DjVu: Analyzing and Compressing Scanned Documents for Internet Distribution

Patrick Haffner, Léon Bottou, Paul G. Howard, and Yann LeCun
AT&T Labs-Research
100 Schultz Drive
Red Bank, NJ 07701-7033
{haffner,leonb,pgh,yann}@research.att.com

Abstract

DjVu is an image compression technique specifically geared towards the compression of scanned documents in color at high resolution. Typical magazine pages in color scanned at 300dpi are compressed to between 40 and 80 KB, or 5 to 10 times smaller than with JPEG for a similar level of subjective quality. The foreground layer, which contains the text and drawings and requires high spatial resolution, is separated from the background layer, which contains pictures and backgrounds and requires less resolution. The foreground is compressed with a bi-tonal image compression technique that takes advantage of character shape similarities. The background is compressed with a new progressive, wavelet-based compression method. A real-time, memory efficient version of the decoder is available as a plug-in for popular web browsers.

1. Introduction

Document images have often become easier and cheaper to manipulate in electronic form than in paper form. Traditional libraries are becoming increasingly digital as the costs of scanning and storage are declining. With the generalized use of email and the Internet, the preferred way to communicate documents is electronic, and the preferred display medium is fast becoming the computer screen.

The most computer-friendly representation of a document is symbolic: a string of ASCII text with tags such as SGML/HTML, or a page description language such as Adobe's PDF. Unfortunately, while these formats are appropriate for digitally produced documents, they are not appropriate for scanned documents with significant visual content. Even if Optical Character Recognition (OCR) accuracy were perfect, the visual aspect would be lost in the translated document. Visual details, including font irregularities, paper color, and paper texture, are particularly important for historical documents, but visual content is also crucial in documents with tables, illustrations, mathematical or chemical formulae, and handwritten text. A simple alternative would be to scan the original page and simply compress the image. Several authors have proposed image-based approaches to digital libraries [9, 10] that are restricted to black-and-white images. What is required is a format for efficient storage, retrieval, and transmission of high-quality document images in color. A standard color image compression algorithm produces very 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) would be around 100 KB to 200 KB, and would be barely readable. The same page at 300dpi would be of acceptable quality, but would occupy around 500 KB.

Pages must appear on the screen after only a few seconds delay. Assuming a 56 kilobits per second (kbps) connection, this means that the most relevant parts of the document (the text) must be compressed down to about 20 to 30 KB. With a progressive compression technique, the text would be transmitted and displayed first. Then the pictures, drawings, and backgrounds would be transmitted and displayed, improving the quality of the image 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.

Another peculiarity of document images, their large size, makes current image compression techniques inappropriate. A magazine-size page at 300dpi is 3300 pixels high and 2500 pixels wide. Uncompressed, it occupies 25 MB of memory; more than what the average PC can properly handle. A practical document image viewer would need to 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] responds to all the problems mentioned above. 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 (44 KB according to the latest statistics). DjVu pages are displayed within the browser window through a plug-in, which allows easy panning and zooming of very large images. This is made possible by an on-the-fly decompression method that allows images that would normally require 25 MB of RAM once decompressed to require only 2 MB of RAM.

This paper gives a brief overview of the DjVu technique, which is described in more details in [2]. Sections 2. and 3. show how DjVu makes a highly efficient use of the foreground/background representation of the image, especially for Internet applications that require extremely high compression ratio and progressive decoding. This paper also presents new contributions to DjVu: Section 4. describes a new MDL based post-segmentation filter and Section 6. shows how the combination of DjVu and OCR allows for complex indexing tasks.

2. The DjVu Compression Method

The basic idea behind DjVu is to separate the text from the backgrounds 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 (CCITT G3, G4, and JBIG1). The DjVu technique improves on both and combines the best of both approaches. A foreground/background separation algorithm generates and encodes three images separately 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), 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 upsampled background image. If the mask pixel is 1, the pixel color is chosen as the color of the connected component (or taken from the foreground image). The foreground/background representation is also a key element of the MRC/T.44 standard[7].

The mask image can be encoded with a bi-level image compression algorithm. The bi-level image compression algorithm used by DjVu to encode the mask is dubbed JB2. It is a variation on AT&T's proposal to the upcoming JBIG2 fax standard. The key to the compression method is a method for making use of the information in previously encountered characters (marks) without risking the introduction of character substitution errors that is inherent in the use of OCR [1]. The marks are clustered hierarchically. Some marks are compressed and coded directly using arithmetic coding (this is similar to the JBIG1 standard). Others marks are compressed and coded indirectly based on previously coded marks, also using a statistical model and arithmetic coding. The previously coded mark used to help in coding a given mark may have been coded directly or indirectly. There are many ways to achieve the clustering and the conditional encoding of marks, the algorithm that we currently use is called "soft pattern matching" [5].

The background image can be encoded with a method suitable for continuous-tone images. DjVu uses a progressive, wavelet-based compression algorithm called IW44 for this purpose. In addition, a new 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.

3. The DjVu Browser

The digital library user experience depends critically on the performance of the browsing tools. Much more time is spent viewing documents than formulating queries. As a consequence, 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 MBytes of memory. Decompressing such images before displaying them would exceed the memory limits of average desktop computers.

We developed a plug-in for Netscape Navigator and Internet Explorer. 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 Separation

The first phase of the foreground/background separation is described in [2]. This hierarchical color clustering algorithm attempts to retain as much information as possible about the original image while quantizing the colors on two levels only: background for the light colors and foreground for the dark colors. This is not exactly our objective, since the foreground is the part of the image where a high resolution is necessary for visual understanding. For example, the separation algorithm 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.

