The `swfigure` package
Managing large and spread wide figures

Claudio Beccari*

Version v.0.9.14 — Last revised 2020-11-10

Contents

1 Introduction **1**

2 Installation **3**

3 Usage **3**

4 Display mode peculiarities **5**

5 Acknowlegements **6**

6 The code **8**

6.1 The `\SWfigure` macro for Spread Wide images . 11

6.2 The `\NFfigure` macro for normal full page figures 13

6.3 The `\RFfigure` macro for rotated figures 13

6.4 The `\VSfigure` macro for tall and slim figures . 14

6.5 The `\HSfigure` macro for spread wide slim figures 15

6.6 The `\FSfigure` macro for Full Spread figures . . 16

Abstract

This package defines a single command that with different options can insert large images into the document; those that occupy an entire spread s get split into two halves that are inserted on two facing pages in such a way that they merge when the document is read on the screen (set to double page view) or is printed in a well sewn book so that facing pages join correctly at the spine. This documentation is accompanied by another file `swfigure-examples.pdf` containing examples of this package several uses.

1 Introduction

Sometimes it is necessary to insert in a given document very large images; either they are larger than a spread, or they do not exceed the width of a spread. In the first case it is necessary to use to large sheets of paper folded in the proper way and inserted in the printed document as special inserts; or for documents to be read on the screen such large sheets have to be attached to the document as separate files to be viewed in windows different from that where the document is being read.

In the other case it is possible to manage such large figures in different ways; we define a single command that, according to different settings, can insert such figures in six different display modes.

*E-mail: `claudio dot beccari at gmail dot com`

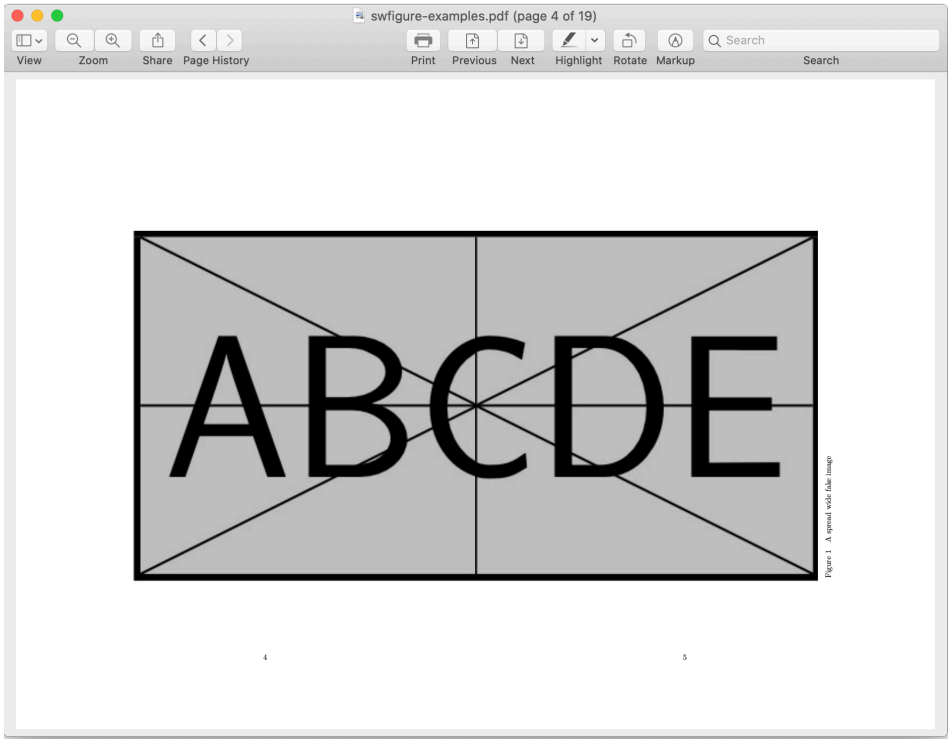

Figure 1: Example of a large fake image that occupies a spread; screenshot of a spread view from the companion file swfigure-examples.pdf.

1. The normal LaTeX kernel full page display mode, but using the `figure` environment with the `[p]` placing option. Apparently there is no need to define a new command for this display mode, except the advantage of using the same command as the other modes and a few small further functionalities that the new command has available.
2. The landscape display mode available with the `lscape` package and other similar ones. Again this display mode appears superfluous, but the advantage is to use the same command as with the other display modes
3. The screen-wide display mode, where two facing pages display a large figure divided in two halves, each one set in its page shifted to the spine so as to occupy on each page the text width plus the internal margin width; with suitable aspect ratio, both facing pages are practically full. No other text appears in this display mode, except the caption in the right page external margin.
4. The slim screen-wide image is similar to the previous mode, but since the figure is slim, it is set at the top of two facing pages, with text underneath; the caption is under the right figure half in the right page, and the left text block height is set so that that the first lines of both text blocks are aligned.
5. The slim tall image is set to the side of the text block that wraps it. Recurse is made to the `wrapfig` package, therefore the software might sometimes hick-up a little bit, because of the idiosyncrasies of `wrapfig` that performs very well in most but not all circumstances; see that package documentation

for such `wrapfig` limitations. With this `wsfigure` package we limit the use of this display mode to images that have a "height over width" ratio not lower than 2; we provide also some option arguments so as to correct small imperfections in the sizing of the text indention

6. A very large image, typically A3 sized, that occupies a full spread including the lateral margins; it is a solution to inputting, for example, a line drawing of a technical device, that otherwise should be attached as a separate document; a very large table that does not fit the usual A4 paper and that cannot be reduced by means of `longtable` or scaled down to unreadable font sizes.

In any case, depending on the page geometry and the image aspect ratio, it is very handy to have available a single command that changes the display mode by just changing a single input optional argument. In facts the user might start with one of the six described display modes; after examining the document daft the user might chose another display mode, and it suffices to change the optional argument, without changing the whole code.

