Zpravodaj Československého sdružení uživatelů TeXu

Timothy Eyre

PDFdiff: A PDF File Comparison Script

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 20 (2010), No. 3, 208–214

Persistent URL: http://dml.cz/dmlcz/150122

PDFdiff: skript srovnávající PDF soubory

Timothy Eyre

Abstrakt

Článek představuje skript naprogramovaný v Pythonu, který na vstupu očekává dva PDF soubory a automaticky je zpracuje nástroji pdftk, Ghostscript, ImageMagick a XeTeX tak, že vznikne nový PDF soubor, který ukazuje rozdíly mezi oběma vstupy.

Klíčová slova: Srovnání souborů, Python, Ghostscript, pdftk, ImageMagick.

doi: 10.5300/2010-3/208

Introduction

Tools that show the difference between two files are a mainstay of software development. They are most commonly used to view the difference between two versions of a source code file; you might even use such a tool when you are developing TeX code. Less common, but still potentially useful, are tools that show the difference between two binary files.

This article describes the motivation for and implementation of a tool that shows the differences between two PDF files.

1. Motivation

Here are three examples where a PDF file difference-finding tool would be useful:

(1) You have two versions of a paper in PDF format. You know that the differences between them are slight but you need to know exactly what the differences are.

(2) You have created a document using the Kanji Stroke Order Font [1] and want to create an equivalent version but without the stroke order numbers. A colleague tells you that the Choumei font (also available from [1]) is simply the Kanji Stroke Order Font with the stroke numbers removed. You suspect that the Choumei font may have different metrics to the Kanji Stroke Order Font and will therefore change the layout of your document. You need a way to test this.

(3) You have re-written a TeX macro used by a large publication to make it more maintainable or to add a new feature. You are fairly sure that the new version of the macro will not change the publication's layout in any way but need a way to verify this.

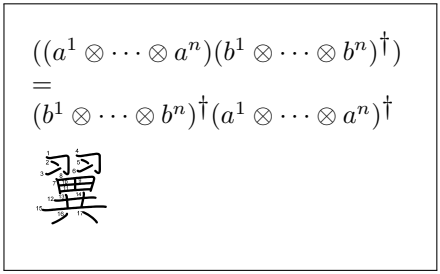

Figure 1: Pre-modification version of the PDF file

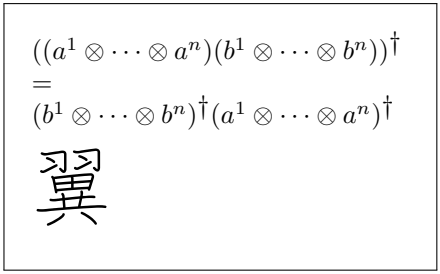

Figure 2: Post-modification version of the PDF file

Figures 1 and 2 illustrate the first two of these three scenarios. The change to the mathematical expression between Figures 1 and 2 is easy to see, but such changes are harder to spot in a a larger article. It is not obvious whether the change from the Kanji Stroke Order Font to the Choumei font would change the spacing of the document.

In both cases, the output of PDFdiff shown in Figure 3 makes the changes easy to see.

2. Implementation

I implemented PDFdiff using Python to stitch together the steps summarized in Figure 4.

This section describes each of these steps in turn. I use the words 'pre' and 'post' to denote the versions of a document before and after some sort of modification that we invoke PDFdiff to determine. In practice it may not be the case that there is a well-defined notion of 'pre' and 'post' version but it is a useful shorthand for identifying the two documents.

