The `subfiglist` package v1.0

Manuel Nutz

March 22, 2015

Contents

1 Package loading

`\usepackage[options]{subfiglist}`

Options have to be given in a key-value comma separated list as in `key1=val1,key2=val2`. After loading the package, options can also be specified or overwritten with the command `\subfiglistset{options}`. Currently the following options are supported.

`caption` Can be set to `true` or `false` and will be set to `true` when specified completely without value. Default value is `true`. This option cannot be changed after `\begin{document}`.

This option decides whether the interface provided by the `caption` package is used for formatting of the subfigure labels. This interface is powerful, but in order to use it some internal commands of the `caption` package have to be modified. Compatibility

was only tested for version 3.3 of the `caption` package and things may fail for any other version.

Note that this option does *not* decide whether the caption package is loaded! The functionality of the caption package is used in any case, but because this tweak is considered particularly "dangerous" it can be turned off explicitly.

`environment-width` Length that specifies the default width of the `subfiglist` environment. Default value is `\textwidth`. Shorthand `ew` can be used instead.

`environment-space` Length that specifies the default space between two images in the `subfiglist` environment. Default value is `1ex`. Shorthand `es` can be used instead.

`file-top` Length that specifies the default additional white space above an image in the `subfiglist` environment. Default value is `0pt`. Shorthand `ft` can be used instead.

`file-bottom` Length that specifies the default additional white space below an image in the `subfiglist` environment. Default value is `0pt`. Shorthand `fb` can be used instead.

`file-left` Length that specifies the default additional white space left of an image in the `subfiglist` environment. Default value is `0pt`. Shorthand `fl` can be used instead.

`file-right` Length that specifies the default additional white space right of an image in the `subfiglist` environment. Default value is `0pt`. Shorthand `fr` can be used instead.

`label-hpos` Default horizontal positioning of label within subfigure. Can be either `l` (left), `c` (centered) or `r` (right). Default is `l`. Shorthand `lh` can be used instead.

`label-vpos` Default vertical positioning of label within subfigure. Can be either `t` (top), `c` (centered) or `b` (bottom). Default is `t`. Shorthand `lv` can be used instead.

`label-xshift` Length that specifies the default additional horizontal shift of the label, where positive values shift to the right and negative values to the left, respectively. Default is `0.5ex`. Shorthand `lx` can be used instead.

`label-yshift` Length that specifies the default additional vertical shift of the label, where positive values shift downwards and negative values upwards, respectively. Default is `0.5ex`. Shorthand `ly` can be used instead.

`label-color` or `label-colour` Default text color of the label. More complex color definitions using e. g. extended color expressions from the `xcolor` package syntax should be enclosed in *double* braces as

```
label-color={{rgb,2:green,0.75;blue,1}}
```

demonstrates. Default color is `.` (period) which means no color change, i. e. current text color. Shorthand `lc` can be used instead.

`label-background` Default background color of the label. As for the `label-color` option, more complex color definitions should be enclosed in *double* braces. Default color is `none`, i. e. fully transparent background. Shorthand `lb` can be used instead.

2 The subfiglist environment

The subfiglist environment is used for specification of the desired figure layout and for loading the corresponding files. It is typically used inside a figure environment. A first simple example is given in listing 1.

```
\begin{subfiglist}{1 2 3 \\ 4 5 6}
  \subfiglistfile{1}{figures/01.png}
  \subfiglistfile{2}{figures/02.png}
  \subfiglistfile{3}{figures/03.png}
  \subfiglistfile{4}{figures/04.png}
  \subfiglistfile{5}{figures/05.png}
  \subfiglistfile{6}{figures/06.png}
\end{subfiglist}
```

Listing 1: Simple example of subfiglist environment usage

In case the subfiglist environment is used outside a figure or any other float environment, the caption package has to be told explicitly that it is supposed to label figures. This can be done with the following command.

```
\captionsetup{type=figure}
```

The subfiglist environment serves as a wrapper for several commands, which can be used to specify images, put labels or image overlays. The general syntax is as follows.

```
\begin{subfiglist}[options]{spec}
  content
\end{subfiglist}
```

The commands to be used as content are discussed in the subsequent sections. The specification spec determines in what layout the subfigures are to be arranged. Within spec the following characters are admissible.

