Sergey Lesenko and Laurent Siebenmann

Viewing DVI files with Acrobat Reader: DVIPDF gives birth to AcroDVI

Sergey Lesenko
Institute for High Energy Physics (IHEP)
Protvino (Moscow Region)
142284 Russia
lesenko@mx.ihep.su

Laurent Siebenmann
Mathématique, Bât. 425
Université de Paris-Sud
91405-Orsay, France
Laurent.Siebenmann@math.u-psud.fr

Abstract

The first author's DVIPDF program converts from DVI, the output format of TEX, to PDF, the input format for Adobe's Acrobat Reader. Although DVIPDF has existed as a prototype for about three years, the uses to which it will be put by the TEX community are only gradually emerging. This article presents one concrete application. DVIPDF has been adapted under the Windows 9X/NT operating systems to allow "drag-and-drop" viewing of DVI files in Acrobat Reader. The resulting viewer is called AcroDVI: it involves DVIPDF and the Acrobat Reader, operating in concert. Intended for viewing legacy DVI files, it aims to support the most common `\special` commands. An evolutive change in electronic publishing practice is proposed in this connection: the conventional EPS graphics format could well be replaced by various optimal formats: JPEG or PNG for bitmaps, and PDF for vectorial graphics. These can then be conveniently exploited in some natural ways hitherto unavailable: shared, re-edited, or directly viewed.

Introductory viewing experience

> *... Egli e' scritto in lingua matematica, e i caratteri son triangoli, cerchi, ed altre figure geometriche ...*
>
> — Galileo, writing on physical science

To view an article, the modern scientist using AcroDVI can simply push its DVI file icon onto the icon of AcroDVI. The DVI file is quickly converted to PDF; then a window pops up for viewing by Acrobat Reader. If you are not already familiar with Acrobat Reader, the biggest thrill will surely be the top-quality graphics and typography, both superior in various respects to what web browsers offer. Worth noting for TEX users are the hypertext features familiar from web browsers. All this is remarkable, but only the notion that DVI can be the root format is new.

In the AcroDVI viewing experience, even those familiar with Acrobat Reader will enjoy one novelty: enhanced visibility of the graphics objects. They appear not only in the PDF page view but also autonomously in native formats suitable for reuse and also for display at an optimal scale. The three formats that AcroDVI deals with directly are PNG (Portable Network Graphics) for bitmapped high contrast graphics, JPEG (Joint Photographic Experts Group) for color photos, and PDF (Portable Document Format) for vectorial graphics (more on these later). One of these formats should be optimal for just about any still (that is, non-moving) graphics object.

If you push the icon of a PNG or JPEG or PDF file onto that of AcroDVI, then it will be immediately viewed in an Acrobat Reader window. Likewise for EPS files, provided Acrobat Distiller or *Ghostscript* is accessible and enough fonts are available. Latent in Acrobat Reader, which does not directly process PNG or JPEG files, are broad graphics viewing capabilities, and DVIPDF has

merely tapped into them; Adobe could have done as much for Acrobat Reader, but chose not to.

There are many specialized tools for both viewing *and* editing PNG and JPEG files, notably the free XNview under Windows and Linux and the shareware program Graphics Converter on the Macintosh. If you take care to view at scale 100%, then you will see the bitmapped graphics at their best possible quality.

The most widely used tools for viewing PNG and JPEG graphics are probably the web browsers. This is an open invitation to make double use of the graphics in an article: first, in an illustrated HTML introduction, and second, in the DVI file for the article's body. Thus, AcroDVI provides polyvalence for graphics. At the same time, it provides a basic polyvalence for text, namely, the possibility to view the same DVI file with Acrobat Reader and with traditional DVI viewers.

The need for polyvalence and low bulk was the immediate motivation for developing AcroDVI. It arose for mathematics journal content in the CD-ROM project called MathCD, for which the second author is managing editor. Indeed, MathCD has an order of magnitude less space available for many journals than a single journal can afford to use on the Internet.

Where space is at a premium, as on some CD-ROMs and in personal electronic libraries, the DVI format plus auxiliary native graphics can now reasonably replace the PDF format. On the other hand, where space is virtually unlimited, as on many Internet sites, expect to see more formats and greater bulk.

There are relatively few hyper-references in the electronic journal articles on MathCD. Currently, DVIPDF does support hyper-references using a `\special` syntax, parallel to Acrobat Distiller's `pdfmark` syntax. However, it does not yet support the most common `\special` syntax of today's DVI files, namely the one introduced by *xhdvi* and paralleling HTML.

What is AcroDVI really?

The technologically aware user will tend to see AcroDVI as the sum of its parts: DVIPDF plus Acrobat Reader plus some Windows programming using the Dynamic Data Exchange (DDE) protocol as the framework for collaboration between DVIPDF and Acrobat Reader.

However, to the passing user, the whole will be more important than the parts. That is, AcroDVI acts as a viewer that directly accepts most DVI files (*modulo* font availability), as well as graphics files in the PNG, JPEG, or PDF formats — it is the first viewer to do all this.

