The PSfrag system, version 3

Michael C. Grant and David Carlisle
psfrag@rascals.stanford.edu

26 November 1996

Contents

1 What is PSfrag? **1**

2 PSfrag necessities **2**
2.1 Choosing a PostScript driver 2

3 Installing PSfrag **3**

4 Usage **3**

5 Commands and Environments **4**
5.1 Embedding PSfrag operations into EPS files 6

6 Package Options **7**

7 An Example **7**
7.1 Figure scaling and resizing 8

8 Known problems **9**

9 The PSfrag mailing list **10**

1 What is PSfrag?

Many drawing and graphing packages produce output in the Encapsulated PostScript (EPS) format, but few can easily produce the equations and other scientific text of which LaTeX is capable. On the other hand, many LaTeXbased drawing packages are not as expressive or easy-to-use as the many advanced drawing packages that produce EPS output.

PSfrag provides the best of both worlds by allowing the user to precisely overlay Encapsulated PostScript (EPS) files with arbitrary LaTeX constructions. In order to accomplish this, the user places a simple text "tag" in the graphics file, as a "position marker" of sorts. Then, using simple LaTeX commands, the user instructs PSfrag to remove that tag from the figure, and replace it with a properly sized, aligned, and rotated LaTeX equation.

This version of PSfrag is significantly easier to use than previous versions, because it eliminates the preprocessing step required in those versions. As a result, the new PSfrag should also prove to be significantly more portable, reliable, and flexible.

Veteran PSfrag users need to know that the `\tex` command, while still supported, has been deprecated. For details, consult section 5.1. In addition, because the engine is so new, its compatibility with many DVI-to-PostScript drivers has yet to be determined; consult section 2.1 for details.

Dr. Craig Barratt wrote the original version of PSfrag as a student at Stanford University.[1] The interface has deviated very little since then, but the internals have been completely re-written. The current version of PSfrag is maintained by Michael C. Grant and David Carlisle. Many thanks go to Ted Stern, who also provides diligent assistance.

2 PSfrag necessities

In order to use PSfrag, you will need the following tools:

- A recent version of LaTeX 2ε and the graphics package. PSfrag currently requires the 1995/12/01 version or later of these packages, but it is always best to have the most recent release. PSfrag will now work with the old epsf macros, but the graphics package is still used internally.
- A compatible DVI-to-PostScript driver (see below). dvips is the primary choice of the PSfrag developers, and is certainly the most widely-used.

The latest versions of LaTeX 2ε, the graphics package, PSfrag, and dvips can all be found on CTAN, the Comprehensive TeX Archive Network. The CTAN cites, and their mirrors, include:

Name	IP address	Location
`ftp.dante.de`	129.206.100.192	Germany
`ftp.tex.ac.uk`	128.232.1.87	England
`ftp.cdrom.com`	165.113.58.253	USA

2.1 Choosing a PostScript driver

PSfrag relies on some sensitive PostScript tricks to accomplish its goals. Due to limited and time and resources, the authors could not confirm that PSfrag works properly on every available PostScript driver. We have attempted to insure that it will *eventually* work on every driver that is fully comaptible with the graphics package (*i.e.*, one for which a `.def` file is provided.)

The drivers which have been confirmed to work with PSfrag are:

Driver	Tested by	Compatibility
Thomas Rokicki's dvips	the authors	fully compatible
Y&Y's DVIPSONE	the authors	*Level 2 printers only.*

[1] Craig no longer participates in the development of PSfrag; however, heaps of praise may still be sent to him at `psfrag@rascals.stanford.edu`!

As the table shows, some drivers, like DVIPSONE, will work properly with PSfrag but will produce output that prints properly only on newer, *Level 2* PostScript printers. If you try to print such a document on a Level 1 printer using one of these drivers, the replacements will not show up. Hopefully, the incidence of Level 1 printers is diminishing, especially with the emergence of Level 3 PostScript on the horizon.

Please help us add entries to this compatibility list! If PSfrag works with your driver, please let us know, so we can add it to the list. If PSfrag does *not* work, please submit a bug report. Consult section 9 for contact information. If possible, test your PSfrag output on both Level 1 and Level 2 printers, so we can make a distinction here if necessary.

