The `tikz-imagelabels` package*

Tobias Plüss

2019/01/29

Abstract

This package allows to put annotations (arrows, labels) on top of images using Ti*k*Z.

Contents

Change History

v0.1
General: Submission to CTAN . . . 1

*This document corresponds to `tikz-imagelabels` v0.1, dated 2019/01/29.

1 Introduction

For manuals, scientific reports and the like, one often needs to add annotations to an image (mostly a photograph) to label different items. An example of this is shown in Figure 1, which shows the names of the different stars in a star constellation. The package `tikz-imagelabels` allows to produce this kind of illustration.

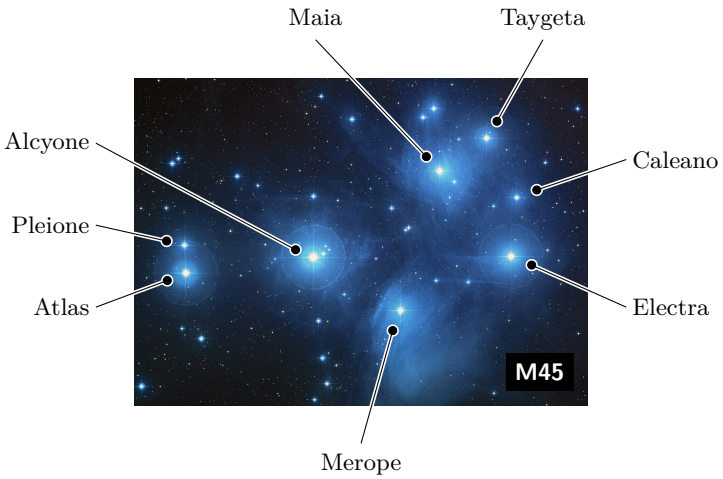

Figure 1: The Pleiades, also known as M45. Image source: `http://hubblesite.org/newscenter/archive/releases/2004/20/image/a/`

But why does this task deserve its own LaTeX package? there are several reasons:

- One does not need to edit the image in an external graphics program. They can be input directly into your LaTeX document.
- Since the labels and annotations are processed by LaTeX, all the font settings and the like remain consistent through the whole the document. However, of course the `tikz-imagelabels` package allows to configure the style.
- The annotations stay rock-solid if the image needs to be rescaled later.
- Since the arrows are processed by Ti*k*Z, they are vector graphics and thus, issues with scaling or unsharp/blurry labels, which would result if one labels the image in a graphics software, are avoided.

Figure 1 was created with the following code:

```
\begin{annotationimage}{width=6cm}{pleiades.jpg}
  \draw[annotation left = {Atlas at 0.3}] to (0.11,0.4);
  \draw[annotation left = {Pleione at 0.55}] to (0.11,0.49);
  \draw[annotation left = {Alcyone at 0.8}] to (0.39,0.45);
  \draw[annotation below = {Merope at 0.5}] to (0.58,0.28);
  \draw[annotation right = {Electra at 0.3}] to (0.84,0.45);
  \draw[annotation right = {Caleano at 0.75}] to (0.85,0.64);
  \draw[annotation above = {Maia at 0.4}] to (0.67,0.72);
  \draw[annotation above = {Taygeta at 0.9}] to (0.78,0.82);
  \draw[image label = {M45 at south east}];
\end{annotationimage}
```

2 Usage

2.1 Inclusion of the image

`annotationimage`

To include an image, the `annotationimage` environment is used. It has the following syntax:

`annotationimage[`⟨*grid*⟩`]{`⟨*options*⟩`}{`⟨*file name*⟩`}`

The ⟨*grid*⟩ is an optional parameter. If this parameter is present, i.e. if it has the value `[grid]`, then a coordinate grid is visible. The coordinate grid is used to find the coordinates of the points to be labelled. If the parameter ⟨*grid*⟩ is omitted, no coordinate grid is drawn. The code

```
\begin{annotationimage}[grid]{width=6cm}{pleiades.jpg}
\end{annotationimage}
```

produces the image shown in Figure 2.

