The `pdfrack` manual

Marc Boyer
ONERA
France
Marc.Boyer@onera.fr

March 26, 2010

Contents

1 Introduction

What is `pdfrack`? The `pdfrack` is a *hack* that allow to use `psfrag` and `pdflatex`.

The `pdfrack` is not a *complete* solution: it does not handle all LaTeX files, and you will sometimes need to modify your LaTeX source files if you want to use it. But it can help you to save time.

1.1 Why pdfrack?

If you want to generate a `pdf` file from a LaTeX one, you either can do it directly, using `pdflatex`, or by first generating a `postscript`[1] file and converting this `postscript` to a `pdf` file.

The `psfrag` package allow the user to replace some text in a `postscript` figure by another LaTeX text. To use `psfrag`, you must use a filter from `dvi` to `postscript`. So, if you compile with `pdflatex`, the text substutions are lost. Nevertheless, you sometimes want to use `pdflatex`, to be able to add hyperlinks in your `pdf`, or because your `postscript` to `pdf` filter produces a ugly text.

1.2 Basic idea

The basic idea of `pdfrack` is, from your LaTeX source file, to produce the figures in `pdf` format, *with the `psfrag` replacements*. Then, you can compile with `pdflatex`, including the `pdf` figures.

2 Using it

2.1 Installation

Put the script `pdfrack.sh` in a directory included in your path, and `pdfrack.sty` somewhere where TeXwill find it[2].

Moreover, the `pdfrack.sh` script uses a lot of other scripts and software. (`cut, dvips, epstopdf, grep, head, latex, ps2ps, ps2epsi, sort, tail`, and, first of all, a bourne shell). They all are installed with common Unix/Linux distribution, so, you should not have to care about it.

2.2 A very short tutorial

MAKE A BACKUP OF YOUR FILES BEFORE USING THIS SCRIPTS !

Assuming you have a LaTeX file, with figures included with

```
\begin{figure}[htbp]
  \centering
  \psfrag{Fp(x)}{$\mathcal{F}(x)$}
  \psfrag{Gp(x)}{$\mathcal{G}(x)$}
  \includegraphics[width=\textwidth]{BasicIdea}
  \caption{The basic idea of \pdfrack}
  \label{fig:BasicIdea}
\end{figure}
```

you simply have to:

[1] With LaTeX; you generate a `dvi` file, and, with some filter, like `dvips`, you get a `postscript` one.

[2] The variable `TEXINPUTS` defines where your TeXtool looks for inputs.

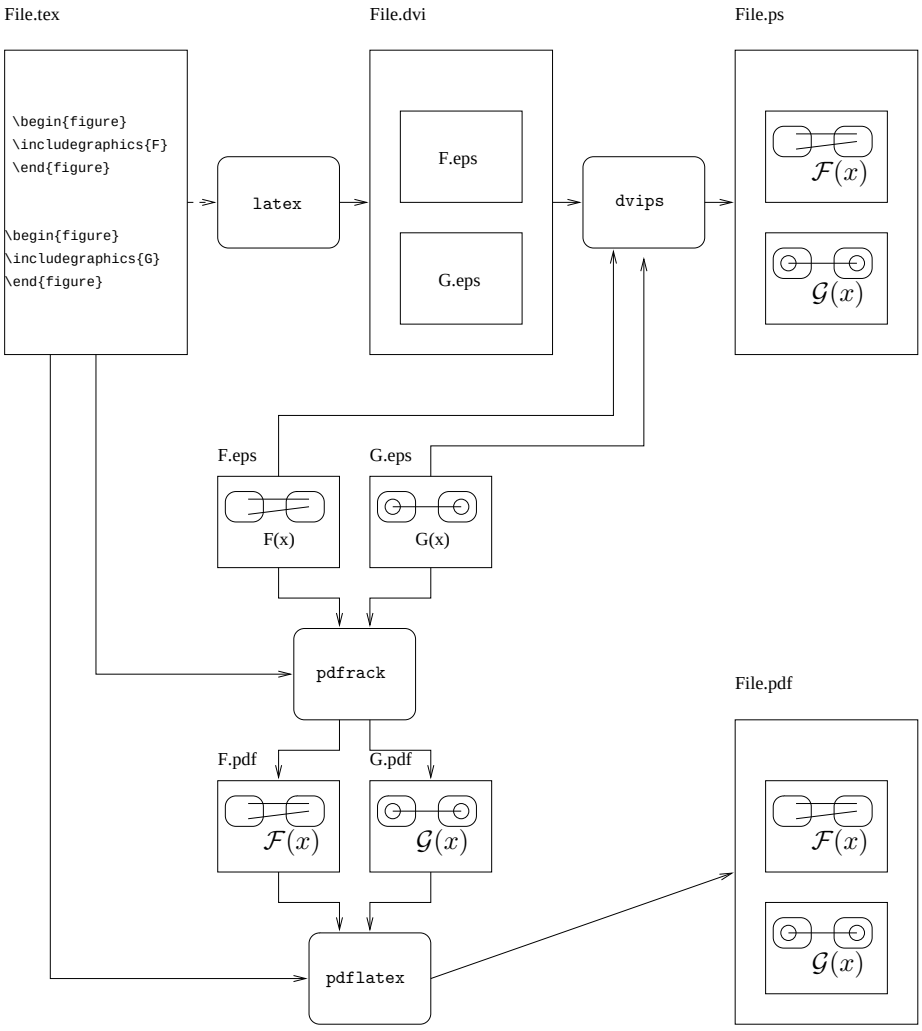

Figure 1: The basic idea of `pdfrack`

- include the `pdfrack` package `\usepackage{pdfrack}`
- replace the `\includegraphics` commands by `\pdfrackincludegraphics` (or use the `-a` option)