We have decided to dignify the whole with the the acronym AcroDVI. A viewing eye is what you should see in the logo:

AcroDVI

that is put together by a TEX macro `\AcroDVI` from pieces of standard TEX fonts.

The Windows icon is a colored iris (an "eye-con"); files to be viewed are dragged and dropped on top of the icon. (See Fig. 1 for black-and-white renditions of the current icon.)

In its present provisional state, AcroDVI involves a single binary executable called `dvipdf.exe` while the shortcut icon to it and the distribution directory are called AcroDVI. This makes DVIPDF and AcroDVI rather like a marsupial 'mother-with-baby-in-pouch'.

To facilitate portability, source code is divided into modules of C++ source code devoted exclusively to the AcroDVI viewer functions and modules of C code that can hopefully be compiled as a "black box" processor to implement DVIPDF as a stand-alone DVI-to-PDF converter. Incidentally, most of the `\special` features developed recently for viewing legacy DVI files (see below) have become permanent additions to the "black box" part of the DVIPDF program.

What shape will maturity bring to AcroDVI? Two options currently hold our attention.

In Lesenko (1997), it was proposed to build a DVIPDF plug-in for Acrobat Reader. With this approach, to view a given DVI file in Acrobat Reader, one would push its icon onto the Acrobat Reader icon rather than onto the DVIPDF icon. The plug-in architecture promises to promote portability of AcroDVI.

A second reasonable option would make AcroDVI an autonomous Windows application distinct from DVIPDF. This architecture promises to facilitate orchestration by AcroDVI of *multilateral* collaborations among DVIPDF, Netscape, Zip, Acrobat Reader, *Ghostscript*, and so on.

As soon as *Ghostscript/Ghostview* under Windows provide support for the key functions "Open Doc" and "Close Doc" of DDE, we will make available a new AcroDVI configuration that replaces Acrobat Reader by *Ghostscript/Ghostview*. It will probably be less luxurious than with Reader, but it will, in addition, accept EPS files.

Fonts

DVI files do not contain fonts — that is one basic reason why they are so compact. The question then arises: where are fonts for AcroDVI to come from? The best one can hope is that, in practice, enough Type 1 fonts will be in AcroDVI's expansible repertoire, which is based chiefly on B.K. Malyshev's BaKoMa Type 1 font collection, covering essentially all fonts commonly used in freely distributed electronic science publications.

Adobe's Type 1 is currently the only font format supported by DVIPDF; TrueType fonts are not accepted. Nor are Adobe Type 3 fonts allowed, bitmapped or not; Acrobat Reader would in any case handle them poorly.

The Adobe Type Manager, which first made screen viewing with scalable (vectorized) fonts a significant reality is *not* needed by AcroDVI since the relevant functions have been absorbed into Acrobat Reader.

On MathCD, there are just a few DVI files that call for commercial Adobe Type 1 fonts. DVIPDF will not currently handle these unless you have them installed in Type 1 format. Since many of these have acceptable TrueType versions preinstalled by Windows, more support for TrueType would be desirable.

Until then, we recommend Malyshev's own *DView* for such fonts. It offers essentially universal font support — although different graphics support.

Installing AcroDVI

AcroDVI (including DVIPDF) currently runs under all recent versions of Microsoft Windows (not under version 3.x). It is freely available on the Internet (see Resources).

Currently, both AcroDVI and DVIPDF are presented as a directory of approximately 1.5 megaoctets (Mo), not including the BaKoMa font collection, which is another few megaoctets. As for many Windows programs, an installer program is used.

The installed system is largely *autonomous* in that it requires only the prior presence of the Acrobat Reader (v. 4.0 or higher), and *non-invasive* in that it alters the behavior of nothing outside its own installed directory (currently called `dvipdf`). To deinstall it, one just deletes that directory.

Hopefully, this means that AcroDVI will be as simple to use as Acrobat Reader itself. For sophisticated users, there is an extensive configuration file to play with.

Performance testing

The following performance figures are for a 1997 PC with a Pentium I processor operating at clock speed 200 Mhz under Windows 95. For other Windows environments, a simple correction for clock speed should give a good first approximation to performance. The standard warning that "your mileage may vary" is appropriate. The programs, like vehicles, are extensively configurable, and the files, like terrain, are diverse.

For a typical mathematics article, the conversion to PDF format goes at about 15 pages per second, about 4 times greater than with Acrobat Distiller, or with *Ghostscript* in its PS2PDF mode. Comparison is relevant since it would be possible to publish compressed PostScript files without included fonts while giving Acrobat Distiller or *Ghostscript* access to the same BaKoMa font collection.

For its PDF output, DVIPDF does both font subsetting and stream compression. (The new compressed Type 1 font format has yet to be exploited by DVIPDF.) The efficiency of its default PDF output is thus respectable but not yet optimal. For example, it is comparable to that of the PDF files currently published by the American Mathematical Society, for the electronic research journal *ERA* (*Electronic Research Announcements*). However, by playing with the settings of Distiller, we were usually able to do better with Distiller, typically by a 3:2 ratio, particularly for small files. Do not rush to conclude that this ratio in favor of Distiller applies to all math journals. Indeed, the advantage swung in favor of DVIPDF for the next test by (not quite) a 2:3 ratio. It seems that both these PDF compilers could still reduce PDF bulk somewhat, in spite of many years of effort in this direction. From this point, however, we will focus on AcroDVI as a viewer of DVI, while ignoring its role as a compiler of PDF.