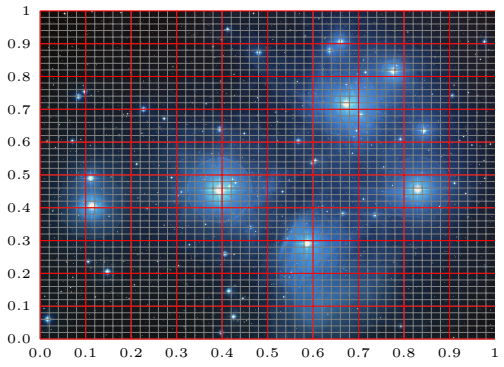

Figure 2: Example of an image with coordinate grid

⟨*options*⟩ is any set of options understood by the `\includegraphics` command,

e.g. `width=`, `height=` and so on. It may also be left empty, but in this case, the curly braces need to be there, though.

The ⟨*file name*⟩ is, obviously, the file name of the image. Like for the ⟨*options*⟩, any image format supported by `\includegraphics` may be used.

2.2 Adding a label

A label (like the "M45" in Figure 1) can be added to the image using following `\draw` macro:

`\draw[image label = {`⟨*text*⟩ `at` ⟨*placement*⟩`}];`

The ⟨*text*⟩ parameter is obvious. It contains the text to be put into the label.

The ⟨*placement*⟩ dictates the placement of the label. It may be one of `north west`, `north`, `north east`, `east`, `south east`, `south`, `south west` or `west`. Also `center` is possible, even though it possibly doesn't make a lot of sense. Figure 3 shows an example with several labels.

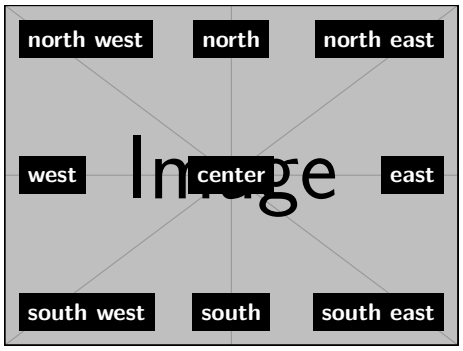

Figure 3: Example for the placement labels

The label in Figure 1 was drawn using the following code:

```
\draw[image label = {M45 at south east}];
```

2.3 Adding annotations

An annotation is added with the aid of the Ti*k*Z macro `\draw`. The syntax is as follows:

`\draw[annotation` ⟨*placement*⟩ `= {`⟨*text*⟩ `at` ⟨*position*⟩`}] to (`⟨*x*⟩`,` ⟨*y*⟩`);`

The ⟨*placement*⟩ is one of: `above`, `right`, `below` or `left`. It tells on which side of the image the annotation will appear. ⟨*above*⟩ and ⟨*below*⟩ basically determine the y coordinate of the text, while ⟨*left*⟩ and ⟨*right*⟩ determine the x coordinate of the text. The remaining coordinate is determined using the ⟨*position*⟩.

The ⟨*text*⟩ is the actual text of the annotation.

The ⟨x⟩ and ⟨y⟩ parameters are the actual coordinates where the arrow should point to. Note that `tikz-imagelabels` will automatically insert a small distance between the arrow's tip and the coordinate given, such that the arrow is close to the coordinate but does not cover it.

For example, the code

```
\draw[annotation left = {Atlas at 0.3}] to (0.11,0.4);
```

draws the text "Atlas" on the left-hand side of the image, at $y = 0.3$. The arrow will point towards coordinate $(0.11, 0.4)$ but ends shortly before this coordinate such that the interesting feature to be labelled is not covered by the arrow's tip.

3 Configuring styles

`\imagelabelset` Various options, like font size and so on, can be configured with the `\imagelabelset` macro. It uses the key-value syntax from Ti*k*Z, e.g.:

`\imagelabelset{`⟨*key*⟩ `=` ⟨*value*⟩`, ...}`

Multiple ⟨*key*⟩ and ⟨*value*⟩ pairs may be combined. The following sections list all possible configurations.

