The pdfpages Package*

Andreas Matthias
andreas.matthias@gmail.com

2017/06/14

Abstract

This package simplifies the insertion of external multi-page PDF or PS documents. It supports pdfTeX, VTeX, XeTeX, LuaTeX, and pTeX.

Contents

1 Introduction **1**

2 Usage **2**
2.1 Package Options 2
2.2 Commands 2
2.3 The Layout 11
2.4 Hints for Users of platex 11
2.5 Pitfalls 12

3 Required Packages **12**

4 Acknowledgment **12**

1 Introduction

When creating PDF documents, it is sometimes useful to insert pages of external PDF documents. This can be done with the `\includegraphics` command from the `graphics` package. But a simple `\includegraphics{doc.pdf}` normally produces '`Overfull \hbox`' and '`Overfull \vbox`' warnings, because the size of the inserted pages does not match the print space.

The `pdfpages` package makes it easy to insert pages of external PDF documents without worrying about the print space. Here are some features of the `pdfpages` package: Several logical pages can be arranged onto each sheet of paper and the layout can be changed individually. A lot of hypertext operations are supported, like links to the inserted pages, links to the original PDF document, threads, etc. When working with VTeX the same is possible with PostScript documents, too. Note that PostScript documents are only supported by VTeX and *not* by pdfLaTeX.

*This file has version number v0.5i, last revised 2017/06/14.

When producing DVI output `pdfpages` cannot insert pages of a PDF documents. But instead of interrupting execution `pdfpages` will insert empty pages. This feature is important when using packages like `pst-pdf`, which need to produce DVI output at the first run.

Links and other interactive features of PDF documents When including pages of a PDF only the so called content stream of these pages is copied but no links. Up to now there are no TeX-engines (pdfTeX, XeTeX, ...) available that can copy links or other interactive features of a PDF document, too. Thus, all kinds of links[1] will get lost during inclusion. (Using `\includepdf`, `\includegraphics`, or other low-level commands.)

However, there's a glimmer of hope. Some links may be extracted and later reinserted by a package called *pax* which can be downloaded from CTAN [3]. Have a look at it!

Help I really enjoy working on the pdfpages package in my spare time and I appreciate your great feedback. You, the users, encouraged me to start with pdfpages and to continue working on it for more than 15 years now. It's great to be part of this magnificent TeX community. I enjoyed every minute and the pdfpages project itself does not need any donations.

However, I'd be very grateful if you made a donation to Kira Grünberg at http://www.donationkira.com/. Kira is an Austrian pole vaulter who had a terrible training accident in July 2015. Since then she is paralysed. But with an irresistible smile and an amazing attitude towards life she's accepted what happened and is now following new goals and dreams. Nevertheless life after a cervical spinal cord injury is not at all easy and continuous physiotherapy to improve arm and hand functioning is essential. I'd be very pleased to hear from you making a donation to Kira. Thank you!

2 Usage

2.1 Package Options

`\usepackage[`⟨*options*⟩`]{pdfpages}`

⟨*option*⟩ –

`final`: Inserts pages. This is the default.

`draft`: Does not insert pages, but prints a box and the filename instead.

`demo`: Inserts empty pages instead of the actual PDFs.

`nodemo`: Disables 'demo'.

`enable-survey`: Activates survey functionalities. (*experimental, subject to change*)

2.2 Commands

`\includepdf` Inserts pages of an external PDF document.

[1] Actually not only links but all kinds of *PDF annotations* will get lost.

`\includepdf[`⟨*key=val*⟩`]{`⟨*filename*⟩`}`

⟨*key=val*⟩ – A comma separated list of options using the ⟨*key*⟩=⟨*value*⟩ syntax.

⟨*filename*⟩ – Filename of the PDF document. (The filename *must not* contain any blanks!)

The following list describes all possible options of `\includepdf`. All options are using the ⟨*key=value*⟩ syntax.

- Main options:

  pages Selects pages to insert. The argument is a comma separated list, containing page numbers (`pages={3,5,6,8}`), ranges of page numbers (`pages={4-9}`) or any combination. To insert empty pages use `{}`.