The DVI files used by AcroDVI are far less bulky than the PDF files used by Acrobat Reader. As evidence, here are a few examples from the first 1999 issue of the *Electronic Journal of Probability*, which added the PDF format compiled by Distiller to its web offerings in 1998:

Article	*Pages*	`.pdf`	`.pdf.gz`	`.dvi`	`.dvi.gz`	*Adv*
1	11	402	284	48	22	12.9
2	19	437	320	78	27	11.8
3	19	459	343	85	35	9.8
4	81	1162	960	412	154	6.2
5(?)	12	251	112	55	33	3.4

All file sizes are given in kilo-octets (Ko). Notice that DVI files regularly compress to about 40% of their original size while PDF files compress far less (as big internal chunks are precompressed). The last column of the table, the DVI advantage, gives the size ratio of compressed PDF files to compressed DVI files. This is an accurate measure of modem transfer speed ratios — whether the files are compressed or not — because during modem transfer, all material is compressed. The same ratio will be roughly the file size advantage of DVI files on a CD-ROM such as MathCD, which attempts to make the best use of available space. Indeed, a thoroughly precompressed form of PDF would be chosen for such a CD-ROM while the DVI files would probably be zip-compressed, along with any auxiliary graphics files.

The fifth article was anomalous in a number of respects. It had `\special` commands; there were two `.eps` figures, and these were complemented by their `.pdf` versions from Distiller, and the total of these graphics inclusions was less than 12 Ko of insertions (compressed). The explanation for PDF being only 3.4 times less efficient than DVI turned out, on investigation, to be mostly due to a common error in the production of PDF; namely, it was made with bitmapped TEX fonts, which perform disastrously in Acrobat Reader. When this is corrected, one can expect a PDF size similar to that of the first article.

Here are a couple of further examples, the shortest and longest available articles from a 1999 issue of *ERA*:

Article	*Pages*	`.pdf`	`.pdf.gz`	`.dvi`	`.dvi.gz`	*Adv*
1	3	105	84	12	5.5	15.3
2	12	248	215	66	27	8.0

These examples were reworked by us using well-tuned settings for Distiller (Windows version); the results (see below) are more flattering for the PDF format while leaving substantial advantage to DVI. Note that the *difference* between efficient and inefficient PDF is often many times greater than the total size of a DVI version.

Article	*Pages*	`.pdf`	`.pdf.gz`	`.dvi`	`.dvi.gz`	*Adv*
1	3	62	50	12	5.5	9.1
2	12	144	118	66	27	4.4

In the same vein, we note that the electronic journal *Geometry and Topology* (see `www.emis.de`) posts no TEX format whatsoever, just the Adobe formats PS and PDF. For their first 1000 pages the average PDF bulk per page is 10 Ko (fonts included) or about 8 Ko compressed. Thus, the DVI advantage would probably be somewhat less than 4. Expert use of PDF does make a big difference.

The modem bottleneck. Modem transfer speed is an important time factor. With a good telephone modem and a good line one can hope to get a transfer rate of about 5 Ko per second of compressed material. Now, a mathematics article in DVI format is about 2 Ko per compressed page, and thus the transfer rate is about 2.5 pages per second. This is about the speed at which one can scroll through the article with the 200 MHz PC used for these tests. Note that DVIPDF converts to PDF format at 6 times this speed. The time taken is perhaps time lost, but it is negligible.

With poor telephone lines or modems, or again congested web conditions, transfers that last more than a minute or two are likely to be broken; clearly the large PDF files are the ones at greatest risk, and with present web protocols, partial transfers are completely wasted.

Article transport costs. To get a very rough cost estimate, consider a mathematics article of 100 pages posted on the Internet and ultimately downloaded by a thousand readers (the typical number of subscriptions to a paper journal). Let is assume, to get an easily calculated figure, that everyone uses a contempory 56K baud modem with telephone charges of $2 per hour and in compensation let us neglect all other charges. With these figures, the telephone cost for delivering the article is about $22 for DVI format and between $70 and $250 for PDF format. Such figures suggest that use of DVI does reduce data transport costs significantly.

Improving the AcroDVI environment. First, the problem to be solved: in browsing the literature, it is not uncommon to look quickly at dozens of articles. This can quickly eat up many megaoctets of disk space if PDF format is involved. Now, one of Murphy's computing laws asserts that any hard disk that isn't new is surely nearly full, no matter what its capacity, since "data expands to fill any void". Thus, DVIPDF constantly risks running out of space.

To largely eliminate this overflow risk, there will be a setting for AcroDVI that makes the PDF file ephemeral and invisible. As soon as the next DVI file is processed, the previous invisible PDF will be erased. (That is no loss, since it can be regenerated quickly.) With this scheme, it suffices to verify at the beginning of a browsing session that your hard disk has enough space for the largest

single PDF file you expect to read, plus enough space for the relatively small DVI files.

