

The Scanpages Package

Michael Sharpe

December 1, 2016

I Briefly

This package is intended for production of documents based on scanned material in any format acceptable to `pdflatex` as a graphic inclusion—eg, `pdf`, `png`, `jpg`, though `pdf` works best with this package. For me, the format has been useful when trying to archive pre-`TeX` documents without converting them to `LaTeX` source documents but inputting the scanned pages one by one and adding indexing, hyperlinks, a table of contents (You'll have to do this using `\addcontentsline{toc}...`, and if using `hyperref`, you may need to add `\phantomsection` immediately before `\addcontentsline`), footnotes, marginal notes and the like. What makes scanned documents bothersome is the irregularities introduced by the scanning process. Pages are sometimes skewed and often offset horizontally and/or vertically from one another, and extraneous marks will likely appear, so each page may need adjustments. The methods of this package are of two types. First, the package `scanpages.sty` contains macros to make it more convenient to perform those adjustments and add adornments. Second, some kind of script is very useful for making the relevant part of the `tex` source, if you are handling more than a few pages of scanned material. For this, there are two almost equivalent scripts included to automate this process as much as possible, one written in `python`, the other in `AppleScript`. (The latter may be installed in `TeXShop`'s macro menu for completely self-contained usage.)

2 The `LaTeX` package

Starting with a scanned document, you need to measure four dimensions to specify the region you wish to import, and decide on a magnification factor to apply. The region to import from the scan should ideally be a bit larger (`10pt` is a good starting value) than the area typically containing all the data, and perhaps excluding original page numbers. (The part of the scanned page lying outside the region is cropped.) The four critical dimensions are:

- The x and y coordinates `llx`, `lly` of the lower left corner of the image region (the part you want to import) relative to the lower left corner of the scanned page;
- the width `w` of the image region;
- the height `h` of the image region.

The package is called using these items as options to the package.

```
\usepackage[scale=.9,llx=8cm,lly=12cm,w=3.5in,h=3.5in]{scanpages}
```

It is not an error to omit one or more of these values, as default values will be substituted:

- `llx` and `lly` default to `1in`.
- `scale` defaults to `1.0`.
- `w` defaults to `400pt`.
- `h` defaults to `600pt`.

The package creates then a destination box centered horizontally and vertically in the page and runs `\includegraphics` with appropriate values set for the `viewport` and `crop` whenever it sees entries like

```
\scanpage[rot=-1,page=1,dx=20,dy=15]{scan-0} % page= only for pdf files
%\index{}
%\put(450,250){Is this assertion correct?}
\endpicture\endgroup\newpage
```

which it interprets as follows:

- if `scan-0` is a blank string, a blank picture is produced, otherwise, if no corresponding graphic file can be found, `graphicx` will stop with an error message;
- `page=` will be understood only by `.pdf` inclusions;
- `rot` is an angle of rotation (degrees) in the mathematically positive sense (counter-clockwise) about the center. You may find it easier to enter `tanrot=0.175`, the tangent of the rotation angle, as this is more easily estimated from the picture; `page=1` selects the first page of the file `scan-0.pdf`. This option may be omitted if the file contains only one page;
- `dx` nudges the resulting picture to the right by `20bp`, and similarly for `dy`. (Actually, `dx` nudges the viewport to the left by `20bp/scale`.) If no unit is provided, `bp` is assumed;
- any material following the `\scanpage` line and before `\endpicture` can be used for index entries, table of contents entries, footnotes, additional graphic inclusions, annotations and the like. As the action is all taking place within a `LATEX` picture environment with `unitlength` set to `1bp`, each visible item must be placed in an instruction of the form

```
\put(x,y){...}
```

where `x` and `y` are purely numeric coordinates with implied unit `1bp`, which matches that required for dimensions in `\includegraphics`.

The page it produces contains, in addition to the scan material and other embellishments, a superimposed grid with unit `1bp` and a black box marking the edge of outline of the destination box. After all adjustments are complete, the grid may be suppressed by adding the option `nogrid` to the package option list.

2.1 Options to include the original page numbers

The `\scanpage` macro also accepts three additional options that allow the page number from the original scanned document to be displayed.