3 Installing PSfrag

Installing the various PSfrag files is quite simple:

1. Run LaTeX on `psfrag.ins` to extract `psfrag.sty` and `psfrag.pro`.
2. Install `psfrag.sty` in a standard location for LaTeX 2ε macros. For kpathsea-based systems such as teTeX, this path is determined by the `TEXINPUTS` variable.
3. Install `psfrag.pro` wherever your PostScript driver looks for header files. For kpathsea-based systems, this is determined by the `DVIPSHEADERS` varaible. For dvips in particular, the most logical choice would be the same directory in which `tex.pro` and `special.pro` are located.
4. If you have an older version of PSfrag, you may delete the following files, if they exist: `ps2frag.ps`, `ps2frag` or `ps2psfrag` (the processing scripts), and `epsf.sty` (the one provided by PSfrag, *not* the dvips version!). System managers may wish to replace `ps2frag` with a script which notifies users of the upgrade.

4 Usage

Here is a quick summary of the usage of PSfrag:

- Make sure that your EPS figure contains simple "tag" words in the same positions in which you would like the LaTeX replacements. Some effort has been made to allow for more arbitrary tag text; but it is still more reliable to use simple, short, alphanumeric tags.
- If you wish to use `epsf.sty` to include EPS files, it must be loaded with the `\usepackage` command *before* `psfrag.sty`. Other packages that are based upon `graphics.sty`, such as `graphicx.sty` or `epsfig.sty`, can be loaded before or after `psfrag.sty`.
- Load `psfrag.sty` with a `\usepackage` command.
- For each tag word in your EPS file, add a command to your your LaTeX document to specify how this tag is to replaced, as follows:

  `\psfrag{`*tag*`}[`⟨*posn*⟩`][`⟨*psposn*⟩`][`⟨*scale*⟩`][`⟨*rot*⟩`]{`*LaTeX text*`}`

The tag will be replaced by the LaTeX text. Certain complicated values for the tag word might confuse TeX so it's best to use simple alphanumeric names.

Example: in a drawing program like xfig, you place the text

`xy`

at a particular point. To replace this with $x + y$, one possible macro would be

```
\psfrag{xy}{$x+y$}
```

All `\psfrag` calls that precede the `\includegraphics` (or equivalent) in the same or surrounding environments will be utilized for a given PostScript file. So, you can define global `\psfrag`s as well as those that are local to a figure.

- Any text that is not mentioned in a `\psfrag` command will not be replaced; hence, PostScript and LaTeX text can be freely mixed.

When viewing the output with a DVI previewer such as dviwin or xdvi, a vertical list of the replacements will be placed on the left side of each figure. This list allows you to check the typesetting of your replacements; it disappears in the final PostScript version. Unfortunately, DVI drivers are incapable of *placing* the PSfrag replacements on top of the figure, so for that you will need to print it out or use a PostScript previewer like GhostView.

This version of PSfrag *should* run properly in the compatibility mode of LaTeX2.09. Let us know if you find otherwise (see section 9).

5 Commands and Environments

`\psfrag{`*tag*`}[`⟨*posn*⟩`][`⟨*psposn*⟩`][`⟨*scale*⟩`][`⟨*rot*⟩`]{`*replacement*`}`
`\psfrag*{`*tag*`}[`⟨*posn*⟩`][`⟨*psposn*⟩`][`⟨*scale*⟩`][`⟨*rot*⟩`]{`*replacement*`}`

The `\psfrag` macro defines which LaTeX-typeset text {*replacement*} is to be place at the same position as the PostScript {*tag*}. The command should be placed before the call to `\epsfbox`, `\includegraphics`, or equivalent. It matches *all* occurrences of {*tag*} in the figure.

A `\psfrag` replacement will remain in effect until its surrounding environment is exited. Therefore, you can define global `\psfrag`s which will apply to every figure, or define `\psfrag`s inside a single environment (*e.g.*, a `figure` environment) which apply to a single EPS file.

The optional positioning arguments [⟨*posn*⟩] and [⟨*psposn*⟩] specify how the bounding box of the LaTeX text and the bounding box of the PostScript text line up, respectively. Some drawing packages would refer to these as "control points" or "alignment points."

