The PSfrag system

Craig Barratt [*] Michael C. Grant [†]

10 March 1995

Contents

1 What is PSfrag?

Many drawing and graphing packages produce output in PostScript, but do not support the inclusion of equations and other scientific text of which LaTeX is capable. Likewise, many LaTeX users simply find the various LaTeX graphics packages too clumsy, and prefer the familiar GUI of a PostScript-generating graphics tool.

PSfrag provides the best of both worlds, by allowing the user to replace arbitrary text in Encapsulated PostScript (EPS) files with arbitrary LaTeX constructions. For example, one can

[*] Craig, the original author of PSfrag, is no longer participating in its development. Heaps of praise may still be sent to `craig@arraycomm.com`.

[†] Bug reports, suggestions, *etc.* go to `psfrag@rascals.stanford.edu`. See section 11.

place a simple text "anchor" in the graphics file to denote the position of a desired LaTeX equation, and PSfrag will automatically remove that anchor and replace it with a properly sized, aligned, and rotated LaTeX equation.

2 PSfrag necessities

In order to view documents whose figures have already been processed by PSfrag, then you must have the following tools:

- LaTeX 2_ε and the graphics package. Note that to achieve the same functionality as found in the original LaTeX 2.09 version of PSfrag, it will be necessary to use `graphicx.sty`, or the `epsf.sty` or `epsfig.sty` wrappers for `graphicx.sty`. See section 9.3 for a full explanation.
- A DVI-to-PostScript driver that is well-supported by the graphics package. Specifically, this means that a `.def` file must have been written for that package which properly defines the macro `\Gin@PS@raw` for your driver. See that package's documentation for details.

  Currently, `dvips` is the best-supported such driver.

If you wish to generate new PSfrag figures, you will also need the following utilities:

- Your favorite drawing or graphing program, which must be able to output your figures in Encapsulated PostScript format.
- Perl 4.036 (or older) to run the `ps2frag` processing script supplied with PSfrag. The Perl version was originally created by Piet Tutelars (`rcpt@urc.tue.nl`), and greatly improved the portability of PSfrag, since Perl is available for many platforms.[1] For MS-DOS users, a batch file `ps2frag.bat` is provided which will run the script for you.
- Ghostscript, a PostScript interpreter which `ps2frag` uses to locate and document the positions of the text in an EPS file. It is free for personal use and available for most platforms.

The latest versions of LaTeX 2_ε, the graphics package, PSfrag, and `dvips` can all be found on CTAN, the Comprehensive TeX Archive Network, composed of three FTP sites:

Name	IP address	Location
`ftp.dante.de`	129.206.100.192	Germany
`ftp.shsu.edu`	192.92.115.10	Texas, USA
`ftp.tex.ac.uk`	128.232.1.87	England

Both Perl and GhostScript can be found at many of the major FTP sites, probably including `gatekeeper.dec.com`, `ftp.uu.net`, `src.doc.ic.ac.uk`, and others.

[1] The C-version of `ps2frag` is not supported for LaTeX 2_ε, because of the changes to the way the script must operate.

3 Installing PSfrag

Installing the various PSfrag files is a bit more complicated than installing most macro packages.

1. Run LaTeX on psfrag.ins to extract psfrag.sty, epsf.sty, and ps2frag.ps. It will ask you a single question concerning the way your system handles long file names.

2. Install psfrag.sty and epsf.sty in a standard location for LaTeX 2ε macros. epsf.sty will serve as a drop-in replacement for both epsf.sty and epsfrag.sty for LaTeX 2ε. (Actually, it does nothing more than load epsfig.sty from the graphics package.)

3. Install ps2frag.ps somewhere in your TeX tree, preferably with your DVI-to-PostScript driver files (*e.g.*, /usr/local/lib/texmf/dvips).

4. Edit the ps2frag script. You will need to change (or confirm the correctness of) the following things:

   - the type of platform you have. Currently, the only two supported types are 'MSDOS' and 'UNIX'.
   - the method the &fragfile command uses to determine the name of a LaTeX 2.09 PSfrag file. For example, the usual method in Unix is to simply add frag to the end of the name; in MS-DOS, the file extension is replaced with .fra.
   - the value of the $PS2FRAG variable, which is the location of the file ps2frag.ps.

   Reasonable defaults and customization instructions are provided in the script itself. You shouldn't need to be a Perl whiz to make these customizations.

5. If you are running UNIX, install ps2frag somewhere in your path, such as /usr/local/bin. If you are using MS-DOS, edit the ps2frag.bat script to reflect the desired location of ps2frag and Ghostscript, and install ps2frag.bat somewhere in your path.

