

# The auto-pst-pdf package

Will Robertson & Johannes Große

wspr 81 at gmail dot com

2009/04/26    v0.6

## 1 Future plans

This package is no longer being actively developed (although I'm happy to add small features and fix bugs). I (Will Robertson) first wrote auto-pst-pdf to aid the use of psfrag in pdf $\LaTeX$  documents. The newer pstool package does this in a more efficient and convenient manner, and I suggest using that package instead if that's what you're using this package for. However, pstool does not yet support full pst-pdf usage, so auto-pst-pdf is certainly not deprecated yet.

## 2 Basic usage

This package provides a wrapper around pst-pdf to automatically accomodate for typesetting either with DVI or PDF output. With default package option [on], typesetting under pdf $\LaTeX$  will automatically initiate an auxiliary compilation of  $\LaTeX \rightarrow \text{dvips} \rightarrow \text{ps2pdf} \rightarrow \text{pdfcrop}$  to generate the required PDF figures for the document.

After this has been done and the figures no longer need to be re-generated, the package can be given the [off] option to save compilation time:

```
\usepackage[off]{auto-pst-pdf}
```

If the extension of your  $\LaTeX$  document is not .tex, then it must be declared when the package is loaded (*e.g.*, I like to use .ltx to distinguish between Plain  $\TeX$  and  $\LaTeX$  files):

```
\usepackage[ext=ltx]{auto-pst-pdf}
```

## 3 Requirements

pdf $\TeX$  must be called with the -shell-escape option. Requires the following packages: ifplatform, pst-pdf, xkeyval.

Heiko Oberdiek's `pdfcrop` Perl script<sup>1</sup> must be installed for the default `crop=on` option (see section 5). Under Windows, a Perl installation<sup>2</sup> will also need to be installed even though `pdfcrop` itself is part of MiKTeX.

## 4 Provided macros for including graphics

Macros are provided to easily facilitate figures created by the `MATLAB` package `laprint`<sup>3</sup> and the Mathematica package `MathPSfrag`<sup>4</sup>. Also, a generic `psfrag`<sup>5</sup> wrapper is provided.

<code>\mathfig{&lt;filename&gt;}</code>	insert a Mathematica graphic from <code>MathPSfrag</code> (without <code>-psfrag</code> suffix)
<code>\matlabfig{&lt;filename&gt;}</code>	insert a <code>MATLAB</code> graphic from <code>laprint</code>
<code>\psfragfig{&lt;filename&gt;}</code>	insert an <code>EPS</code> with <code>psfrag</code>

The above commands all accept an optional argument which is passed to the underlying `\includegraphics` macro.

The `\matlabfig` command meddles slightly with the output of `laprint`; the font sizes in the figure will always be as originally defined. (This is unavoidable I'm afraid.)

For the `\psfragfig` command, `psfrag` statements are input from either or both of the files `<document>-psfrag.tex` and `<filename>-psfrag.tex` if they exist. Furthermore, supplementary `\psfrag` statements can be added in a trailing optional argument:

```
\psfragfig[<graphics options>]{<filename>}[<psfrag statements>]
```

Manual `\psfrags` override those in `<filename>-psfrag.tex` which in turn override those in `<document>-psfrag.tex`.

## 5 Advanced package options

Better results are obtained by using `pdfcrop` in the auxiliary compilation process, and this is used by default. It is not installed by default, however, and will not always be required. Cropping with this tool can be controlled with the `crop` option:

```
\usepackage[crop=off]{auto-pst-pdf}
```

The package automatically deletes the files generated during the auxiliary `LATEX` compilation. Which files are deleted are chosen by passing a list of file extensions

<sup>1</sup><http://www.ctan.org/tex-archive/support/pdfcrop/>

<sup>2</sup>Freely available: <http://www.activestate.com/Products/activeperl/index.plex>

<sup>3</sup><http://www.uni-kassel.de/fb16/rat/matlab/laprint/>

<sup>4</sup><http://wwwth.mppmu.mpg.de/members/jgrosse/mathpsfrag/>

<sup>5</sup><http://www.ctan.org/tex-archive/help/Catalogue/entries/psfrag.html>

to the `cleanup` option (no error message or warning is produced if a file is specified that does not exist). The default list is:

```
\usepackage[cleanup={log,aux,dvi,ps,pdf}]{auto-pst-pdf}
```

