When (image) size matters

Peter Willadt

Abstract

For space and performance reasons, scaling of images to be included into a PDF document down to a certain resolution is often desirable. This article describes a halfway automatic method to achieve this goal with pdfTEX.

1 Basics

TEX output is by default device independent, and this is fine. In ancient implementations, adoption to different previewers or printing devices resulted from the work of DVI processing software, in recent years TEX has been widely replaced by pdfTEX and other software that produces PDF output which can be processed by a wide range of devices (the *P* in PDF stands for *portable*, after all).

With scalable fonts and scalable inline graphics produced by software packages like Ti*k*Z or METAPOST, still everything is fine. Problems arise when raster graphics are embedded into a PDF file. pdfTEX includes raster graphics at their natural size. Especially with the megapixel mania of digital cameras this leads to bloated files. Download time and processing effort at the printing device increase; perhaps your printer will give up with an out-of-memory error for rendering a stamp-sized photograph.

Rescaling bitmap images to a size that suffices for usual post-processing help keep file size and processing complexity small while retaining expected image quality. The most important question is: what resolution will suffice? There are two answers: With pure black-and-white pictures ('line art'), the printing devices' native resolution (e.g. 1200 dpi) is fine. With grayscale or color pictures, a resolution of 1/4 of that will easily do. The reason is that colored 'pixels' are formed by combining several dots.[1] For 16 distinct gray tones, one 'pixel' consists in theory of a 4×4 matrix of black or white pixels. In practice, the real resolution may be even coarser, as effects like bleeding of ink or surface roughness of paper have also to be considered. With professional printing equipment, true output resolution is measured in lpi (lines per inch) and grayscale or color images scaled to their dpi equal to the printing devices' lpi should be fine. If you read this article in the print version of *TUGboat*, figure 1 gives you the chance to see what professional printing equipment can achieve on plain paper. Printing this page on your own device will probably be even more sobering: On my 1200 dpi laser printer, I can't see any difference among the images. With special paper and techniques like duotone printing, there is room at the top.

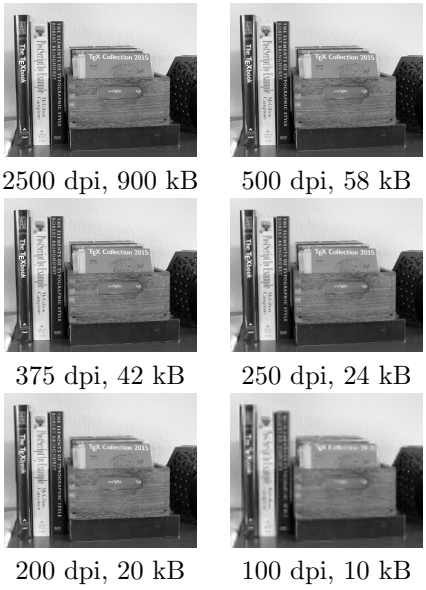

Figure 1: The same photograph in several resolutions. See for yourself where quality degradation becomes perceptible.

For online viewing, one pixel of an image corresponds to one pixel on screen. If you zoom in even more pixels may be needed. Considering high-res devices and moderate zooming, 300 dpi will probably suffice for the next few years.

2 Looking outside the box

In the early times of desktop publishing, when disk space was costly and memory size and local network bandwidth were seriously limited, layout designers often worked with low-resolution preview pictures and final images would be inserted on the way to the imagesetter. There even existed a standard called *open prepress interface* (OPI) [1] for automatic image replacement in PostScript files.

Adobe software (at least InDesign and Distiller) will rescale images during PDF production to an appropriate resolution chosen for the intended target you choose (e.g. print or web). OpenOffice and LibreOffice leave images untouched, while Microsoft Word treats pictures, without any chance to intervene, in a way that causes considerable grief to people in printing offices.

[1] It does not matter if the device does halftoning by rastering or dithering. Special printing devices which are capable of producing ink drops of varying size or using thermal sublimation are not covered by these thoughts.

3 Including pictures for different output devices into a single PDF file

Probably for purposes similar to OPI, the PDF specification allows the inclusion of different images for viewing on screen and printing. Unfortunately, there are several restrictions and drawbacks:

- both images have to have the same dimensions
- both images should have the same resolution
- both images will be included within the PDF file, so file size gets bloated.
- Software support is rare.

Alternative print images are enabled as an experimental feature in `pdftex.def` and can be used with the `graphicx` package out of the box. You just say

```
\includegraphics
    [print=imgPrint.jpg,...]{imgView}
```

instead of

```
\includegraphics[...]{myimgView}
```

and you're done.

It only works with bitmap graphics and you have to specify the full filename. Another drawback is that image reuse does not work with this option for screen images, so your file gets bloated even more. With Adobe Acrobat, I have been able to use it, but most other PDF processing software fails. On my GNU/Linux system, I ended up with a file I could view but not print. Considering all this, I can unfortunately see no good use for this technology apart from playing pranks.

4 Ways of attack

There are three possible hooks to scale images: before the pdfTEX run; while pdfTEX is processing pictures; and as a postprocessor on the finished PDF.

Googling for the problems aforementioned, you will find a postprocessing solution using Ghostscript on the final PDF file [2] and another one using a Python tool [3].

There also exists a LATEX package and corresponding ConTEXt module, both called `degrade` [4], which shrink image files on the fly using ImageMagick in the background. Both of these packages require a Unix-ish operating system and `\write18` has to be enabled.

Beyond downscaling, there are other ways of getting smaller image files. For one, increasing JPEG compression allows drastic reductions in space. This can be done when there will be no further image processing involved, but it requires careful visual checking for compression artifacts.