0 to 9 All subfigures are assigned a number, which is used as reference later. When n subfigures are to be arranged, the specified numbers have to be 1 to n with no number appearing twice or being omitted. However, it is not strictly necessary (but recommended) to specify the numbers in ascending order. Any numbers may be enclosed in braces, which is necessary for numbers greater than 9, as any spaces are ignored. Hence, 12 is interpreted as one followed by two, while {12} is interpreted as twelve.

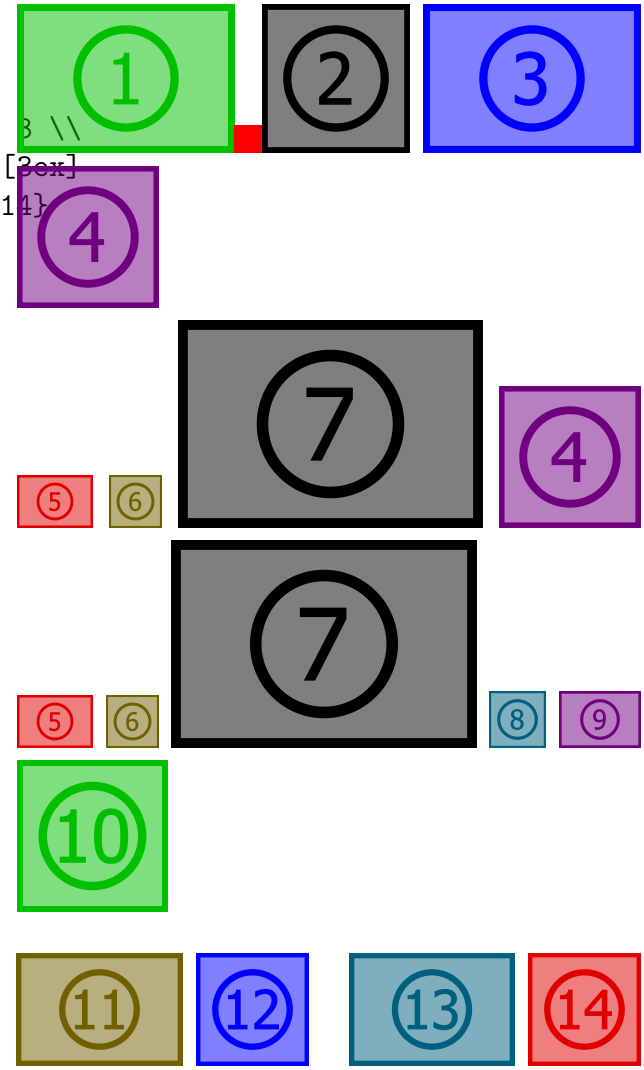
\\[length] Indicates a line break just as in usual text. The optional argument [length] can be used to specify a vertical space that is different from the default distance between two images.

@{...} Can be used to typeset the argument {...} between two images instead of the default spacing. For example, 1@{TEXT}2 will omit any space between image 1 and 2 and typeset TEXT instead. In particular, @{\hspace*{length}} can be used to put a space between two images that differs from the default value.

{...} Braces are used for grouping content in the usual way. This mechanism can be used to create subblocks for a more advanced positioning of subfigures.

The more complicated example in listing 2 illustrates the usage of the spec argument and the meaning of the individual parts.

```
\begin{subfiglist}
  {
    1 @{\color{red}\rule{2ex}{2ex}} 2 3 \\
    { 4 \\ 5 6 } 7 { 8 9 \\ {10} } \\[3ex]
    {11} {12} @{\hspace*{3ex}} {13} {14}
  }
  \subfiglistfile{1}{figures/01.png}
  \subfiglistfile{2}{figures/02.png}
  \subfiglistfile{3}{figures/03.png}
  \subfiglistfile{4}{figures/04.png}
  \subfiglistfile{5}{figures/05.png}
  \subfiglistfile{6}{figures/06.png}
  \subfiglistfile{7}{figures/07.png}
  \subfiglistfile{8}{figures/08.png}
  \subfiglistfile{9}{figures/09.png}
  \subfiglistfile{10}{figures/10.png}
  \subfiglistfile{11}{figures/11.png}
  \subfiglistfile{12}{figures/12.png}
  \subfiglistfile{13}{figures/13.png}
  \subfiglistfile{14}{figures/14.png}
\end{subfiglist}
```



Listing 2: More complicated example to demonstrate the usage of the spec argument.

The options have to be given in a key-value comma separated list as in `key1=val1,key2=val2`. Currently the following options are supported.

