Color Documents on the Web with DjVu

Patrick Haffner, Yann LeCun, Leon Bottou, Paul Howard, Pascal Vincent, Bill Riemers

AT&T Labs - Research
100 Schulz Drive, Red Bank, NJ 07701, USA
{haffner,yann,leonb,pgh,vincent,bcr}@research.att.com

Abstract

We present a new image compression technique called "DjVu" that is specifically geared towards the compression of scanned documents in color at high resolution. With DjVu, a magazine page in color at 300dpi typically occupies between 40KB and 80KB, approximately 5 to 10 times better than JPEG for a similar level of readability. Using a combination of Hidden Markov Model techniques and MDL-driven heursitics, DjVu first classifies each pixel in the image as either foreground (text, drawings) or background (pictures, photos, paper texture). The pixel categories form a bitonal image which is compressed using a pattern matching technique that takes advantage of the similarities between character shapes. A progessive, wavelet-based compression technique, combined with a masking algorithm, is then used to compress the foreground and background images at lower resolution while minimizing the number of bits spent on the pixels that are not visible in the foreground and background planes. Encoders, decoders, and real-time, memory efficient plug-ins for various web browsers are available for all the major platforms.

1 Introduction

With the generalized use of the Internet and the declining cost of scanning and storage hardware, documents are increasingly archived, communicated, and manipulated in digital form rather than in paper form. The growing need for instant access to information makes the computer screen the preferred display medium.

Compression technology for bitonal (black and white) document image archives has a long history (see [9] and references therein). It is the basis of a large and rapidly growing industry with widely accepted standards (Group 3, MMR/Group 4), and less popular and emerging standards (JBIG1, JBIG2).

The last few years have seen a growing demand for a technology that could handle *color* documents in a effective manner. Such applications as online digital libraries with ancient or historical documents, online catalogs for e-commerce sites, online publishing, forms processing, and scientific publication, are in need of an efficient compression technique for color documents. The availability of low-cost, high quality color scanners, the recent emergence of high-speed production color scanners, and the appearance of ultra high resolution digital cameras opens the door to such applications.

Standard color image compression algorithms are inadequate for such applications because they produces excessively large files if one wants to preserve the readability of the text. Compressed with JPEG, a color image of a typical magazine page scanned at 100dpi (dots per inch) is around 100-200KB, and is barely readable. The same page at 300dpi has acceptable quality, but occupies 300-600 KB. These sizes are impractical for online document browsing, even with broadband connections.

Preserving the readability of the text and the sharpness of line art requires high resolution and efficient coding of sharp edges (typically 300dpi). On the other hand preserving the appearance of continuous-tone images and background paper textures does not require as high a resolution (typically 100dpi) An obvious way to take advantage of this is to segment these elements into separate layers. The foreground layer would contain the text and line drawings, while the background layer would contain continuous-tone pictures and background textures.

The separation method brings another considerable advantage. Since the text layer is separated, it can be stored in a chunk at the beginning of the image file and decoded by the viewer as soon as it arrives in the client machine.

Overall, the requirements for an acceptable user experience are as follows: The text should appear on the screen after only a few seconds delay. This means that the text layer must fit in 20-40KB assuming a 56Kb/sec connection. The pictures, and backgrounds would appear next, improving the image quality as more bits arrive. The overall size of the file should be on the order of 50 to 100 KB to keep the overall transmission time and storage requirements within reasonable bounds.

Large images are also problematic during the decompression process. A magazine-size page at 300dpi is 3300 pixels high and 2500 pixels wide and occupies 25 MB of memory in uncompressed form, more than what the average PC can properly handle. A practical document image viewer should therefore keep the image in a compressed form in the memory of the machine and only decompress on-demand the pixels that are being displayed on the screen.

The DjVu document image compression technique [2] described in this paper addresses all the above mentioned problems. With DjVu, pages scanned at 300dpi in full color can be compressed down to 30 to 80 KB files from 25 MB originals with excellent quality. This puts the size of high-quality scanned pages in the same order of magnitude as an average HTML page (which are around 50KB on average). DjVu pages are displayed progressively within a web browser window through a plug-in, which allows easy panning and zooming of very large images without generating the fully decoded 25MB image. This is made possible by storing partially decoded images in a data structure that typically occupies 2MB, from which the pixels actually displayed on the screen can be decoded on the fly.