Going one step further, the speed of AcroDVI can now be *doubled* by switching off compression of the the PDF output. At this point AcroDVI has been nicely optimized as a DVI viewer — at the cost of temporarily neglecting its role as a PDF compiler.

Comparing PDF and DVI formats

Adobe's Acrobat Reader has PDF as its native file format. This format is very autonomous:

- Graphics objects are always embedded within the PDF file.
- Fonts are usually embedded as well (the alternative, to use system fonts, has proved somewhat unreliable).

These positive features bring some disadvantages:

- Bitmapped graphics are unlikely to be displayed on-screen at optimum quality since that means no scaling. Vectorial graphics may not be seen in their full glory since that often requires the full screen.
- It is difficult to export graphics objects from the PDF file in the most useful formats.
- PDF files tend to be many times larger than DVI files. This is, of course, partly because of the font burden,[1] but the complexity of the PDF file structure brings substantial hidden costs.

Besides its space economy, the DVI format has other virtues worth mentioning. Like all of TEX, the DVI format is very stable, in spite of (and even because of) its `\special` appendages. This is important for archiving. Second, DVI is simple: only a few pages in Knuth's book on the TEX program (Knuth, 1986) are needed to define it adequately. Finally, one can derive from DVI all formats currently used for mathematics, excepting TEX source (i.e. the `.tex` file).

The strongest argument for PDF format has been the wide availability and high performance of the Acrobat Reader. Particularly outstanding are the user interface, the graphics quality, and the graphics speed. Adding to this: search, hypertext, copy-and-paste to text files, annotations, printing facilities, and PostScript (or EPS) export, it is clear that Acrobat Reader is a major contender for the affections of the reading public.

[1] The journal *Geometry and Topology* posts PDF format both with and without included fonts; omission of fonts economizes 25% over the first thousand pages of articles.

This does not prove that Acrobat Reader has no rivals among DVI readers. For example, xdvi (under unix) is by far the fastest viewer; the recent *BaKoMa DView* can do better in quality and scope of typography; emTEX provides better search and text export. Interesting new DVI viewers continue to appear: for example, *tkdvi* and *nDVI* (see Resources for details). It would be destructive not to serve such DVI readers. And ultimately destructive of TEX itself since TEX systems are typically built around them.

EPS, and now PDF, PNG and JPEG

The schemes to be described follow proposals in (Siebenmann, 1996); they are just some of many that have been elaborated for integration of graphics into the PDF output of DVIPDF (see Lesenko, 1997, 1998).

Let us begin by considering the graphics integration issue that arose for electronic journal articles to appear on MathCD. The DVI format is usually one of two or three presented, and it entails, for each article, one DVI file accompanied perhaps by some EPS graphics files. For MathCD it was important that the TEX version of the articles *not* be necessary for the integration of the reformatted graphics files.

Since DVIPDF cannot, on its own, convert EPS graphics to PDF, it was initially decided to provide PDF versions of the graphics objects via Distiller. One reason for this decision was that the PDF versions of vectorial graphics files are of optimal quality and often quite efficient, provided that font subsetting is used in creating them. Inasmuch as these PDF files can be immediately viewed by Acrobat Reader on most platforms, this conversion is of immediate benefit to almost all users.

Gradually, it became apparent that conversion to PNG and JPEG formats by various methods sometimes offers greater advantages. Fortunately, the solution to be described for PDF extends to PNG and JPEG graphics files.

The `\special` syntax in the DVI files used for EPS integration was most often the one used by Tomas Rokicki's `epsf.tex`. Aiming to exploit pre-existing DVI files using with Rokicki's *dvips*, we decided to have DVIPDF interpret the existing Rokicki syntax:

```
\special{Psfile=test.eps llx=11 lly=22
         urx=33 ury=44 rwi=550 rhi=660}
```

This is probably the world's most common `\special` syntax.[2]

The unit for the first four "bounding box" entries is 1 bp ("big point"). `llx` is the x-coordinate of the lower left corner of the bounding box, etc. Most often (but not always), this bounding box has simply been copied by TEX from the bounding box indicated in the EPS file header.

The last two entries, tagged by `rwi` (for real width) and by `rhi` (for real height), specify, in units of 0.1 bp, the width and height of the integrated bounding box on the output page. Either of these two entries, may be absent, in which case uniform scaling is used. By convention, the integrated box has its lower left corner placed at the DVI insertion point.

The (expanded) argument of this `\special` command is passed intact into the DVI file. Beware that it is normally generated inside of TEX, so that the author sees only some high-level commands, as those found in `epsf.tex`.

For both the `.eps` file and its derived `.pdf` file, the figure is located on a coordinate plane with unit of length = 1 bp; also, the scale and orientation are the same for both planes. In the event the `.pdf` file was created by *Ghostscript*, the two coordinate systems will be exactly the same. Then the *dvips* rules of integration from *dvips* are applied without modification and the results are identical.

