Compressed Images in PostScript and PDF with LaTeX HOWTO

Daniel Käps

last revised: 2002-11-24

Abstract

It is a common problem that PostScript files (as generated by TeX/LaTeX users) become really large when users try to include pixel mapped images (also referred to as "sampled images", "pixmaps" or "bitmaps"). The huge size of the resulting PostScript files is due to the fact, that many converters to Encapsulated PostScript decompress the images.

However, with the right tools and specifying the correct options it is possible to include certain image types (such as JPEG or PNG) retaining compression. This can greatly reduce the size of the resulting PostScript files and thus speed up downloading and displaying.

The article explains which tools and image formats can be used to put images into PostScript without decompression. Moreover, some considerations have been made about which image formats can be used when generating other output formats, namely PDF (Portable Document Format) and HTML, from a LaTeX source.

Contents

Generation of DVI/PS Files (e.g. using 'latex')

Pixel mapped images can be included via LaTeX's '`\includeimage`' (see [1]) into PostScript output without problems when they are available in (Encapsulated) PostScript format.

Anyway, to put *compressed* images into the (Encapsulated) PostScript Format, (at least) PostScript *Level II* must be used. Decompression has to be done by the PostScript viewer/interpreter. Many PostScript viewers seem to be able to do this. Specifically, the following programs/versions were tested. All of them were able to display the PostScript files containing the compressed images (or the DVI files referencing the EPS files):

- GNU Ghostscript 5.10 (1998-12-17) (for e.g. gv 3.5.8)
- gv 3.5.8 (Debian 2.2)
- gnome-gv 0.82 (Debian 2.2)
- xdvi (Debian 2.2)
- Aladdin Ghostscript 5.50 (1998-9-11) (for GSview 2.6)
- GSview 2.6 (1998-09-17) (Win95b)
- Yap 0.96a (Win95b)

Basically, the conversion of the images to Encapsulated PostScript can be accomplished in different ways:

- **invoking external tools before calling 'latex'**
  External tools may be called manually or from a shell script to create the EPS files before '`latex`' is invoked.
- **invoking external tools while running 'latex'**
  Using the '`\DeclareGraphicsRule{...}`' macros, LaTeX can be told to call the specified external tool while compiling the LaTeX source.
- **using a modified 'dvips'**
  The default '`dvips`' cannot deal with image formats such as JPEG or PNG. However, there are some patches available to make it capable of dealing with some image formats. [1] [2]

The following will concentrate on the usage of different external tools to convert images to PostScript Level II or III.

Inclusion of JPEG images

PostScript Level II supports direct inclusion of JPEG images using the 'DCT' (discrete cosine transform) filter [3]. To display a PostScript file containing images represented with the DCT-filter, the PostScript interpreter must support (at least) PostScript Level 2.

[1] For example, if '`dvips`' is compiled with the option '`-Demtex`', it will support inclusion of PCX, BMP, MSP bitmaps [2]. However, this is not very portable, the compression ratio with those image formats is usually poor compared to JPEG or PNG compression., and the driver probably decompresses the images when generating PostScript anyway.

[2] The 'bmeps' package ([6]) contains, besides the command line tool '`bmeps`', modified sources for '`dvips`' so that the 'bmeps'-library can be used implicitely. However, since the 'bmeps' library doesn't support JPEG(DCT) compression (at least not in version 1.0.4), one still had to use a different external tool to create EPS files with good compression for included JPEG/photographic images.

JPEG-to-EPS conversion using 'jpeg2ps'

JPEG images can be converted to PostScript with the 'jpeg2ps' program [4]:

```
jpeg2ps IMAGE.jpg >IMAGE.jpg.eps
```

This way the compressed image data is included into the PostScript file as is (*without decompressing it*), which reduces the size of the resulting PostScript file considerably.

Inclusion of PNG and TIFF images

PostScript *Level 3* supports inclusion of TIFF and PNG images by the following means: The 'LZW' (Lempel-Ziv-Welch) and 'Flate' (zlib/deflate) filters are available for (de)compression. The PNG and TIFF pixel-predictor functions are supported [5]. However, while e.g. PNG images can have Alpha-transparency, PostScript seems to support only Bi-level transparency (using the 'imagemask'-operator).

In contrast, PostScript *Level 2* only supports 'LZW'-compression and apparently it doesn't have support for pixel-predictor-functions. Thus, many images won't be compressed as well as it is possible with Level 3's features. Also note that the 'LZW' algorithm is patented, and using it may require paying license fees to the company holding the patent.

