The returntogrid package, v0.2

Ulrike Fischer*

2018/08/21

1 Introduction

This package offers a few commands to get something like an simple, semi-automatic grid typesetting. It does more or less what the existing `gridset` package does. The main differences to `gridset` are that `returntogrid` works also with lualatex, that it has also a command to do some horizontal movements, that it uses internally expl3 and that it does much less "security" checks: It actually started only as internal code, where I had good control about the use of the commands. Then I thought it would be useful to externalize it in case I needed it for more projects and added a thin wrapper of user commands.

It probably also has more bugs – report these at the issue tracker at github **https://github.com/u-fischer/returntogrid**.

2 Vertical grid moves

2.1 Restrictions / Deficiencies: Grid typesetting is not easy

The problem is at first that there are so many things that can destroy the grid:

- different font sizes with their different leading,
- all the various skips around and in lists, around math, around floats,
- pictures and tabulars (e.g. because of lines) often don't fit the grid,

*fischer@troubleshooting-tex.de

- the way TEX handles large boxes: People e. g. quite often believe that a tabular with three rows of normal height shouldn't disturb the grid but it does:

  some text
  tabular row 1
  tabular row 2
  tabular row 3
  some text not on the grid
- and more

Getting back to the grid isn't trivial either. In many cases it can be done by inserting a suitable vertical space. But due to the way TEX builds the page the needed amount can normally only be calculated in a second compilation. As the inserted space changes the following text probably new compilations are needed. So it can take a long time until everything is stable – if a stable solution actually exists: stretchable space, floats that move around and problems at page breaks can actually prevent a stable state.

So you want to use this package be aware of the following restrictions:

1. It will only work with documents/pages which uses `\raggedbottom`.
2. Many `\returntogrid` commands means many compilations. So use the command only when really needed. Setup as many spaces of lists and sectioning commands so that `returntogrid` is not needed to keep the grid.
3. Objects at/after page breaks needs special care, see section 2.3.
4. Top floats could be a real problem.
5. The code uses for comparision the y-coordinate of a point in the text upper corner of the first page. (Such coordinates are measured from the *bottom* of the page.) You will get nonsensical results if you change the geometry in the middle of your document unless you setup a new reference point and tell `\returntogrid` to use it.
6. The skips to return to the grid should normally be inserted only on the main vertical list, not in boxes. But the code doesn't check for inner mode as there could be special cases where it works (e.g. in a tcolorbox). If you use `\returntogrid` in such a place think carefully if this can lead to a stable state.

2.2 The main command

The main command is called \returntogrid and it takes one *optional* argument:

\returntogrid[⟨*key-val-list*⟩]

The command is used for both type of "moves": vertical and horizontal. In this section I'm describing the vertical movements, for horizontal movements see section 3.

Without the optional argument the command should normally be used only at the begin of paragraphs. It will force a new paragraph by issuing a \par, start the paragraph with a \leavevmode, then store the current position of the baseline and at the next compilation insert after the \par a hopefully suitable vskip to move this position to the grid.

It will always insert a *positive* vskip, even if the previous grid position is much nearer – I thought about small negative spaces but I'm not sure that this leads to a stable state.

In the optional argument the following keys can be used:

save Value is a label name. With this option \returntogrid only issues a \leavevmode and stores the position but doesn't insert any space and doesn't force a new paragraph. With this option a use inside boxes makes sense (see below).

use Value is a label name. With this option \returntogrid only inserts the vskip needed to move the position stored with the **save** option to the grid.

This can be e.g. used to move a section title or a line of a tabular to the grid

```
\returntogrid[use=mysec]
\section[xxxabc]{\returntogrid[save=mysec,strut=1]xxxabc}

\returntogrid[use=mytab]
\begin{tabular}[t]{l}
\hline
\returntogrid[save=mytab]xxxabc
\end{tabular}
```

`label` Value is a label name. By default `\returntogrid` generates numbered label names. This means that if you insert a new `\returntogrid` command at the begin of the document all following label names have to be renumbered and all skip calculated anew. As this can need lots of compilations you can avoid this by giving the label a unique name manually. Internally a package specific prefix is always added to the label.