WARNING This package performs as expected with documents typeset in `twoside` mode, and with a page design where the internal margins of both the odd and even pages are equal (symmetrical page design). For example, it does not work with this document designed to work in `oneside` mode and where the the left page margin is always larger than the right one of each page; this page design is functional to the documentation of the TEX system software.

2 Installation

This package is already installed with any complete and up to date TEX system distribution, TEX Live or MiKTEX.

Should the user have available a basic or a partial TEX system installation, the simplest way to install this package is to download the `swfigure.zip` file from the Comprehensive TEX Archive Network (CTAN), and decompress it either in the very folder where there is the document main or single document file or, for a general use, in the user `texmf` personal tree; it might be necessary that the user should directly create this personal tree; how to do it is described in the documentation of the user TEX system distribution.

The same holds true if the user employs a vintage TEX system distribution; this package requires the LATEX3 modern language functionalities, therefore a LATEX kernel with a date after 2019-01-01. Lacking this modern kernel, the package does not work, because packages `xparse` and `xfp` are used for its internal workings.

3 Usage

This package defines a single user command/environment, `DFimage`. The command is usable when the image fills up the page or the spread, so that it contains no text, except the caption; the environment is recommended when the large image is in a page or a spread that contains also some text.

Warning! This particular environment cannot be nested and cannot contain a `\DFimage` command.

The syntax is the following:

```
\DFimage[⟨display mode⟩]{⟨image file name⟩}%
     [⟨lof entry⟩]{⟨caption⟩}[⟨label⟩]%
     (⟨height correction⟩)<⟨line
correction⟩>\meta{width test}!⟨precaption⟩!

or

\begin{DFimage}[⟨display mode⟩]{⟨image file name⟩}%
     [⟨lof entry⟩]{⟨caption⟩}[⟨label⟩]%
     (⟨height correction⟩)<⟨line
correction⟩>\meta{width test}!⟨precaption⟩!
⟨environment textual contents⟩
\end{DFimage}
```

where:

⟨*display mode*⟩ is one of the following uppercase acronyms: `NF` (Normal Figure), `RF` (Rotated Figure), `SW` (Spread Wide image), `HS` (Horizontal Spread-wide image), `VS` (Vertical Slim image), and `FS` (Full Spread image).

⟨*image file name*⟩ is the name of the image graphic file; remember that the LaTeXbased TeX system typesetting programs accept graphic files in the formats with extensions `.pdf`, `.eps`, `.jpg`, `.png`, and few other less common ones. It is not necessary to specify the extension, but it is not forbidden.

⟨*lof entry*⟩ is the optional entry to the List Of Figures; it defaults to the ⟨*caption*⟩ text, but if the latter is sort of lengthy, it is better to enter a shorter text in that list.

⟨*caption*⟩ is the caption text.

⟨*label*⟩ this optional argument is the string that forms the `\label` command argument; of course, if this argument is not specified, the figure number and its page cannot be referenced with `\ref`, `\pageref` and other similar commands.

⟨*height correction or color*⟩ is a round parenthesis delimited optional argument; as a *height correction* it is a fractional number lower than 1 (default 0.8) with which to further scale the scaled image height to be included; it is used by some display modes, and it is ignored by others.

⟨*line correction*⟩ is an angle bracket delimited optional argument preset to zero. It is relevant only with the slim vertical image display mode. Sometimes the `wrapfig` package indents the wrapping text in such a way as to leave to much or too little vertical space around the image and its caption; by examining the document drafts it is possible to correct the predetermined number of lines by increasing or decreasing the vertical space by any (integer) number of lines.

⟨*width test*⟩ this further vertical bars-delimited argument is used only when dealing with vertical slim images; when they are scaled down in order to fit in the available space, their width may become too small to allow a decently typeset caption below the image. A test is made to compare the scaled image width with a fraction of the text block width in order to skip the image insertion if the image width becomes too small; the default value for this ⟨*width test*⟩ fraction is `0.25`, but the user can specify a different value, even zero; with the zero value this width test is skipped.

⟨*precaption*⟩ is an optional color declaratio delimited by exclamation points; the default value is 'empty'; it is used only by the `FS` display mode in order to

typeset a caption, for example, with a contrasting color over the background image average color, and/or with a displacement of the first or only caption line, and/or with a different font.

4 Display mode peculiarities

Each of the listed display modes has its own pros and cons. In the following the phrase "aspect ratio" refers to the "height over width" ratio either of the image or of the available space where to put the image and possibly its caption.

`NF` This display mode is convenient when the image aspect ratio is close to the text block one. Of course the user does not know in advance the vertical space occupied by the caption, therefore the optional ⟨*height correction*⟩ comes handy for small adjustments; this display mode is fully and freely floating, although its positioning option is fixed to `[p]`, an only-float page. Therefore don't use it it the image is not really such as to occupy most of text block area.

`RF` This display mode is convenient when the image aspect ratio is close to the reciprocal of the text block one. This mode accepts the ⟨*height correction*⟩ in order to leave space for the caption. It is a fully and freely floating object with the same pros and cons as the `NF` display mode.

`VS` This mode is convenient for tall and slim figures with aspect ratio not lower than 2; but for obvious reasons, it should not be too large, let's say, not larger than about 3 or 4. As always this depends on the page design and the caption size. The limitations of the underlying `wrapfig` package forbid its usage too close to explicit lists and texts typeset in special modes with a different measure from normal text. Again it is up to the user to chose where to insert the `DFfigure` environment or the `\DFimage` command. The `wrapfig` package documentation and our experience, show that the best position is just before a new paragraph; the environment end should be placed after a suitable number of full paragraphs, even if not all of them are involved with wrapping.