If the `.pdf` file was created by Distiller, the two coordinate systems are related by a translation and some care is required required to make it predictable. We omit the details.

In fact, MathCD has used Distiller mainly, encountering only occasional problems. Fortunately, the reader of an article will be completely oblivious to such complications; only website editors, CD-ROM editors, and conscientious authors are concerned.

Generalizing to bitmapped graphics. There is an important variant of the above mechanism that is optimal for bitmaps. EPS and PDF are very general formats that can accommodate vectorial or bitmapped images; however, for bitmaps, EPS tends to be bulky and slow, and both seem to obstruct the recovery of embedded bitmaps. On the other hand, bitmap manipulation tools such as XNview under Windows and Linux/UNIX or *Graphics Converter* for Macintosh are easy to obtain and can generate an EPS format at any time. Thus, to the extent that you wish to grant full control of bitmapped graphics to the reader of your article, you may wish to use a native bitmapped norm.

The converse applies too: one can lock a PDF file or restrict its use in various ways. And it must be conceded that PDF manages to inherit the space efficiency of both leading public bitmapped formats: PNG and JPEG.

Recall that PNG (Portable Network Graphics) is the most efficient contemporary norm for faithful bitmap compression and is suitable for scientific figures and for most of the myriad uses which the commercial GIF format enjoys on the web. PNG is typically 15% more compact than GIF.[3] JPEG is the dominant "lossy" format for compression of low-contrast color images such as photos. JPEG (like GIF) is well supported by current versions of the Web browsers.

Hopefully, the above considerations will encourage more TEX users to exploit PNG and JPEG bitmaps. Those who are still restricted to vector graphics in TEX should be reminded, every time they see a web browser, that the full gamut of (still) bitmapped images, color included, as seen on the web, are begging to be used in TEX.

What we have said about PNG and JPEG being native or editable graphics formats is to some extent true for PDF. Indeed, the Windows graphics program Mayura Draw uses it as its storage format; however, it does not read arbitrary PDF files.

It is clear from the above conversion, that the preparation *ab initio* of a manuscript in DVI format with PNG bitmapped graphics inclusions can use the conventional EPS integration mechanism. We summarize since the process applies with little change also to JPEG and PDF:

- convert the PNG graphics to EPS, in XNview or similar program;
- integrate the `.eps` file using the consensual `special` command; and finally,
- replace the `.eps` file by the original `.png` file (the `.eps` file is usually bulky and is perhaps best discarded since it can be regenerated if the need arises).

The end user then just pushes the `.dvi` file onto the AcroDVI icon and viewing in Acrobat Reader will begin — using the original PNG graphics.

[2] It is not, however, the simplest for the job; indeed, one could get by without the "bounding box" entries (cf. Siebenmann, 1996).

[3] Unfortunately, the leading browsers, Netscape and Internet Explorer, have been tardy and half-hearted in their support for PNG. It may become necessary for AcroDVI to support GIF.

But there is a shortcut; it is unnecessary to generate an EPS file. Specifying

```
DoBBoxFile  =YES
```

in a configuration file, preview the PNG by pushing its icon onto that of AcroDVI. As a by-product, this previewing creates an auxiliary file (extension `.bb`), which contains the BoundingBox comment as in an EPS file header. With a suitable macro package such as `boxedeps.tex` (version for year 2000) or the LaTeX packages `graphics` or `graphicx`, the `.bb` file can be used *in lieu of* an EPS file.

Auxiliary roles for *Ghostscript*. The first is to allow on-the-fly integration by AcroDVI of EPS files into DVI format; optional settings of AcroDVI enable this when *Ghostscript* is present. This is very useful for viewing legacy DVI-plus-EPS postings prevalent on the Internet.

When an author or publisher is preparing an article for publication in DVI format with graphics inclusions, the strategy should be to vary the graphics format: maximize image quality while minimizing bulk. Effort spent on this often leads to surprising but useful results (see the 1999 documentation for `boxedeps`). Thus, it is advisable to urge authors to present originals of all graphics objects.[4]

Secondly, *Ghostscript* is a valuable converter to bitmap formats from PS, EPS, and even PDF; it has command-line options for parameters such as resolution. Unfortunately, *Ghostscript* has its quirks as a rasterizer. On the other hand, we have mentioned that the Adobe PS- and PDF-based systems seem loath to surrender internally stored bitmaps; they can be likened to a bank so eager for deposits that it has forgotten to provide for withdrawals. When need for withdrawals comes, *Ghostscript* may be your best friend.

The medium molds the message. One has to bear in mind that the various graphics formats and the various viewing mechanisms may influence what ultimately reaches the human eye. The pages of *TUGboat*, for example, are printed in black and white by photo-offset methods and will never faithfully render the colored iris that is the icon for AcroDVI.