`strut` Value is an integer. This will insert a strut of height $n \times$ `\baselineskip` + `\topskip`. See section 2.3 why this could be needed.

`debug-vgrid` This adds the command `\showdebugpagegrid` to the background of the current page.

You should ensure that `\returntogrid` isn't use in arguments that wander to the toc or the header as this would give multiply defined labels or insert the vskip in places where you don't want it.

Also it is up to you to use the save-use system in a sensible way – if the vskip doesn't move the position or if you use the vskip twice you probably will never get a stable position.

2.3 The page break problem

The space inserted to force an object onto the grid can push the object to a new page.

This is a problem if the object doesn't fall naturally on the grid there. In this case one would need a second (non-discardable) space that moves the object further down.

Currently the code doesn't do this. Instead it expects you to set the height of such problematic objects so that they keep the grid naturally.

2.4 Top floats

Floats at the top of the page can disturb the grid too. There is normally no sensible place to insert a vskip in the text running below such a float to return to the grid. This means that you should either avoid top floats, or give them a height that pushes the text below onto the grid without manual adjustments.

2.5 Debugging: Showing the grid

`\showdebugpagegrid`

This command requires that you load the package `tikz`. You can then use it e.g. like this to see the vertical grid lines:

```
\AddToShipoutPictureBG {\AtTextUpperLeft{\showdebugpagegrid}}
```

3 Horizontal grid: going to the next tab position

The `\returntogridsetup` command described in the following section has a key to define lists of "tab positions". It takes two argument: an (unique) name for the tab list and a list of dimension expressions[1].

`settabpositions`={⟨*name*⟩} {⟨*dim expression*⟩,⟨*dim expression*⟩,...}

The tab positions are set from the position you are issuing the command, and the dimensions describe the distance between two tabs. The command doesn't insert spaces but it uses a zero-width box, so use it at a places where a such a box doesn't affect your layout. The tabs must be set before the first use.

So

```
abc%
\returntogridsetup
 {
  settabpositions={demo}{1cm,2\fboxsep +1cm, 2cm}
 }%
text
```

sets the following tab positions for the list "demo":

abctext

[1] This means that e. g. sums of length are allowed.

3.1 Moving to the next tab position

You can move to the next tab position by using the `tab` key in the optional argument of `\returntogrid`:

`\returntogrid`[tab=⟨*name*⟩]

The name refers to the name of the tab list from which you want to use tab positions. If you don't give a name, whatever is the currently active tab list name will be used e.g. the one set with the `tab-list` option described in section 4. If no list with the name exists an error will be raised.

The code is *very* simple: It issues a `\leavevmode`, stores the current position in a label and at the next compilation calculates the distance to the next tab position and inserts a suitable `\hspace*`. If there is no next tab position it inserts a `\hfill` unless the key `hfill` has been set to false. It doesn't try to avoid line breaks or overfull lines.

The current position is stored with a automatically generated numbered label. Like in the case of the vertical movements you can avoid that this names changes all the time you add a command by setting a manual name with the key `label`.

As with vertical movements `\returntogrid` has a number of restrictions:

- Pay attention to stretchable space! If you move something to the end of the line, if you have a positive `\parfillskip` value, if the movement gives an overfull hbox, LaTeX will perhaps change the word spaces and then no stable hspace value can be found. Use `\raggedright` or end the lines with `\\`.

  I advise against setting an explizit tab position at the right margin, it is often unstable (and as the code inserts an `\hfill` after the last tab not needed anyway).

- Don't misuse the command! It is not a replacement for a real tabular or a tabbing as many `\returntogrid` in one line or one paragraph can mean many compilations. But it is useful if you want to move e.g. graphics in a three column layout.

3.2 The twoside problem

Like with vertical grid movements changes of the page geometry can lead to nonsensical results: The tab positions are stored in coordinates which are measured from the left side of the page.

Unlike as the situation with vertical grids geometry changes are actual common: a twoside document has different margins on the odd and even page. Normally you want the tab position to be relative to the text margins, so an offset is needed.

It `returntogrid` detects the twoside mode, or you use the **`twoside`** key in the setup command, `returntogrid` will add the difference between oddsidemargin and evensidemargin as an offset when needed. The code assumes that the page geometry of odd and even pages don't change in the middle of the document!