If you're using cross-references of any kind within the graphics being processed by `pst-pdf`, it will be necessary to perform the auxiliary compilation more than once to resolve them. The exact number will vary by exact application, and must be set explicitly:

```
\usepackage[runs=2]{auto-pst-pdf}
```

The options passed individually to `latex`, `dvips`, `ps2pdf`, and `pdfcrop` in the auxiliary compilation process may all be customised, if you know what you're doing. The defaults for the latter three are

```
\usepackage[dvips={-o -Ppdf},  
             pspdf={-dAutoRotatePages=/None}},  
             pdfcrop={}]{auto-pst-pdf}
```

The  $\LaTeX$  auxiliary compilation has some hard-coded options (see the source if you're interested), and further options can be appended if you wish. For example, to run the auxiliary compilation with more information written to the console, use the following package option:

```
\usepackage[latex={-interaction=nonstopmode}]{auto-pst-pdf}
```

Any package options that are not recognised are passed on to `pst-pdf`. As an example,

```
\usepackage[final]{auto-pst-pdf}
```

will load `pst-pdf` with the `final` package option, possibly overriding a global `draft` option from the class loading.

## 6 Acknowledgements

Many thanks to the authors of `pst-pdf`, `psfrag`, `laprint`, `MathPSfrag`, and `pdfcrop`. This package could not exist without their combined efforts over many years. Finally, Gernot HASENFPLUG deserves special mention for extensive testing, feature suggestions, and moral support :) Thanks, mate.

## File I

# auto-pst-pdf implementation

## 7 Setup code

This is the package.

```
1 \ProvidesPackage{auto-pst-pdf}[2009/04/26 v0.6 Wrapper for pst-pdf]
```

### Change History

```
vo.3
  General: Too many changes to list. Command execution totally re-written. 4
vo.4
  General: Johannes tinkered with the code. Will will improve. :-)      4
  Will sorted it all out.                                                4
vo.5
  General: delay option removed. Keep it simple, stupid!                 4
  Removed \ifdefined to avoid e-TeX.                                     8
  Removed mucking about with image extensions.                          8
  \app@convert: Fix PackageError (should have been a warning).          6
  \matlabfig: Redefine \resizebox for lprint.                             9
  \psfragfig: Extend \psfragfig to accept arbitrary input for extra \psfrag
  commands.                                                                10
vo.6
  General: runs option added (thanks Joseph!)                             4
  Pass unknown options to pst-pdf.                                       4
```

**Required packages** pst-pdf is loaded later on.

```
2 \RequirePackage{ifpdf,xkeyval,ifplatform}
```

### Things we need

```
3 \newif\if@app@off@
4 \newif\if@app@crop@
5 \newcounter{app@runs}
6 \def\app@sufffix{autopp}
7 \edef\app@jobname{\jobname-\app@sufffix}
8 \edef\app@pics{\jobname-pics.pdf}
```

## Option processing

```
9 \DeclareOptionX{off}[]{\@app@off@true}
10 \define@choicekey{auto-pst-pdf.sty}{crop}[\@tempa\@tempb]{on,off}{%
11 \ifcase\@tempb\relax
12 \@app@crop@true
13 \or
14 \@app@crop@false
15 \fi}
16 \DeclareOptionX{on}[]{\@app@off@false}
17 \DeclareOptionX{ext}{\def\app@ext{#1}}
18 \DeclareOptionX{latex}{%
19 \def\app@latex@opts{%
20 \ifwindows
21 -disable-write18
22 \else
23 -no-shell-escape
24 \fi
25 -jobname="\app@jobname"
26 -interaction=batchmode
27 #1}}
28 \DeclareOptionX{dvips}{\def\app@dvips@opts{#1}}
29 \DeclareOptionX{pspdf}{\def\app@pspdf@opts{#1}}
30 \DeclareOptionX{pdfcrop}{\def\app@pdfcrop@opts{#1}}
31
32 \DeclareOptionX{cleanup}{%
33 \let\app@rm@files\@empty
34 \@for\@ii:=#1\do{%
35 \edef\app@rm@files{\app@rm@files,\app@jobname.\@ii}}
36
37 \DeclareOptionX{runs}{%
38 \setcounter{app@runs}{#1}% support calc
39 \ifnum\c@app@runs > \z@
40 \else
41 \app@PackageWarning{The number of runs must be at least one.}%
42 \c@app@runs\@ne
43 \fi}
44
45 \DeclareOptionX*{\PassOptionsToPackage{\CurrentOption}{pst-pdf}}
46
47 \ExecuteOptionsX{%
48 ext=tex,
49 crop=on,
50 latex={},
51 dvips={-Ppdf},
52 pdfcrop={},
```

```

53     cleanup={log,aux,dvi,ps,pdf},
54     runs=1
55 }
56 \ifwindows
57   \ExecuteOptionsX{pspdf={}}
58 \else
59   \ExecuteOptionsX{pspdf={-dAutoRotatePages=/None}}
60 \fi
61 \ProcessOptionsX