- `origpgnum=104` sets the current scanned page number to `104`, and specifies that this number should appear only when the `grid` is in use—that is, except when the option `nogrid` is in force.

Unless modified by an option in a future page, the scanned page number will increment by one and display on all subsequent pages.

- `origindest` specifies that the current scanned page number should be visible in the final document.
- `origpgnumoff` prevents the scanned page number from appearing in either the final document or the grid, from this page forward.
- The format governing the display of the current scanned page number is set by the macro `\origpgcmd`, whose default value is

```
\newcommand{\origpgcmd}{\put(400,-30){\texttt{Orig. page \# \the\origpgnum}}}
```

which specifies that entries like

```
Orig. page # 104
```

will appear with baseline 30bp (about 3/8 in) below the target box and 400bp (about 5.6in) from its left edge. You may change this in your preamble, after loading `scanpages`, with a similar line, like

```
\renewcommand{\origpgcmd}{...}
```

2.2 Resetting the initial choices

The options you chose when loading the package may be changed in the middle of a document. Just insert

```
\initviewport{<scale>}{<llx>}{<lly>}{<w>}{<h>}
```

to change your initial choice of options. E.g.,

```
\initviewport{.95}{3cm}{4cm}{8cm}{12cm}
```

will in effect make `scale=.95`, `llx=3cm`, `lly=4cm`, `w=8cm` and `h=12cm`.

3 Making a source entry for each page

Making more than a few entries by copy and paste, updating the indices, is quite boring, and I don't see how to manage this in T_EX, hence the need for the external scripts. The scripts operate on a file containing a small template that can generate what you need. I create a block of text like this:

```
%Repetitions=100
%Variables={NNN,0:1+1,1:2+-2,2:[-200+300],3:1+1(3)}
%%Begin page NNN0
%\scanpage[rot=0,dx=NNN2,dy=0,page=NNN0]{pic}
%%\index{}
%\endpicture\endgroup\newpage
```

The `%Repetitions=` line describes the maximum value of a counter starting at 1 that controls the iteration. The `%Variables=` line is less obvious. The fragment `{NNN,0:1+1,1:2+-2,2:[-200+300],3:1+1(3)}` means that variables are named `NNN0`, `NNN1`, `NNN2`, `NNN3`. Variable `NNN0` is initially 1, increments by

1 and so takes successive values 1, 2, . . . , 100. Variable NNN1 is initially 2 and increments by -2, while variable NNN2 alternates between -200 for odd counter values and 300 for even counter values. Variable NNN3 takes the same value as NNN0 but prints in a field of length 3, padding as necessary on the left with 0's so it substitutes successively 001, 002, . . . , 099, 100. (Note that commas, colons and plus signs are simply separators and have no arithmetic significance.) Running the script on this file will append 100 copies, the first two lines omitted, and with one % stripped and variables replaced by their successive values. The first two resulting items appearing as

```
%Begin page 1
\scanpage[rot=0,dx=-200,dy=0,page=1]{pic}
%\index{}
\endpicture\endgroup\newpage
```

```
%Begin page 2
\scanpage[rot=0,dx=300,dy=0,page=2]{pic}
%\index{}
\endpicture\endgroup\newpage
```

(The variable NNN1 was created with descriptor 1:2+-2, has initial value 2, decreasing by 2 at each iteration, but was never used.)

Alternating variables can be useful with a scan taken from a two-sided document, where offsets may be quite different for odd and even pages. Note too that where variables that increment must be integer valued, alternating variables can be alphanumeric. For example, the descriptor 2:[odd+even] would work as expected.

Two other special forms are available.

