Electronic Document Publishing Using DjVu

Artem Mikheev[1], Luc Vincent[1], Mike Hawrylycz[1], and Léon Bottou[2]

[1] Lizardtech Software, 1008 Western Avenue, Seattle WA 98105, USA
lvincent@lizardtech.com
http://www.lizardtech.com

[2] NEC Research Institute, 4 Independence Way, Princeton, NJ 08540–6634, USA
leon@bottou.org

Abstract. Online access to complex compound documents with client side search and browsing capability is one of the key requirements of effective content management. "DjVu" (Déjà Vu) is a highly efficient document image compression methodology, a file format, and a delivery platform that, when considered together, has shown to effectively address these issues [1]. Originally developed for scanned color documents, the DjVu technology was recently expanded to electronic documents. The small file sizes and very efficient document browsing make DjVu a compelling alternative to such document interchange formats as PostScript or PDF. In addition, DjVu offers a uniform viewing experience for electronic or scanned original documents, on any platform, over any connection speed, which is ideal for digital libraries and electronic publishing. This paper describes the basics of DjVu encoding, with emphasis on the particular challenges posed by electronic sources. The DjVu Virtual Printer Driver we implemented as "Universal DjVu Converter" is then introduced. Basic performance statistics are given, and enterprise workflow applications of this technology are highlighted.

1 Scanned Document Publishing and DjVu

Document image processing for the Internet, including storage, presentation, and transmission systems for complex compound documents is one of the fundamental challenges of current content management systems. Enterprise applications, including corporate Intranet usage and internal workflow management systems require rapid transmission and feature-rich viewing that enable users to quickly access and browse important documents. The advent of wireless communications only further stresses the technology required to address these problems as transmission costs remain significant and the processing capacity of portable client devices is limited.

In parallel, digital libraries are becoming a more accepted form of information archiving, search, retrieval, and taxonomy. However, online document libraries will succeed only to the extent that they prove adequate and effective for common and mass usage. A key to surmounting these technological hurdles is understanding the workflow of the document storage, transmission, and viewing and the adoption of standards and applications that perform naturally at each stage of the process.

The last several years have seen a growing demand for technology that could handle *color* documents, whether scanned or electronic, in an 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 open the door to such applications.

Compression technology for bitonal (black and white) document images has a long history [2]. It is the basis of a large and fast growing industry with widely accepted standards (Group 3, MMR/Group 4), as well as still emerging standards (JBIG1, JBIG2). Standard color image compression algorithms are grossly inadequate for the type of applications mentioned earlier because associated file sizes are still excessively large 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 is of acceptable quality, but occupies at least 500KB to 1 MB. These sizes are impractical for online document browsing, even with broadband connections.

A notable observation about document encoding is that preserving the readability of the text and the sharpness of line art requires high resolution (typically 300–600dpi) and the preservation of sharp edges. On the other hand, preserving the appearance of continuous-tone images and background paper textures does not require as high a resolution (typically, 100dpi is sufficient). 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. This multi-layer raster representation is a key element in various emerging document interchange products and standards such as MRC, TIFF-FX, JPEG2000 (part 6), and of course DjVu! The separation method brings another considerable advantage to DjVu: 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. Along with the DjVu “Hidden Text Layer”, this enables immediate access for text search and retrieval applications.

The general requirements for an acceptable user experience may be described as follows: The text and line diagrams should appear on the display after only a few seconds delay. This means that the text layer must fit within 20–40KB, under the assumption of a 56KB/sec modem connection. The pictures and other background items would subsequently appear, improving the image quality as more bits arrive. The overall size of the file should be in the order of 50–100KB to bound the overall transmission time and storage requirements.

Thanks to its outstanding compression capabilities, excellent image quality, simple and effective file format, and highly efficient viewing and browsing, DjVu has proven to provide a content delivery platform that can successfully meet all the challenges mentioned above [1,3,4,5,6]. With DjVu, pages scanned at 300dpi in full color can be compressed down to 30–80KB files from 25MB originals with excellent quality. This puts the size of high-quality scanned pages in the same order of magnitude as a typical HTML page. Like HTML pages, DjVu pages are displayed progressively within a web browser window, via a custom plugin. Panning, zooming, and navigation is highly efficient, even though pages are internally kept in a partially decoded representation, which requires only about 2MB of cache. In short, the DjVu technology has all the features required to provide an excellent end-user experience when browsing collections of complex color documents.