```

### Shorthands

```

62 \def\app@exe{\immediate\write18}
63 \def\app@nl{^^J\space\space\space\space}
64 \newcommand\app@PackageError[2]{%
65   \PackageError{auto-pst-pdf}{\app@nl #1^^J}{#2}}
66 \newcommand\app@PackageWarning[1]{%
67   \PackageWarning{auto-pst-pdf}{\app@nl #1^^JThis warning occurred}}
68 \newcommand\app@PackageInfo[1]{\PackageInfo{auto-pst-pdf}{#1}}

```

These are cute:

```

69 \newcommand\OnlyIfFileExists[2]{\IfFileExists{#1}{#2}{}}
70 \newcommand\NotIfFileExists[2]{\IfFileExists{#1}{}{#2}}

```

`\app@convert` #1 : command name  
#2 : source file  
#3 : destination file

Check if the source file exists and calls the command to generate the destination file. If the final file is not created, generate an error.

```

71 \def\app@convert#1#2#3{%
72   \OnlyIfFileExists{#2}{%
73     \app@exe{\csname app@cmd@#1\endcsname{#2}{#3}}%
74     \NotIfFileExists{#3}{\app@PackageWarning{Creation of #3 failed.}}}}

```

`\app@compile` First we define the entire latex → dvips → ps2pdf (→ pdfcrop) command sequence. The actual call to the compilation macro follows thereafter. This macro contains the actual creation of the pdf container. Each processing step is in a separate macro to allow simple modification.

```

75 \def\app@compile{%
76   \app@cleanup
77   \app@remove@container
78   \loop\ifnum\c@app@runs > \@ne
79     \app@convert{extralates}{\jobname.\app@ext}{\app@jobname.dvi}%
80     \advance\c@app@runs\m@ne
81   \repeat

```

```

82 \app@convert{latex}{\jobname.\app@ext}{\app@jobname.dvi}%
83 \app@convert{dvips}{\app@jobname.dvi}{\app@jobname.ps}%
84 \if@app@crop@
85   \app@convert{pstopdf}{\app@jobname.ps}{\app@jobname.pdf}%
86   \app@convert{pdfcrop}{\app@jobname.pdf}{\app@pics}%
87 \else
88   \app@convert{pstopdf}{\app@jobname.ps}{\app@pics}%
89 \fi
90 \IfFileExists{\app@pics}
91   {\app@cleanup}
92   {\app@PackageWarning{Could not create \app@pics.
93     Auxiliary files not deleted.}}

```

Command-line program to delete files:

```

94 \edef\app@rm{\ifwindows del \else rm -- \fi}

```

`\app@try@rm` Macro to delete files (comma-separated) if they exist:

```

95 \newcommand\app@try@rm[1]{%
96   \@for\@tempa:=#1\do{%
97     \OnlyIfFileExists{\@tempa}{\app@exe{\app@rm "\@tempa"}}}}

```

Remove pdf picture container:

```

98 \def\app@remove@container{\app@try@rm{\app@pics}}

```

Clean up auxiliary files: (`\app@rm@files` defined by the `cleanup` package option)

```

99 \def\app@cleanup{\app@try@rm{\app@rm@files}}

```

**L**A<sub>T</sub>E<sub>X</sub>:

```

100 \def\app@cmd@latex#1#2{latex \app@latex@opts\space
101   "\unexpanded{\let\APPmakepictures\empty\input} #1"}
102 \def\app@cmd@extralatem#1#2{latex \app@latex@opts\space
103   "\unexpanded{\let\APPmakepictures\undefined\input} #1"}

```

dvips:

```

104 \def\app@cmd@dvips#1#2{dvips \app@dvips@opts\space -o "#2" "#1"}

```

ps2pdf:

```

105 \def\app@cmd@pstopdf#1#2{ps2pdf \app@pspdf@opts\space "#1" "#2"}

```

pdfcrop:

```

106 \def\app@cmd@pdfcrop#1#2{pdfcrop \app@pdfcrop@opts\space "#1" "#2"}

```

## 7.1 Base functionality

For compilation, we use the [notightpage] option of pst-pdf and the pdfcrop Perl script because EPS figures can have elements that extend outside their bounding boxes, and end up with clipped content after ps2pdf. Otherwise the script ps4pdf would be sufficient.