If the aspect ratio of the image to include is lower than 2, the following message is printed in the document where the image should appear (of course with `#1` replaced by the actual image file name):

```
========================================
Figure #1 is not tall enough.
Consider using a display mode different
from VS; may be NF or RF are better suited.
========================================
Nothing done!
========================================
```

A similar action and a similar message is output if the *scaled* image width becomes too small.

`SW` This display mode is convenient when a really large image requires two pages to display all its details; its aspect ratio should be close to the aspect ratio of the total space available for the two page spread; depending on the page design this total space approximate value is probably around 0.5. The object to be included in the document should start on an even page, therefore it floats to the first even page available; this may force a page break where

the current page is not fully completed and may be shorter than the other text pages; the user then should help the automatic positioning by choosing very carefully where to insert the `\DFfigure` command in the source text; possibly the user should chose an end of paragraph close to the end of a page.

`HS` This display mode is convenient when the initial image aspect ratio is very small; if it is smaller than 0.3 its insertion leaves enough space beneath the two image halves (plus caption) so that normal text can fill the space under such spread wide slim figure. The constrains described for display mode `SW` apply also to this mode.

`FS` The `SW` mode occupies a spread without invading the lateral margins; it simply occupies only the internal margin. On the opposite, the `FS` Full Spread mode occupies also the lateral margins; typically a full spread of two A4 pages, can contain a rotated A3 page, without scaling down its contents. With this full occupation of both facing pages, there is no room for any caption; therefore the actual image, besides occupying an A3 page, should have a minimum white margin around, so that a caption can be superimposed to the white part if its original bottom margin that, upon rotation, becomes the right one. If the original image has no margins, the caption may be typeset with contrasting different color from the default black one.

5 Acknowlegements

This package would not exist if Heinrich Flech did not need to insert several types of large images in his documents; he asked me to create suitable commands and/or environments and he tested all the series of progressive developments of this package; believing that this package might be useful to other TEX users, he invited me to submit the result to the TEX community, and here it is. Heinrich did not want his name as an author, because he insists that he was "just" a tester. He was precious and I warmly thank him.

Thanks also to Benedict Wilde who spotted some errors and suggested the use of specific settings for the PDF viewer.

Thanks also to the unknown person nicknamed "dylan.bacc" on the GUIT forum, who induced me to work on the sixth display mode. I knew that using the `memoir` class, with suitable settings, and having available an advanced printer, it was possible to print documents with pages of different formats and page designs; but this was not my aim: I wanted something possibly independent from the document class and not requiring any particular printer.

References

[1] The LaTeX3 Project, *The `xparse` package* — Document command parser. Release date 2020-05-15. PDF document readable with `texdoc xparse`.

[2] The LaTeX3 Project, *The `xfp` package* — Floating Point Unit. Release date 2020-05-15. PDF document readable with `texdoc xfp`.

[3] The LaTeX3 Project, *The LaTeX3 interfaces*. Release date 2020-09-24. PDF Document readable with `texdoc interface3`.

[4] D.P. Carlisle and The LaTeX3 Project, *Packages in the 'graphics' bundle*. Release date 2020-08-21. PDF document readable with `texdoc grfguide`.

[5] P. Lehman and J Wright, *the `etoolbox` Package* — An e-TeX Toolbox for Class and Package Authors. version 2.5j, released on 2020-08-24. PDF document readable with `texdoc etoolbox`.

[6] D.P. Carlisle, *The `lscape` package*. Version 3.02 released on 2020-05-28. PDF document readable with `texdoc lscape`.

[7] D.P. Carlisle, *The `afterpage` package*. Version 1.08 released on 2014-10-28. PDF document readable with `texdoc afterpage`.

[8] D. Arseneau, *The `wrapfig` package*. Version 3.6 released on 2003-01-31.

[9] P. Wilson, *The Memoir Class for Configurable Typesetting* —User guide. Version 3.7m released 2020-09-10. PDF document readable with `texdoc memman`.

6 The code

The required TEX format and is date, and the identification of this package have already ben inserted by the initial commands of this documented file.

Here we call the initial required packages and, since the necessary software to define some LATEX3 (L3) language "functions" are already part of the LATEX kernel, we just use them in order to define a robust LATEX 2_ε/L3 interface to create some testing macros that accept many logical operators that act on items that, besides boolean variables, are also numerical expressions connected with relation operators; for further details it suffices to examine the `interface3.pdf` file, that is integral part of any recent TEX system distribution. We neede also some commands to compare strings or to check if a given string was included into an L3 sequence (list) of strings.

The syntax of the new testing macro is the following:

`\fptest{`⟨*test*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

The logical operators that are usable in the ⟨*text*⟩ phrase are `||` (`OR`), `&&` (`and`), and `!` (`NOT`), plus other less common ones; the `!` negates the status of boolean variables, but also the numerical relation operators; these, on turn, may be paired so that, for example `>=` is the same as `!<`. The comparisons of strings with the following syntax

`\CompStrings{`⟨$string_1$⟩`}{`⟨$string_2$⟩`}`

Produces a boolean result that may be used as a ⟨*test*⟩ argument to `\fptest` The `\SetList` macro defines a named L3 sequence variable containing the reference list of valid display modes, while the `\TestList` is an L3 function that tests if a given string is listed in a named sequence list. Their syntax is the following:

`\SetList{`⟨*list name*⟩`}{`⟨*comma separated string list*⟩`}`
`\TestList`⟨*string*⟩`{`⟨*list name*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

The fields ⟨*true*⟩ and ⟨*false*⟩, as usual, are the action to be executed if the test is true or false; the test is true if ⟨*string*⟩ is listed into the ⟨*named list*⟩.