[⟨*posn*⟩] the LaTeX text reference point. The syntax of this argument is identical to that of the `\makebox` command. Up to two letters may be chosen, one from the list {`t`,`b`,`B`}, (top, bottom, baseline) and another from {`l`,`r`} (left, right). If a letter from either list is omitted, the behavior is to place the point in the center of the appropriate direction; Together, these specify one of 12 anchor points. If the argument is omitted altogether, then `[Bl]`, or left baseline positioning, is assumed; but note that supplying `[]` specifies centered positioning.

When running in LaTeX 2.09 compatibility mode, the default alignment is `[bl]`, in order to support legacy documents. For most text, however, this should not make an appreciable difference.

`[`⟨*psposn*⟩`]` the PostScript text reference point. The possible arguments are identical to that of `[`⟨*posn*⟩`]`, as is the default value, `[Bl]` (`[bl]` in LaTeX 2.09 compatibility mode.)

The LaTeX replacement text may be optionally scaled and rotated about its reference point:

`[`⟨*scale*⟩`]` Scaling factor (default 1). It's best if you use font size changes in the LaTeX text rather than scale, but you can use the scale to tweak its size. Default is `[1]`.

`[`⟨*rotn*⟩`]` Extra rotation of the text around its reference point, in degrees. The nominal rotation of the LaTeX text matches that of the PostScript text it replaces. The total rotation is this nominal value plus `[`⟨*rotn*⟩`]`. The default is `[0]`.

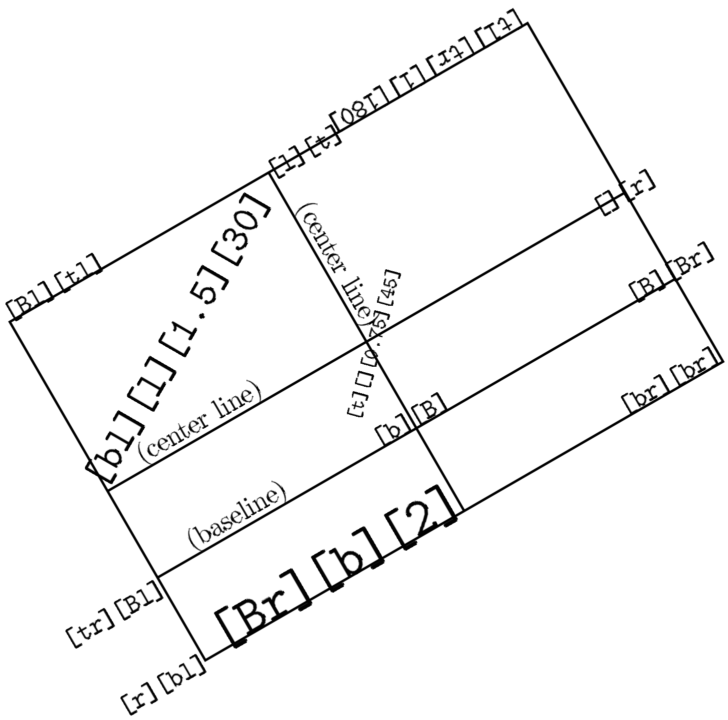

Figure 1: An illustration of various options for the `\psfrag` command.

Figure 1 provides an illustration of the effects of the use of various arguments above (and it happens to be a good exerciser for the package, too). If you're viewing the DVI file with a previewer such as xdvi, you should see the PSfrag replacements lined up to the left of the figure; and, if your previewer can display EPS files, a large, rotated `gA`. If you have printed this out, or are viewing the PostScript file with a viewer like GhostView, then the replacements should be properly superimposed on a graphical representation of the bounding box, center lines, and baseline of the tag `gA`. (This graphical box is provided only in debug mode.)

If a replacement for `{`*tag*`}` already exists, the unstarred command `\psfrag` will replace it without warning. The starred version `\psfrag*`, however, will *add* the new replacement to a list. Using the starred command, a single piece of PostScript text could trigger several replacements. I can't think of a reason why most users would use the starred version, but it was used in Figure 1 above.

```
\begin{psfrags} \end{psfrags}
```

The `psfrags` environment may be used, if necessary, to delimit the scope of the `\psfrag` calls. As we said before, `\psfrag` commands retain their effect until the most immediate surrounding environment is exited. *Any* environment will do: `center`, `figure`, *etc.*. Therefore, it may never be necessary to use this environment, and the environment has no other effect on the document.