`\imagelabelset` can be put anywhere, but it makes sense to put it into the preamble of a document to ensure all images have the same look.

There is also a default style. If no `\imagelabelset` command is present, the default values for all options are taken. The default style used is as follows:

```
\imagelabelset{
  coarse grid color = red,
  fine grid color = gray,
  image label font = \sffamily\bfseries\small,
  image label distance = 2mm,
  image label back = black,
  image label text = white,
  annotation font = \normalfont\small,
  arrow distance = 1.5mm,
  border thickness = 0.5pt,
  arrow thickness = 0.4pt,
  tip size = 1.2mm,
  outer dist = 0.5cm,
}
```

The individual keys are described in the following sections.

3.1 Grid color

In most cases, it will not be necessary to adjust the grid colors. However, depending on the image, it may be desirable to do so. This is exactly what the ⟨*coarse grid color*⟩ and ⟨*fine grid color*⟩ are used for. Any color specification compatible

to Ti*k*Z may be used. The defaults are red for the coarse grid and gray for the fine grid.

3.2 Font and color for the labels

The font for the image labels may be configured with ⟨*image label font*⟩. By default, the image labels are typeset with bold, small, sans-serif font.

The background color of the image labels may be set using the ⟨*image label back*⟩ kay, whereas the text color is specified with the ⟨*image label text*⟩ key. Defaults for the background color and for the text color are black and white, respectively.

3.3 Distance of image labels to the image border

The ⟨*image label distance*⟩ key configures the distance, d, from the image's border to the border of the image label, as shown in Figure 4. By default, it is set to 2 mm.

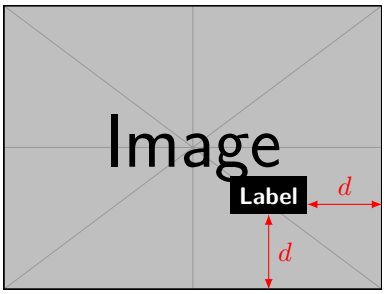

Figure 4: Illustration of the ⟨*image label distance*⟩

3.4 Font for annotations

The font used for annotations is set by ⟨*annotation font*⟩. By default, the `\normalfont` is used with small size.

3.5 Distance of arrow tips

As mentioned earlier, the arrows are shortened such that their tips don't cover the desired point. Figure 5 illustrates this. All the arrows point to the same coordinate, $(0.5, 0.5)$, but they end at the distance x away from the point. This distance may be configured using the ⟨*arrow distance*⟩. By default, this distance is set to 1.5 mm. This ensures that the arrow tips are close enough to the interesting features, but not so close that they cover important parts of the image.

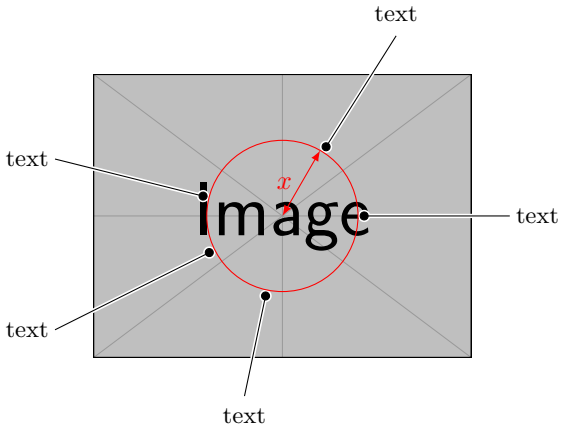

Figure 5: Illustration of the ⟨*arrow distance*⟩

3.6 Thickness and size of the arrows

The arrows themselves consist of two parts: the inner part, which is the actual arrow, and the border, which is, by default, a white border around the arrow. The border is required to ensure that each arrow is visible, no matter on what background it is drawn. The thickness of the black line can be configured using the ⟨*arrow thickness*⟩, whose default value is 0.4 pt. The thickness of the border around the arrow is configured with the ⟨*border thickness*⟩, having a default value of 0.5 pt. These two values correspond to the Ti*k*Z line widths `thin` and `semithick`.