For the reader's amusement, Fig. 1 shows in black-white several rather different renditions of the iris, all of which derive from one multicolored pastel original contributed by Tina and Keira Miyata. This

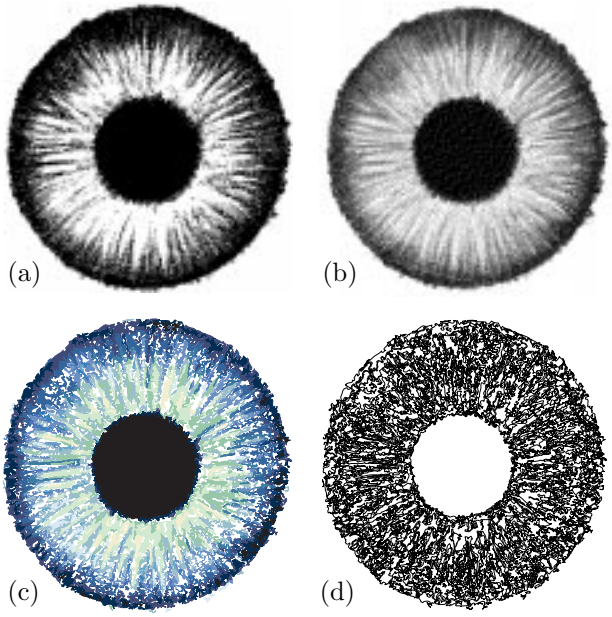

Figure 1

was scanned in 32-bit color to a 450×450-pixel bitmap, and stored as a 57 Ko file in JPEG format. This *TUGboat* article used the `.eps` versions.

(a) a suitable projection to black-white (= b/w) by *Graphics Converter*; size 30 Ko as `.eps.gz`, 20 Ko as `.pdf.gz`, 14 Ko as .png.

(b) is derived by Floyd-Steinberg filtering by *Graphics Converter*; size 36 Ko as `.eps.gz`, 26 Ko as `.pdf`, and 18 Ko as `.png`,

(c) (colored) arises from vectorization, by 16 regions of flat color, using Adobe's Streamline (v. 3); size 144 Ko as `.eps.gz`, 146 Ko as `.pdf.gz`, and 30 Ko as `.jpg`.

(d) (b/w) is the 1000 or so curves that are the boundaries of the 16 colors of (c); size 203 Ko as `.eps.gz`, 186 Ko as `.pdf.gz`, and 15 Ko as `.png`. This last bitmap blurred the curves; doubled resolution gave a 53 Ko `.png` file.

The quest for image clarity and beauty is an empirical art; with no testing, the results of photo-offset printing may be disappointing. We apologize in advance.

On the need for DVI efficiency

There is currently lukewarm support for the use of efficient methods. What can be done efficiently by astute programming is by preference done by the liberal expenditure of RAM or disk space or processor power. Thus, for AcroDVI to be taken seriously, cogent evidence is wanted that the resources economized through maintaining and

[4] For example, in preparing MathCD, the lack of such originals has been more of a vexation than the lack of TeX source files!

developing the efficient DVI format can be used decisively.

This is perhaps most evident with CD-ROMs. A CD-ROM contains about 650 Mo of data. If exploited to archive mathematics in compressed DVI form (and no other) a CD-ROM could contain about 300,000 pages of mathematics. That is enough space to record all the mathematics currently on the `xxx.lanl.gov` "e-print" archive (recently named "arXiv"), which is said to amount to about 200,000 pages. Alternatively, it is enough to distribute all the mathematics research articles published in one year (on paper *or* electronically). Again, it is enough space to reprint the whole of the *Annals of Math* (the most prestigeous math journal) plus the whole of *Crelle* (the oldest math journal).

Going beyond mathematics, it might be possible to present a complete encyclopedia on a single CD (or two), using compressed DVI (and graphics) files for storage and AcroDVI for viewing. Currently, the favored storage format for encyclopedias is RTF (Rich Text Format) and the usual viewer is MSWord. RTF enjoys efficiency comparable to that of HTML and DVI; it allows the same enviable flexibility of line length as HTML, and it is somewhat more expressive than HTML but less so than DVI. AcroDVI (allied with Acrobat Reader) offers the best typography and $peed.

Although such projects may not be realized in the immediate future, MathCD is intended to hint at them all.

There should also be evidence that CD-ROM capacity will not grow so fast that it outstrips the increasing demand for such permanent storage. If it does, then it is plausible that there is room for waste. The spectacular 1000-fold growth of the capacity of inexpensive hard disks in the last dozen years has fed wild expectations of storage technologies. But the reality for CD-ROMs is sobering. It is now known that the next (second) generation of CD-ROMs, called DVDs (Digital Versatile Disc) coming about 15 years after the first, will be based on a simple evolution of the current CD-ROMs: a rough doubling of density is involved, along with use of both sides of the disk. The capacity gain to 4.5 Gig[5] will be somewhat less than 10-fold (not 1000-fold). This is a factor frighteningly similar to the wastage factor that would be imposed by general adoption of the bulky PDF format. Furthermore, it could be a decade before the new CD-ROM format is sold at the affordable prices of today's CD-ROMs, since that is the time it took for today's CDs to reach mass consumer prices. This is one of the strongest arguments for retaining the efficient DVI norm. Fortunately, DVD readers will accept today's CD-ROMs.