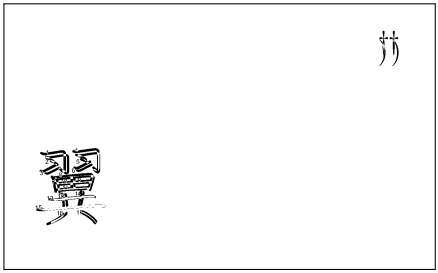

Figure 3: Difference between the two PDF files

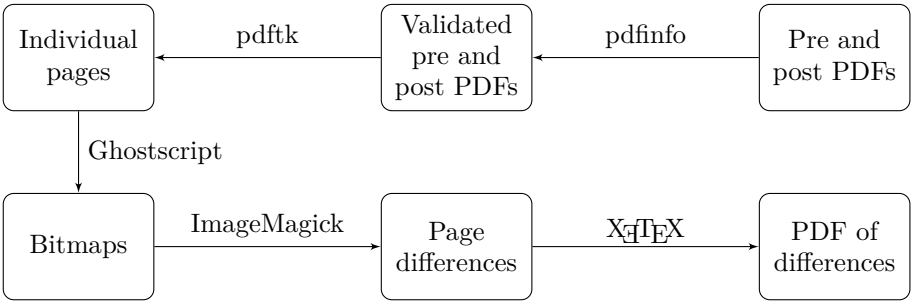

Figure 4: PDFdiff workflow

2.1. Validation

To be able to compare the pre and post versions of the document sensibly, the two versions of the document must have the same dimensions. Therefore the first thing that PDFdiff does is to call out to the tool `pdfinfo` [2] to extract the dimensions of the document under scrutiny.

If the dimensions of the pre and post versions of the document differ then PDFdiff reports this and exits.

2.2. Burst

To simplify the process of generating the bitmaps with Ghostscript, PDFdiff first bursts the input PDF files into multiple PDF files, each containing a single page of either the pre or post version of the document. PDFdiff does this by calling out to the tool `pdftk` [3] as follows:

```
pdftk pre.pdf burst
pdftk post.pdf burst
```

After each call to pdftk, PDFdiff renames the files. This is necessary because pdftk chooses its own output filenames and so we need to prevent the output of the second burst from overwriting the output of the first.

PDFdiff then checks that the pre and post versions of the document have the same number of pages. If the two documents differ in length then PDFdiff will inform you of the fact. If you are then determined to compare the two versions of your document regardless of their differing number of pages then it is a simple matter to concatenate a few extra pages to the end of the shorter document to even up the lengths.

At the end of this burst process, we should have $2n$ new PDF files, where n is the number of pages in the document under scrutiny.

2.3. Pages to Bitmaps

PDFdiff is now ready to convert the pages it has extracted into bitmap files. It does this using Ghostscript [4], as follows:

```
gswin32 -dNOPAUSE -r600 -dBATCH -sDEVICE=bmp256
                                    -sOutputFile=xxx.bmp xxx.pdf
```

Here `xxx` denotes the name of a single-page PDF file that PDFdiff produced at the burst stage. `-dNOPAUSE` tells Ghostscript that it should not wait for user input. `-r600` tells Ghostscript to use a resolution of 600dpi, which I judge to be a reasonable compromise between quality and resource use. `-dBATCH` tells Ghostscript to exit after processing the supplied PDF file rather than leaving the Ghostscript environment open. `-sDEVICE=bmp256` tells PDFdiff to write an 8-bit colour bitmap file. This allows some degree of colour to be preserved in the final difference file.

2.4. Subtract Bitmaps

Having called out to Ghostscript $2n$ times to produce $2n$ bitmaps, PDFdiff calls out to the ImageMagick software suite [5] to subtract the bitmaps from one another, as follows:

```
composite -compose difference pre_i.bmp post_i.bmp diff_i.bmp
```

In this command, `pre_i.bmp` denotes the bitmap generated from page i of `pre.pdf`, `post_i.bmp` denotes the bitmap generated from page i of `post.pdf` and `diff_i.bmp` represents the file generated by the `composite` command of Imagemagick that shows the difference between the two input bitmap files. In all three cases, $1 \leq i \leq n$.