  E.g.: `pages={3,{},8-11,15}` will insert page 3, an empty page, and pages 8, 9, 10, 11, and 15.

  Page ranges are specified by the following syntax: ⟨*m*⟩`-`⟨*n*⟩. This selects all pages from ⟨*m*⟩ to ⟨*n*⟩. Omitting ⟨*m*⟩ defaults to the first page; omitting ⟨*n*⟩ defaults to the last page of the document. Another way to select the last page of the document, is to use the keyword `last`. (This is only permitted in a page range.)

  E.g.: `pages=-` will insert *all* pages of the document, and `pages=last-1` will insert all pages in reverse order.

  (Default: `pages=1`)

  nup Puts multiple logical pages onto each sheet of paper. The syntax of this option is: `nup=`⟨*xnup*⟩`x`⟨*ynup*⟩. Where ⟨*xnup*⟩ and ⟨*ynup*⟩ specify the number of logical pages in horizontal and vertical direction, which are arranged on each sheet of paper. (Default: `nup=1x1`)

  landscape Specifies the format of the sheet of paper, which is rotated by 90 degrees. This does *not* affect the logical pages, which will *not* be rotated by the 'landscape' option. To rotate the logical pages use the 'angle' option (e.g. 'angle=90'). Either 'true' or 'false' (or no value, which is equivalent to 'true'). (Default: `landscape=false`)

- Layout options:

  delta Puts some horizontal and vertical space between the logical pages. The argument should be two dimensions, separated by space. See Chapter 2.3 and Figure 1. (Default: `delta=0 0`).

  offset Displaces the origin of the inserted pages. The argument should be two dimensions, separated by space. In 'oneside' documents positive values shift the pages to the *right* and to the *top* margin, respectively, whereas in 'twoside' documents positive values shift the pages to the *outer* and to the *top* margin, respectively. See Chapter 2.3 and Figure 1. (Default: `offset=0 0`)

  frame Puts a frame around each logical page. The frame is made of lines of thickness `\fboxrule`. Either 'true' or 'false' (or no value, which is equivalent to 'true'). (Default: `frame=false`)

column Pdfpages normally uses 'row-major' layout, where successive pages are placed in rows along the paper. The `column` option changes the output into a 'column-major' layout, where successive pages are arranged in columns down the paper. Either 'true' or 'false' (or no value, which is equivalent to 'true'). (Default: `column=false`)

columnstrict By default the last page is not set in a strict 'column-major' layout, if the logical pages do not fill up the whole page. The `columnstrict` option forces a strict 'column-major' layout for the last page. Either 'true' or 'false' (or no value, which is equivalent to 'true'). (Default: `columnstrict=false`)

1	4	
2	5	
3		

`columnstrict=true`

1	3	5
2	4	

`columnstrict=false`

openright This option puts an empty page before the first logical page. In combination with `nup=2x1`, `nup=2x2`, etc., this means that the first page is on the right side. The same effect can be achieved with the `pages` option, if an empty page is inserted in front of the first page. Either 'true' or 'false' (or no value, which is equivalent to 'true'). (Default: `openright=false`)

pagecommand Declares LaTeX commands, which are executed on each sheet of paper. (Default: `pagecommand={\thispagestyle{empty}}`)

turn By default pages in landscape format are displayed in landscape orientation (if the PDF viewer supports this). With `turn=false` this can be prohibited. Either 'true' or 'false' (or no value, which is equivalent to 'true'). (Default: `turn=true`)

noautoscale By default pages are scaled automatically. This can be suppressed with the `noautoscale` option. In combination with the `scale` option (from graphicx) the user has full control over the scaling process. Either 'true' or 'false' (or no value, which is equivalent to 'true'). (Default: `noautoscale=false`)

fitpaper Adjusts the paper size to the one of the inserted document. Either 'true' or 'false' (or no value, which is equivalent to 'true'). (Default: `fitpaper=false`)

reflect Reflects included pages. Either 'true' or 'false' (or no value, which is equivalent to 'true'). (Default: `reflect=false`)

signature Creates booklets by rearranging pages into signatures and setting `nup=1x2` or `nup=2x1`, respectively. This option takes one argument specifying the size of the signature, which should be a multiple of 4.

