The auto-pst-pdf package-Dokumentation

Will Robertson & Johannes Große
(übersetzt von Gérome Bochmann und Christine Römer)
wspr 81 at gmail dot com

2009/0426 v0.6

Dieses Paket ist „maintained“ (per LPPL maintenance status) von Will Robertson.

Es arbeitet mit auto-pst-pdf.dtx
und den abgeleiteten Dateien auto-pst-pdf.pdf,
auto-pst-pdf.sty und auto-pst-pdf.ins.

Teil I

auto-pst-pdf User Guide

1 Zukünftige Entwicklung

Dieses Paket wird nicht aktiv weiterentwickelt (obwohl ich (Will Robertson) gern kleinere Funktionen einfügen und Fehler behebe). auto-pst-pdf wurde von mir geschrieben, um die Benutzung von psfrag in pdfLaTeX-Dokumenten zu erleichtern. Diese Funktion wird durch das neuere pstool-Paket viel effizienter und bequemer umgesetzt. Ich empfehle daher die Benutzung von pstools, sofern Sie auto-pst-pdf wegen dieser Funktion verwenden. Allerdings unterstützt pstool die Verwendung von pst-pdf noch nicht ausreichend, so dass auto-pst-pdf bis jetzt noch nicht völlig überflüssig ist.

2 Grundlagen

Dieses Paket bietet einen Wrapper (eine „Hülle“) für pst-pdf, damit das Setzen von DVI oder PDF erleichtert wird. Wenn man mit pdfLaTeX generiert, sorgt die standardmäßige Paketoption `[on]` automatisch dafür, dass die zusätzlichen Schritte LaTeX→ `dvips` → `ps2pdf` → `pdfcrop` in der Kompilierung ausgeführt und so die benötigen PDFBilder für das Dokument erzeugt werden.

Sobald die Bilder einmal generiert wurden und bereits vorliegen, kann diese Option im Paket mit `[off]` abgeschaltet werden, um Zeit beim Kompilieren zu sparen:

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

Sollte die Dateierweiterung ihres LaTeXDokumentes nicht `.tex` sein, dann muss dies beim Laden des Paketes angegeben werden (Ich benutze z. B. `.ltx`, um zwischen TeX und LaTeXDateien zu unterscheiden):

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

3 Vorraussetzungen

pdfTeX muss mit der Option `-shell-escape` aufgerufen werden. Dazu werden die folgenden Pakete benötigt: ifplatform, pst-pdf, xkeyval.

Für die Default-Option `crop=on` muss das `pdfcrop`- Perlscript[1] von Heiko Oberdiek installiert sein. Obwohl pdfcrop zur MiKTeX-Distribution gehört, wird unter MS Windows außerdem eine Perl-Installation[2] nötig sein.

4 Verfügbare Makros zum Einfügen von Grafiken

Darstellungen, die mit dem MATLAB-Paket labprint[3] und dem Mathematica-Paket MathPSfrag[4] erstellt wurden, können mithilfe der Makros sehr leicht eingefügt werden. Es steht außerdem ein Wrapper für ein generisches psfrag[5] zur Verfügung.

`\mathfig{`⟨*filename*⟩`}`	eine Mathematica-Grafik von MathPSfrag einfügen (ohne `-psfrag` Suffix)
`\matlabfig{`⟨*filename*⟩`}`	Eine MATLAB-Grafik aus laprint einfügen
`\psfragfig{`⟨*filename*⟩`}`	Ein EPS mit psfrag einfügen

[1] `http://www.ctan.org/tex-archive/support/pdfcrop/`
[2] Frei verfügbar unter: `http://www.activestate.com/Products/activeperl/index.plex`
[3] `http://www.uni-kassel.de/fb16/rat/matlab/laprint/`
[4] `http://wwwth.mppmu.mpg.de/members/jgrosse/mathpsfrag/`
[5] `http://www.ctan.org/tex-archive/help/Catalogue/entries/psfrag.html`

Die oben stehenden Befehle akzeptieren alle ein optionales Argument, das an das darunterliegende Makro `\includegraphics` weitergereicht wird.