However, the `diff_i.bmp` files produced by `composite` are not ideal. They are large and the text (if any) will appear as white on a black background. To solve both these problems, PDFdiff runs the bitmap files through another ImageMagick command, as follows:

```
convert -negate diff_i.bmp diff_i.png
```

This produces a much smaller PNG version of the bitmap file and the `-negate` option reverses the colours so that we have black on a white background again.

The ImageMagick processing is the most time-consuming part of the workflow, typically requiring around two-thirds of the total time taken to create the difference file.

2.5. Regenerate Document

Now we have n `.png` files, it is a simple matter to create a new PDF file of n pages with each page containing a single bitmap file as a graphics inclusion. I chose to use XeTeX for this purpose simply because that is the flavour of TeX I use most of the time rather than because I needed a specific feature of XeTeX.

To achieve this, PDFdiff generates some TeX source of the following form:

```
\pdfpagewidth=155pt\pdfpageheight=93pt
\hsize=155pt\vsize=93pt\voffset=-1in\hoffset=-1in
\hbox{\XeTeXpicfile "diff_i.png" width 155pt height 93pt}
\vfill\eject
\bye
```

Naturally, there is an `\XeTeXpicfile` inclusion for each $i \in \{1, \ldots, n\}$.

PDFdiff then calls out to XeTeX with the auto-generated TeX source file as the parameter, as follows:

```
xetex diff.tex
```

Once XeTeX completes its processing, we have the desired final PDF file that shows the differences between the pre and post versions of the document.

3. Deficiencies

There are four deficiencies that users may notice with this script: speed, size of output, resolution and input parameters.

3.1. Speed

To run the script to compare the 8 pages of this paper takes 215 seconds on my laptop. It would be nice if the script ran faster. Testing shows that most of the time is taken at the ImageMagick stage, so there is not much scope for increasing the speed.

3.2. Size of Output

The original version of this document is about 280 kB in size. Running PDFdiff over two versions of this document produced an output file 2 MB in size. In general,

the output file produced by PDFdiff is much larger than the input files. This is simply because the page is described as a raster graphic rather than using the font mechanism provided by the PDF standard. Given the size of modern disks this is only likely to be a problem when sending the output of a large difference over a communications link.

3.3. Resolution

PDFdiff relies on raster graphics to create the difference output. Therefore, by definition, the resolution of the output must be finite, unlike the smooth lines of the original vector graphics. Of course, if you are using the old TEX `.pk` files you will be using rasters anyway.

It is easy enough to modify PDFdiff to produce output of arbitarily high resolution (subject to the limitations of Ghostscript and Imagemagick) but this is inevitably at the expense of longer run times and larger output files.

3.4. Input Parameters

Purely out of laziness, I have written the script to assume that the 'pre' version of the PDF document is called `pre.pdf` and the 'post' version of the document is called `post.pdf`. It would be simple to change this but I prefer the discipline of being sure which file is which and the simplifying effect on the Python code.

Conclusion

Using readily-available tools, it is possible to create a workflow to compare PDF documents. Comparing PDF documents is relatively resource-hungry but can be invaluable. Not only is PDFdiff useful for developing TEX documents, TEX and related tools also make up important parts of the workflow.

References

[1] Kanji Stroke Order Font and Choumei Font: http://www.nihilist.org.uk/

[2] pdfinfo: http://www.glyphandcog.com/

[3] pdftk: The PDF Toolkit: http://www.accesspdf.com/pdftk/

[4] Ghostscript: http://pages.cs.wisc.edu/~ghost/

[5] ImageMagick: http://www.imagemagick.org/

Summary: PDFdiff: A PDF File Comparison Script

A Python script can be used to take two PDF files and automatically process them with pdftk, Ghostscript, ImageMagick and XeTeX to produce a PDF file that shows the differences between the two input files.

Keywords: PDF, file difference, Python, Ghostscript, pdftk, ImageMagick.

Timothy Eyre, mail@nihilist.org.uk
CSTUG c/o FEL ČVUT, Technická 2
Prague, CZ-166 27, Czech Republic