An example for documents in portrait orientation:

```
\includepdf[pages=-, signature=8,
            landscape]{portrait-doc.pdf}
```

An example for documents in landscape orientation:

```
\includepdf[pages=-, signature=8]{landscape-doc.pdf}
```

signature* Similar to `signature`, but now for right-edge binding.

booklet This option is just a shortcut of the 'signature' option, if you choose a signature value so large that all pages fit into one signature. Either

'true' or 'false' (or no value, which is equivalent to 'true'). (Default: `booklet=false`)

picturecommand Declares picture commands which are executed on every page within a picture environment with the base point at the lower left corner of the page. (The base point does not change if the page is rotated, e.g. by the `landscape` option.) (Default: `picturecommand={}`)

picturecommand* Like `picturecommand`, but with the restriction that `picturecommand*` executes its picture commands only on the very first page. (Default: `picturecommand*={}`)

pagetemplate By default the first inserted page will be used as a template. This means that all further pages are scaled such that they match within the contour of this first page. This option allows to declare another page to be used as a template; which is only useful if a PDF document contains different page sizes or page orientations. The argument should be a page number. (Default: `pagetemplate=`⟨*first inserted page*⟩)

templatesize This option is similar to the `pagetemplate` option, but its arguments specify the size of the template directly. Its syntax is: `templatesize={`⟨*width*⟩`}{`⟨*height*⟩`}` Note: The two lengths should be a bit larger than desired, to keep away from rounding errors.

(Default: `templatesize=`⟨*size of the first inserted page*⟩)

rotateoversize This option allows to rotate oversized pages. E.g. pages in landscape orientation are oversized relatively to their portrait counterpart, because they do not match within the contour of a portrait page without rotating them. By default oversized pages are scale and are *not* rotated. Either 'true' or 'false' (or no value, which is equivalent to 'true'). (Default: `rotateoversize=false`)

doublepages Inserts every page twice. This is useful for 2-up printing, if one wants to cut the stack of paper afterwards to get two copies. Either 'true' or 'false' (or no value, which is equivalent to 'true'). (Default: `doublepages=false`)

doublepagestwist Whereas with `doublepages` the cutting edge is once on the inner side and ones on the outer side, `doublepagestwist` turns the pages such, that the cutting edge is always on the inner side. Either 'true' or 'false' (or no value, which is equivalent to 'true'). (Default: `doublepagestwist=false`)

doublepagestwistodd Turns the pages such, that the cutting edge is always on the outer side. Either 'true' or 'false' (or no value, which is equivalent to 'true'). (Default: `doublepagestwistodd=false`)

doublepagestwist* Like `doublepagestwist` but for double side printing. Either 'true' or 'false' (or no value, which is equivalent to 'true'). (Default: `doublepagestwist*=false`)

doublepagestwistodd* Like `doublepagestwistodd` but for double side printing Either 'true' or 'false' (or no value, which is equivalent to 'true'). (Default: `doublepagestwistodd*=false`)

duplicatepages Duplicates each page n times, with n being the argument to this option. (Default: `duplicatepages=2`)

- Miscellaneous options:

  lastpage In DVI mode pdfpages cannot determine the number of pages of the included document. So this option is suitable to specify the number of pages. This option is only used in DVI mode and has no meaning in any other mode. The argument should be a page number. (Default: `lastpage=1`)

- Hypertext options:

  link Inserted pages become a target of a hyperlink. The name of the link is '⟨*filename*⟩.⟨*page number*⟩'. The filename extension of ⟨*filename*⟩ *must not* be stripped. Either 'true' or 'false' (or no value, which is equivalent to 'true'). (Default: `link=false`)

  linkname Changes the default linkname created by the option `link`. Instead of ⟨*filename*⟩ the value of this option is used. E.g. `linkname=mylink` produces the linknames '`mylink.`⟨*page number*⟩'.