The `trace` package is still present in this beta version of the package; it will not be present any more in the future stable versions. Packages `graphicx`, `afterpage`, and `wrapfig` are functional for this package. In order to avoid "Option Clash" messages, such packages are loaded without any option; should the user load some or all of them with options, the user should load them *before* loading this package. This particular documentation does not require the `graphicx` package, because it is already loaded by `swfigure`.

```
\RequirePackage{etoolbox}
\RequirePackage{xfp}

\ExplSyntaxOn
\ProvideExpandableDocumentCommand\CompStrings{m m}{%
  \str_if_eq_p:nn{#1}{#2}}

  \ProvideExpandableDocumentCommand\fptest{m m m}{%
     \fp_compare:nTF{#1}{#2}{#3}}
```

```
10
11  \ProvideExpandableDocumentCommand\SetList{m m}{%
12  \seq_clear_new:N #1
13  \seq_set_from_clist:Nn \DisplayModeList {#2}}
14
15  \ProvideExpandableDocumentCommand\TestList{m m m m}{%
16  \seq_if_in:NnTF #1 {#2}{#3}{#4}}
17
18 \ExplSyntaxOff
19
20 \RequirePackage{trace}
21
22 \RequirePackage{graphicx}
23 \RequirePackage{afterpage}
24 \RequirePackage{wrapfig}
25
```

We define some dimension and some counter registers; to some dimensions we assign parameters relating to the page design. We also compute the dimensions of the available spread space so as to have available the elements to compare the aspect ratios of the images and of the available spaces. The `\if@SWtmode` switch is used only by the `SW` display mode in order to save the typesetting mode that was active when the `\DFimage` command was used in the source file; if it was in text mode, the switch is set to `true` and no spaces should be present after the last argument of that macro; the result is that the `\SWfigure` service macro shifts to vertical mode, but as soon as it is completed the switch test restores the text mode.

```
26 \newdimen\internalmargin
27     \internalmargin=\dimexpr\oddsidemargin+1in+1bp\relax
28 \newdimen\externalmargin
29     \externalmargin=\dimexpr\evensidemargin+1in
30 \newdimen\spreadwidth
31     \spreadwidth=\dimexpr 2\textwidth+2\internalmargin\relax
32 %
33 \newdimen\DFwidth
34 \newdimen\DFheight
35 \newdimen\DFhalfwidth
36 \newdimen\DFheight
37 \newdimen\DFhalfheight
38 \newdimen\FigSpace
39 \def\SWcaptionShift{1em}%
40 \def\FScaptionShift{2em}%
41 \newsavebox\DFtotalimage
42 \newsavebox\DFimageI
43 \newsavebox\DFimageII
44 \newsavebox\RFbox
45 \newdimen\VS@textwidth
46 \newcount\VS@lines
47 \newif\if@SWtmode
48
```

The following caption-like commands recompile the captions within a zero height vertical box and deal with the mandatory and optional arguments according to the following syntax:

\DFcaption[⟨*lof entry*⟩]{⟨*caption*⟩}[⟨*label*⟩]

Notice that the ⟨*lof entry*⟩ defaults to the full caption text if a different text is not explicitly entered. This commands are specific to those display modes that compose the caption in vertical mode, therefore the caption measure is the `\textheight`.

```
\NewDocumentCommand\DFcaption{O{#2} m o}{\refstepcounter{figure}%
  \vtop to 0pt{\hsize=\textheight\parindent=0pt\leavevmode
  Figure \thefigure\quad #2\vss}%
  \addcontentsline{lof}{figure}{\protect\numberline{\thefigure}#1}
  \IfValueT{#3}{\label{#3}}\relax%
}
\NewDocumentCommand\DFcaptionP{O{#2} m o D!!{black}}%
  {\refstepcounter{figure}%
  \vbox to 0pt{\vss\hsize=\textheight\parindent=0pt\leavevmode
  #4\relax Figure \thefigure\quad #2}%
  \addcontentsline{lof}{figure}{\protect\numberline{\thefigure}#1}
  \IfValueT{#3}{\label{#3}}\relax%
}


```

The `\cleartoeven` command may already exist in the document class. We prefer to make an absolute redefinition so as to be sure that it performs as we need in this package. In any case this new definition may be used in such a way that the `\DFimage` user command may be used also while typesetting in text (horizontal) mode, resuming such mode after complete expansion. This is useful, especially for spread wide display modes, to chose the most suitable place in the source file to place the `\DFimage` command with its arguments.

```

\newcommand\set@tmode@newpage{%
\ifvmode
  \@SWtmodefalse
\else
  \unskip\@SWtmodetrue\linebreak
  \vadjust{\vspace{0pt plus1fill}\newpage}%
\fi}

\newcommand\reset@tmode{\if@SWtmode\expandafter\noindent\ignorespaces\fi}

\newcommand\cleartoeven[1]{%
    \set@tmode@newpage\clearpage
    \ifodd\c@page\afterpage{#1}\else#1\fi%
    }

```

Now we define the "real" user macro necessary to chose the ⟨*display mode*⟩ and the various parameters necessary for the desired one. See section 3 on page3 for its syntax and the other necessary arguments. Notice that only two arguments are mandatory: the ⟨*image file name*⟩ and the ⟨*caption*⟩ text. All the other five optional parameters are delimited by various delimiters; the first one, ⟨*display mode*⟩ has as default setting the acronym `SW` for a "Spread-Wide" display mode; this mode was the one this package was initially conceived for, and its initials recall

this priority. All other modes came afterwards. the other optional parameters may have a default value, but they are ignored by certain modes that don use those parameters. As for the previous macro `\DFimage`, here the ⟨*lof entry*⟩ defaults to the full ⟨*caption*⟩ text.