2 DjVu Encoding Basics

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 combines the best of both approaches. A foreground/background separation algorithm segments images into separately compressed layers:

– The **background layer** contains a low resolution (typically 100 dpi.) background image representing details that are best encoded using a continuous tone technique. This usually includes the document background and the pictures. For the background and foreground images, DjVu uses a progressive, wavelet-based compression algorithm called *IW44* or *DjVuPhoto*. IW44 offers many key advantages over existing continuous-tone image compression methods that are particularly useful for efficient panning and zooming of large images.
– The **foreground layer** contains a high resolution bitonal *mask image* (typically 300 to 600 dpi) that accurately defines the shape of details with sharp edges such as text and line-art. The optional color information is either encoded as a solid color per connected component in the mask, or as a very low resolution *foreground image* (typically 25 dpi) whose colors are applied using the mask as a stencil. The bitonal mask image is encoded with a new bi-level image compression algorithm dubbed *JB2* or *DjVuBitonal*. 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 to use a shape clustering algorithm to find similarities between shapes [7,8].

The DjVu encoder was designed to be as generic as possible. The only information it requires about the document is its scanning resolution. This puts an extraordinary constraint on the segmentation algorithms used to obtain the foreground/background separation. Most of the details are beyond the scope of this aritcle and the reader is referred to [9].

3 DjVu for Electronic Documents

Scanned documents and documents of electronic source origin present distinct challenges for encoding. While DjVu was initially designed for scanned documents, it soon became apparent that it could also significantly improve the compression rate and speed of rendering of electronic documents such as PostScript, PDF or MSWord. These document description languages are generally slow to render, may produce very large files and are often platform dependent (mostly because of the fonts and character sets they rely on).

Converting such documents into DjVu can be done by first rendering them into a high-resolution color raster image and then converting it to DjVu. However, relying on a purely pixel-based intermediate representation is less than ideal in that it disregards essential information about the document, such as what is text and what is not. This information could be used to significantly improve conversion quality, which, for electronic documents, is essential. In addition, intermediate pixel-based representations are expensive to compute and require large amounts of RAM. One would therefore like the conversion process to work directly on the existing *structured page information* [10], that is, high level objects such as text, fonts, colors, embedded images, etc., instead of pixels.

Structured page information for electronic documents is used in a large variety of file formats such as the MSWord `.doc` files, PDF files, or PostScript files. Printing such files converts the structured information into a list of drawing operations such as “fill a rectangle”, “draw a line”, “draw an image” or “draw a piece of text”. This can be interpreted as an ordered list of predefined graphic objects, which can—and often do—overlap. The challenge is to efficiently turn this ordered list into only two layers, for subsequent DJVu compression.

A novel and general framework for unifying the concepts underlying scanned and electronic DjVu segmentation was recently presented in [9]. The authors show how a concept known as the Minimum Description Length (MDL) principle [11] can be used as a type of universal segmentation tool to determine which objects are defined in the foreground and background for both scanned and electronic documents. In essence the MDL principle works by minimizing the overall coding cost. This coding cost is the sum of the number of bits necessary to encode the image (*the encoding bit cost*) and the number of bits necessary to encode the discrepancy between the encoded image and the original image (*the discrepancy bit cost*). This general approach can be successfully used on raster data (scanned documents) as well as electronic documents, albeit using substantially different algorithms.

Though DjVu converted file sizes vary depending on the type of electronic document converted to DjVu, file sizes generally compare favorably. As a rule of thumb, one can expect a file size reduction of a factor 2 to 50. This is somewhat remarkable considering that the originals were electronic. The resulting DjVu files are a completely portable version of their original electronic counterparts: they do not depend on any system-specific objects such as fonts; like all DjVu files, they are very efficient to transmit, display, store, and they can be viewed on any platform. In addition, they contain a *hidden text layer* that is a faithful representation of the text in the original electronic document, and is key for search and retrieval. This makes DjVu an excellent document interchange format.