This paper gives an overview of the DjVu technology, which is described in more details in [2]. The first two sections explain the general principles used in DjVu compression and decompression. The remaining two sections 4 and 5 study the behavior of the the foreground/background segmentation algorithm. The last section details unique features that contribute to the performance of DjVu.

2 The DjVu Compression Method

The basic idea behind DjVu is to separate the text from the background and pictures and to use different techniques to compress each of those components. Traditional methods are either designed to compress natural images with few edges (JPEG), or to compress black and white document images almost entirely composed of sharp edges (Group 3, MMR/Group 4, and JBIG1). The DjVu technique improves on both and combines the best of both approaches. A foreground/background separation algorithm generates three images from which the original image can be reconstructed: the background image, the foreground image and the mask image. The first two are low-resolution color images (generally 100dpi for the background and 25dpi for the foreground), and the latter is a high-resolution bi-level image (300dpi). A pixel in the decoded image is constructed as follows: if the corresponding pixel in the mask image is 0, the output pixel takes the value of the corresponding pixel in the appropriately up-sampled background image. If the mask pixel is 1, the pixel color is taken from the foreground image.

The mask image is encoded with a new bi-level image compression algorithm dubbed JB2. It is a variation on AT&T's proposal to the emerging JBIG2 standard. The basic idea of JB2 is to locate individual shapes on the page (such as characters), and use a shape clustering algorithm to find similarities between shapes [1, 5]. Shapes that are representative of each cluster (or in a cluster by themselves) are coded as individual bitmaps with a method similar to JBIG1. A given pixel is coded with arithmetic coding using previously coded and neighboring pixels as a context (or predictor). Other shapes in a cluster are coded using the cluster prototype as a context for the arithmetic coder, thereby greatly reducing the required number of bits since shapes in a same cluster have many pixels in common. In lossy mode, shapes that are sufficiently similar to the cluster prototype may be substituted by the prototype. A another chunk of bits contains a list of shape indices together with the position at which they should be painted on the page. all of this is coded using arithmetic coding.

For the background and foreground images, DjVu uses a progressive, wavelet-based compression algorithm called IW44. IW44 offers many key advantages over existing continuous-tone images compression methods. First, the wavelet transform can be performed entirely without multiplication operations, relying exclusively on shifts and adds, thereby greatly reducing the computational requirements. Second, the internal memory data structure for IW44 images allows in-place progressive decoding of the wavelet coefficients (without copies), and uses an efficient sparse tree representation. Third, the data structure allows efficient on-the-fly rendering of any sub-image at any prescribed resolution, in a time proportional to the number of rendered pixels (not image pixels). This last feature is particularly useful for efficient panning and zooming through large images. Lastly, a masking technique based on multiscale successive projections [4] is used to avoid spending bits to code areas of the background that are covered by foreground characters or drawings. Both JB2 and IW44 rely on a new type of adaptive binary arithmetic coder called the ZP-coder[3], that squeezes out any remaining redundancy to within a few percent of the Shannon limit. The ZP-coder is adaptive and faster than other approximate binary arithmetic coders.

The idea of foreground/background representa-

Compression	None	GIF	JPEG	DjVu
hobby p15	24715	1562	469	58
medical dict.	16411	1395	536	110
time zone	9174	576	249	36
cookbook	12128	1000	280	52
hobby p17	23923	1595	482	52
U.S. Constit.	31288	2538	604	134
hobby p2	23923	1213	383	68
ATT Olympic	23946	955	285	41

Table 1: *Compressed files sizes (in KB) for 8 documents using the following compression methods: no compression, GIF on the 150dpi image, JPEG on the 300dpi image with quality 20 and DjVu with 300dpi mask, and 100dpi backgrounds. The visual quality of the compressed images can be examined at* `http://www.djvu.att.com/examples/comp`

tion is a key element of the MRC/T.44 standard proposal[8]. Section 6 reports the gain from using JB2 and IW44 rather than the traditional MMR/Group 4 and JPEG which are part of the T.44 standard.