Actually `\DFimage` just examines the ⟨*display mode*⟩, and accordingly passes the necessary parameters to the actual macros that implement the various mode displays. It outputs an error message if the ⟨*display mode*⟩ acronym was misspelt.

```
\NewDocumentEnvironment{DFimage}%
  {O{SW} m O{#4} m o D(){0.8} D<>{0} D||{0.25} D!!{}}%
{%
  \SetList{\DisplayModeList}{SW,HS,VS,FS,NF,RF}%
  \TestList{\DisplayModeList}{#1}%
  {\csuse{#1figure}{#2}[#3]{#4}[#5](#6)<#7>|#8|!#9!%
    \fptest{\CompStrings{\@currenv}{DFimage}}{}{\reset@tmode}%
  }%
  {%
    \DFwarning[#1]{#2}[#3]{#4}
  }%
}%
{%
  \aftergroup\reset@tmode
}%

\NewDocumentCommand\DFwarning{ o m o m o d() d<> d|| d!!}%
{\PackageWarning{swfigure}%
    {*****************************************\MessageBreak
      Option #1\space is not valid. Nothing done \MessageBreak
                                                 \MessageBreak
      Image #2 was not processed                 \MessageBreak
     *****************************************\MessageBreak
    }%
}


```

In the following subsections the six specific display mode macros are examined and commented.

6.1 The `\SWfigure` macro for Spread Wide images

The specific macro for spread-wide images has the following syntax:

`\SWfigure{`⟨*image file name*⟩`}[`⟨*lof entry*⟩`]{`⟨*caption*⟩`}[`⟨*label*⟩`]`
`(`⟨*height correction*⟩`)<`⟨*lines*⟩`>\meta{width thest}!precaption!`

The various arguments have already been described; as you see the ⟨*display mode*⟩ argument is not there anymore, as well as the last two optional arguments of `\DFfigure`.Notice that the ⟨*caption*⟩ text, the third argument of this macro, is used also to initialise the ⟨*lof entry*⟩, so that if the user doe not specify such short caption entry, the default is identica to the full caption. Notice the the last arguments are not actually used; they are needed in the definition due to the way `DFimage` transfers its arguments to the called macros.

The working strategy is the following: the initial image to be divided in two halves is stored in a box; this box is examined in order to find its total width and heigh, so as to be able to compute the aspect ratio. All other dimensions are examined to have available all the necessary data to display the image halves in the requested way.

Notice that the `trim` option to `\includegraphics`, that is being used to divide the initial image in two exact halves, maybe should be used without specifying other options; but, most important, it requires actual numbers, not macros; therefore we have to make use of the usual "dirty" trick of defining an expanded macro `\x` within a group that the macro itself closes upon its expansion; in this way the optional argument of the `\includegraphics` macro contains only keywords and numbers and may be executed without errors.

Let W be the total spread width and w be the initial image width; let H be the text height and h the initial image height; compute the A_{W} to be the aspect ratio of the spread available space H/W. Similarly let A_{fig} be the initial image aspect ratio; we have to scale the image with a scaling factor such that the image keeps its aspect ratio, but its scaled height does not exceed the text block height, and its scaled width does not exceed the spread width; of course the aim is to maintain the scaled image as large as possible therefore we have to chose the most suited of the two scaling factors that can be computed from the given data and the inequalities described above. In mathematical terms we have to chose the initial image scaling factor S such that:

$$S = \begin{cases} w/W & \text{if} \quad A_{\mathrm{fig}} \leq A_{\mathrm{W}} \\ h/H & \text{if} \quad A_{\mathrm{fig}} > A_{\mathrm{W}} \end{cases} \tag{1}$$

This choice is made by testing the various ratios by means of the L3 `\fptest` function, that computes such ratios and compares them. The ⟨*test*⟩ shows exactly which lengths are used to compute the ratios, and which comparison is executed; the commented line over the `\fptest` macro emphasises this correspondence. Once the scaling factors are found the boxes containing the half figures are scaled and further on are inserted in floating figure environments; we use une of the dirty tricks: the matrioska dolls; this metaphor describes nested `\makebox`es used to mask their actual widths, but capable to push the images towards the spread spine. Eventually the caption is set in the right margin of the right spread page, rotated 90° counterclockwise. The `cleartoeven` guaranties that the first float is set into an even numbered page. A final `\clearpage` ensures that both floats are output.

```
\NewDocumentCommand\SWfigure{m o m o d() d<> d|| d!!}{%
\setbox\DFtotalimage=\hbox{\includegraphics{#1}}%
\DFwidth=\wd\DFtotalimage \DFhalfwidth=0.5\DFwidth
\DFheight=\ht\DFtotalimage
\FigSpace=\dimexpr\textwidth+\internalmargin\relax
%
\setbox\DFimageI\hbox{\bgroup
\edef\x{\egroup\noexpand\includegraphics*[%
trim = 0 0 \the\DFhalfwidth\space 0]}\x{#1}}%
%
\setbox\DFimageII\hbox{\bgroup
\edef\x{\egroup\noexpand\includegraphics*[%
trim = \the\DFhalfwidth\space 0 0 0]}\x{#1}}%
```

```
%
%               h/w            >           H/W
\fptest{\DFheight/\DFhalfwidth > \textheight/\FigSpace}%
  {\edef\DFscalefactor{\fpeval{\textheight/\DFheight}}}%
  {\edef\DFscalefactor{\fpeval{\FigSpace/\DFhalfwidth}}}%
%
\setbox\DFimageI=\hbox{\scalebox{\DFscalefactor}{\usebox{\DFimageI}}}%
\setbox\DFimageII=\hbox{\scalebox{\DFscalefactor}{\usebox{\DFimageII}}}%
%
\cleartoeven{%
  \begin{figure}[p]%
  \vbox to\textheight{\vss\hsize=\textwidth%
  \makebox[\hsize][l]{\makebox[\FigSpace][r]{\box\DFimageI}}\vss}%
  \end{figure}\newpage
%
  \begin{figure}[p]%
  \vbox to\textheight{\vss\hsize=\textwidth
  \makebox[\hsize][r]{\makebox[\FigSpace][l]{\box\DFimageII}%
% Rotated caption in the right page
  \makebox(0,0)[lb]{\hspace*{\SWcaptionShift}\raisebox{0.5\textheight}{%
  \rotatebox[origin=tc]{90}{\DFcaption[#2]{#3}[#4]%
  %
  }}}}\vss}%
  \end{figure}%
}\clearpage\reset@tmode}