Our electronic to DjVu document conversion framework and algorithms are referred to as *DjVuDigital*. Current implementations include a Ghostscript driver [12], which enables the creation of DjVu from PostScript and PDF, and a Windows Virtual Printer Driver, which we now describe.

4 The DjVu Virtual Printer Driver

An important class of a file format conversion applications are called Virtual Printer Drivers (VPD). These are applications that configure as if they were standard system or network printers, but serve the role of file format interchange devices. There are many brands and variations of VPDs currently on the market that allow conversion between most of the commonly used graphic file formats. Adobe PDFWriter^{TM} is essentially a VPD converting Postscript or MSWord documents into PDF. Below we describe the steps and workflow of the Lizardtech Software DjVu VPD. The details are given with respect to an implementation on the Microsoft Windows architecture, but the principles are the same for most operating systems.

4.1 Representation of Printed Output in Metafile Form

Most popular Windows applications have printing functionality: the user is able to specify the specific printer device and output the document to that device. The first step in converting an electronic document to DjVu is the representation of the structured page information as an Enhanced Metafile. Enhanced Metafiles are used in the DjVu VPD to store a picture created by using the Win32 GDI functions.

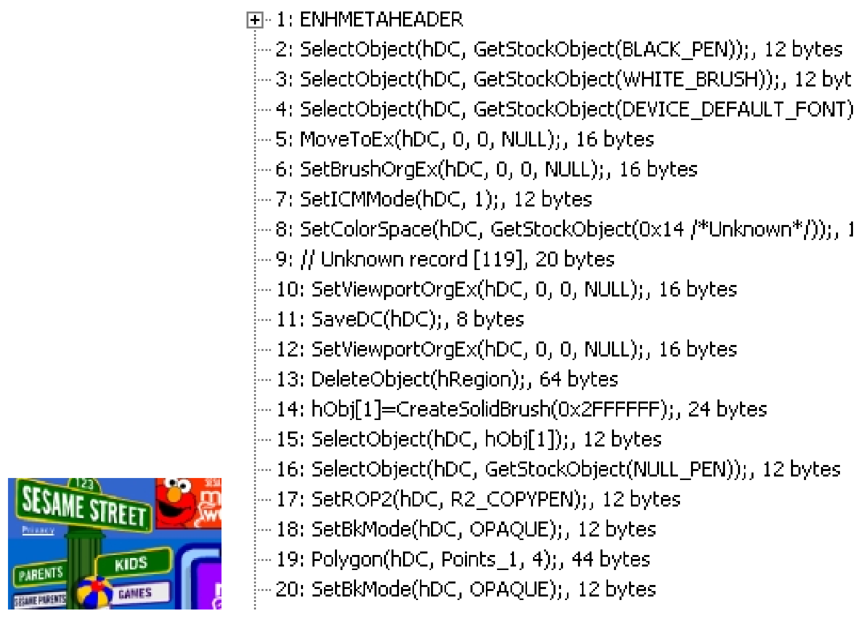
The Windows Enhanced Metafile format (EMF) is used to represent both scalable vector and bitmap graphics under Windows. This format is widely used in Windows applications for clipboard, client-server data exchange and within the Windows spooler. An EMF file consists of the following elements: a header, a table of handles to GDI objects, a private palette, and an array of metafile records. EMF facilitates a device independence description of objects. It gives a uniform object representation whether it appears on a printer, a plotter, the desktop, or in the client area of any Win32-based application. In this respect, one can think of EMFs as the Windows equivalent of Adobe PostScript files. An example is shown in Fig. 1.

Although there are many ways for the application to perform printing to a document, the two main methods are:

1. Print by rasterizing consecutive vertical bands of the document. Then send the representation of those bands in the bitmap form to the printer.
2. Print by drawing separate drawing elements on the printer device context.

The second method leads to much faster printing in most cases and is used by most modern applications. One consequence of this method is that the corresponding generated EMF contains a sequence of overlapping drawing elements that need to be rendered. While, at first, this may seem like a drawback, it is quite advantageous to the algorithm used by the VPD.