`width` Length that specifies the width of the `subfiglist` environment. Default value is `\textwidth` if not specified otherwise in the package options. Shorthand `w` can be used instead.

`space` Length that specifies the space between two images in the `subfiglist` environment. Default value is `1ex` if not specified otherwise in the package options. Shorthand `s` can be used instead.

Alongside with the `subfiglist` environment also the `subfiglist*` environment exist, which is identical in usage but internally uses the `\hspace*` command instead of the `\hfill` command for creating horizontal white space between images. This should not make any visible difference in any situation I could think of. But I couldn't decide which version to use anyway, so here it is. Maybe it's helpful to have it in some way.

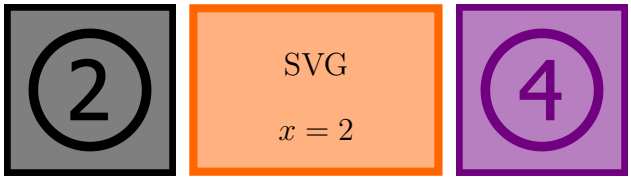
3 The command `\subfiglistfile`

`\subfiglistfile[options]{image-number}{file-name}`

The argument `image-number` is used to reference the numbers used in the `spec` argument of the `subfiglist` environment. For `file-name` either a simple file name as in `file` or a full or relative path as in `path/to/file` can be used. The file extension can be omitted if it belongs to the usual set that is checked by the `\includegraphics` command.

If the file extension `pdf_tex` is given as in `file/svg.pdf_tex`, then it is automatically assumed that the corresponding file was created with the 'PDF + LaTeX' option of Inkscape, which allows for separation of drawings and text. The text inside a SVG image is then typeset directly by LaTeX with full support of formatting and math mode as shown in listing 3.

```
1 \begin{subfiglist}{1 2 3}
2   \subfiglistfile{1}{figures/02.png}
3   \subfiglistfile{2}
4     {figures/svg.pdf_tex}
5   \subfiglistfile{3}{figures/04.png}
6 \end{subfiglist}
```



Listing 3: Compatibility with both bitmap images and the 'PDF + LaTeX' option of Inkscape

When an image with the file extension `pdf_tex` is loaded, it is put inside the `subfiglistsvgenv` environment, which can be used to change the default formatting of the text in SVG images. By default the `subfiglistsvgenv` environment expands to nothing. In order to set all text on SVG images in footnotesize sans serif font, the following code can be used. First, a sans serif version of the math fonts has to be defined in the preamble.

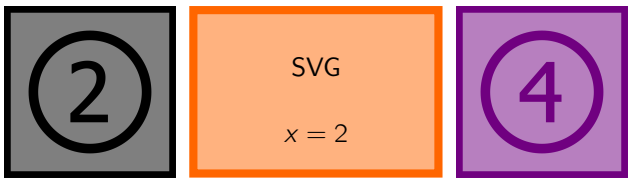
```
\DeclareMathVersion{sans}
  \SetSymbolFont{operators}{sans}{OT1}{cmbr}{m}{n}
  \SetSymbolFont{letters}{sans}{OML}{cmbrm}{m}{it}
  \SetSymbolFont{symbols}{sans}{OMS}{cmbrs}{m}{n}
  \SetMathAlphabet{\mathit}{sans}{OT1}{cmbr}{m}{sl}
  \SetMathAlphabet{\mathbf}{sans}{OT1}{cmbr}{bx}{n}
  \SetMathAlphabet{\mathtt}{sans}{OT1}{cmtl}{m}{n}
  \SetSymbolFont{largesymbols}{sans}{OMX}{iwona}{m}{n}
```

Afterwards, the `subfiglistsvgenv` environment can be redefined in the desired way. Special care has to be taken to terminate every line properly by `%` or otherwise spacing might be messed up.

```
\renewenvironment*{subfiglistsvgenv}{%
  \begin{sffamily}%
  \mathversion{sans}%
  \footnotesize%
}{%
  \end{sffamily}%
}
```

After this setup the same code as above yields the result shown in listing 4. Note that the text color cannot be changed globally this way, since the file generated by Inkscape explicitly sets it.

```
\begin{subfiglist}{1 2 3}
  \subfiglistfile{1}{figures/02}
  \subfiglistfile{2}
    {figures/svg.pdf_tex}
  \subfiglistfile{3}{figures/04}
\end{subfiglist}
```



Listing 4: Footnotesize sans serif font in SVG image

The options have to be given in a key-value comma separated list as in `key1=val1,key2=val2`. Currently the following options are supported.