5.1 Embedding **PSfrag** operations into EPS files

```
\tex[⟨posn⟩][⟨psposn⟩][⟨scale⟩][⟨rot⟩]{⟨LaTeX text⟩}
\psfragscanon  \psfragscanoff
```

PSfrag 3.0 supports the embedded `\tex` commands found in previous release of PSfrag, but it has been deprecated somewhat because of its reliance on a pre-processing step. Unlike previous versions of PSfrag, support for the `\tex` command must be *explicitly requested*, as described below.

As you can see, the syntax of the `\tex` command is very similar to the `\psfrag` command. However, instead of adding the `\tex` command to your LaTeX file, the `\tex` command is *embedded in the EPS file itself.* In other words, the command becomes its own replacement tag.

For example, you might place the text

```
\tex[bl][bl]{$\alpha$}
```

at a particular point in your PostScript file to have LaTeX replace it with α. Many PSfrag users find this feature useful for the axis labels and titles of MATLAB graphs.

The advantage to this approach is that changes can be made to the EPS file without having to modify any `\psfrag` commands in the LaTeX file. (It is still necessary to *re-compile* the LaTeX file in such cases, however.)

There are cautions and disadvantages to this approach, including:

- Changing the labels created by `\tex` commands requires editing the figure; if you use `\psfrag` instead, you need only to edit the document, which might be less cumbersome. (You must run LaTeX again in both cases.)
- Because `\tex` commands are long strings, they can extend past the other graphics in your EPS file. As a result, they can modify the EPS bounding box in an undesired way. This problem can be mitigated by reducing the font size of the `\tex` string, since this does not affect the size of its PSfrag replacement.
- The `\tex` command is not supported in compressed PostScript files.
- The TeX engine must scan the PostScript file for these strings, which can add to the processing time of your document. (To be honest, we have yet to encounter a case where this is a significant concern.)

- *Important!* Whenever a file is scanned by PSfrag, it generates a file with the name `\jobname.pfg`, where `\jobname` is the base name of the master LaTeX file. It will overwrite, without warning, any file with that name.

Since it was impossible to improve the `\tex` command in the same way as the `\psfrag` command, this feature is no longer enabled automatically except in LaTeX 2.09 compatibility mode. So, for LaTeX 2_ε documents, you must activate it in one of two ways:

- To turn on scanning for a single figure, precede the `\epsfbox` or `\includegraphics` command with a call to the command `\psfragon`. Scanning will be turned off again when the surrounding environment is exited; or, you can turn it off explicity with a call to `\psfragscanoff`.
- To turn on scanning for the entire document, pass the option `scanall` to `psfrag.sty` in the `\usepackage` command.

The `\tex` scanner will continue to be supported in this form. So, if you do find applications where you prefer the `\tex` command, do not hesitate to use it! For best results, however, we recommend that you use `\psfrag` exclusively instead.

6 Package Options

There are only four package options for PSfrag. Any other options that are not handled by PSfrag will be forwarded to `graphics.sty`.

`209mode` (LaTeX 2_ε native mode only) forces PSfrag to operate exactly as if LaTeX 2.09 compatibility mode was enabled. As a result, `bl` alignment is the default, and `\tex` scanning is enabled for all EPS files. This option is useful if you are trying to convert old LaTeX 2.09 documents to LaTeX 2_ε.

The LaTeX2.09 version of PSfrag generated an auxiliary file for each EPS figure containing important replacement information. These files are no longer used and can be deleted.

`2emode` (LaTeX 2.09 compatibility mode only) forces PSfrag to remain in LaTeX 2_ε mode, even in the presence of a LaTeX 2.09 document; this is the direct opposite of `209mode`. When enabled, the default alignment is `Bl`, and `\tex` scanning is turned off by default.

`scanall` turns on `\tex` scanning by default. Use this option if most your figures use embedded `\tex` commands.

`debug` turns on some of the debugging features of PSfrag. It inserts extra code into the PostScript file that draw the bounding boxes of each piece of text that is replaced. It is probably not useful to anyone but the developers of PSfrag.