  thread Combines inserted pages to an article thread. Either 'true' or 'false' (or no value, which is equivalent to 'true'). (Default: `thread=false`)

  threadname Several threads are distinguished by their threadnames. By default the threadname is equal to the filename (plus filename extension), but it can be changed with this option. This is useful if the same file is inserted twice or more times and should not be combined to one single thread. Or the other way round if pages from different documents should be combined to one single thread. (Default: `threadname=`⟨*filename.ext*⟩)

  linktodoc Lets the inserted pages be hyperlinks to the document from which they were extracted. Note that the PDF-Viewer will not find the file, if ⟨*filename*⟩ has not filename extension (.pdf). Either 'true' or 'false' (or no value, which is equivalent to 'true'). (Default: `linktodoc=false`)

- Additional hypertext options:

  linkfit Specifies, how the viewer displays a linked page. This option changes the default behavior of the option `link`. Possible values are: `Fit`, `FitH` ⟨*top*⟩, `FitV` ⟨*left*⟩, `FitB`, `FitBH` ⟨*top*⟩, `FitBV` ⟨*left*⟩, and `Region`.

  See [2] for a details description of these PDF destinations. The `region` destination was added by pdfpages and is not a real PDF destinations. It scales a page such that the included page fits exactly into the window of the PDF viewer.

  Note that not all of these options are supported by all TEX-engines or drivers, respectively. (Default: `linkfit=fit`)

  linktodocfit By default the option `linktodoc` opens the page in 'Fit in Window' view. Another view can be specified with this option. Possible values are the legal PDF tokens: `/FitH` ⟨*top*⟩, `/FitV` ⟨*left*⟩, etc. (See [2] for more details.) (Default: `linktodocfit=/Fit`)

  newwindow By default option `linktodoc` opens a new window. This can be changed with option `newwindow`. Either 'true' or 'false' (or no value, which is equivalent to 'true'). (Default: `newwindow=true`)

linkfilename Sets the name (with path) of the file to be linked to by the option `linktodoc`. You will hardly ever need this option. (Default: `linkfilename=`⟨*filename.ext*⟩)

- Experimental options: (Syntax may change in future versions!)

  addtotoc Adds an entry to the table of contents. This option requires five arguments, separated by commas:

  `addtotoc={`⟨*page number*⟩`,`⟨*section*⟩`,`⟨*level*⟩`,`⟨*heading*⟩`,`⟨*label*⟩`}`

  ⟨*page number*⟩: Page number of the inserted page.

  ⟨*section*⟩: LaTeX sectioning name – e.g., section, subsection, ...

  ⟨*level*⟩: Number, denoting depth of section – e.g., 1 for section level, 2 for subsection level, ...

  ⟨*heading*⟩: Title inserted in the table of contents.

  ⟨*label*⟩: Name of the label. This label can be referred to with `\ref` and `\pageref`.

  Note: The order of the five arguments must not be mixed. Otherwise you will get very strange error messages.

  The `addtotoc` option accepts multiple sets of the above mentioned five arguments, all separated by commas. The sets must be sorted such that the ⟨*page number*⟩s are in ascending order. (Strictly speaking they must have the same order as the page numbers specified by the `pages` option.)

  The proper recursive definition of the `addtotoc` option is:

  `addtotoc={`⟨*toc-list*⟩`}`

  ⟨*toc-list*⟩ → ⟨*page number*⟩,⟨*section*⟩,⟨*level*⟩,⟨*heading*⟩,⟨*label*⟩[,⟨*toc-list*⟩]

  addtolist Adds an entry to the list of figures, the list of tables, or any other list (e.g. from *float.sty*). This option requires four arguments, separated by commas:

  `addtolist={`⟨*page number*⟩`,`⟨*type*⟩`,`⟨*heading*⟩`,`⟨*label*⟩`}`

  ⟨*page number*⟩: Page number of the inserted page.

  ⟨*type*⟩: Name of a floating environment. (`figure`, `table`, etc.)

  ⟨*heading*⟩: Title inserted into LoF, LoT, etc.

  ⟨*label*⟩: Name of the label. This label can be referred to with `\ref` and `\pageref`.