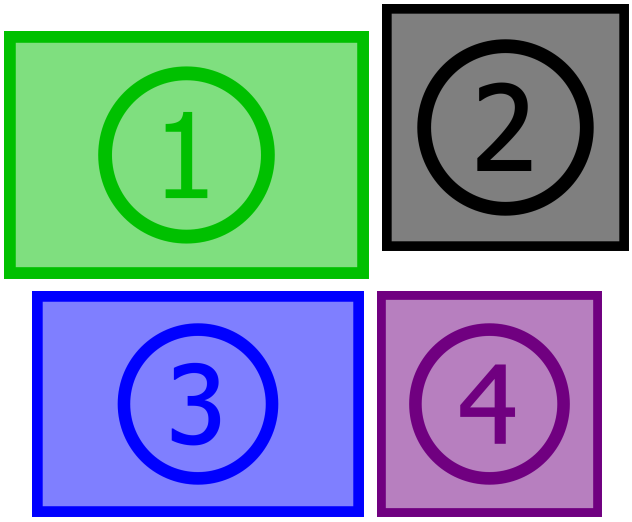
`top` Length that specifies the additional white space above the specified image. Default value is `0pt`. Shorthand `t` can be used instead.

`bottom` Length that specifies the additional white space below the specified image. Default value is `0pt`. Shorthand `b` can be used instead.

`left` Length that specifies the additional white space left of the specified image. Default value is `0pt`. Shorthand `l` can be used instead.

`right` Length that specifies the additional white space right of the specified image. Default value is `0pt`. Shorthand `r` can be used instead.

```
\begin{subfiglist}{1 2 \\ 3 4}
  \subfiglistfile[t=2ex]{1}
    {figures/01}
  \subfiglistfile[b=2ex]{2}
    {figures/02}
  \subfiglistfile[l=2ex]{3}
    {figures/03}
  \subfiglistfile[r=2ex]{4}
    {figures/04}
\end{subfiglist}
```



Listing 5: Additional spacing around images

4 The command \subfiglistlabel

`\subfiglistlabel[options]{image-number}{label-hook}`

Places a label for the image referenced by `image-number`. The argument `label-hook` can be used to pass a `\label` command in order to reference the image by a `\ref` like command. If `label-hook` is left empty, a label is still typeset but it cannot be referenced by `\ref`.

The appearance of the label can be changed both with and without the package option `caption`. As an example it shall be demonstrated how to obtain a bold sans-serif lowercase letter in braces as label, while a reference using `\ref` produces the image number followed by a thin space `\,` and a normal font lowercase latter in braces like in 1 (a).

When the option `caption` is used, the font of the label can be set using the `caption` package mechanism using `\captionsetup`. Braces and thin space for the `\ref` command can be obtained by redefinition of `\thesubfigure`. However, also the caption package makes use of `\thesubfigure` and the thin space is unwanted there, so it has to be canceled by putting everything inside a `\makebox`. The full code may look as follows.

```
\DeclareCaptionLabelFormat{killspace}{%
  \makebox[\widthof{#2}-\widthof{\,}][r]{#2}%
}
\captionsetup{labelfont={bf,sf}}
\captionsetup[subfigure]{labelformat=killspace}
\renewcommand*\thesubfigure{\,(\alph{subfigure})}
```

Without the `caption` option the desired behavior of `\ref` is again obtained by redefiniton of `\thesubfigure`. The format of the label, however, has to be defined explicitly via a redefinition of `\subfiglistlabelformat` in the following way.

```
\renewcommand*\subfiglistlabelformat{\textbf{\textsf{(\alph{subfigure})}}}
\renewcommand*\thesubfigure{\,(\alph{subfigure})}
```

The options have to be given in a key-value comma separated list as in `key1=val1,key2=val2`. Currently the following options are supported.

`hpos` Horizontal positioning of label within subfigure. Can be either `l` (left), `c` (centered) or `r` (right). Default is `l` if not specified otherwise in the package options. Shorthand `h` can be used instead.

`vpos` Vertical positioning of label within subfigure. Can be either `t` (top), `c` (centered) or `b` (bottom). Default is `t` if not specified otherwise in the package options. Shorthand `v` can be used instead.