According to an independent test by Inglis [6] with bitonal scanned documents, JB2 in lossless mode achieves an average compression ratio of 26.5, which can be compared to 13.5 for MMR/Group 4, and 19.4 for JBIG1. The lossy mode brings another factor of 2 to 3 over lossless, with more improvement on mostly textual images, and less on images with pictures and in low-quality images.

The performance of IW44 is very similar to the best published wavelet-based encoders. The size of IW44 images is typically 50 to 70% that of JPEG for the same signal to noise ratio. IW44 is particularly good at high compression ratios, and for images with few highly textured areas.

On color documents, the full DjVu method with foreground/background separation can reach compression ratios of 500:1 to 1000:1. As shown in table 1, typical letter-size color documents at 300dpi (catalog or magazine page) compressed with DjVu occupy 30KB to 80KB. Occasionally, larger documents, or document with lots of highly detailed pictures or handwriting may occupy 80 to 140KB. This is 5 to 10 times better than JPEG for a similar level of legibility of the text. DjVu is particularly good at reproducing ancient documents with textured paper.

Results and examples are available from the DjVu digital library at `http://www.djvu.att.com/djvu`. More examples are available from many commercial and non-commercial users of DjVu on the Internet.

3 The DjVu Browser Plug-in

Browsers must provide a very fast response, smooth zooming and scrolling abilities, good color reproduction and sharp text and pictures. These requirements impose stringent constraints on the browsing software. The full resolution color image of a page requires about 25 MB of memory. Decompressing such images before displaying them would exceed the memory limits of average desktop computers.

We developed a web browser plug-in that supports all the major browsers on all the major OS platforms. Downloaded images are first pre-decoded into an internal memory data structure that occupies approximately 2MB per page. The piece of the image displayed in the browser window is decoded on-the-fly from this data structure as the user pans around the page. Unlike many document browsers, each page of a DjVu document is associated with a single URL. Behind the scenes, the plug-in implements information caching and sharing. This design allows the digital library designer to set up a navigation interface using well-known Web technologies like HTML, JavaScript, or Java. This provides more flexibility than other document browsing systems where multi-page documents are treated as a single entity, and the viewer handles the navigation between the pages. The DjVu plug-in supports hyperlinks in DjVu documents by allowing the content designer to specify active regions on the image which links to a given URL when clicked upon.

4 The Foreground/Background Segmenter

The first phase of the foreground/background separation is based on two-dimensional Hidden Markov models[1] with two states (foreground and background) and single Gaussian distributions. While the Gaussian parameters (means and covariances) are estimated on local regions of the image, the "consolidated" [2] transition probability between the foreground state and the background state is regarded as an external constant, the choice of which will be discussed in the next section.

This initial foreground separation stage is designed to prefer *over-segmentation*, so that no characters are dropped. As a consequence, it may erroneously put highly-contrasted pieces of photographs in the foreground. A variety of filters must be applied to the resulting foreground image so as to eliminate the most obvious mistakes.

[1] Equivalent to causal Markov Random Fields.

[2] As we use 2x2 pixel cliques, there are in fact 16 transition patterns, whose probabilities are expressed as a function of one common transition probability.

The main filter is designed to be as general as possible and avoids heuristics that would have to be tuned on hundreds of different kinds of documents. Since the goal is compression, the problem is to decide, for each foreground blob found by the previous algorithm, whether it is preferable to *actually* code it as foreground or as background. Two competing strategies are associated with data-generating models. Using a Minimum Description Length (MDL) approach [7], the preferred strategy is the one that yields the lowest overall coding cost, which is the sum of the cost of coding the model parameters and the cost of coding the error with respect to this "ideal" model. Like most MDL approaches used for segmentation [7], the motivation is to obtain a system with very few parameters to hand-tune. However, the MDL principle is used here to make only *one* decision, thus avoiding the time consuming minimization of a complex objective function. To code the blob as part of the "smooth" background only requires a background model. To code the blob as a piece of foreground that sticks out of the background requires a foreground model, a background model and a mask model.