Der Befehl `\matlabfig` „vermurkst“ die Ausgabe von labprint ein wenig, sodass die Schriftgrade in der Abbildung entsprechend der ursprünglichen Definition ausfallen. (Ich fürchte, das lässt sich nicht vermeiden.) Die psfrag-statements für den Befehl `\psfragfig` werden entweder mithilfe einer von beiden oder mit den beiden Dateien ⟨*document*⟩`-psfrag.tex` und ⟨*filename*⟩`-psfrag.tex` übergeben, sofern diese existieren. Ansonsten können zusätzliche `\psfrag`-Statements mithilfe eines angehängten, optionalen Arguments hinzugefügt werden:

`\psfragfig[`⟨*graphics options*⟩`]{`⟨*filename*⟩`}[`⟨*psfrag statements*⟩`]`

5 Fortgeschrittene Paketfunktionen

Man kann bessere Ergebnisse erzielen, wenn man `pdfcrop` während der Hilfskompilierung verwendet, was eigentlich standardmäßig auch getan wird. `pdfcrop` wird jedoch nicht standardmäßig mit installiert und auch nicht immer gebraucht. Das „Cropping“ mit `pdfcrop` kann mithilfe der entsprechenden Option gesteuert werden:

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

Das Paket löscht die während der LaTeX-Hilfskompilierung generierten Dateien automatisch. Welche Dateien gelöscht werden, kann mit einer Liste von Dateierweiterungen gesteuert werden, die an die `cleanup`-Option übergeben wird (es wird keine Warnung oder Fehlermeldung ausgegeben, wenn ein Dateityp angegeben wird, der nicht existiert). Dies ist die Standardliste:

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

Wenn innerhalb der Grafiken, die von pst-df verarbeitet werden, Querverweise verwendet werden, ist es nötig, die Zusatzkompilierung mehr als einmal durchzuführen, damit die Querverweise übernommen werden. Die Zahl hängt von der Anwendung ab und muss explizit definiert werden:

```
{\color{gray}|
|\usepackage[|\color{niceblue}|runs=2|\color{gray}|]{auto-pst-pdf}|}
```

Die Argumente, die für die Zusatzkomplierung an `latex`, `dvips`, `ps2pdf`, und `pdfcrop` übergeben werden können alle – entsprechendes Wissen vorausgesetzt – angepasst werden. Die Standardwerte für die letzten drei lauten wie folgt:

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

Die Zusatzkompilierung von LaTeX besitzt einige hardcoded-Optionen (der Quellcode gibt hier Aufschluss) und einige Optionen können bei Bedarf angehängt werden. Um beispielsweise während der Zusatzkompilierung mehr Informationen auf der Konsole ausgeben zu lassen, muss die folgende Paketoption verwendet werden:

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

Alle unbekannten Paketoptionen werden an pst-pdf weitergegeben. Beispielsweise lädt `\usepackage[final]{auto-pst-pdf}` pst-pdf mit der Paketoption `final`, wodurch die `draft`-Option vom Laden der Klasse möglichst aufgehoben wird.

6 Danksagung

Vielen Dank an die Autoren von pst-pdf, psfrag, laprint, MathPSfrag und pdfcrop. Ohne ihre vereinten Bemühungen über Jahre hinweg würde dieses Paket nicht existieren. Schließlich gebührt Gernot Hassenfplug besonderer Dank für ausführliche Prüfungen, Vorschläge zu Paketfunktionen und die moralische Unterstützung :) Danke Dir.

Teil II
auto-pst-pdf Implementation

7 Setup Code

Das ist das Paket.

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

Change History

- v0.3
  - Zuviel, um es hier aufzulisten. Ausführung von Befehlen komplett neu geschrieben.
- v0.4
  - Johannes hat an dem Code rumgebastelt. Will wirds richten. :-)
  - Will hat alles hingebogen.

- v0.5
  - Allgemein: `delay`-Option entfernt. Es soll einfach sein, Mensch!!
  - `\ifdefined` entfernt, um e-TEXzu vermeiden.
  - Das Herumgekaspere mit den Image-Erweiterungen wurde unterbunden.
  - `matlabfig` `\resizebox` für `laprint` neu definiert.
  - `app@convert`: Package Error gefixt (sollte eine Warnung gewesen sein.)
  - `\psfragfig` erweitern, damit willkürlicher Input für zusätzliche `\psfrag` Befehle akzeptiert wird.
- v0.6
  - Unbekannte Optionen werden an pst-pdf weitergegeben.
  - `runs` Option hinzugefügt (Danke Joseph!!)

Benötigte Pakete pst-pdf wird später geladen.