6. (optional) Install the man page for the ps2frag script, called ps2frag.1, into your local man tree (*e.g.*, /usr/local/man).

The ps2frag script supports the generation of old-style LaTeX 2.09 PSfrag information files through the -209 command-line flag. So, it is reasonable to completely replace the old versions of this script and accompanying ps2frag.ps file, even if you need to support both LaTeX 2ε and LaTeX 2.09 documents.

4 Usage

Here is a quick summary of the usage of PSfrag:

- psfrag.sty loads graphics.sty by default. If you prefer to use another package, such as graphicx.sty, epsfig.sty, or epsf.sty, load it with \usepackage in the preamble. Only packages which use graphics.sty will work with PSfrag, however.

- Load psfrag.sty with a \usepackage command.

- For each PostScript file you wish to include, run the ps2frag script:

  ```
  ps2frag example.eps
  ```

This uses Ghostscript to generate the information necessary for PSfrag processing, and embeds this information in the preamble of the EPS file. This will not alter the appearance of the document in any way, because this information appears as a comment to any PostScript interpreter. Specifically, it is represented as an extended Document Structuring Convention comment in the header of the EPS file, where the `%%BoundingBox` comment is also located.

Every time an .eps is updated, `ps2frag` must be re-run. Makefiles can simplify this process! The `ps2frag` script will not re-process any file that already contains PSfrag comments (unless the `-f` flag is supplied), so running `ps2frag *.eps` will process only those files in the current directory which have not yet been processed.

- Use one of following two methods for replacing PostScript text with LaTeX fragments. Both methods may be used in the same document. Further description of the commands is given in Section 5.

Method 1: In the EPS figure, insert actual PostScript text that looks like

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

PSfrag will replace this entire text with a typeset version of `{`*LaTeX text*`}`. 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. Explanation of the optional arguments is deferred to Section 5.

Method 2: In the EPS figure, place a simple "tag" word in the same position in which you would like the LaTeX replacement. In your LaTeX document, 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. Note that certain complicated values for "PS text" 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.

Method 2 has the distinct advantage that you can use EPS files which were designed without consideration for PSfrag. In addition, long `\tex` macros embedded in an EPS file may distort the bounding box of the figure. Using Method 1, on the other hand, allows text to be modified in the EPS file without forcing changes in the LaTeX source as well.

- Any text that is not specified by a `\psfrag` or by `\tex` will not be replaced; hence, PostScript and LaTeX text can be freely mixed.

Please see Section 7 for illustrations of both methods.

5 Commands and Environments

`\tex[⟨posn⟩][⟨psposn⟩][⟨scale⟩][⟨rot⟩]{LaTeX text}`
`\psfrag{name}[⟨posn⟩][⟨psposn⟩][⟨scale⟩][⟨rot⟩]{LaTeX text}`

These macros declare the LaTeX text to be placed at the same position as the PostScript marker. The first form is embedded directly in a PostScript text command (`show`, `ashow`, `widthshow`, `awidthshow`, `kshow`), and serves as its own marker. The second form is placed in a surrounding LaTeX environment before the `\includegraphics` (or equivalent) call, and uses the text in `{`*name*`}` as the PostScript marker. (It matches *all* occurrences of `{`*name*`}`, so multiple substitutions can take place.)

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.

Note: In the LaTeX 2.09 version, the default was `[bl]`, and baseline alignment was not supported. For most text, however, this should not appreciably affect the output.

`[`⟨*psposn*⟩`]` The PostScript text reference point. The possible arguments are identical to that of `[`⟨*posn*⟩`]`, as is the default value, `[Bl]`.