  Like `addtotoc`, `addtolist` accepts multiple sets of the above mentioned four arguments, all separated by commas. The proper recursive definition is:

  `addtolist={`⟨*lof-list*⟩`}`

  ⟨*lof-list*⟩ → ⟨*page number*⟩,⟨*type*⟩,⟨*heading*⟩,⟨*label*⟩[,⟨*lof-list*⟩]

  survey Creates a survey of those pages of the document, which are marked with `\AddToSurvey`. (`\AddToSurvey` is a simple command with no arguments. It just writes out labels to the `.aux` file.) This option may be used when preparing slides to create a survey of only 'finished' pages – if pages are build up incrementally.

  To use this option a special sequence of production steps must be obeyed. Here is a small example:

```
--- slides.tex ---
\documentclass{article}
\usepackage[draft,enable-survey]{pdfpages}
\begin{document}
  ... some text ...
  \AddToSurvey
  ... some text ...
  \includepdf[survey,nup=2x2]{slides-tmp.pdf}
\end{document}
```

This is the outline of a document, called `slides.tex`. Run it through pdfLaTeX several times until all cross-references are solved. (LaTeX will produce a warning, if cross-references are not solved, yet.) Now copy the file `slides.pdf` to `slides-tmp.pdf` and rename `draft` (package option of pdfpages) to `final`. The next and final run through pdfLaTeX will actually insert the desired pages, whereas the former runs with `draft` did just insert blank pages. The inserted pages are hyperlink to the orignial pages.

Experienced users would certainly call

```
pdflatex '\PassOptionsToPackage{final}{pdfpages} \input{slides}'
```

instead of exchanging `draft` for `final`.

Either 'true' or 'false' (or no value, which is equivalent to 'true'). (Default: `survey=false`)

survey-nolink Same as option `survey` except that the inserted pages do not become hyperlinks. This option may be used to create an external survey. To continue the example above (`slides.tex`), it is now possible to create handouts of the 'finished' slides as an external document.

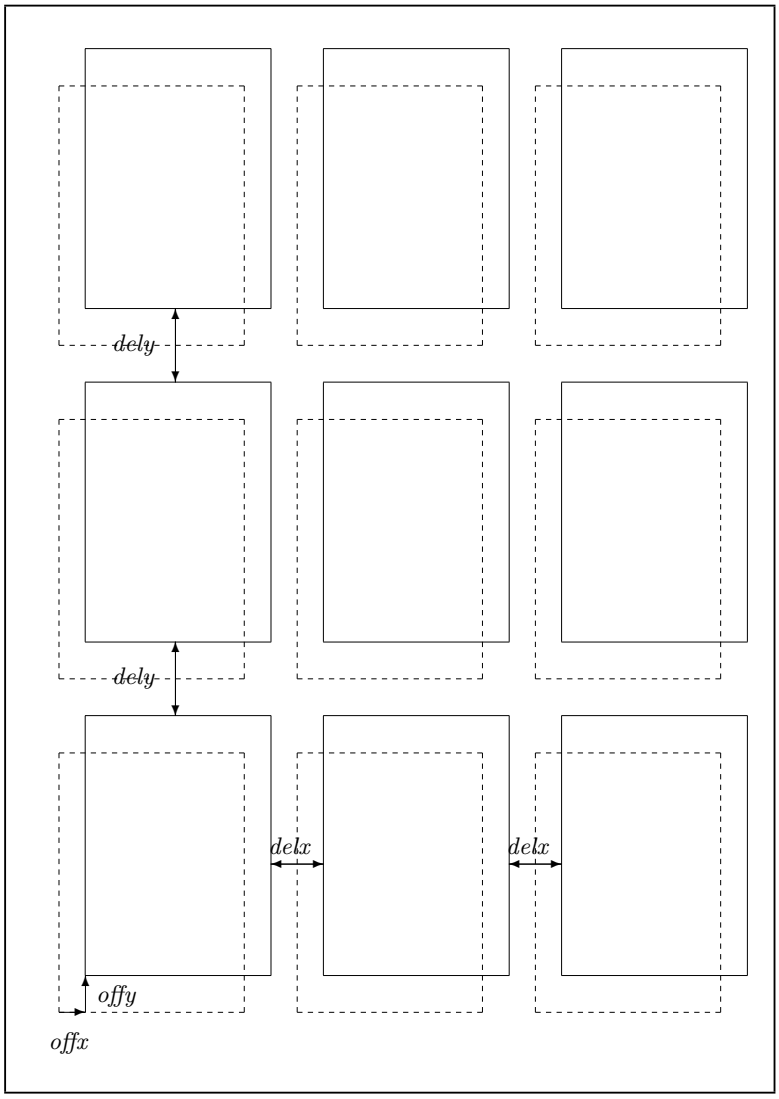
```
\documentclass{article}
\usepackage{xr}
\externaldocument{slides}
\usepackage[enable-survey]{pdfpages}
\begin{document}
  \includepdf[survey-nolink, nup=1x2]{slides.pdf}
\end{document}
```