7 An Example

In the following example, we demonstrate how to use PSfrag with the MATLAB package. The following MATLAB commands generate a plot of both a sine wave and a cosine wave, places both simple tags and `\tex` replacements into the figure, and saves the result as an EPS file `example.eps`.

```
t = 0:.1:10;
plot(t,sin(t),t,cos(t));
axis('square'); grid;
title('\tex[B][B]{Plot of $\sin(t)$ and $\cos(t)$}');
xlabel('\tex[t][t]{$t$}');
ylabel('\tex[B][B]{$\sin(t)$, $\cos(t)$}');
text(t(30),sin(t(30)),'p1');
text(t(60),sin(t(60)),'p2');
text(t(90),sin(t(90)),'p2');
tt=text(t(50),cos(t(50)),'p3');
set(tt,'HorizontalAlignment','center','VerticalAlignment',...
    'bottom','Rotation',atan2(-sin(t(50))*10,2)*180/pi);
print -deps example
```

(In MATLAB, the 'text' command defaults to a left-center alignment, corresponding to a `[`⟨*psposn*⟩`]` argument of `[l]`.)

The code below includes `example.eps` into the current document, resizing it to a width of 3.5 inches. Several `\psfrag` commands are used to replace the tags `p1`, `p2`, and `p3` in the figure, and the command `\psfragscanon` command is used to notify PSfrag that it must scan `example.eps` for the `\tex` tags.

```
\begin{figure}[tbh]
    \unitlength=1in
    \begin{center}
        \psfragscanon
        \psfrag{p1}[l]{\begin{picture}(0,0)
            \put(0.15, 0.2){\makebox(0,0)[l]{$\sin(t)$}}
            \put(0.1,0.2){\vector(-1,-2){0.1}}
            \end{picture}}
        \psfrag*{p1}[][l]{$\ast$}
        \psfrag{p2}[][l]{$\ast$}
        \psfrag{p3}{$\cos(t)$}
        \includegraphics[width=3.5in]{example.eps}
     \end{center}
     \caption{A \textsf{psfrag} example.}
\end{figure}
```

Note the use of a `picture` environment within the replacement for `p1`.

The result of these two steps is shown in Figure 2.

7.1 Figure scaling and resizing

There are two ways to resize EPS figures with the graphics package, and each has as different effect on PSfrag replacements. If you are used to using `epsf.sty`, you will be accustomed to only one such behavior.

If you use the `\scalebox` or `\resizebox` macros of `graphics.sty`, then the PSfrag replacments *will* scale with the figure. This effect is illustrated in 3 below. Figure 3 uses the following command to scale the figure to 3.5 inches in width:

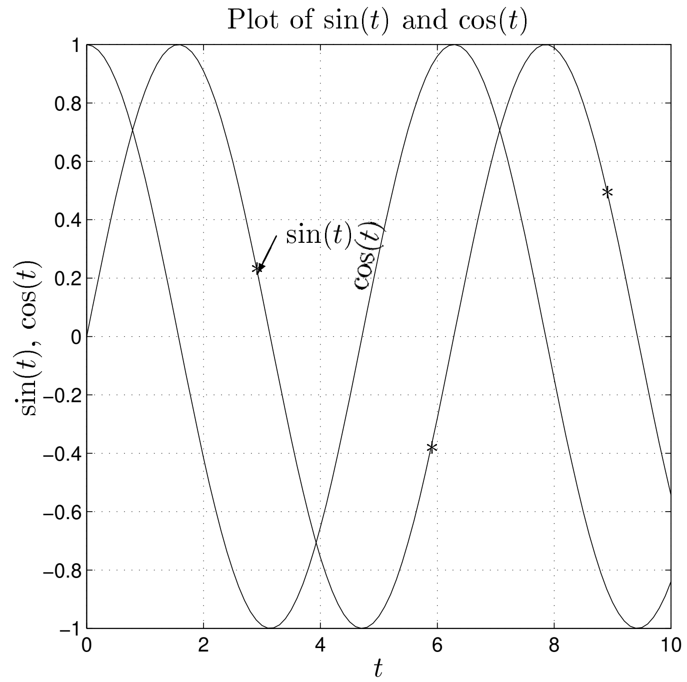

Figure 2: A PSfrag example.

```
\resizebox{3.5in}{!}{\includegraphics{example.eps}}
```

This is in direct contrast to Figure 2, which uses the `width=` keyword from the `graphicx.sty`, as follows:

```
\includegraphics[width=3.5in]{\includegraphics{example.eps}}
```

Figure 2 also reflects the behavior that you would see when using the `epsf.sty` macros `\epfxsize`, `\epsfysize`, *etc.* In these cases, the PSfrag text does not scale with it. to resize the figure.