`xshift` Length that specifies the additional horizontal shift of the label, where positive values shift to the right and negative values to the left, respectively. Default is `0.5ex` if not specified otherwise in the package options. Shorthand `x` can be used instead.

`yshift` Length that specifies the additional vertical shift of the label, where positive values shift downwards and negative values upwards, respectively. Default is `0.5ex` if not specified otherwise in the package options. Shorthand `y` can be used instead.

`color` or `colour` Text color of the label. More complex color definitions using e. g. extended color expressions from the `xcolor` package syntax should be enclosed in *double* braces as

```
color={{rgb,2:green,0.75;blue,1}}
```

demonstrates. Default color is `.` (period) if not specified otherwise in the package options, which means no color change, i. e. current text color. Shorthand `c` can be used instead.

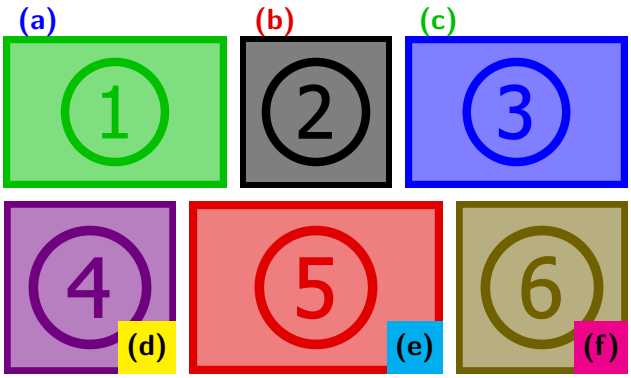
`background` Background color of the label. As for the `label-color` option, more complex color definitions should be enclosed in *double* braces. Default color is `none`, i. e. fully transparent background, if not specified otherwise in the package options. Shorthand `b` can be used instead.

Listing 6 illustrates these options. Note that the `top`, `bottom`, `left` and `right` options of the `\subfiglistfile` command do not influence the label positioning. They can hence be used to move the label off a subfigure.

5 The command \subfiglistoverlay

```
\subfiglistoverlay{image-number}{content}
```

```
\begin{subfiglist}{1 2 3 \\ 4 5 6}
  \subfiglistfile[t=3ex]{1}{figures/01}
  \subfiglistfile[t=3ex]{2}{figures/02}
  \subfiglistfile[t=3ex]{3}{figures/03}
  \subfiglistfile{4}{figures/04}
  \subfiglistfile{5}{figures/05}
  \subfiglistfile{6}{figures/06}
  \subfiglistlabel
    [y=0pt,c=blue]
    {1}{\label{foo}}
  \subfiglistlabel
    [y=0pt,c={{rgb,256:red,224}}]
    {2}{\label{bar}}
  \subfiglistlabel
    [y=0pt,c={{rgb,1:green,.75}}]
    {3}{\label{baz}}
  \subfiglistlabel
    [h=r,v=b,x=-0.5ex,y=-1ex,b=yellow]
    {4}{}
  \subfiglistlabel
    [h=r,v=b,x=-0.5ex,y=-1ex,b=cyan]
    {5}{}
  \subfiglistlabel
    [h=r,v=b,x=-0.5ex,y=-1ex,b=magenta]
    {6}{}
\end{subfiglist}
```



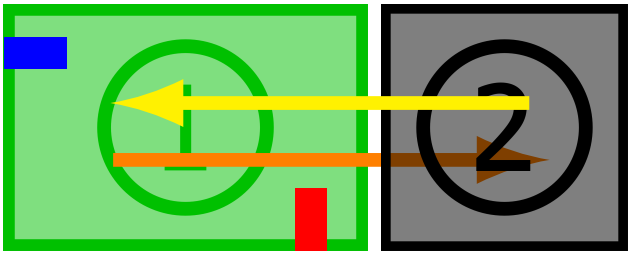
Listing 6: Demonstration of the different options for the `\subfiglistlabel` command

This command can be used to place objects on top of the image referenced by `image-number`. The argument `content` can contain arbitrary commands that are placed inside the environment `subfiglistoverlayenv`. By default, `subfiglistoverlayenv` expands to a `picture` environment with `\unitlength` being equal to the width of the image referenced by `image-number`.

When `\subfiglistoverlay` in its default setup, i. e. for `subfiglistoverlayenv` expanding to a `picture` environment, is used for drawing lines or shapes, loading of the `pict2e` package is strongly recommended. The standard `picture` environment suffers from several severe limitations, which are lifted by this package.