Then, just run the command `pdfrack.sh` on your file. It should produce some output like:

```
-------------------------------------------
  ---> Converting figure BasicIdea
Using LaTeX: producting dvi
This is e-TeX, Version 3.14159-2.1 (Web2C 7.4.5)
entering extended mode
Method 2: Trying conversion with dvips -|->  ps2eps -|-> eps2eps -|-> epstopdf
Converting dvi -> ps (dvips)
Converting ps -> eps (ps2eps)
     Cleaning eps with eps2eps filter
Converting eps -> pdf (epstopdf)
Removing temporary files
```

for each figure, and generates a standalone `pdf` figure *with the psfrag replacements*.

Then, you can run `pdflatex` on your file.

Often, the bounding box of the generated file is not very good. Try different values for the `-m` options, and try to use or no the `-p` and `-e` options. If offers you 16 possibilities. If none work, your are unlucky.

Semé also the `README` file to have more details.

2.3 Limitations

The `pdfrack` package is based on a bourne-shell script that parse the file to find the `\begin{figure}` and `\end{figure}` tags. So, you must use this to way of including figure (or change the script).

Note that the `%` is is general seen as a comment, even if prefixed with \. So, avoid to use it.

If your LaTeX code is too complex, then, `pdfrack` will ne be able to handle it. Then, you should write a simpler LaTeXfile, with your figure(s) and use this file to generate the pdf figures.

2.4 Options

2.4.1 Summary

-h help

-m method: integer in 0..3 (default is 2) the number of the method used to convert `dvi` to `pdf` if one method fails, try another, and add -p and/or -e options `:-)`

-p filter each postscript file with `ps2ps`

-e filter each encapsulated postscript file with `eps2eps`

-a auto, translate all figures, avoid the use of `pdfrackinclude`

-i force use of `ps2epsi` instead of `ps2eps`

-k keep (all tempory files are keeped, usefull for debug)

-H own header file (default is to extract from file.tex file)

-M master file (to handle sub files)

2.4.2 Filter options: `-m, -p, -e, -i`

The main source of troubles is the generation of a `pdf` from the `dvi` of `postscript` file, and especially the size of the "Bounding box". If the default method fails, you should try some values of the `-m` option.

You also can add the `-p` and `-e` options. In theory, they are useless, since `-p` force a `postscript` to `postscript` translation, and since `-e` does the same with encapsulated postscript. But in practice, it may help.

In general, the filter `ps2eps` does a better job than `ps2epsi`. Then, by default, the script uses it. The option `-i` is there to force the use of `ps2epsi`.

2.4.3 Auto option

You can avoid the use of the command `pdfrackincludegraphics`, with the option `-a`. It will generate a `pdf` figure for all figures included with `includegraphics`, even those without any placement.

2.4.4 Other header option: `-H`

To generate the `pdf` file with the placements, the script use the header of your LaTeX file. But, it may be insufficient: by example, your placement can use some command defined neither in your header, neither in the `figure` environment. Then, you can write another header that will be used to generate the `pdf` figures.

2.4.5 Handling subfiles: `-M`

If your master file made some input of some other LaTeX code, with `\input`, `\include` or any other command[3], it is not parsed by the `pdfrack` script.

You have to explicitly call the `pdfrack` script on each sub file, with the option `-M` that specify the name of the master file, or the option `-H` to specify another header file.

[3] Like the `\Input` of the `srcltx` package.

3 How does it works?

The core of `pdfrack` is a `pdfrack.sh` script that try, for each figure in your LaTeX source, to produce a small LaTeX file with only the figure. Then, this file is compiled with LaTeX, converted into `postscript`, and then, we have a `postscript` figure, with the replacements.

Now, the system should convert a `postscript` file to a `pdf` one, with the right bounding box... I am not at all a guru of `postscript` (neither `pdf`), so, I try to use some tools like `ps2ps`, `ps2eps`, `ps2pdf` and so on. There is an option in the script `-m` that allow the user to chose one or the other.