```

6.2 The `\NFfigure` macro for normal full page figures

This code is without surprises, except that it receives from the `DFimage` steering macro five usable arguments (plus three more ones that are not actually used); the fifth (optional) one being delimited by round parentheses; by default, it contains the value 0.8, but the user can pass to the steering macro a different value in order to reduce the actual image size so as to leave enough space beneath it for the caption; the user, in facts, does not know in advance how many lines would occupy a structured complex caption.

```
\NewDocumentCommand\NFfigure{m o m o d() d<> d|| d!!}{%
  \begin{figure}[p]
    \includegraphics[width=\linewidth, height=#5\textheight,
                     keepaspectratio]{#1}%
    \caption[#2]{#3}%
    \IfValueT{#4}{\label{#4}}\relax
  \end{figure}
}

```

6.3 The `\RFfigure` macro for rotated figures

There are no new comments to make about this almost standard way of displaying a large figure, except, perhaps, the fact the instead of invoking the `lscape` package, the whole figure is rotated by means of rotating the very box that was used to contain the scaled image; the steering macro does not pass any ⟨*height correction*⟩

argument to this macro, because it automatically scales the box containing the image and the caption so as to fill up the height or the width of the text block. Of course the macro signature includes also the descriptors of unused arguments.

```
\NewDocumentCommand\RFfigure{m o m o d() d<> d|| d!!}{%
      \dimen8=\textwidth\dimen10=\textheight
      \figure[p]\setbox\RFbox=\hbox{%
      \rotatebox[origin=cc]{90}{\parbox[b][\dimen8][c]{\dimen10}%
        {\centering\includegraphics[width=\dimen10, height=\dimen8,
                        keepaspectratio]{#1}%
          \caption[#2]{#3}%
          \IfValueT{#4}{\label{#4}}\relax%
        }}%
        \edef\RFx{\fpeval{\ht\RFbox/\textheight}}%
        \edef\RFy{\fpeval{\wd\RFbox/\textwidth}}%
        \fptest{\RFx > \RFy}%
          {\scalebox{\RFx}{\box\RFbox}}%
          {\scalebox{\RFy}{\box\RFbox}}%
        \endfigure}
```

6.4 The `\VSfigure` macro for tall and slim figures

This is the most critical macro (because it depends on the `wrapfig` package) in order to set the image with its caption on the external side of the text block, with the text flowing around it. Package `wrapfig` has several limitations concerning the text that flows around the image. Nevertheless with a "normal" textual contents of the text block the result may be very good.

The user is recommended to attentively read the `wrapfig` documentation because it has several limitations. here we suggest the user to open the environment just before a paragraph and close it after several paragraphs; remember that since the opening and the closing statements are far away from one another, it is possible to forget this situation and insert another such environment within the former one ,or a `\DFimage` command, and this will produce undesirable effects; never ever nest two `Dfimage` environments and never ever insert a `\DFimage` command within an environment.

The user should help a little bit the correct performance of this macro; for this reason, besides the ⟨*height correction*⟩ factor, the steering macro may pass to it also the ⟨*line correction*⟩ in order to correct the amount of lines the surrounding text should be indented; by playing with both corrections good results may be obtained.

Notice that only with this macro a test on the initial image aspect ratio is performed; should this image have an aspect ratio smaller than 2, the process is aborted and a message is printed in the output document that informs the user and recommends to use another ⟨*display mode*⟩ code.

Furthermore, should the image be very slim and tall, the ⟨*height correction*⟩ may conveniently reduce its height, but, in order to avoid changing its aspect ratio, it reduces also its width; eventually the width is reduced to the point that beneath the image there would not be enough space for a decently typeset caption. A further optional delimited argument is available: its delimiting tokens are two vertical bars, `| |`, and its default value is `0.25`. This means that if the scaled image becomes narrower than this coefficient times the text width, an other message is

typeset in place of the figure. Depending of the various elements that influence these tests, the user can try different values for this coefficient, even reaching the value zero, that eliminates such test on the scaled image width.

```
\NewDocumentCommand\VSfigure{m o m o d() d<> d|| d!!}{%
   \setbox\DFtotalimage=\hbox{\includegraphics{#1}}
   \DFwidth=\wd\DFtotalimage \DFheight=\ht\DFtotalimage
   \edef\VS@aspectratio{\fpeval{\DFheight/\DFwidth}}
   \fptest{ \VS@aspectratio < 2 }%
     {\begin{center}\ttfamily
      ========================================\\
      Image \textit{#1} is not tall enough.\\
      Consider using a display mode different\\
      from VS; may be NF or RF are better suited.\\
      ========================================\\
      Nothing done!\\
      ========================================
      \end{center}
     }%
     {\edef\VS@factor{\fpeval{#5\textheight/\DFheight}}%
      \fptest{\VS@factor>1 || \VS@factor\DFwidth < #7\textwidth}%
       {%
         \begin{center}\ttfamily
         ================================================\\
         The scaled image \textit{#1} is too slim.        \\
         Maybe directly using the wrapfig package might   \\
         solve this problem.                              \\
         ================================================\\
         Nothing done!\\
         ================================================
         \end{center}
       }%
       {\setbox\DFtotalimage=
        \hbox{\scalebox{\VS@factor}{\box\DFtotalimage}}%
        \edef\VS@width{\fpeval{\VS@factor*\DFwidth}\p@}%
        \setbox\DFtotalimage=\vbox{\hsize=\VS@width%
        \def\@captype{figure}\box\DFtotalimage
        \caption[#2]{#3}%
        \IfValueT{#4}{\label{#4}}\relax}
        \VS@lines=
        \fpeval{round(\ht\DFtotalimage/\baselineskip,0)+#6}%
        \begin{wrapfigure}[\VS@lines]{o}[Opt]{\VS@width}%
        \box\DFtotalimage
        \end{wrapfigure}%
       }%
     }%
  }%

```

6.5 The `\HSfigure` macro for spread wide slim figures

The beginning of the macro is not so different from that used for the `SWfigure` macro. But this time there is no alternative to scale the image taking into account only its width and the spread width. The new little problem, now, is to match

the total height of the two halves of the original wide and slim image, so that the texts of the two facing pages have their respective first lines perfectly aligned.