The size of the round dot at the end of the arrows is configured using the ⟨*tip size*⟩. Figure 6 illustrates both, the ⟨*arrow thickness*⟩, and the ⟨*tip size*⟩, as parameters a and b, respectively.

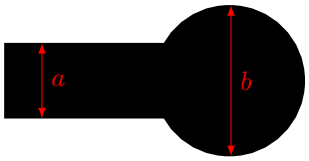

Figure 6: Illustration of the ⟨*arrow thickness*⟩, a, and the ⟨*tip size*⟩, b

3.7 Distance of annotation texts from the image

The parameter ⟨*outer dist*⟩ configures how far away from the image the annotation texts will be positioned. By default, this value is 0.5 cm.

4 Implementation

The only packages required are `tikz` and `xifthen`. If not already loaded, they will be loaded automatically.

```
\RequirePackage{tikz}
\RequirePackage{xifthen}
```

Some Ti*k*Z libraries are also reuiqred for proper operation.

```
\usetikzlibrary{
  arrows.meta,
  calc,
  positioning,
  decorations,
  decorations.markings,
  math,
}
```

4.1 Configuration

For the `\imagelabelset` command, a `pgfkeys` family is defined. All configurations (e.g. ⟨*tip size*⟩ and so on) and styles are stored in the PGF key `/imagelabels`. This ensures that these configurations don't overwrite any other parameters the user may have set elsewhere.

```
\pgfkeys{
  /imagelabels/.is family,
  /imagelabels/.search also={/tikz},
}

\def\imagelabelset{\pgfqkeys{/imagelabels}}
```

Then, a set of macros is created which stores the values for the individual configuration options.

```
\imagelabelset{
  coarse grid color/.store in = \maingridcolor,
  fine grid color/.store in = \finegridcolor,
}

\imagelabelset{
  image label font/.store in = \imagelabelfont,
  image label distance/.store in = \imagelabelsep,
  image label back/.store in = \imagelabelback,
  image label text/.store in = \imagelabeltext,
}

\imagelabelset{
  annotation font/.store in = \annotationfont,
  arrow distance/.store in = \arrowdistance,
  arrow thickness/.store in = \arrowthickness,
  tip size/.store in = \tipsize,
```

```
  border thickness/.store in = \borderthickness,
  outer dist/.store in = \labeloutersep,
}
```

4.2 Default configuration

The default configuration is set. This will ensure that each of the previously defined macros has a valid initial value, which may be overwritten by the user.

```
\imagelabelset{
  coarse grid color = red,
  fine grid color = gray,
  image label font = \sffamily\bfseries\small,
  image label distance = 2mm,
  image label back = black,
  image label text = white,
  annotation font = \normalfont\small,
  arrow distance = 1.5mm,
  border thickness = 0.6pt,
  arrow thickness = 0.4pt,
  tip size = 1.2mm,
  outer dist = 0.5cm,
}
```

4.3 Environment declaration

Next, the `annotationimage` environment is declared. It takes 3 arguments, the first of which is optional. If it is omitted, its default value is empty.

```
\newenvironment{annotationimage}[3][]{
```

Each time a new `annotationimage` environment is opened, this code will ensure that all the definitions stored under the PGF key `/imagelabels` are loaded. Then, a new `tikzpicture` is created.

```
\let\tikzset\imagelabelset
  \begin{tikzpicture}
```

The 2nd and 3rd arguments to the `annotationimage` are the size/scaling options for the image, as well as the actual image file. Thus, a new Ti*k*Z node called `image` is created; the node's content is the image.

```
  \node[anchor=south west, inner sep=0]
    (image) at (0,0) {\includegraphics[#2]{#3}};
```

Using a scope ensures that the top-right corner always has coordinate $(1, 1)$.

```
  \begin{scope}[x={(image.south east)},y={(image.north west)}]
```

Next, the first (optional) argument's value is checked. If the user said `[grid]` to the first argument, the following code is executed.

```
  \ifthenelse{\equal{#1}{grid}}{%
```