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 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). The graphical square denotes the bounding box, center lines, and baseline of an exaxmple piece of replaced text. PSfrag was used to place several LaTeX replacements. In each case, the replacing LaTeX code is actually the set of arguments to produce that particular alignment, rotation, and scaling, encased in an `\fbox`.

```
\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.

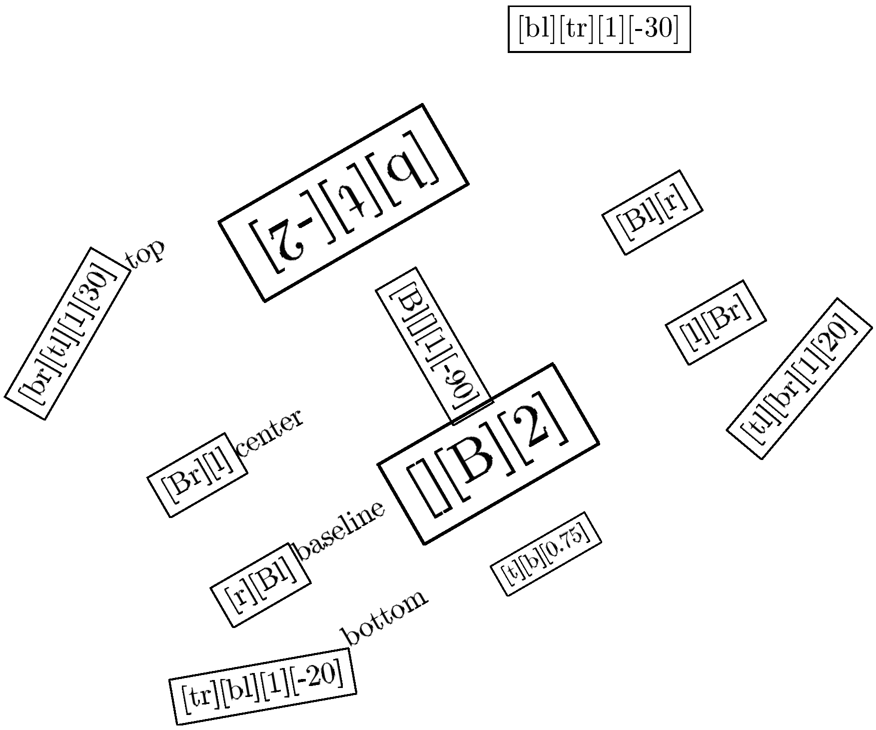

Figure 1: An illustration of various alignment, scaling, and rotation combinations for the `\psfrag` and `\tex` commands.

6 Package Options

There are only three package options for PSfrag.

`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 `Bl`, and PSfrag expects to find the text positioning information in a separate `-frag` file, instead of in the header of the EPS file itself.

This option simplifies the conversion of old documents to LaTeX 2_ε, allows for easy support of LaTeX 2_ε documents that were created before this package was released, or which share figures with LaTeX 2.09 documents. However, it's also just as easy to re-run the `ps2frag` processing script on all of the figures. Plus, the PSfrag comments in the header of the EPS file can easily coexist with those in the `-frag` file, allowing both LaTeX 2_ε and LaTeX 2.09 documents to access and support PSfrag figures in their native mode.

`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 PSfrag looks for text positioning information in the header of the EPS file.

This option should allow people to use the new features of PSfrag when other factors prevent them from using native LaTeX 2_ε mode, such as the need to use old LaTeX 2.09 style files. However, if you are using LaTeX 2.09 compatibility mode because you need to share your document with someone who cannot run LaTeX 2_ε, you must avoid using this option!

showall disables the hiding of the PSfrag tags in the EPS files. This might be useful if you wanted to investigate a potential bug in your `\psfrag` commands or in `psfrag.sty` itself (perish the thought!). When the resulting document is viewed, both the PostScript tags and PSfrag replacements will appear.

7 Examples

A typical figure declaration might look like this:

```
% uses graphicx.sty commands
\begin{figure}[tbh]
\begin{center}
    \psfrag{u}[b][tr]{$u = 1 + \gamma$}
    \psfrag{y}[]{$\hat{y}$}
    \psfrag{z}[][]{output $z$}
    \includegraphics[width=4in]{example.eps}
\end{center}
\caption{This is an example.}
\end{figure}
```

This replaces the PostScript text "u" with $u = 1 + \gamma$. The middle-bottom of $u = 1 + \gamma$ is placed at the top-right of "u". Similarly, center of the $\hat{y}$ is aligned with the left of the baseline of the PostScript text "y" (the default for psposn), and the center of z is aligned with the center of the PostScript text "z". The PostScript file is centered with an x width of 4 inches.

Embedded replacements can be performed in MATLAB (for example) with:

```
t = 0:.1:10;
plot(t,sin(t),t,cos(t));
axis('square'); grid;
title('\tex[B][B]{Plot of $\sin(t)$}');
xlabel('\tex[t][t]{$t$}');
ylabel('\tex[B][B]{$\sin(t)$}');
text(t(30),sin(t(30)),'p1');
text(t(60),sin(t(60)),'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 embedded `\tex` commands specify alignment points appropriate for each label. The LaTeX file might say:

```
\documentclass[12pt]{article}
\usepackage{graphicx,psfrag}
\begin{document}
    This is a test.
    \unitlength=1in
    \begin{center}
        \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{p2}[c][l]{$\ast$}
        \psfrag{p3}[B][B]{$\cos(t)$}
          \includegraphics[width=3.5in]{example.eps}
    \end{center}
\end{document}
```