The size of the metafile generated depends greatly on the content of the source electronic document. For example, let us consider printing the front page of the Web site CNN (`http://www.cnn.com`) using Internet Explorer 5.5. Depending on daily variations on the site, the typical Metafile it produces is 2.5MB in size, which is somewhat typical for letter-size documents printed at 300dpi. This metafile contains about 60,000 different records, however only 8,000 of them represent drawing elements. Rarely, documents can produce metafiles of extremely large size. In one instance, we came across a 1-page document which generated a metafile of close to 100MB! This particular metafile contained 5 Million records, about 450,000 of which corresponding to the actual drawing elements.



```
1: ENHMETAHEADER
2: SelectObject(hDC, GetStockObject(BLACK_PEN));, 12 bytes
3: SelectObject(hDC, GetStockObject(WHITE_BRUSH));, 12 byt
4: SelectObject(hDC, GetStockObject(DEVICE_DEFAULT_FONT)
5: MoveToEx(hDC, 0, 0, NULL);, 16 bytes
6: SetBrushOrgEx(hDC, 0, 0, NULL);, 16 bytes
7: SetICMMode(hDC, 1);, 12 bytes
8: SetColorSpace(hDC, GetStockObject(0x14 /*Unknown*/));, 1
9: // Unknown record [119], 20 bytes
10: SetViewportOrgEx(hDC, 0, 0, NULL);, 16 bytes
11: SaveDC(hDC);, 8 bytes
12: SetViewportOrgEx(hDC, 0, 0, NULL);, 16 bytes
13: DeleteObject(hRegion);, 64 bytes
14: hObj[1]=CreateSolidBrush(0x2FFFFFF);, 24 bytes
15: SelectObject(hDC, hObj[1]);, 12 bytes
16: SelectObject(hDC, GetStockObject(NULL_PEN));, 12 bytes
17: SetROP2(hDC, R2_COPYPEN);, 12 bytes
18: SetBkMode(hDC, OPAQUE);, 12 bytes
19: Polygon(hDC, Points_1, 4);, 44 bytes
20: SetBkMode(hDC, OPAQUE);, 12 bytes
```

Fig. 1. Realization of an Enhanced Metafile (EMF) and representative EMF commands.

DjVu documents being raster-based, they contain a certain resolution value, which needs to be specified before conversion. Currently acceptable values are in the range 50 to 4,800 dpi, with 400 dpi being the default. After conversion, the DjVu document will have physical dimensions defined by the printing application. These dimensions are easily computed from the user-specified resolution and the dimensions stored in the metafile.

4.2 Parsing the Metafile and Record Classification

The enhanced metafile is a sequence of records representing drawing operations applied to a specific Windows Device Context. These records are accessed using standard Win32 API calls. Some EMF records correspond to mode setting operations: they do not modify actual graphic content, but are important in that they determine how subsequent record should be rendered. Records that actually modify pixels of the Metafile Device Context are parsed, rendered, and then a decision is made whether they belong to the foreground or background DjVu layer.

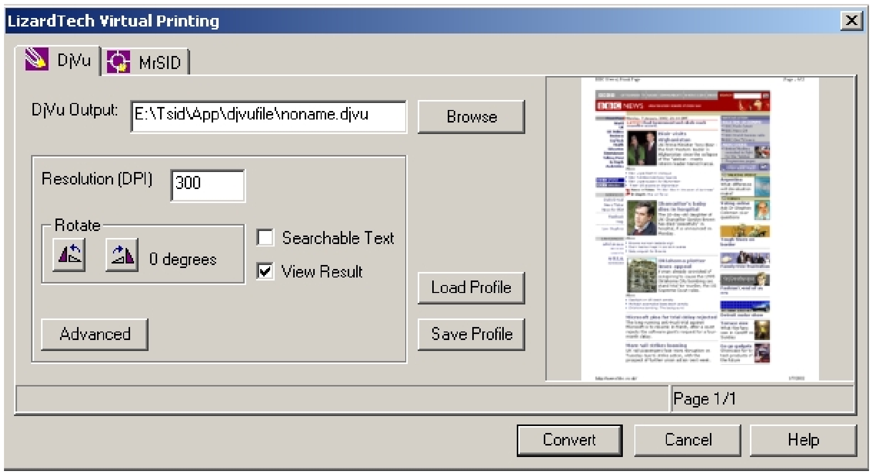
This is complicated by the fact that, in order for overlaps to be accurately processed, EMF records need to be considered in reverse drawing order, that is, starting from the EMF record corresponding to the *last* element that should be drawn (topmost drawing element). For more information on this process, see [10]. For each page, two segmented layers are produced, which can then be sent to the back-end DjVu compressor. This back-end is essentially identical as the one used for scanned documents. In particular, it should be noted that it takes advantage of redundancy across pages in the JB2 compression, and also avoids spending any bits on IW44 compressed background regions that will be covered by foreground pixels.