This actually draws the coordinate grid.

```
    \draw[very thin, draw=\finegridcolor, step=0.02]
      (0,0) grid (1,1);
    \draw[thin, draw=\maingridcolor, xstep=0.1, ystep=0.1]
      (0,0) grid (1,1);
```

then, the labels are put to the coordinate axes.

```
    \foreach \x in {0,1,...,9} {
      \node [anchor=north] at (\x/10,0) {\tiny 0.\x};
    }
    \node [anchor=north] at (1,0) {\tiny 1};

    \foreach \y in {0,1,...,9} {
      \node [anchor=east] at (0,\y/10) {\tiny 0.\y};
    }
    \node [anchor=east] at (0,1) {\tiny 1};
  }{}
}
{
```

Each time the `annotationimage` environment is closed, the previously opened `scope` and `tikzpicture` environments need to be closed as well.

```
\end{scope}
\end{tikzpicture}}
```

4.4 Style definitions for the annotations

What follows is the definition of the styling for the annotations.

```
\imagelabelset{
```

This is the style for the annotation arrow itself.

```
  annotation arrow/.style =
  {
```

The `preaction` first draws a thick white arrow. This arrow will become the border.

```
    preaction =
    {
      draw,
      -{Circle[fill=white, length=\tipsize+2*\borderthickness,
        width=\tipsize+2*\borderthickness]},
      line width = 2*\borderthickness + \arrowthickness,
      white,
      shorten >= \arrowdistance,
    },
```

After the `preaction` has been performed, this will actually draw the “normal” arrow.

```
    draw,
    -{Circle[fill=black, length=\tipsize, width=\tipsize]},
    black,
    line width = \arrowthickness,
    shorten >= \borderthickness + \arrowdistance,
  },
```

All the annotation texts have a common style. This ensures they have the same font etc. Setting the `inner sep` to 0.5 ex ensures that the distance between the text and the arrow is somewhat aesthetic. It is an empirically determined value.

```
  annotation node/.style =
  {
    font=\annotationfont,
    inner sep = 0.5ex,
  },
```

Next comes the styles for the different annotation placements. For an annotation being below the image, this style applies.

```
  annotation below/.style args = {#1 at #2}{
```

Using the annotation arrow style tells Ti*k*Z to draw an arrow as specified above, using the geometry defined with `\imagelabelset`.

```
    annotation arrow,
```

After the arrow has been drawn, a further path is inserted, which is the actual annotation text. For the annotations being above and below the image special care must be taken: a `\strut` is appended to the label text to ensure that texts being on the same side of the image are on the same line. Without the strut, the texts may be differently aligned, depending on their letters – e.g. letters “p” and “g” go slightly further down in the y direction than “a” or “b”.

```
    insert path = {
      (#2,0) ++ (0,-\labeloutersep)
        node[anchor = north, annotation node] {#1\strut}
    }
  },
```

The remaining annotation styles are defined similarly.

```
  annotation above/.style args = {#1 at #2}{
    annotation arrow,
    insert path = {
      (#2,1.0) ++ (0,\labeloutersep)
        node[anchor = south, annotation node] {#1\strut}
    }
  },
  annotation left/.style args = {#1 at #2}{
    annotation arrow,
    insert path = {
      (0,#2) ++ (-\labeloutersep,0)
        node[anchor = east, annotation node] {#1}
    }
```

```
  },
  annotation right/.style args = {#1 at #2}{
    annotation arrow,
    insert path = {
      (1.0,#2) ++ (\labeloutersep,0)
        node[anchor = west, annotation node] {#1}
    }
  },
}
```

4.5 Style definitions for the labels

Next follows the style definition for the image labels. A general style defines the appearance and color.

```
\imagelabelset{
  image label style/.style = {
    rectangle,
    minimum width = 5mm,
    minimum height = 5mm,
    fill = \imagelabelback,
    text = \imagelabeltext,
    font = \imagelabelfont,
  },
```

On the other hand, the `image label` style defines the actual image labels.

```
  image label/.style args = {#1 at #2}{
    insert path = {
      (image.#2) node[outer sep = \imagelabelsep,
        anchor=#2, image label style] {#1}
    }
  },
}
```