- Scanning software often places each scanned page in a separate file with names like scan-001.jpg, scan-002.jpg. In cover this case you need an integer variable padded to three places, which could be produced by the descriptor of the form 4:1+1(3) and a pattern like:

```
%Repetitions=100
%Variables={NNN,0:1+1,1:2+-2,2:[-200+300],4:1+1(3)}
%%Begin page NNN0
%\scanpage[rot=0,dx=NNN2,dy=0]{scan-NNN4}
%%\index{}
%\endpicture\endgroup\newpage
```

- If you are scanning two-sided material, you may end up with odd and even pages each saved in sequences of files like odd-0001.jpg, odd-0002.jpg, ... , even-0001.jpg, even-0002.jpg, ... and in this case it is handy to use a descriptor that “goes up by halves” to give the sequence 0001, 0001, 0002, 0002, The descriptor to use is like 3:2+1/2(4), which starts an internal counter at 2, increments it by 1 at each step, and prints half its value, truncated to an integer, and padded to length 4. (If the (4) had been omitted, there would have been no padding.) So the pattern to replicate would be like:

```
%Repetitions=100
%Variables={NNN,0:1+1,1:[20+40],2:[odd+even],3:2+1/2(4)}
%%Begin page NNN0
%\scanpage[rot=0,dx=NNN1,dy=0]{NNN2-NNN3}
%%\index{}
%\endpicture\endgroup\newpage
```

```
%\endpicture\endgroup\newpage
```

4 Whiting out scan artifacts

Use a white shape to overwrite scan artifacts. The package provides some macros to assist. Both are based on glyphs in a special font included as part of the `scanpages` package, one a square shape and one a circular shape, each 500 units wide and tall, with zero depth and zero side-bearings. (A thousand units is equal to 1em, so at with 10pt typesetting, 500 units amounts to 5pt.)

```
\whitesq(150,200){5} % width and height scaled by 5  
\whitesq(150,200){5}[2] % width scaled by 5, height by 2  
  
\whitecirc(150,200){5} % width and height scaled by 5  
\whitecirc(150,200){5}[2] % width scaled by 5, height by 2
```

the only difference being in the the shape of the of the region whited out. These macros make use of `\scalebox` from the `graphicx` package, and follow the same syntax for scale width and height.

5 Differences between the scripts

The AppleScript is meant to work within TeXShop after installation in the TeXShop Macros Menu—see instructions below. It works on the selected part of the file, and its output is placed in the same file, which can be part of a larger document. The `python` script is meant to run from the command line on a file containing just the pattern text, and produces output in the same file, which can then be copied than into your working `.tex` document. Eg, if you copied the script into a directory on your `PATH` and made it executable

```
replicate.py myfile.txt
```

would read input from and write output to `myfile.txt`. (The script has been tested with `python 2.7.4` under MacOS 10.9.2. It should work without modification under Linux but may require some minor changes in Windows using a `python` from `activestate.com`.) The two scripts give the same output provided you use variable names that are identical in case to those the one in the `%Variables=` line. (In the examples above, this was always `NNN`.) The `python` script is case sensitive, but the AppleScript is not—it will act on any variant like `nnn0` or `NnN1` as well.

5.1 Installation in TeXShop's Macros Menu

Select TeXShop's Macros Menu and choose the top item—`Open Macro Editor ...`. Then, from the same menu, choose the second item—`Add macros from file ...`—and navigate to `replicate.plist` in this distribution. When you choose that file, the AppleScript will be installed under the name `Replicate` in the Macros Menu.

6 Example

The last page of this document shows the result of using `scanpage` with a blank filename, which does nothing but create a blank picture with a grid, then overwrite a part of the page with a portion of another. The illustration will not show the (`nogrid`) final result.

```
\newpage
\initviewport{1}{1in}{1in}{8cm}{12cm}
\scanpage{}
%viewport=xl yl xr yr
\put(100,200){\includegraphics[viewport=165 530 300 700,clip]{pic1}}
\endpicture\endgroup\newpage
```

(A useful variant of this technique was suggested by my colleague Gill Williamson. Suppose your starting point is a hand-sketched graph or some other mathematical illustration. Scan the sketch and create a one-page document as described above on which you will make embellishments such as labels, then save this with no grid and import that document into your final larger document. This protects you from having to make many individual changes if you change the size of your picture so that the `\put()` coordinates all have to be changed.)

400

350

300

250

200

150

100

50

0

-200 -150 -100 -50 0 50 100 150 200 250 300 350 400

Page 1

-50

-100

-150

-200

-250

-300

-350