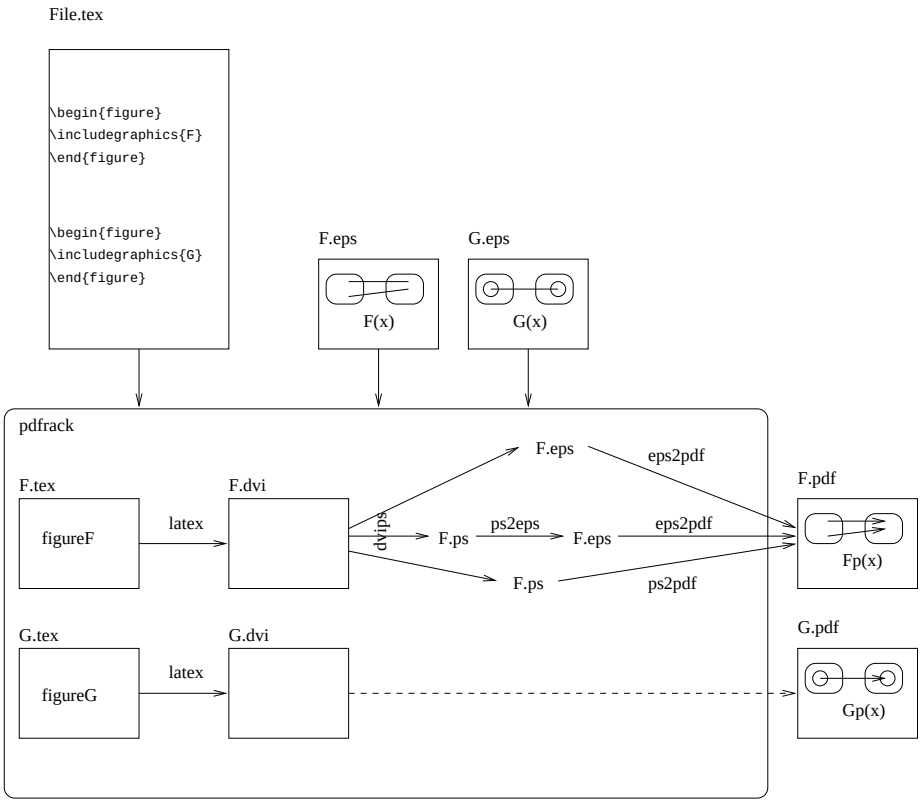

Figure 2: Inside `pdfrack`

4 Knowns bugs and limitations

1. **Only a subpart of my figure appears** The main problem is the computation of the "Bounding box" of the postscript figure. If the default method fails, try the `-m` option, with different values. Adding the `-e` may also help. See paragraph 2.4.2.

2. **The figure is alone on a full page. Its seems to big.**
   See item 1

3. **I am a Windows user**
   Neither am I[4]... This solution is based on some well known Unix command-line tools, like the bourne shell, `ps2eps, cut, tail, grep`... To use my script on windows, you should install a solution like Cygwin.

4. `I uses two times the same file, with different replacements, but only one replacement is made.`
   Yes... For each `postscript` file, a `pdf` file is generated, with the same name. Then, if the same file is used two times, with two different placements, the same name is used twice, and only one `pdf` file rests. The simpler solution is to copy the `postscript` file, or to use some link capacity (like `ln`).

5. `The figures in included files (with \include, \input are not taken into account`
   See paragraph 2.4.5.

6. **My problem is not in the known bugs list !**
   Just email me.

5 Why this name ?

I think a good solution for using `pdflatex` and `psfrag` should be named `pdfrag` or `pdffrag`. I do not think that my solution is really good. It is just a hack. Then `pdfrack` comes from:

```
  pdf
+ psfrag
+    hack
---------
  pdfrack
```

6 Other solutions

There are, of course, other solutions:

DrawAt Matthijs Douze (another member from ENSEEIHT) has developped DrawAt, a solution for MacOS X only. `http://www.enseeiht.fr/~douze/drawat/index.html`

unpsfrag Félix Valado Pomarinho has developped a perl script: `unpsfrag`

[4]This is not always true: when I am forced to write Word document, I use a Windows machine thought an `rdesktop` connection.

fragmaster Tilman Vogel developped `fragmaster`, another perl-oriented solution. `http://www.tat.physik.uni-tuebingen.de/~vogel/fragmaster`

pstoedit and [X]fig With `[X]fig`, you can already make a figure and, setting the special flag to a text zone, you already can put on your figures some LaTeX code, and export in the combined mode either in postcript or pdf.

In the combined Postscript/LaTeX, if your file is named `figure.fig`, it creates a `figure.pstex` file which is the postcript version of your fig figure *without the special-tagged texts*, and a `figure.pstex_t` file which juste include the `figure.pstex` (with the `\includegraphics` command) *and* adds the LaTeX text at the right place.

In the combined Pdf/LaTeX, this is the same except that `figure.pstex` is named `figure.pdf` and `figure.pstex_t` is named `figure.pdf_t` and includes `figure.pdf`.

If you like to avoid the graphic interface, this ca be done in command line with `fig2dev`.

```
fig2dev -L pstex_t figure.fig figure.pstex_t
fig2dev -L pstex figure.fig figure.pstex
```

If you do not have a fig figure, you can transform a postcript file into a fig one with `pstoedit` with command line like:

```
pstoedit -dis -f fig example.eps > example.fig
```

figfrag With `figfrag` (`http://www.ctan.org/tex-archive/graphics/figfrag/` you can use the psfrag feature in your fig picture and create a standalone eps figure with the psfrag replacements.