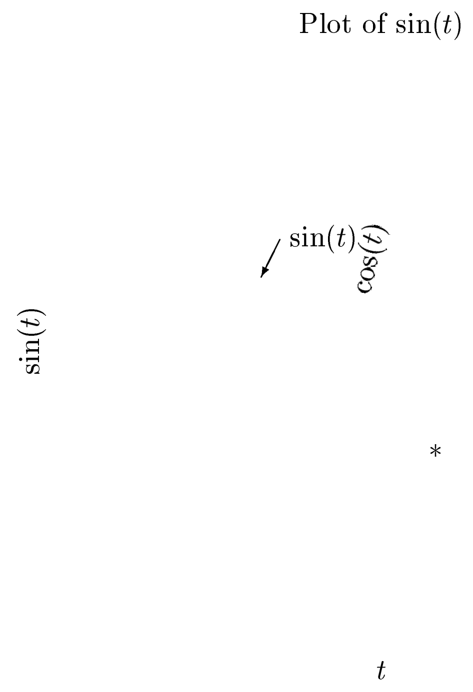

Figure 2: A PSfrag example.

This replaces "p1" with a small picture, "p2" with an asterisk, and "p3" with some text. The output of this example looks something like Figure 2.

8 Using the ps2frag script

`ps2frag [-f] [-209] [-2e] [-x` *pts*`] [-y` *pts*`]` *files...*

The `ps2frag` script is relatively simple to use. Its default behavior will be almost always sufficient. `ps2frag` does have several command-line options, however, which enable you to modify the behavior of the script.

`-f` forces `ps2frag` to update the information in all of the named EPS files, even if they have already been processed. If you have hand-modified an EPS file, you will need to use this option to force `ps2frag` to process it again.

`-209` invokes LaTeX 2.09 mode. In this mode, the text information extracted from the EPS file is placed in a separate PSfrag file with the same base name, as was done in the LaTeX 2.09 version of PSfrag. This option is supplied to allow old documents to be maintained with the same script.

-2e disables LaTeX 2.09 mode. If ps2frag has been configured to run in LaTeX 2.09 mode by default, this option will provide LaTeX 2ε behavior. You can determine the default behavior by running ps2frag with no arguments; if LaTeX2.09 mode is the default, it will respond with

```
This is ps2frag, version 2.0
LaTeX 2.09 compatibility mode enabled.
usage: /local/bin/ps2frag -209 [-f] [-209] [-2e] [-x ⟨pts⟩] [-y ⟨pts⟩] files...
```

-x,-y specify displacements, in points, in the X and Y directions that you wish to add to each text fragment. Some drawing programs add small offsets that cause the actual positions of the text to differ from that which was specified by the user. For example, when idraw is used with "grid gravity" turned on, it shifts 12-point text upwards by 8.33 points; so, for such files, you might want to specify -y -8.33.

You may supply as many file names to a single ps2frag command as you wish. By default, it will processs only those files which do not already have PSfrag information in their headers, quickly skipping over the rest. This allows you to update an entire directory of figures very quickly by simply running ps2frag *.eps.

If you are running ps2frag on a system which can handle long file names, consider linking or copying the script to the name ps2psfrag as well. By default, if ps2frag is run under the name ps2psfrag, it activates LaTeX 2.09 compatibility mode. For many sites, this will allow for smooth support of both modes.

9 Deviations from LaTeX 2.09

Because so many things are changing with LaTeX these days, what with the release of LaTeX2e and its enhanced graphics package, I felt that a major rehaul of PSfrag was justified. Almost all of the internals have been completely rewritten, and much of the external interface has changed.

9.1 Comment-based fragment information

One significant difference is that ps2frag now deposits the fragment information into the document structuring comments of the EPS file itself, instead of placing them in a separate file. They have been structured to appear as comments to PostScript (and to any standards-conforming viewer/processor), but are in fact embedded PSfrag commands. I hope people find this convenient, after an inevitable initial "shock."

Of course, if one must use the old method for some documents, the 209mode package option is provided. I do believe that it should be used only as an exception to the rule, however.

9.2 Using the new graphics package

Because of the LaTeX 2ε upgrade, the the graphics calls used to load the figures have completely changed. In LaTeX 2.09, a modified version of dvips's epsf macros were used; this limited its usefulness to dvips users, and proved a maintenance hassle, since new versions of dvips with clean copies of epsf.sty were released much more frequently than updates to PSfrag.