Text capture is performed concurrently, without the need for any OCR. Indeed, EMF files contain bounding box and text character information. This information is parsed as essentially another object of the structured page information, and enables the DjVu "hidden text layer" to be easily created.

4.3 Overview of Virtual Printer Driver Workflow

The DjVu Virtual Printer Driver we implemented seamlessly installs like any system printer. Once installed, the process of conversion to DjVu starts after the user has selected a "Print" command from the host application's print menu. When the visible local or network printers are displayed, the user selects "DjVu VPD" from the drop down list. This causes the Windows application to generate an EMF file and then brings up the interface shown in Fig. 2. This interface includes a preview pane, which makes it easy to correct any application-specific settings required for producing the desired output.

At this point, the user has to specify the main conversion parameters: resolution, whether or not to include the hidden text layer, and final orientation of the document. The "Advanced Parameters" dialog box, shown in Fig. 3, allows the user to fine tune conversion for the specific document variation. In particular, control is given for the quality of the background encoding, the relative weight of importance of foreground versus background, and the limitations on the color


LizardTech Virtual Printing

DjVu | MrSID

DjVu Output: E:\Tsid\App\djvufile\noname.djvu

Resolution (DPI): 300

Rotate: 0 degrees

[ ] Searchable Text

[✓] View Result

Page 1/1

Fig. 2. Main Screen of DjVu Virtual Printer Driver

palate to be used. In 99% of cases, these advanced options do not need to be changed from their defaults.

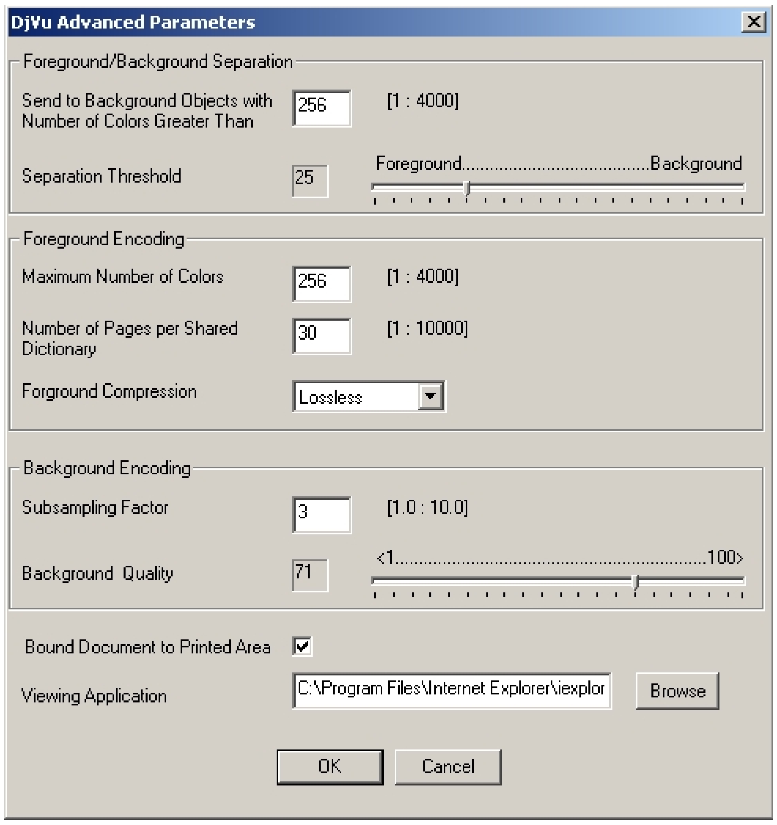
5 Results and Conclusions

Extensive tests were conducted during the development of the VPD, analyzing both image quality upon conversion and printer performance. We found that the VPD performed well on a broad sample of document types, with specialized document types such as high resolution maps, or images with very large numbers of objects sometimes requiring tuning of the parameter settings for optimal results. The bulk of the tests were on desktop workstation running Pentium-4/1.4GHz with 256MB RAM. Documents of many several types were used for these tests, Fig. 4 showing some representative document types.