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

Dinge, die wir brauchen

```
\newif\if@app@off@
\newif\if@app@crop@
\newcounter{app@runs}
\def\app@suffix{autopp}
\edef\app@jobname{\jobname-\app@suffix}
\edef\app@pics{\jobname-pics.pdf}
```

Verarbeitung von Optionen

```
\DeclareOptionX{off}[]{\@app@off@true}
\define@choicekey{auto-pst-pdf.sty}{crop}[\@tempa\@tempb]{on,off}{%
\ifcase\@tempb\relax
\@app@crop@true
\or
\@app@crop@false
\fi}
\DeclareOptionX{on}[]{\@app@off@false}
\DeclareOptionX{ext}{\def\app@ext{#1}}
\DeclareOptionX{latex}{%
\def\app@latex@opts{%
\ifwindows
```

```
-disable-write18
\else
-no-shell-escape
\fi
-jobname="\app@jobname"
-interaction=batchmode
#1}}
\DeclareOptionX{dvips}{\def\app@dvips@opts{#1}}
\DeclareOptionX{pspdf}{\def\app@pspdf@opts{#1}}
\DeclareOptionX{pdfcrop}{\def\app@pdfcrop@opts{#1}}

\DeclareOptionX{cleanup}{%
  \let\app@rm@files\@empty
  \@for\@ii:=#1\do{%
    \edef\app@rm@files{\app@rm@files,\app@jobname.\@ii}}}

\DeclareOptionX{runs}{%
  \setcounter{app@runs}{#1}% support calc
  \ifnum\c@app@runs > \z@
  \else
    \app@PackageWarning{The number of runs must be at least one.}%
    \c@app@runs\@ne
  \fi}

\DeclareOptionX*{\PassOptionsToPackage{\CurrentOption}{pst-pdf}}

\ExecuteOptionsX{%
    ext=tex,
    crop=on,
    latex={},
    dvips={-Ppdf},
    pdfcrop={},
    cleanup={log,aux,dvi,ps,pdf},
    runs=1
}
\ifwindows
  \ExecuteOptionsX{pspdf={}}
\else
  \ExecuteOptionsX{pspdf={-dAutoRotatePages=/None}}
\fi
\ProcessOptionsX
```

Kurzschrift

```
\def\app@exe{\immediate\write18}
```

```
\def\app@nl{^^J\space\space\space\space}
\newcommand\app@PackageError[2]{%
  \PackageError{auto-pst-pdf}{\app@nl #1^^J}{#2}}
\newcommand\app@PackageWarning[1]{%
  \PackageWarning{auto-pst-pdf}{\app@nl #1^^JThis warning occured}}
\newcommand\app@PackageInfo[1]{\PackageInfo{auto-pst-pdf}{#1}}
```

Die hier sind niedlich:

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

1. Name des Befehls
2. Quelldatei
3. Zieldatei

Überprüfe, ob die Quelldatei existiert und den Befehl zur Generierung der Zieldatei aufruft. Wenn die Datei nicht erstellt wird, gib einen Fehler aus.

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

Zuerst wird die Befehlssequenz `latex` → `dvips` → `ps2pdf` (→ `pdfcrop`) im Ganzen definiert. Das Kompilierungsmakro wird hiernach aufgerufen. Innerhalb dieses Makros wird der eigentliche `pdf`-Container erzeugt. Jeder Verarbeitungsschritt ist in einem einzelnen Makro enthalten, um die Modifikation zu erleichtern.

```
\def\app@compile{
  \app@cleanup
  \app@remove@container
  \loop\ifnum\c@app@runs > \@ne
     \app@convert{extralatex}{\jobname.\app@ext}{\app@jobname.dvi}
     \advance\c@app@runs\m@ne
  \repeat
  \app@convert{latex}{\jobname.\app@ext}{\app@jobname.dvi}
  \app@convert{dvips}{\app@jobname.dvi}{\app@jobname.ps}
  \if@app@crop@
    \app@convert{pstopdf}{\app@jobname.ps}{\app@jobname.pdf}
    \app@convert{pdfcrop}{\app@jobname.pdf}{\app@pics}
  \else
    \app@convert{pstopdf}{\app@jobname.ps}{\app@pics}
```

```
  \fi
  \IfFileExists{\app@pics}
    {\app@cleanup}
    {\app@PackageWarning{Could not create \app@pics.
      Auxiliary files not deleted.}}}
```

Kommandozeilenprogramm, mit dem Dateien gelöscht werden:

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

Makro, das Dateien löscht (durch Kommata getrennt) sofern diese existieren:

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

PDF-Bild-Containter entfernen:

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

Hilfsdateien löschen: (\app@rm@files durch die Paketoption `cleanup` definiert)

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

LaTeX:

```
\def\app@cmd@latex#1#2{latex \app@latex@opts\space
  "\unexpanded{\let\APPmakepictures\empty\input} #1"}
\def\app@cmd@extralatex#1#2{latex \app@latex@opts\space
  "\unexpanded{\let\APPmakepictures\undefined\input} #1"}
```