Also, color depth can be decreased (by "posterization" or grayscale conversion). The author believes that this technique is best carried out with interactive software and visual checks for the results. Unfortunately, reducing color depth does not yield large gains in space,[2] but it might be useful to do gray-scale conversion for material to be printed in black and white to get fine control over the results. Gamma correction and adjustment of black and white levels are often helpful to get better printing results, but for file size there is no benefit.

5 Proposal for a perfect solution

A presumably perfect solution would scale pictures on the fly while producing PDF, ideally triggered by a command like `\pdfFinalResolution=300` or `\destination=web` in the document preamble. This command would probably be supported by some bookkeeping to avoid unnecessary computations on already downscaled images. If black-and-white output was intended, all images might be converted to grayscale, also keyword-driven. Of course, when images were to be clipped, only the visible part of these images would be included.

I guess that this could be done with LuaTEX almost out of the box, and as it has now (Sept. 2016) reached a stable state, there should be no obstacles to implementing it.

6 Implementation (less than perfect) and usage

I have resorted to external software that reads a TEX log file to scan for filenames of images and required target resolutions and then builds up scaled images. As you can specify paths for graphic inclusion with the `graphicx` LATEX package, you get a comparatively easy solution if you adopt to some conventions. You should avoid giving path names on individual `\includegraphics` commands and instead use the `\graphicspath` directive. In a first run you will comment the path to the final images out, having generated the downscaled pictures you will comment the original file path out.

```
\graphicspath{{my/hires/images/}}
%\graphicspath{{printimg/}}
% move comment up for final pdfLaTeX run
```

You will have to repeat this procedure as you change image sizes or as you add new images, so it is probably best to start generating scaled pictures when your document is almost done. Really fast previewing can — as you probably know — be done by specifying the `draft` option to the graphics package,

[2] With the example picture, only ten percent reduction of disk space was achieved by grayscale conversion.

where you get only frames instead of pictures in your PDF file.

So, your workflow will look like this:

- Run LaTeX on your file, with `\graphicspath` pointing to the original files.
- Run `pdflatexpicscale` on your LaTeX project.
- Run LaTeX on your file, with `\graphicspath` pointing to your optimized files.
- Repeat if you change picture sizes, add new pictures, or choose a different target resolution.

If you cannot produce PDF files directly, the only change to the workflow will be that you have to additionally call your PDF producing software.

`pdflatexpicscale`[3] is a Perl script. It depends on some standard Perl packages and the presence of ImageMagick software. As these prerequisites are quite common, it should run with your system. You call the script with the name of your LaTeX project and optionally with the desired resolution and desired picture directory. If you omit arguments, reasonable defaults will be assumed. So a typical call would be:

```
pdflatexpicscale --printdpi=200 \
    --destdir=medrespics myarticle
```

If your LaTeX file is called `myarticle.tex`, you have done a LaTeX run, so that the log file exists, you want 200 dpi output and the directory for the scaled pictures is an existing subdirectory of the current directory called `medrespics`, then you may copy the above command verbatim.

The software can be downloaded from CTAN [5], and is included in TeX Live. Documentation is included. You may probably want to read it, as it is not identical with this article.

7 Caveats, limitations and drawbacks

My PostScript printer prints some black-and-white images inverted. I could have inverted them with an image processor, but then they would look wrong on screen. As a workaround I converted them to grayscale. Some provision needs to be made to keep `pdflatexpicscale` from scaling them down like other halftone images. The easiest way is keep them in a separate directory and to include this directory at the beginning of the `\graphicspath` list.

The target resolution you choose may not truly meet the printing devices' needs, especially if you do not know who will print your document. Perhaps the printing device has got fantastic image scaling software that you replace by some inferior software on your computer. Also you probably will not want to recompile all of your documents just because you bought a new printer.

`pdflatexpicscale` changes image size and target resolution. This has serious consequences if you intend to use clipping, or to display pictures in a size dependent on their resolution. Also, anisotropic scaling is not supported.

When a file gets used several times at different sizes, only the largest will be included. The software reads the log file from beginning to the end and starts rendering immediately, so when an image is included at first in thumbnail size and then larger, it will be rendered several times.

The Perl script uses ImageMagick's `convert` software for scaling pictures, so quality of resampling and file compression (most important for lossy compression formats like JPEG) depend upon ImageMagick's algorithms.

Security concerns: ImageMagick has had several security flaws fixed in 2016. So it is probably not a good idea to provide scaling services to anonymous users that might upload a malicious image file.

The solution presented only deals with pure raster graphics (JPEG and PNG). If you include graphics in a mixed format like PDF, rasterization might be beneficial or disadvantageous, depending on the content. Rastering vector graphics is definitely not what you want. Treating your PDF with one of the postprocessing solutions mentioned might help.

A last remark: Having two projects share the same images is a recipe for dissatisfaction. It is quite common to keep, for instance, a presentation and the corresponding handout in the same folder, but graphic requirements are totally different. The best solution is to keep downscaled pictures in separate directories; `pdflatexpicscale` can easily cope with this.

References

[1] `http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/postscript/pdfs/5660.OPI_2.0.pdf`

[2] `http://tex.stackexchange.com/questions/14429/pdftex-reduce-pdf-size-reduce-image-quality`

[3] `http://tex.stackexchange.com/questions/2198/how-to-create-small-pdf-files-for-the-internet`

[4] `http://ctan.org/pkg/degrade`

[5] `http://ctan.org/pkg/pdflatexpicscale`

◇ Peter Willadt
`willadt (at) t-online dot de`

[3] This name was chosen because Google found no hits in July, 2016.