4 Setup and configuration

`\returntogridsetup`{⟨*key-val-list*⟩}

With this command you can change some defaults of the package.

`active` Boolean, default is true. It deactivates following `\returntogrid` commands.

`step` Value is a length, the default is `\baselineskip`. It sets the distance of the vertical grid.

`reference` The name of the point used as reference for the vertical grid. The default is a point at the text upper corner stored like this:

```
\AddToShipoutPictureBG*{%
 \AtTextUpperLeft{%
  \zsaveposy{ufgrid@vpointtextupperleft}}}
```

`offset` Value is a length, the default is `\topskip`. The reference point doesn't need to be on the grid, this key sets the offset.

`settabpositions` This stores tab positions that can be use with the `\nextgridtab` command See section 3.

`tab-list` Value a name. This sets the name of tab-list that will be used if you don't give a name explictely until the enclosing group end.

`hfill` Boolean. Default is true. When set to false no `\ ` is inserted after the last tab position.

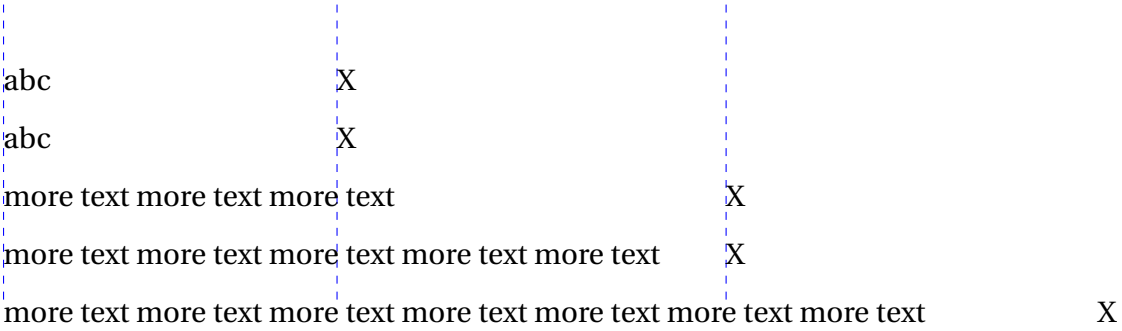
`twoside` Boolean. This activates/deactivates the offset described in 3.2 in case the automatic detection doesn't do what you want.

`debug-tab` This helps to see horizontal tabs when defining them. It should be used before using `settabpositions`. It needs the package `tikz`.

5 Examples

In the following text all parts with XXX are forced by the package to return to the grid.

First some horizontal movements:



The code

```
\noindent
\returntogridsetup{
 debug-tab,
 settabpositions={main}{0.3\textwidth,0.3\textwidth+0.05\textwidth
    },
 tab-list={main}
 }


\noindent abc\returntogrid[tab]X

\noindent abc\returntogrid[tab,label=mine]X

\noindent more text more text more text \returntogrid[tab]X

\noindent more text more text more text more text more text\
   returntogrid[tab]X

\noindent more text more text more text more text more text more
   text more text\returntogrid[tab]X
```

moving something to the next page

6 XXX

The code

```
\vspace*{29,1\baselineskip}
\returntogrid[use=mysec1]
\section[]{\returntogrid[save=mysec1,strut=1]XXX}
\rule{0.4pt}{3cm}
```

- abc

- abc

- abc

The code

```
\begin{itemize}[itemsep=3pt]
\item abc

\item \returntogrid abc

\item \returntogrid abc
\end{itemize}
```

a

xxxa

xxxabc

xxxsome text
some text
some text
some text
some text
some text
some text
some text
some text

7 xxxabc

abc
abc

The code

```
a

\vspace{3pt}

\returntogrid
xxxa



\returntogrid
\LARGE xxxabc
\normalsize



\returntogrid[use=mytab]
\begin{tabular}[t]{l}
\returntogrid[save=mytab]xxxsome text\\
some text\\
some text\\
some text\\
some text\\
some text\\
some text \\some text\\some text
\end{tabular}

\returntogrid[use=mysec2]
\section[]{\returntogrid[save=mysec2,strut=1]xxxabc}

abc \\abc
```