`dvips`:

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

`ps2pdf`:

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

`pdfcrop`:

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

7.1 Grundfunktionalität

Für die Kompilierung wird `[notightpage]` als Option von `pst-pdf` und das `pdfcrop`-Perlscript verwendet, da EPSAbbildungen Elemente außerhalb ihrer Bounding-Box enthalten können und dabei abgeschnittener Inhalt nach `ps2pdf` entstehen kann. Andernfalls würde das script `ps4pdf` vollkommen ausreichen.

pdfLaTeX

Kompilierung Zusätzliche Verarbeitung mit pst-pdf ist nötig:

```
\ifpdf
  \if@app@off@\else
    \ifshellescape
      \app@exe{echo "  "}
      \app@exe{echo "----------------------------------------------------------------------"}
      \app@exe{echo "auto-pst-pdf: Auxiliary LaTeX compilation"}
      \app@exe{echo "----------------------------------------------------------------------"}
      \app@compile
      \app@exe{echo "----------------------------------------------------------------------"}
      \app@exe{echo "auto-pst-pdf: End auxiliary LaTeX compilation"}
      \app@exe{echo "----------------------------------------------------------------------"}
    \else
      \app@PackageError{%
        "shell escape" (or "write18") is not enabled:\app@nl
        auto-pst-pdf will not work!}
       {You need to run LaTeX with the equivalent of
        "pdflatex -shell-escape"\app@nl
        Or turn off auto-pst-pdf.}%
    \fi
  \fi
  \if@app@crop@
    \PassOptionsToPackage{notightpage}{pst-pdf}
  \fi
```

LaTeX Compilation Entweder `latex` wird innerhalb eines pdfLaTeXDurchlaufs aufgerufen (siehe oben) oder das Dokument wird ganz normal kompiliert.

```
\else
```

LaTeX Kompilierung bei Null anfangen (wie in '`latex` ⟨*document*⟩`.tex`') — in diesem Fall tut die `postscript`-Umgebung nichts und das Dokument wird "normal"verarbeitet:

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

LaTeX Kompilierung wird durch dieses Paket induziert:

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

```
    \fi
  \fi
\fi
```

Nachdem die erforderlichen Paketoptionen deklariert wurden, je nach Modus der Applikation, ist es nun an der Zeit das Paket zu laden:

```
\RequirePackage{pst-pdf}
```

7.2 Extras für externe Pakete

Es werden Befehle zur Verfügung gestellt, die `\includegraphics` für die Ausgabe verschiedener `psfrag`-bezogener Pakete imitieren (und ebenso wahlweise ein Argument akzeptieren). Dadurch wird ein konsistenter und einfacher Weg geboten, der solche Abbildungen in das Dokument einbindet. Vorschläge für Wrapper anderer Pakete, die `psfrag`-Abbildungen ausgeben, sind gewünscht (z. B.: SciLab, R, Maple, LabView, Sage, ... ?)

Die Skalierung, die `labprint` hier für `\includegraphics` verwendet, muss deaktiviert werden, andernfalls werden Label, die die Bounding-Box überschreiten, die angegebene Breite der Grafik verändern.

```
\let\app@ig\includegraphics
\newcommand\matlabfig[2][]{%
  \begin{postscript}
    \renewcommand\resizebox[3]{##3}%
    \renewcommand\includegraphics[2][]{\app@ig[#1]{##2}}%
    \input{#2}%
  \end{postscript}}
```

Für Mathematicas MathPSfrag-Ausgabe

```
\newcommand\mathfig[2][]{%
  \begin{postscript}
    \input{#2-psfrag}%
    \includegraphics[#1]{#2-psfrag}%
  \end{postscript}}
```

EPS Grafiken via psfrag. Einfach die psfrag-Befehle in die Datei ⟨*document*⟩-psfrag.tex und/oder ⟨*figname*⟩-psfrag.tex einfügen. ⟨*document*⟩ steht hier für den Dateinamen des Hauptdokuments und ⟨*figname*⟩ ist der Dateiname, der eingefügten Grafik.

```
\newcommand\psfragfig[2][]{%
  \@ifnextchar[
    {\app@psfragfig[#1]{#2}}
```

```
    {\app@psfragfig[#1]{#2}[]}}
\def\app@psfragfig[#1]#2[#3]{%
  \begin{postscript}
    \InputIfFileExists{#2-psfrag}{}{}%
    #3
    \includegraphics[#1]{#2}%
  \end{postscript}}
```

Zu guter Letzt, alle psfrag-Befehle, die mit dem Dokument in Verbindung stehen eingeben:

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