Note that the first image together with overlay is typeset before the second image. Hence the orange arrow is hidden behind the second (half-transparent PNG) image, while the yellow arrow appears in front of everything. To influence the behavior of overlays, the `subfiglistoverlayenv` environment can be redefined. It takes two arguments, the first being the width and the second the height of the current image.

```
\begin{subfiglist}{1 2}
  \subfiglistfile{1}{figures/01}
  \subfiglistfile{2}{figures/02}
  \subfiglistoverlay{1}{%
    \put(0.8,0){%
      \color{red}%
      \rule{1em}{2em}%
    }%
    \put(0,0.5){%
      \color{blue}%
      \rule{2em}{1em}%
    }%
    \linethickness{1ex}%
    \put(0.3,0.25){%
      \color{orange}%
      \vector(1,0){1.2}%
    }%
  }
  \subfiglistoverlay{2}{%
    \linethickness{1ex}%
    \put(0.6,0.6){%
      \color{yellow}%
      \vector(-1,0){1.7}%
    }%
  }
\end{subfiglist}
```



Listing 7: Overlays using the default `picture` environment

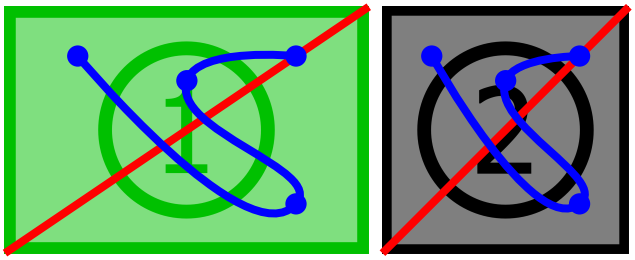
The following example shows how to employ PSTricks and use a `pspicture` environment as overlay, which has image width and image height as horizontal and vertical unit length, respectively. With this method the top right corner of the image can be addressed by the coordinate $(1, 1)$. Note that this example requires XeLaTeX, since PSTricks is incompatible with pdfLaTeX by design and the usual workarounds using the packages `auto-pst-pdf` or `pdftricks` do not seem to get along with the use of `pspicture` nested within self-defined environments properly.

```
\renewenvironment*{subfiglistoverlayenv}[2]{%
  \psset{xunit=#1,yunit=#2}%
  \begin{pspicture}(1,1)%
}{%
  \end{pspicture}%
}
```

The automatic rescaling behavior of this approach depending on image size and aspect ratio

is shown in listing 8.

```
\begin{subfiglist}{1 2}
  \subfiglistfile{1}{figures/01.png}
  \subfiglistfile{2}{figures/02.png}
  \subfiglistoverlay{1}{%
    \psset{%
      linewidth=3pt,%
      linecolor=blue%
    }%
    \psline[linecolor=red]%
      (0,0)(1,1)%
    \pscurve[showpoints=true]%
      (0.2,0.8)(0.8,0.2)%
      (0.5,0.7)(0.8,0.8)%
  }
  \subfiglistoverlay{2}{%
    \psset{%
      linewidth=3pt,%
      linecolor=blue%
    }%
    \psline[linecolor=red]%
      (0,0)(1,1)%
    \pscurve[showpoints=true]%
      (0.2,0.8)(0.8,0.2)%
      (0.5,0.7)(0.8,0.8)%
  }
\end{subfiglist}
```



Listing 8: Overlays using the `pspicture` environment

6 Known issues

- A newline command `\\` apparently produces at least `1pt` of vertical space. In fact, the command `\\[\dim]` produces a vertical distance of `\dim + 1pt` between the two lines it separates. This difference is automatically corrected for by the package. However, requesting vertical space of *less* than `1pt` will mess up vertical spacing.
- All images are placed on lines just like normal text. This works fine, as long as the images are higher than the minimum line height, because then the line height will automatically be increased to match the exact image height. However, for images with less height, the lines will keep their minimum height and thereby mess up vertical

spacing. This minimum height is probably given by `\baselineskip`, `1em`, `\ht\strutbox` or something related. Anyway, all these values are more or less the same.

7 Rough description of the algorithm

For every image – or more general for every "object" like images, spaces or newlines – several properties like width, height, aspect ratio and others have to be saved. In any object oriented programming language this would probably be solved by writing a class, which can store the various properties in fields, and creating instances for any of the considered objects.