2.5 times faster than JBIG when decompressing the entire image because there are less pixels to be processed by the time-consuming context modeling.

4. CONCLUSION

A new storage system was proposed for document images. Only small modifications were needed for JBIG to achieve quick preview and spatial access to the image. These properties greatly improves the interactivity of any document imaging applications.

5. ACKNOWLEDGEMENTS

The work of Pasi Fränti was supported by a grant of the Academy of Finland, and the work of Eugene I. Ageenko by a grant of Centre for International Mobility (CIMO).

6. REFERENCES

- [1] P. Fränti, "A Fast and Efficient Compression Method for Binary Images", *Signal Processing: Image Communication*, **6**, 1994, pp. 69-76.
- [2] P. Fränti and E.I. Ageenko, "Semi-adaptive modelling for JBIG", *Manuscript*, 1997. (submitted)
- [3] JBIG, Progressive Bi-level Image Compression, ISO/IEC International Standard 11544, ITU Recommendation T.82, 1993.
- [4] R. Pajarola and P. Widmayer, "Spatial Indexing into Compressed Raster Images: How to Answer Range Queries Without Decompression". *Proc. Int. Workshop on Multimedia DBMS*, Blue Mountain Lake, NY, 1996, pp. 94-100.
- [5] W.B. Pennebaker, J.L. Mitchell, *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, 1993.
- [6] W.B. Pennebaker, J.L. Mitchell, G.G. Langdon, R.B. Arps, "An Overview of the Basic Principles of the Q-coder Adaptive Binary Arithmetic Coder". *IBM Journal of Research and Development*, **32**, 1988, pp. 717-726.