Either 'true' or 'false' (or no value, which is equivalent to 'true'). (Default: `survey-nolink=false`)

xr-prefix Adjusts a prefix to the labels `\includepdf` is looking for. The name of the prefix must be the same as the name of the optional argument of `\externaldocument`. (Default: `xr-prefix={}`)

Internally the command `\includepdf` makes use of the `\includegraphics` command from the `graphicx` (actually `graphics`) package. Hence it is possible to use all the options of `\includegraphics`, too. Options which are not interpreted by `\includepdf` are passed directly to `\includegraphics`.

Especially the 'trim' and 'clip' options of `\includegraphics` are quite useful, if only parts of a page should be inserted. (Maybe to cut off the header and footer of the inserted pages.) Just use the 'trim' and 'clip' options as if they were options of `\includepdf`. They will be passed to `\includegraphics` internally.



— physical sheet of paper
— logical pages
--- centered position

`delta=`⟨*delx*⟩ ⟨*dely*⟩
`offset=`⟨*offx*⟩ ⟨*offy*⟩

Figure 1: Layout

`\includepdfmerge` Inserts pages of several external PDF documents.

`\includepdfmerge[`⟨*key=val*⟩`]{`⟨*file-page-list*⟩`}`

⟨*key=val*⟩ – A comma separated list of options using the ⟨*key*⟩=⟨*value*⟩ syntax.

⟨*file-page-list*⟩ – ⟨*filename*⟩[,⟨*page spec*⟩][,⟨*file-page-list*⟩]
A comma separated list of filenames and optional ⟨*page spec*⟩ specifiers. A ⟨*page spec*⟩ can be everything the option `pages` accepts. Leading and trailing spaces of items in the list is stripped.

The `\includepdfmerge` command uses the same options as `\includepdf` with one exception. The option `pages` has no meaning for `\includepdfmerge`. Instead the ⟨*page spec*⟩ specifier is used to specify which pages should be inserted. The ⟨*page spec*⟩ specifier accepts the same values as the `pages` option. If no ⟨*page spec*⟩ specifier is given, only the first page will be inserted.

Examples: To create a kind of summary of three PDF documents, it might be nice to insert just the first page of each document and to provide links to the original documents:

```
\includepdfmerge[nup=1x3, landscape, linktodoc]
                {doc1.pdf, doc2.pdf, doc3.pdf}
```

But sometimes the title page of a document is not the first page. So it would be more pleasant to insert the title page of each document than the first page. This can be done with the ⟨*page spec*⟩ specifier. The following example inserts the second page of *doc1.pdf* and the third page of *doc2.pdf* and *doc3.pdf*:

```
\includepdfmerge[nup=1x3, landscape, linktodoc]
                {doc1.pdf, 2, doc2.pdf, 3, doc3.pdf, 3}
```

Here is an example of more complex ⟨*page spec*⟩ specifiers:

```
\includepdfmerge[nup=1x3, landscape, linktodoc]
                {doc1.pdf, 1-3,
                 doc2.pdf, 3, 5, 9,
                 doc3.pdf, 3-5, 7}
```

`\includepdfset` If you need the same options for `\includepdf` all the time, it is possible to define global options with `\includepdfset`. The argument of `\includepdfset` is a comma separated list of options, using the ⟨*key*⟩=⟨*value*⟩ syntax. These options are processed each time `\includepdf` is called. Local options (passed as an optional argument directly to `\includepdf`) are overwriting global options:

```
\includepdfset{⟨global options⟩}
\includepdf[⟨local options⟩]{pdf-file}
```

Only options specific to this package can be made global by `\includepdfset`. Options of the graphicx package are not concerned.