PNG-to-EPS and TIFF-to-EPS conversion using 'bmeps'

PNG and TIFF images can be converted into PostScript Level I, II or III with the program 'bmeps' [6]. To create an EPS file with maximum compression, use:

```
bmeps -p3 -c -e8f -tpng IMAGE.png >IMAGE.png.eps
```

which creates a Level 3 EPS file using 'flate'-compression for a colored PNG file. Only some small problems occured with the used version of 'bmeps':

- **image extension:**
  Note that, unless the file comes from 'stdin', the image extension *must* be .png, .tif, .tiff or another to 'bmeps' known extension, otherwise the file will be silently ignored, even if using the switch '-t').
- **problems with Yap 0.96a:**
  Yap 0.96a (a DVI viewer for Windows coming with MiKTeX) had some (correctable) problems with viewing DVI files.[3]

For a TIFF file, the call is simply:

```
bmeps -p3 -c -e8f -ttiff IMAGE.tiff >IMAGE.tiff.eps
```

'bmeps' is also capable of mapping TIFF and PNG alpha channels to EPS level 3 (Bi-level) image masks.

PNG-to-EPS and TIFF-to-EPS conversion using ImageMagick's 'convert'

PNG and TIFF images can be converted into PostScript Level III using the program 'convert' which is part of the ImageMagick graphics tools [7]:

[3] Yap wasn't able to handle DVI files referencing image files converted to EPS using 'bmeps 1.0.4', with the options '-p3 -c -e8f'. GSview 2.6, however, had no problem with displaying the final PostScript file. The problem can be simply corrected by manually moving in the generated EPS file the 'tilde–greater' end sequence of the ASCII-85 encoded image data to the preceding line.

`convert -compress zip IMAGE.png eps2:IMAGE.png.eps`

This includes the PNG file without decompressing it into the PostScript file using 'flate'-compression. However, things are not as straightforward as they are for '`jpeg2ps`' or '`bmeps`':

- **ImageMagick version and platform:**
  The used version of ImageMagick was: 4.2.8 99/08/01 under a Debian Linux 2.2 and ImageMagick 5.1.0 00/01/01 under SuSE Linux 6.4. The call didn't work for ImageMagick version 5.4.9 under Windows, nor did it work (under Windows) with any other combination of options that were tried .

- **PostScript level (version):**
  One must specify '`eps2:`' as PostScript output format (not just '`eps:`'), otherwise the image is put uncompressed into the EPS file. (Actually, the option should have been named 'eps3:', not 'eps2:'.)

- **option** `-compress`**:**
  For ImageMagick version 4.2.8 99/08/01 one must specify the option '`-compress zip`', otherwise the image will be put uncompressed into the EPS file. However, for ImageMagick version 5.1.0 00/01/01, '`-compress zip`' is used by default.

- **problems with some images:**
  For http://www.libpng.org/pub/png/img˙png/AlphaBall.png, an RGB-type PNG image with an Alpha channel, the call produced an EPS file 10 times as big as the original PNG file (with ImageMagick 4.2.8 99/08/01). However, for another RGB-type PNG image with Alpha channel (specifically http://www.libpng.org/pub/png/img˙png/imgcomp-440x330.png), the size of the EPS file was ok (about 1.25 times the size of the PNG image), as it was for almost any other tested image.

To reduce the size of a PNG image itself, and to convert from a RGB-type to a palette-type PNG, the program '`pngquant`' [9] can be used.

For a TIFF file, the call is:

`convert -compress zip IMAGE.tiff eps2:IMAGE.tiff.eps`

TIFF-to-EPS conversion using libtiff's 'tiff2ps'

TIFF images can be put compressed into a PostScript Level II file with the program '`tiff2ps`', which is part of the libtiff tools [10]:

`tiff2ps -2 -e -s IMAGE.tif >IMAGE.tif.eps`

However, the following things should be taken into account:

- **libtiff version and platform:**
  The used version of libtiff was 3.5.4 (under a Debian 2.2 Linux).

- **decompression of 'deflate'-encoded TIFF images:**
  It seems that for 'deflate'-encoded TIFF images (check with '`tiffinfo`'), '`tiff2ps`' unfortunately *decompresses* the images so the generated Encapsulated PostScript files become really huge (scope: Version 3.5.4 under a Debian 2.2 Linux).

- **Patent problems with LZW encoding:**
  The only compression method supported for EPS by '`tiff2ps`' seems to be the LZW compression. Since this is legally encumbered, it should not be used, anyway.

- **compression ratio:**
  The compression ratio obtained by 'tiff2ps' with LZW is usually poorer than the compression ratio obtained when using 'bmeps' or ImageMagick's 'convert' using 'flate'-compression as described above.

Generation of PDF Files (using 'pdflatex')

As with 'latex', pixel mapped images can be included with '\includeimage' (see [1]). 'pdflatex' supports inclusion of JPEG, PNG and TIFF *directly*, without the need to convert them before to something like Encapsulated PostScript [11].

Moreover, PDF can include EPDF (Encapsulated PDF), but cannot include EPS files. However, EPS files may be converted to EPDF using external tools, such as 'ps2pdf' (TODO: must test this).