As you can see, the text in the second figure is decidedly smaller than the first. This is because `\resizebox` uses PostScript tricks to scale *all* of the contents of its argument. Since the `\psfrag` commands are not actually typeset until *within* the `\includegraphics` command, they are resized as well.

The `graphicx.sty` key-value pairs `width=`, `height=`, and `scale=` scale the figure without scaling the replacement text, as long as they are supplied *before* an `angle=` rotation key. Of course, the `\resizebox` and `\scalebox` macros are still available in `graphicx.sty`, so you can mix and match both behaviors as you see fit. See the graphics documentation for more details.

If you are still unsure about these distinctions, then try both methods for scaling your figures until you find a convention that works best for you.

8 Known problems

PSfrag is bug-free.

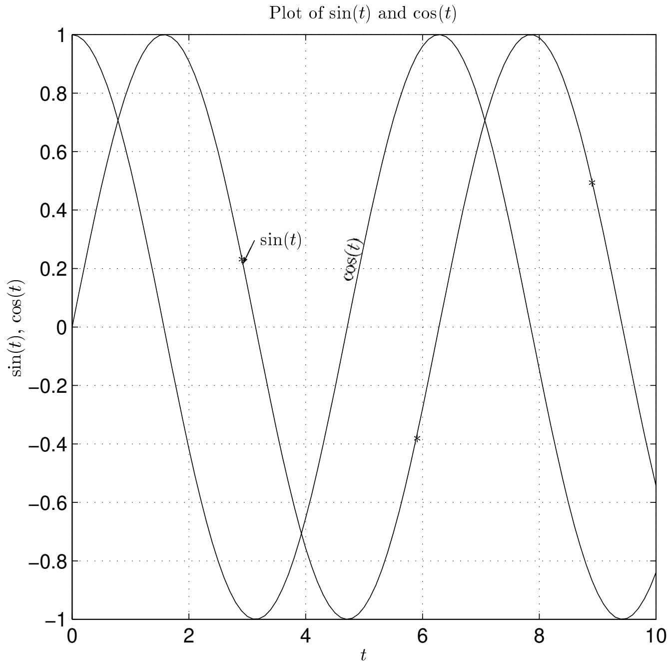

Figure 3: The same PSfrag example as Figure 2, using `\resizebox` to set the width.

Well, of course we're kidding. If you find any problems, please confirm they are not mentioned below; and, if not, report them to te PSfrag mailing list (see below).

The PostScript text must be displayed using a single call to `show`, `ashow`, `kshow`, `widthshow`, or `awidthshow`. Some programs, however, place each character individually, with individual calls to `show`. This technique is incompatible with PSfrag, and always will be. Experience has shown that this excludes only a very small number of drawing packages.

The LaTeX replacements are rendered after the figure has been drawn; therefore, effects such as partial obscuring are not possible.

9 The PSfrag mailing list

There is a Majorodomo mailing list for purposes of PSfrag maintenance. It *is not* intended to replace this manual or a small amount of educated guesswork. But, it *is* the perfect place for bug reports, development ideas, and so forth. Anyone who wishes to assist in PSfrag's evolution may subscribe; to do so, just send mail to

`majordomo@rascals.stanford.edu`

with the line `subscribe psfrag` in the *body* of the text.

Bug supports, ideas, *etc.* should go to

`psfrag@rascals.stanford.edu`.

If you have found a bug to report, please provide us with the necessary files (a LaTeX file, the EPS figures, *etc.*) so we can test it out ourselves! Try to provide us with the shortest self-contained example that demonstrates your bug. If this is not possible, drop us a line first.