This version of PSfrag however, attempts to be well-integrated with the standard graphics package. Currently, it does require the ability to pass direct PostScript; since graphics.sty does not provide a standard way to perform this, `dvips` is the only DVI-to-PostScript driver that PSfrag currently supports.

In the future, when `graphics.sty` provides a "raw PostScript" macro, PSfrag should work on a much wider variety of dvi-to-PostScript drivers.

9.3 Figure scaling and resizing

There is one important caveat that must be heeded when using the `graphics.sty` package, especially if you have used PSfrag before.

In the past, when using the epsf macros `\epfxsize`, `\epsfysize`, *etc.* to resize the EPS figure, the PSfrag text did not scale with it. If you use the `\scalebox` or `\resizebox` macros of `graphics.sty`, then the PSfrag replacments *will* scale with the figure.

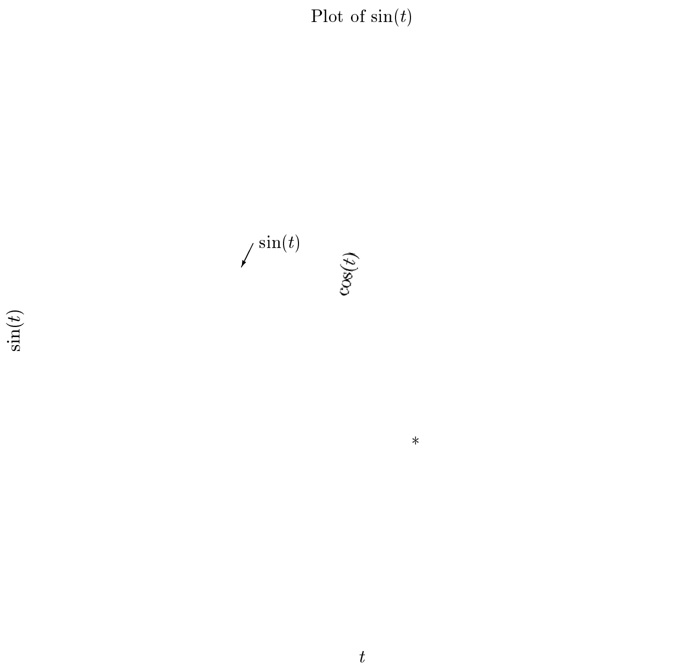

Figure 3: The same PSfrag example as Figure 2, using `\resizebox` to set the width.

If this distinction isn't clear, consider Figure 3, which imports the same EPS file as Figure 2, but uses the command

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

to resize the figure. For comparison, Figure 2 uses

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

As you can see, the text in the second figure is decidedly smaller than the first. The `\resizebox` command 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=` will all scale the figure without scaling the replacement text, as long as they are supplied *before* an `angle=` rotation key. figure. 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.

I suspect that everyone who prefers the behavior of the old PSfrag will be quite happy simply using the key-value options in `graphicx.sty`. Some may suggest, however, that the text scaling produced by `\scalebox` and `\resizebox` is *precisely* the way PSfrag should have always behaved. Now, *both* behaviors are supported.

9.4 The `ps2frag` script

In previous versions of PSfrag, the processing script was called `ps2psfrag` and was written in the UNIX C-Shell. In order to support non-Unix platforms, the script was re-written in Perl, and its name shortened to `ps2frag`.

The `ps2frag` script supports both LaTeX 2.09- and LaTeX2e-style processing, so you can safely use it for both old and new documents. If you are running `ps2frag` on Unix, you may wish to symbolically link it to `ps2psfrag` as well. The script will recognize that it is being called by the old name, and automatically invoke LaTeX2.09 compatibility features. Otherwise, the `-209` option will enable these features. See Section 8 for details.

10 Bugs

The PostScript text must be displayed using a single call to `show`, `ashow`, `kshow`, `widthshow`, or `awidthshow`. More esoteric methods employed by some PostScript-generating programs will not work with psfrag.

This is not a problem with MATLAB, Xmath, xfig, and idraw, which all use `show`. Aldus Freehand uses `awidthshow`, so it is also fine.

LaTeX fragments always appear on top of the EPS file.

Some of the improvements in bounding-box handling may affect the way LaTeX 2.09 documents are processed. In addition, a rare bugs in the old `ps2frag.ps` file have been corrected, which will change the way some EPS files are proceesed.

11 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's 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

`psfrag-request@rascals.stanford.edu`

with the single word `subscribe` in the *body* of the text.

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

`psfrag@rascals.stanford.edu`.

If you have a bug to report, please provide us with an example that we can run LaTeX on it ourselves! Please provide us with the shortest self-contained example that demonstrates your bug. If this is not possible, drop us a line first.