Since the right half of the figure contains also the caption it is necessary to box again the left page box so as to have the same hight and depth as the right page one.

```
\NewDocumentCommand\HSfigure{m o m o d() d<> d|| d!!}{%
  \setbox\DFtotalimage=\hbox{\includegraphics{#1}}%
  \DFwidth=\wd\DFtotalimage \DFhalfwidth=0.5\DFwidth
  \FigSpace=0.5\spreadwidth%
%
  \setbox\DFimageI\hbox{\bgroup
  \edef\x{\egroup\noexpand\includegraphics*[%
          trim = 0 0 \the\DFhalfwidth\space 0]}\x{#1}}%
%
  \setbox\DFimageII\hbox{\bgroup
  \edef\x{\egroup\noexpand\includegraphics*[%
          trim = \the\DFhalfwidth\space 0 0 0]}\x{#1}}%
%
  \edef\DFscalefactor{\fpeval{\FigSpace/\DFhalfwidth}}%
%
  \setbox\DFimageI=\hbox{%
    \scalebox{\DFscalefactor}{\usebox{\DFimageI}}}%
  \setbox\DFimageII=\hbox{%
    \scalebox{\DFscalefactor}{\usebox{\DFimageII}}}%
  \setbox\DFimageII=
    \hbox{\dimen10=\linewidth\dimen8\internalmargin
           \vbox{\hsize\DFhalfwidth\parindent\z@
             \box\DFimageII\par
             \leavevmode\hspace*{\dimen8}%
             \vtop{\hsize\dimen10\parindent\z@
               \textwidth=\hsize
               \DFcaption[#2]{#3}[#4]%
              }\vspace*{2\baselineskip}%
            }%
          }%
  \setbox\DFimageI=\vbox to\ht\DFimageII{\box\DFimageI\vss}%
%
  \cleartoeven{%
    \hb@xt@\textwidth{%
      \makebox[\DFhalfwidth][l]{\box\DFimageI}\hss}%
%
     \afterpage{\hb@xt@\textwidth{%
         \hss\makebox[\DFhalfwidth][r]{\box\DFimageII}}%
    }%
  }%
}
```

6.6 The `\FSfigure` macro for Full Spread figures

Although using certain classes, such as `memoir`, it is possible to typeset documents on pages of different sizes and to print them on advanced printers that use different sized papers, this package aims to be independent form any class and any printer; it is limited to the ordinary default paper size A4, as it is usual in Europe, but

letter paper is also valid); ideally it would typeset an A3 sized image on a full spread, i.e. by using also the space of all the margins.

Actually with American paper sizes there is not a series of sizes that behaves as the ISO A series, but this macro tries to adjust the large image to the available full spread; this means an image as large as two standard pages.

If the available image is taller than large, with an aspect ratio around $\sqrt{2}$, it turns the image 90° counterclockwise; cuts it in two identically sized halves, and sets them on two facing pages joining them at the spine but covering the greatest part of both pages, so as to avoid exceeding either the paper height or twice the paper width. If the image has an aspect ratio close to $\sqrt{1/2}$, it does the same, except rotating the image.

In any case the user creates a separate PDF file to be imported into the document that is supposed to contain it; it is better if this PDF is of vectorial type, i.e. any drawing and all fonts should should be vectorial; this PDF might contain photos or similar images of rasterised type, but the pixel density should be suitably high in order to preserve the image quality.

The user has two choices: (*a*) either the image has its own margins, or (*b*) the image is cropped and has no white margins.

Normally this display mode does not leave space for a caption; therefore if a caption is needed, it is overprinted on the image; in case (*a*) there are no problems because the caption goes over the white right margin (rotated 90° counterclockwise); while in case (*b*) the caption is over printed un the right side of the image; depending on the image "background" a different color might be more suited compared to black (the default color). The last macro argument takes care of inserting before the caption any declaration that may involve color, displacement, font encoding (seldom necessary), family, series, shape and size;

With all the above constraints the syntax of this `\FSimage` macro is the following; it uses one of the main macro delimited optional arguments in a different way:

```
\FSimage{⟨image file name⟩}[⟨lof entry⟩]{⟨caption text⟩}[⟨label string⟩]
    (⟨height correction⟩)<⟨line correction⟩>\meta{width text!⟨precaption⟩!
```

The above is the full definition, but the fifth, sixth and seventh arguments are not used; they are necessary for the special mechanism of argument transfer from the environment or command opening statement arguments to this actual macro.

Moreover, since the included image overlaps the top and bottom margins, if these contain the header and footer such elements would show if the image is not opaque enough; in any case the footer would show because it overlaps the image. Since most images are not opaque at all it is better to locally use the `empty` page style, but without using the `\thispagestyle` command, because it hold for just one page, while the spread occupies two pages. The macro provides this setting but it has to create a group in case the `\DFimage` command is used, while it is not necessary when using the `DFimage` environment, therefore the current environment is tested and the opening `\bgroup` and closing `\egroup` commands are used onlY if the code is used without the environment.

Therefore the code steps are the following:

1. The initial figure is loaded into a box from which the necessary measures are taken; in particular the its width and its height in order to check the aspect

ratio, or better, which dimension is greater than the other. If the height is smaller than the width the image is sort of panoramic and we proceed splitting the total image box into two equal halves, with a vertical cut in the middle.