For each document class the original and converted sizes were recorded as well as the time to convert. The results shown in Table 1 were found to be typical expected performance.

Our results indicate that the DjVu VPD successfully converts almost all Windows type documents. Long PowerPoint documents including complex backgrounds sometimes take a long time to convert, owing to the inefficient way shaded backgrounds are represented in EMF records. This type of driver could have enormous benefit to the digital libraries and in enterprise applications where archival and business critical documents need to be published in a readily accessible manner.

The ability to capture searchable text is an extremely attractive feature in the above applications. As the text layer can be carried along with the encoded document, DjVu is easily integrated with document search and content management systems.



DjVu Advanced Parameters

Foreground/Background Separation

Send to Background Objects with Number of Colors Greater Than: 256 [1 : 4000]

Separation Threshold: 25 Foreground.......................................Background

Foreground Encoding

Maximum Number of Colors: 256 [1 : 4000]

Number of Pages per Shared Dictionary: 30 [1 : 10000]

Forground Compression: Lossless

Background Encoding

Subsampling Factor: 3 [1.0 : 10.0]

Background Quality: 71 <1...100>

Bound Document to Printed Area: ✓

Viewing Application: C:\Program Files\Internet Explorer\iexplor

Fig. 3. Advanced Parameter Screen of Virtual Printer Driver

Table 1. Comparative size and conversion times using the VPD for several common document types, shown in Fig. 4. Performance data was obtained on a Pentium 4, 1.4GHz.

Document Name	Format	Number of Pages	Original Size	**DjVu** Size	Time to Convert
NY Times	PDF	92	25.9 MB	8.57 MB	12 m 32 s
HBS Review	PDF	132	8.75 MB	4.63 MB	11 m 5 s
Vector Map	PDF	1	1.67 MB	117 KB	19 s
Slides	PowerPoint	23	1.7 MB	1.05 MB	5 m 32 s
BBC Web page	HTML	3	174 KB	116 KB	14 s

A remarkable property of DjVu is the extreme file size reduction possible. This makes DjVu a natural choice in wireless and very constrained bandwidth situations. The DjVu VPD and its underlying technology could play an important role in the large scale preparation of documents for this use. With the recent