The current pause in progress of telephone modem speeds gives additional arguments. The 56 kilobaud telephone modems of today are considered to be the last gasp of a tired technology up against what is called the "Shannon limit". In this case, a dramatic switch to ADSL (Asymetrical Digital Subscriber Lines) is being promoted with great speed gains: nominally 1.5 megabits/sec download and .5 megabits for upload (but, in practice, perhaps only one third or one quarter of that). There remains the question whether and when this technology will be as widely available and as affordable as the present modem technology.

Although hard disk capacity has been growing prodigiously, electronic libraries such as ELibMath EMS (`www.emis.de`) could come to need DVI's polyvalence and efficiency. Thus far, a hard disk of a few gigaoctets is sufficient to store the entire library of a few dozen journals. At current affordable prices for storage, dozens of mirror copies of the library have been established worldwide. As time passes, journals are not only multiplying and individually growing but are offering more and more formats for downloading, notably the bulky PDF. If and when this causes overflow of the current generation of the ELibMath hard disks, DVI format could offer an attractive remedy.

The `xxx.lanl.gov` e-print server has shown the way on economy by deriving essentially all formats from a `.tex` source on demand. This server successfully deploys immense expertise and resources under UNIX systems and manages to compile any document from `.tex` files to derive on-the-fly any other format the user requests. To do much the same on a CD-ROM, but using `.dvi` format, seems just within the realm of possibility — relying heavily on the greater simplicity and wide acceptance of DVI format. The first edition of MathCD will nevertheless be far more liberal (heteroclite) than the `xxx.lanl.gov` server.

We conclude that the economy and polyvalence of TEX's original DVI norm may indeed be the magical stuff from which dreams can be woven.

Is AcroDVI in the lead?

As a front end to Acrobat Reader for DVI viewing, how well does AcroDVI face competition?

[5] Double that for two-layer versions — whose durability is, unfortunately, in doubt.

There are several interesting *indirect* competitors that we merely mention in historical order: *Ghostscript/Ghostview*, then Distiller teamed with Acrobat Reader, and most recently pdfTEX (see Thành, 1998), also for use with the Reader.

Potentially, the strongest *indirect* competitor would be Acrobat Reader itself using a more compact and agile version of PDF format — but there is no sign of that.

One *direct* competitor of AcroDVI is Malyshev's *BaKoMa DView*, which not only has the broadest typographic capabilities in the TEX world of 1999, but also the ability to output PDF files. We leave the user to judge the relative virtues. Both will be provided on MathCD.

A second direct competitor is dvipdfm by Mark A. Wicks, which surfaced in 1998. It is an autonomous converter quite similar in concept to DVIPDF. Executable binaries are available on CTAN for 2 platforms, W9X/NT and i386 Linux. The reviewer of our paper informed us of many compiled dvipdfm binaries on the TEX-Live 4 CD-ROM. The platform/OS combinations served include: DEC alpha/OSF4, HP/HPUX10, i386/Linux, SGI/IRIX6.2, RS6000/AIX4.1.4, Sparc/Solaris 2.5–2.6, and Windows (32-bit).

Thus far, neither of these direct competitors has provided close integration with Acrobat Reader. It is probably fair to say that both are presently aiming at PDF publication, *not* DVI viewing. They are not yet competing frontally — but they soon could.

As for support of the most frequently used `\special` commands, *BaKoMa DView* is well advanced, thanks to adherence to *dvips* syntax. AcroDVI has some catching up to do here because it originally fashioned its own `\special` syntax; basic functionality for color and hyper-references are, however, present. Least adapted for viewing legacy DVI files is dvipdfm — because of its reliance on 'pdfmark' syntax; however, it has good basic `\special` functionality.

On the other hand, the recent wide porting of dvipdfm and distribution via the TEX-Live CD could well eclipse DVIPDF, and with it, AcroDVI. If that is our fate, we hope that both DVIPDF and AcroDVI will nevertheless be remembered as seminal proofs of feasibility.

Acknowledgements and History

The second author is grateful for an invitation from Stanislas Klimenko to visit IHEP in Protvino for several weeks in the autumn of 1997 to work with the first author, and also with Basil Malyshev. Basil has very kindly permitted us to distribute a version of his BaKoMa font collection with AcroDVI.

The idea of exploiting DVIPDF and Acrobat Reader together as a feature-rich DVI reader has been a subject of discussion between us (Lesenko and Siebenmann) since the 1996 TUG meeting in Dubna, Russia. For a long time, this project remained on a back burner while basic features of DVIPDF were perfected by the first author. As MathCD project took shape, it offered many stimulating design challenges, and the last year has brought substantial progress that seems to justify our early optimism.

Resources

Acrobat: a series of products by Adobe Inc., including Acrobat Reader, and Acrobat Distiller; the former is free while the latter is sold (but low prices for Distiller are available to academic users in many countries). Supported platforms include: Windows 3.x, Windows 9X, Windows NT, Macintosh, OS/2-Warp, Linux, IBM-AIX, SunOS, Solaris, SGI-IRIX, HP-UX, and Digital UNIX. Adobe's website address: `www.adobe.com`.
There is an active news list (`comp.text.pdf`) that can provide user support. See also EMJ, below, in particular Nelson Beebe's comments of 23 April 1999.