2. Otherwise the image is as a portrait, and we split the total image block into two halves with an horizontal cut in the middle; such two boxes need a rotation of 90° anticlockwise due to the horizontal cut of an original higher than wide image. With this operation we simulate a rotation of the original image, because it is not allowed to simultaneously rotate and trim the original image; the trimming action apparently cannot be executed together with any other one among the many other that the `\includegraphics` command can do[1]. Of course, with this rotation of the half pictures we have to redefine the height and width to use for scaling.
3. The `\fptest` is used to compare the aspect ratios of any of the two equal halves with the aspect ratio of the paper. If the former aspect ratio is higher than the latter, the scaling factor is based on the heights, otherwise on the widths, so as to resize both half picture boxes to fit within the page.
4. The `\cleartoeven` command allows to use the main macro even within a paragraph; in any case it possibly inserts a blanc page and restarts working on an even page. First is sets the left half picture box; that complicated series of vertical and horizontal boxes of different specified dimensions contains displacements and the material to output; That complicated alternating set is to guarantee that no overfull vertical or horizontal boxes are created, in spite of the fact that an object larger than the text block is being output; at the same time the box must be flush right in order to reach the spine.
5. The right half is even more complicated to build, because there is also the caption to set on the right of the page, overwriting the right part of the image, but the idea is always to close correctly every box putting the right spacing commands at the beginning or at the end of each box.
6. Eventually a `\clearpage` flushes out everything that is still in the figure stack and possibly restores the vertical or horizontal typesetting mode depending on the status that was in force when the `\DFimage` command or the `DFimage` environment started its expansion. Due to the size of the images that completely fill up all the margins, probably the best way to output such huge images is at the end of a chapter, or at the end of the whole document.

```
\NewDocumentCommand\FSfigure{m o m o  d() d<> d|| d!!}{%
  \fptest{\CompStrings{\@currenv}{DFimage}}{}{\bgroup}%
  \pagestyle{empty}%
  \setbox\DFtotalimage=\hbox{\includegraphics{#1}}%
  \DFwidth=\wd\DFtotalimage \DFheight=\ht\DFtotalimage
  \ifdim \DFheight < \DFwidth
    \DFhalfwidth=0.5\DFwidth
    \setbox\DFimageI\hbox{\bgroup
    \edef\x{\egroup\noexpand\includegraphics*[%
    trim = 0 0 \the\DFhalfwidth\space 0]}\x{#1}}%
    \setbox\DFimageII\hbox{\bgroup
    \edef\x{\egroup\noexpand\includegraphics*[%
    trim = \the\DFhalfwidth\space 0 0 0]}\x{#1}}%
```

[1] Actually it can, provided that the trim option with its arguments is specified before any other option.

```
    \DFwidth=\DFhalfwidth \DFheight=\ht\DFtotalimage
  \else
    \DFhalfheight=0.5\DFheight
    \setbox\DFimageII\hbox{\bgroup
    \edef\x{\egroup\noexpand\includegraphics*[%
    trim = 0 0 0 \the\DFhalfheight\space]}\x{#1}}% bottom half
    \setbox\DFimageI\hbox{\bgroup
    \edef\x{\egroup\noexpand\includegraphics*[%
    trim =  0 \the\DFhalfheight\space 0 0]}\x{#1}}% tophalf
    \setbox\DFimageI=\hbox{\rotatebox[origin=cc]{90}{\box\DFimageI}}
    \setbox\DFimageII=\hbox{\rotatebox[origin=cc]{90}{\box\DFimageII}}
    \DFwidth=\DFhalfheight \DFheight=\wd\DFtotalimage
  \fi
\fptest{\DFheight/\DFwidth > \paperheight/\paperwidth}%
  {\edef\DFscalefactor{\fpeval{\paperheight/\DFheight}}}%
  {\edef\DFscalefactor{\fpeval{\paperwidth/\DFwidth}}}%
  \setbox\DFimageI=\hbox{\scalebox{\DFscalefactor}{\usebox{\DFimageI}}}%
  \setbox\DFimageII=\hbox{\scalebox{\DFscalefactor}{\usebox{\DFimageII}}}%
\cleartoeven{%
  \begin{figure}[p]%
  \vbox to\textheight{\vss\hsize=\textwidth%      vbox 1
    \hbox to\hsize{\hspace*{-\externalmargin}%    hbox 1
      \vbox to\paperheight{\vss%                  vbox 2
        \hbox to\paperwidth{\hss\box\DFimageI}%   hbox 2 end hbox 2
      \vss}%                                      end vbox 2
    \hss}%                                        end hbox 1
  \vss}%                                          end vbox 1
  \end{figure}\newpage
%
  \begin{figure}[p]%
    \vbox to\textheight{\vss\hsize=\textwidth%    vbox 1
      \hbox to\hsize{\hspace*{-\internalmargin}%  hbox 1
        \vbox to\paperheight{\vss%                vbox 2
          \hbox to\paperwidth{%                   hbox 2
            \box\DFimageII\makebox(0,0)[lb]{%     mbox 3
              \hspace*{-\FScaptionShift}%
              \raisebox{0.5\textheight}{%         raisebox
                \rotatebox[origin=bc]{90}{%       rotatebox
                  \DFcaptionP[#2]{#3}[#4]!#8!%
                 }%                               end rotatebox
              }%                                  end raisebox
            }%                                    end mbox 3
          }%                                      end hbox 2
        \vss}%                                    end vbox 2
      \hss}%                                      end hbox 1
    \vss}%                                        end vbox 1
  \end{figure}%
}\clearpage
\fptest{\CompStrings{\@currenv}{DFimage}}{}{\egroup}%
\reset@tmode}
```

HAPPY LaTeX-ing