`\threadinfodict` When using the option `thread` to create an article thread, it may be useful to create a thread information dictionary, too, which contains informations about the thread, such as its title, author, and creation date. The macro `\threadinfodict` is used to set these informations. It can be redefined and may contain entries of a thread information dictionary in low-level PDF commands. (See [2] for more information.)

```
\renewcommand*{\threadinfodict}
  {/Title (My first thread) /Author (That's me!)}
```

2.3 The Layout

The default layout can be changed by the options `delta` and `offset`. Figure 1 shows the meaning of these options.

The inserted logical pages are being centered on the sheet of paper by default. To displace them use the `offset` option, which argument should be two dimensions. E.g. `offset=10mm 14mm` means that the logical pages are displaced by 10 mm in horizontal direction and by 14 mm in vertical direction. In 'oneside' documents positive values shift the pages to the *right* and to the *top* margin, respectively, whereas in 'twoside' documents positive values shift the pages to the *outer* and to the *top* margin, respectively.

By default logical pages are being arranged side by side. To put some space between them, use the `delta` option, whose argument should be two dimensions. Figure 1 shows the meaning of `delta`.

The layout options `delta` and `offset` *always* refer to a sheet of paper in portrait orientation. No matter whether you have set the `landscape` option to `true`, or not.

If you are confused about horizontal (`x`) and vertical (`y`) directions, just set the option `turn=false`. Now your PDF viewer shows the pages in the *same* orientation as in Figure 1. And the options `delta` and `offset` have the *same* meaning as in Figure 1. Regardless of any other options.

2.4 Hints for Users of platex

- The recommended way to tell `pdfpages` that you are using `platex` is to add `dvipdfmx` as a class option, e.g.:

  ```
  \documentclass[dvipdfmx]{article}
  \usepackage{pdfpages}
  ```

- Pdfpages calls program `extractbb` to get the total number of pages of a PDF. But if an `xbb` file (output of `extractbb`) exists, pdfpages will not call `extractbb` but use this file. However, be very cautious with `xbb` files: *Do not use `xbb` files for PDFs with varying page sizes.* Because an `xbb` file contains only the page size of a single page.

2.5 Pitfalls

pagecolor When setting the background color with `\pagecolor` (a command from *color.sty*), the first `\pagecolor` *must* precede `\usepackage{pdfpages}`.

```
\usepackage{color}
\pagecolor{white}
\usepackage{pdfpages}
```

The color is nonrelevant, it can be changed afterwards by using `\pagecolor` again. Just the order (first `\pagecolor` before `\usepackage{pdfpages}`) is important. – This is not needed when using VTeX.

3 Required Packages

The `pdfpages` package requires the following packages:

eso-pic `CTAN:macros/latex/contrib/eso-pic/`

everyshi `CTAN:macros/latex/contrib/ms/`

pdflscape `CTAN:/macros/latex/contrib/oberdiek/`

graphicx, ifthen, calc These packages belong to the standard LaTeX distribution.

Furthermore it requires a recent version of:

pdftex.def `http://www.tug.org/applications/pdftex/`

Since pdfTeX, Version 3.14159-1.00a-pretest-20010806, PDF import has improved a lot. This results in much smaller file sizes, faster processing and the intuitively correct treatment of landscape pages. The latest version of pdfTeX can be found at: `ftp://ftp.muni.cz/pub/tex/local/cstug/thanh/pdftex`.

4 Acknowledgment

I would like to thank Rolf Niepraschk and Heiko Oberdiek for their useful hints and suggestions. As well as Ross Moore, who encouraged me to implement the hypertext features.

References

[1] Hàn Thế Thành, Sebastian Rahtz, Hans Hagen, *The pdfTeX user manual*, http://www.tug.org/applications/pdftex

[2] *PDF Reference*, Adobe Systems Incorporated, http://www.adobe.com/devnet/pdf/pdf_reference.html

[3] Heiko Oberdiek, *pax: Extract and reinsert PDF annotations with pdfTeX* http://www.ctan.org/pkg/pax