Generation of HTML Files (using 'tex4ht')

Using 'tex4ht', an image can be embedded using the following LaTeX code (see [12]):

```
\Picture[ALT.-TEXT]{IMAGEFILENAME HTML-IMAGE-ATTR.}
```

Note that the HTML image attributes should define 'width' and 'height' so that Web-browsers can render the page correctly before all images have been loaded.

Most Web browsers can display JPEG and GIF, newer browsers can also display PNG (see [13]). To achieve compatibility also with older browsers, PNG images could be converted into GIF images. (Note that the 'LZW'-algorithm used to compress GIF images is patented). Formats like TIFF, EPS, EPDF and anything else usually cannot be displayed by Web browsers.

Conclusions

JPEG images can be included without decompression into PostScript Level II, III and PDF and can be displayed by most Web-Browsers.

PNG and TIFF images can be included without de-compression into PostScript Level III and PDF (and with lower compression also in PostScript Level II), but only newer Web-Browsers support displaying PNG images.

TIFF images can be included into PDF, but normally cannot be displayed by Web browsers.

Other image formats may require conversion to be usable for inclusion into PostScript, PDF or HTML. Conversion between various image formats can be done e.g. using ImageMagick's 'convert', which supports hundreds of different graphics formats.

JPEG and PNG generally provide better compression than most other image formats, so it seems reasonable to keep the 'source' images in either of these two formats, depending on which is better suited.

Note that some older printers may not be capable of directly printing PostScript files making use of Level III features. To print Level III files on such a printer, e.g. 'GSView' can be used.

For HTML output, one may choose to use GIF instead of PNG for compatibility with older Web-Browsers.

Compilation Platform

The versions of the tools to convert images were:

- jpeg2ps V1.8 (under Linux)
- bmeps 1.0.4 (under Linux),
  bmeps W32:2002/08/02 (under Win95b)
- ImageMagick 4.2.8 99/08/01 (under Linux)
- ImageMagick 5.1.0 00/01/01 (under Linux)
- tiff2ps from libtiff v3.5.4 (under Linux)

The used TeX versions were:

- TeX (Web2C 7.3.1) 3.14159 (under Linux)
- pdfTeX (Web2C 7.3.1) 3.14159-0.13d (under Linux)
- tex4ht.c (1999-07-18-00-02) (under Linux)
- MiKTeX-TeX 1.15 (TeX 3.14159) (MiKTeX 1.20d build 110) (under Win95b)

References

[1] Packages in the 'Graphics' Bundle,
http://www.ctan.org/tex-archive/macros/latex/required/graphics/grfguide.ps

[2] Keith Reckdahl: *Using Imported Graphics in LaTeX 2e* (Version 2.0, December 15, 1997),
http://www.ctan.org/tex-archive/info/epslatex.pdf [1.5 MB]

[3] Adobe *PostScript Language Reference Manual*, Second Edition, 1990,
http://partners.adobe.com/asn/developer/pdfs/tn/psrefman.pdf [3.3 MB]
Sections 3.8.4 (Filters) and 3.13.3 (Details of individual filters, p. 127)

[4] jpeg2ps utility – a JPEG to PostScript Level 2 (or PostScript 3) wrapper,
http://www.pdflib.com/jpeg2ps/index.html

[5] Adobe *PostScript Language Reference*, Third Edition, 1999,
http://partners.adobe.com/asn/developer/pdfs/tn/PLRM.pdf [7.4 MB]
Section 3.13.3 (Details of individual filters, p. 129)

[6] bmeps – command line tool, library and dvips patch for bitmap-to-EPS conversion,
http://www.e-technik.fh-schmalkalden.de/personen/dhp/krause/frames/index.php3?fc=entire&f=/personen/dhp/krause/pages/software/bmeps/index.html

[7] ImageMagick – Convert, Edit, and Compose Images,
http://www.imagemagick.org/

[8] libpng – A Basic Introduction to PNG Features,
http://www.libpng.org/pub/png/pngintro.html

[9] pngquant – utility to quantize and dither 32-bit RGBA PNGs down to 8-bit (or smaller) RGBA-palette PNGs,
http://www.libpng.org/pub/png/apps/pngquant.html

[10] TIFF Software – library, tools for TIFF images,
http://www.libtiff.org/

[11] The pdfTEX manual,
http://www.tug.org/applications/pdftex/pdftex-a.pdf

[12] tex4ht – From LaTeX to HTML – Pictures,
http://www.cis.ohio-state.edu/%7Egurari/TeX4ht/mn8.html

[13] libpng – Browsers with PNG Support,
http://www.libpng.org/pub/png/pngapbr.html

If you have comments or suggestions, mail to:
kaeps@-spam-away-.informatik.uni-leipzig.de
(remove the "-spam-away-." spam protection).