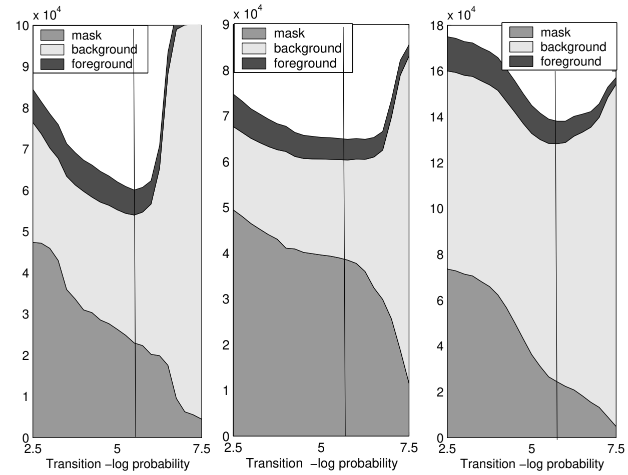
The background model assumes that the color of a pixel is the average of the colors of the closest background pixels that can be found up and to the left. The foreground model assumes that the color of a blob is uniform. What remains to be coded is the boundaries of the mask: the model we use tends to favor horizontal and vertical boundaries.

Thus, in the main filter, the background model allows a slow drift in the color, whereas the foreground model assumes the color to be constant in a connected component. This difference is critical to break the symmetry between the foreground and the background.

5 Optimizing the segmenter

This section shows, on one example, how we defined a semi-automatic tuning procedure for the segmenter.

In the Markov model that drives the foreground/background separation, the transition probability between the foreground and the background states could not be estimated on the data, and was considered as a heuristic choice. The proportion of foreground decreases monotonously as this transition probability decreases. When it reaches zero, no foreground is left. Figure 1 shows the impact of this transition probability on three very different types of documents. The area plots show how the DjVu file size, which is the sum of the number of bytes used to encode the mask, background and foreground layers, evolve as a function of this probability. When it is high, the document is over-segmented and bits are wasted in



Mail order catalog | XVII century book | Textured document

Figure 1: *DjVu files sizes as a function of the negative log probability of the foreground/background transition, reported for three documents. The first two are browsable on the Internet at* www.djvu.att.com/djvu/cat/sharperimage/p0009.djvu *and* www.djvu.att.com/djvu/antics/pharm/p0001.djvu.

encoding a mask that includes images elements, texture and noise. As it converges to zero, the text gets encoded as background. Note that, to avoid interferences, the filtering stage was not applied. In each area plot, the vertical line corresponds to the value of the transition probability for which a human observed the best document quality. It has been observed that, for most document, this point corresponds to the best compression rate. This property greatly facilitates the tuning of the segmenter.

6 Improvements due to JB2 and wavelet compressions

Assuming a satisfactory foreground/background separation, we can focus on the separate compression of the three sub-images. Our test sample is composed of 70 document images and contains highly textured backgrounds, handwriting, mathematical symbols, hand drawings, and musical scores.

For each image, our foreground/background separation algorithm produces 3 sub-images. The raw mask with 1 bit/pixel is 24 times smaller than the original . The raw background image is $3 \times 3 = 9$ times smaller. The raw foreground image is $12 \times 12 = 144$ times smaller. The *raw multi-layer image* yields a compression rate of 6.25:1, as shown on the left of Figure 2. On these sub-images, an objective comparison between DjVu and traditional approaches is possible.