**pdf $\LaTeX$  compilation** Requires supplementary processing with pst-pdf:

```
107 \ifpdf
108   \if@app@off@else
109     \ifshellescape
110       \app@exe{echo " "}
111       \app@exe{echo "-----"}
112       \app@exe{echo "auto-pst-pdf: Auxiliary LaTeX compilation"}
113       \app@exe{echo "-----"}
114       \app@compile
115       \app@exe{echo "-----"}
116       \app@exe{echo "auto-pst-pdf: End auxiliary LaTeX compilation"}
117       \app@exe{echo "-----"}
118     \else
119       \app@PackageError{%
120         "shell escape" (or "write18") is not enabled:\app@nl
121         auto-pst-pdf will not work!}
122       {You need to run LaTeX with the equivalent of
123         "pdflatex -shell-escape"\app@nl
124         Or turn off auto-pst-pdf.}%
125     \fi
126 \fi
127 \if@app@crop@
128   \PassOptionsToPackage{notightpage}{pst-pdf}
129 \fi
```

**$\LaTeX$  compilation** Either we're calling latex from within a pdf $\LaTeX$  run (see above) or the document is being compiled as usual.

```
130 \else
```

$\LaTeX$  compilation from scratch (as in 'latex <document>.tex') — here the postscript environment does nothing and document is processed 'normally':

```
131 \ifx\APPmakepictures\undefined
132   \PassOptionsToPackage{inactive}{pst-pdf}
```

$\LaTeX$  compilation induced by this package:

```
133 \else
134   \if@app@crop@
135     \PassOptionsToPackage{notightpage}{pst-pdf}
```



```

136 \fi
137 \fi
138 \fi

```

After the requisite package options have been declared depending on the execution mode, it's now time to load the package:

```

139 \RequirePackage{pst-pdf}

```

## 7.2 Extras for external packages

Commands are provided that mirror `\includegraphics` (and similarly accept an optional argument) for the output of different psfrag-related packages. This provides a consistent and easy way to include such figures in the document.

Please suggest wrappers for other packages that output psfrag figures (for example: SciLab, R, Maple, LabView, Sage, ... ?)

`\matlabfig` We need to disable the scaling that `laprint` applies to `\includegraphics` in here, because otherwise labels that extend outside the bounding box of the generated PostScript file will change the intended width of the graphic.

```

140 \let\app@ig\includegraphics
141 \newcommand\matlabfig[2] []{%
142   \begin{postscript}
143     \renewcommand\resizebox[3]{##3}%
144     \renewcommand\includegraphics[2] [] {\app@ig[#1]{##2}}%
145     \input{#2}%
146   \end{postscript}}

```

`\mathfig` For Mathematica's MathPSfrag output

```

147 \newcommand\mathfig[2] []{%
148   \begin{postscript}
149     \input{#2-psfrag}%
150     \includegraphics[#1]{#2-psfrag}%
151   \end{postscript}}

```

`\psfragfig` EPS graphics via psfrag. Include your psfrag commands in the files `<document>-psfrag.tex` and/or `<figname>-psfrag.tex`, where `<document>` is the filename of the main document and `<figname>` is the filename of the graphics inserted.

```

152 \newcommand\psfragfig[2] []{%
153   \@ifnextchar[
154     {\app@psfragfig[#1]{#2}}
155     {\app@psfragfig[#1]{#2} []}}
156 \def\app@psfragfig[#1]#2[#3]{%
157   \begin{postscript}
158     \InputIfFileExists{#2-psfrag}{-}{-}%

```

```
159     #3
160     \includegraphics[#1]{#2}%
161 \end{postscript}}
```

Finally, input any psfrag commands associated with the document:

```
162 \InputIfFileExists{\jobname-psfrag}{-}{-}
```