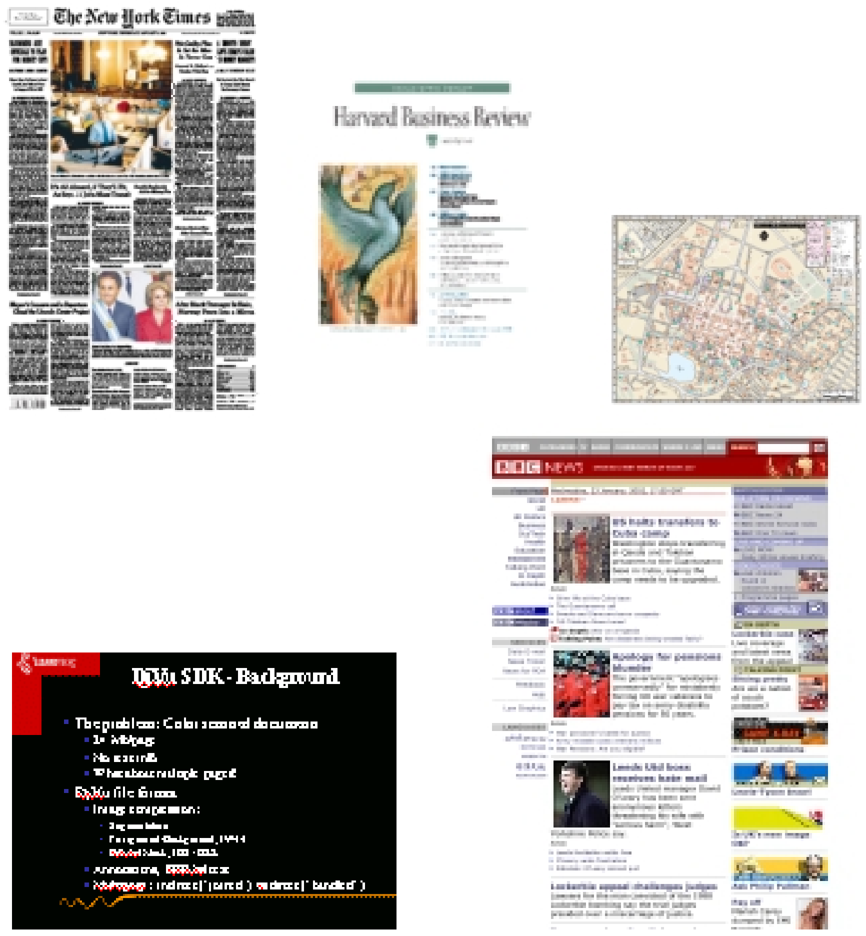


Fig. 4. Sample documents printed using the VPD

addition of electronic document conversion to its portfolio, DjVu becomes a serious, powerful document interchange format, able to address the needs of a wide range of users.

A full range of DjVu authoring products, from non-commercial software packages to full-feature, advanced packages designed for enterprise users, are available from Lizardtech Software, `http://www.lizardtech.com`. The technology is also partly available as open source at `http://djvu.sourceforge.net.` In addition, a growing number of third-party tools, from imaging packages such as *IrfanView* (`http://www.irfanview.com`)to searching and indexing engines such as the *Realview DjVu Index Filter* (`http://www.realview.com.au`),are supporting DjVu. The web site `http://www.djvuzone.org` is dedicated to the DjVu community, with news, documentations and many links to Digital libraries which use DjVu.

References

1. Bottou, L., Haffner, P., Howard, P., Simard, P., Bengio, Y., LeCun, Y.: High quality document image compression with DjVu. Journal of Electronic Imaging **7** (1998) 410–428
2. Witten, I.H., Moffat, A., Bell, T.C.: Managing Gigabytes: Compressing and Indexing Documents and Images. Van Nostrand Reinhold, New York (1994)
3. LeCun, Y., Bottou, L., , Haffner, P., Howard, P.: DjVu: a compression method for distributing scanned documents in color over the internet. In: Proceedings of Color 6, IST. (1998)
4. Bottou, L., Haffner, P., Howard, P., Simard, P., Bengio, Y., LeCun, Y.: Browsing through high quality document images with DjVu. In: Proceedings of IEEE Conference on Advanced in Digital Libraries. (1998)
5. Haffner, P., LeCun, Y., Bottou, L., Howard, P., Vincent, P.: Color documents on the web with DjVu. In: Proceedings of IEEE International Conference on Image Processing, Kobe, Japan (1999) 239–243
6. LeCun, Y., Bottou, L., Haffner, P., Triggs, J., Riemers, B., Vincent, L.: Overview of the djvu document compression technology. In: SDIUT'01, Symposium on Document Image Understanding Technologies, Columbia, MA, University of Maryland (2001) 119–122
7. Ascher, R.N., Nagy, G.: Means for achieving a high degree of compaction on scan-digitized printed text. IEEE Trans. Comput. **C-23** (1974) 1174–1179
8. Howard, P.G.: Text image compression using soft pattern matching. Computer Journal **40(2/3)** (1997) 146–156
9. Haffner, P., Bottou, L., LeCun, Y., Vincent, L.: A general segmentation scheme for DjVu document compression. In Talbot, H., Berman, M., eds.: ISMM'02, International Symposium on Mathematical Morphology, Sydney, Australia, CSIRO Publications (2002)
10. Bottou, L., Haffner, P., LeCun, Y.: Conversion of digital documents to multilayer raster formats. In: ICDAR'2001, International Conference on Document Analysis and Recognition, Seattle, WA (2001)
11. Rissanen, J.: Stochastic complexity and modeling. Annals of Statistics **14** (1986) 1080–1100
12. GhostScript: Home page at http://www.ghostscript.com (2002)