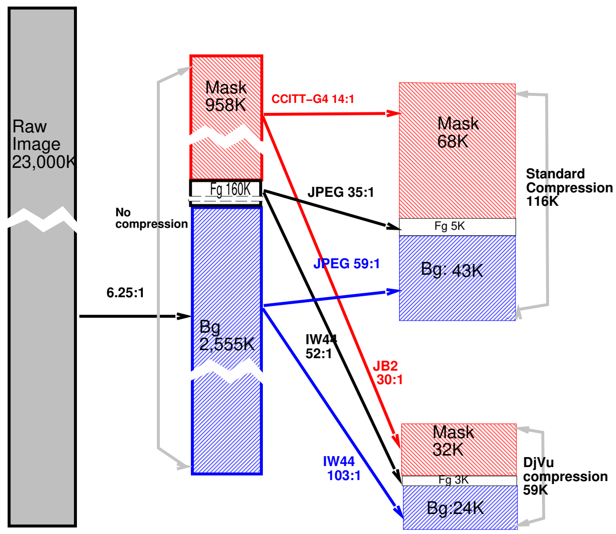


Figure 2: *For a given foreground/background separation, this figure compares, for the three sub-images, the following configurations: (i) no compression, (ii) standard compression techniques such as JPEG and MMR/Group 4 and (iii) compression techniques used in DjVu. Assuming we start from a 23MBytes raw document image, the average size we can expect for each sub-image is reported in the corresponding block. The arrows show the techniques used and the compression rates obtained.*

We measure, for each image, the gain in compression rate brought by the algorithms used in DjVu when compared to the best widely available compression standard for this type of image (MMR/Group 4 for the mask, JPEG for the foreground and background). Figure 2 reports compression rates for the 70 images grouped together.

Comparing JPEG with IW44 for the background layer is performed by imposing the same mean square error on *visible* (non-masked) pixels. Figure 2 shows that the overall compression ratio for the background sub-images improves from 59:1 with JPEG to 103:1 with IW44. Overall, Figure 2 shows that the novel compression techniques used in DjVu to compress the mask, foreground and background sub-images compress our typical 23MBytes image into a 59 KBytes DjVu file. This is about half what we would have been obtained by combining the foreground/background segmentor with JPEG and MMR/Group 4.

7 Conclusion

DjVu, as a new compression technique for color document images, fills the gap between the world of paper and the world of bits. It allows high-quality scanned document to be easily published on the Internet, without implying prohibitive download times for the end-user (with full-color 300dpi pages occupying only 40 to 80 KB, black and white pages only 15 to 40 KB, and ancient books, where most of the color is on the background, 30 to 60 KB)

The UNIX version of the DjVu compression/decompression software is available free for non-commercial use at `http://www.djvu.att.com`. The file format specification and the DjVu reference library is available in source form at the same URL. The DjVu browser plug-in is available for Linux, Windows 95/98/NT, Mac, and various UNIX platforms. The above web site also contains a digital library with over 1000 pages of scanned documents from various origins.

DjVu is already used by a wide variety of commercial and non-commercial users on the web, including University Microfilm Inc. for their 22 million page Early English Book Online service.

References

[1] R. N. Ascher and G. Nagy. A means for achieving a high degree of compaction on scan-digitized printed text. *IEEE Trans. Comput.*, C-23:1174–1179, November 1974.

[2] L. Bottou, P. Haffner, P. G. Howard, P. Simard, Y. Bengio, and Y. LeCun. High quality document image compression with djvu. *Journal of Electronic Imaging*, 7(3):410–428, 1998.

[3] L. Bottou, P. G. Howard, and Y. Bengio. The Z-coder adaptive binary coder. In *Proceedings of IEEE Data Compression Conference*, pages 13–22, Snowbird, UT, 1998.

[4] L. Bottou and S. Pigeon. Lossy compression of partially masked still images. In *Proceedings of IEEE Data Compression Conference*, Snowbird, UT, March-April 1998.

[5] P. G. Howard. Text image compression using soft pattern matching. *Computer Journal*, 40(2/3):146–156, 1997.

[6] Stuart Inglis. *Lossless Document Image Compression*. PhD thesis, University of Waikato, March 1999.

[7] W. Niblack J. Sheinvald, B. Dom and D. Steele. Unsupervised image segmentation using the minimum description length principle. In *Proceedings of ICPR 92*, 1992.

[8] MRC. Mixed rater content (MRC) mode. ITU Recommendation T.44, 1997.

[9] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Van Nostrand Reinhold, New York, 1994.