AcroDVI (with DVIPDF): by S. Lesenko and L. Siebenmann. Alpha versions are posted by anonymous `ftp` in Europe and N. America:
`topo.math.u-psud.fr/pub/tex/`
`cmstex.maths.umanitoba.ca/pub/acrodvi`
When AcroDVI is reasonably stable, it will be submitted to the CTAN archive.

BaKoMa TEX: by Basil K. Malyshev. A TEX implementation for the Microsoft Windows OS that appeared in 1998. Includes an advanced version of the BaKoMa font collection, the DVI viewer DView, and a DVI-to-PDF converter. Available from CTAN and from `ftp://ftp.mx.ihep.su`.

`boxedeps`: by Laurent Siebenmann. A macro package for EPS graphics integration that is valid for all PS printer drivers. Available from CTAN. The year 2000 version co-operates with some PDF compilers to integrate PDF, PNG, JPG graphics using `.bb` files.

dvips: by Tomas G. Rokicki; available from CTAN in the `dviware` directory.

dvipdfm: by Mark A. Wicks. A DVI-to-PDF converter that appeared in 1998:
`http://odo.kettering.edu/dvipdfm/`
Currently available on CTAN for i386 Linux, and for Windows 9X/NT as part of the MikTEX and fpTEX distributions.

EMJ: the Electronic Math Journals discussion list: `http://math.albany.edu:8800/hm/emj`.

`graphics`, `graphicx`: LaTeX2$_\varepsilon$ packages by David Carlisle and Sebastian Rahtz, on CTAN.

Graphics Converter: by Thorsden Lemke. bitmap editor and converter for Macintosh; shareware. `www.lemkesoft.de`.

Ghostscript: by Peter L. Deutsch.
A PostScript and PDF interpreter, that provides bitmapped or PDF output; the latter function is called PS2PDF.
`ftp.cs.wisc.edu/pub/ghost/aladdin`.

GSview: by Russell Lang. A viewer based on *Ghostscript*
`ftp.cs.wisc.edu/pub/ghost/rjl/`.

MathCD: CD-ROM (in prep.) devoted chiefly to journals and software for mathematics.
Go to `MathCD.html` at the editors' web sites: `www.math.washington.edu/~burdzy`, `topo.math.u-psud.fr/~lcs`, and `rsp.math.brandeis.edu`.

Mayura Draw: a graphics program by Karunakaran Rajeev; its native format is a dialect of PDF. `www.wix.com/mdraw210.zip`.

nDVI: a DVI viewer by K. Peeters. `norma.nikhef.nl/~t16/ndvi_doc.html`.

tkdvi: a DVI viewer by A. Lingnau. `www.tm.informatik.uni-frankfurt.de/~lingnau/tkdvi`.

XNview: by Pierre-E. Gougelet, bitmap editor and converter for Windows, Linux, etc.: `latour.univ-paris8.fr/~pierre`.

References

Bienz, Tim; Richard Cohn; and James Meehan. *Portable Document Format Reference Manual*. Addison-Wesley, Reading, Massachusetts, 1993.

Knuth, Donald. *TEX The Program*. Addison-Wesley, Reading, Mass., 1986.

Lesenko, Sergey. "The DVIPDF Program." *TUGboat* **17**(3), 252–254 (1996).

Lesenko, Sergey. "DVIPDF and Graphics." *TUGboat* **18**(3), 166–169 (1997).

Lesenko, Sergey. "DVIPDF and Embedded PDF." Proceedings of Euro-TEX Conference, St. Malo. *Cahiers GUTenberg* **28–29**, 231–241 (1998). `www.gutenberg.eu.org/pub/GUTenberg/`

Malyshev, Basil. "Problems of the conversion of METAFONT fonts to PostScript Type 1". *TUGboat* **16**(1), 60–68 (1995).

Siebenmann, Laurent. "DVI-based Electronic Publication." *TUGboat* **17**(2), 206–214 (1996).

Sojka, Petr, Hàn Thế Thành and Jiří Zlatuška. "The joy of TEX2PDF — Acrobatics with an Alternative to DVI Format." *TUGboat* **17**(3), 244–251 (1996).

Thành, Hàn Thế. "The pdfTEX Program." Proceedings of Euro-TEX Conference, St. Malo. *Cahiers GUTenberg* **28–29**, 197–210 (1998). `www.gutenberg.eu.org/pub/GUTenberg/`

Post-Conference Addendum

With reference to the discussion on modem downloading speeds (section "On the need for DVI efficiency"), Michael Doob reports top speeds near 500 Ko/sec on an optical cable network installed originally for cabled home television. This is stunning progress; even the authors' institutional ethernet LANs have never offered speeds quite so high. Curiously, the slower ASDL technology is attracting more investment. One should bear in mind that better Internet access may well increase 'peak time' Internet congestion, at which times effective throughput is often less than for a simple telephone modem.

We authors thank Michael Doob for hosting our alpha version, and also for making the oral presentation in Vancouver, when, at the last minute, the first author was unable to attend.