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 [8, 6], 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 [6], 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 assymetrical* 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. This model ignores segmentation: in regions which are masked by the foreground, the background color must be interpolated in a way to minimize the background reconstruction error. The foreground model assumes that the color of a blob is uniform. For both models, the noise around the estimated pixel values can be either Gaussian or Laplacian. To use the same type of noise with identical covariances is preferable to allow small perturbations and dithering effects to cancel out when comparing the models. What remain to be coded are the boundaries of the mask: the model we use tends to favor horizontal and vertical boundaries.

In summary, 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. Occasionally, the text segmented by the separation algorithm will appear inverted in the foreground image (as holes of a large connected component). Another filter detects those occurrences and corrects them.

JPEG at 100dpi. DjVu.

Figure 1. *Comparison of JPEG at 100dpi (left) with quality factor 30% and DjVu (right). The file sizes for the complete pages are 82 KB for JPEG and 67 KB for DjVu*

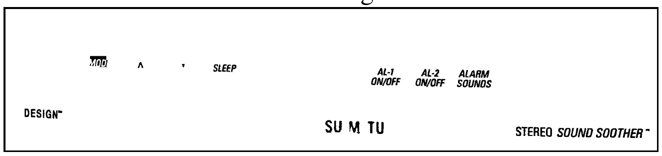
5. Compression Results

Figure 1 shows a comparison between DjVu and JPEG applied to 100dpi images on a segment of an image. Note that the rendering of the photographs is also better with DjVu than with JPEG, even though DjVu uses the same 100dpi for the background image.

More results are available in our experimental digital library which is available online at `http://www.djvu.att.com`. This page shows the performance of the algorithm on a large variety of document types (it is important to stress that no hand-correction was applied to any of these images). From the feedback of users, we are in the process of gathering typical segmentation errors that result in artifacts. For instance, in a photograph of a person, the eyes or the eyebrows may be classified as foreground. It is fair to say that, on the data we have collected so far, these segmentation errors are infrequent.

To test bi-level document images, we downloaded a standard benchmark for bi-level document images, the OCR Lab 10 sample images (available at `http://imagebiz.com/PaperWeb/ocrlab.htm`). This benchmark has been designed to test various methods to digitize bi-level images, mostly with the help of OCR. On bi-level images, DjVu has only one layer: the mask coded with JB2. DjVu requires 268 KBytes to code the 10 images: this is one fourth of what G4 requires (1056

Color Image



Bilevel Mask

`DESIGN SLEEP AL-I AL-i ALARM ON/OFF ON/OFF SOUNDS SU TV STEREO SOUND SOOTHER`

OCR output

Figure 2. *Foreground separation followed by OCR*

KBytes) and half of what PDF with Acrobat Capture 2.0 requires (498 KBytes).

In summary, on bi-level and color images, the DjVu compression rate is 4 to 10 times higher than traditional techniques such as CCITT-G4 or JPEG, at the same level of readability. Typical magazine or catalog pages in color at 300dpi compressed with DjVu occupy 40 to 80 KB. Black and white pages such as technical papers, are between to 15 and 40 KB. Ancient books, where most of the color is on the background, occupy 30 to 60 KB.

6. Indexing DjVu Documents

This section shows that it is possible to apply commercial OCR solutions to the compressed mask with satisfactory retrieval performances, and discusses how to further improve this OCR. Good OCR performance relies on the facts that the foreground/background algorithm properly extracts the text, without omissions, and that lossy JB2 transforms characters in ways that do not affect their readability. In fact, testing the quality of the OCR is another way to check that DjVu achieves its goals.

Figure 2 shows how the foreground/background algorithm is able to segment characters on the photograph of the product itself. This enables to retrieve the document by using the product name “STEREO SOUND SOOTHER”. Recognition experiments were also carried on the University of Washington database. 125 pages of scientific journals were scanned and thresholded to bilevel images. Using a widely available commercial software package, the word substitution rate is around 1% (mostly words that are not found in the standard English dictionary). Lossy JB2 compression increased this error rate to 1.5%, which is acceptable.

7. Conclusion

DjVu, a new compression technique for color document images is described. It fills the gap between the world of paper and the world of bits by allowing scanned document to be easily published on the Internet. With the same level of legibility (300 dots per inch), DjVu achieves compression ratios 5 to 10 times higher than JPEG. DjVu can also be considered as an enabling technology for many document analysis techniques. To achieve optimal compression, it justifies the development of complex *text/image separation* algorithms. The addition of *text layout analysis* and *optical character recognition* (OCR) will make it possible to index and edit text extracted from DjVu-encoded documents. The DjVu compression software is available free for research, evaluation and non-commercial use on various UNIX platforms at `http://www.djvu.att.com`. The DjVu reference library is available in source form at the same URL. The DjVu plug-in is available for Linux, Windows 95, NT, Mac, and various UNIX platforms. The above web site also contains a digital library with over 800 pages of scanned documents from various origins.

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

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

[8] J. Rissanen. Stochastic complexity and modeling. *Annals of Statistics*, 14:1080–1100, 1986.

[9] G. Story, L. O'Gorman, D. Fox, L. Shaper, and H. Jagadish. The RightPages image-based electronic library for alerting and browsing. *IEEE Computer*, 25(9):17–26, 1992.

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