Bijlage 3

Color in professional print production

Siep Kroonenberg
Faculteit der Economische Wetenschappen
Rijksuniversiteit Groningen
n.s.kroonenberg@eco.rug.nl

abstract

This paper takes a look at issues arising in color printing, such as color models, color conversion and color separation. Increasingly, it is feasible to perform these functions on existing PostScript files, independent of the authoring software. The pdf format plays a key role in this trend.

keywords

Color printing, color model, device-independent color, color conversion, color separation, pdf

Can TEX produce professional-level color output? We shall see that, as to color support, TEX lags far behind commercial desktop-publishing software. But we shall also see that there are some viable options, and that recent developments tend to move advanced color support back to where it belongs, viz. to the printer.

Although TEX doesn't have built-in support for color, the `\special`-mechanism provides hooks for adding driver-dependent color support[1].

Color perception

Our retina contains three types of light-sensitive cone. Color vision depends on these types of cone being especially sensitive to resp. the red, green and blue part of the visible spectrum. If light is composed of wavelengths distributed evenly across the visible spectrum, we perceive it as white. In other words, red, green and blue (1,1,1) add up to white. The absence of all three (0,0,0) means black, or no light. Rgb is an *additive* color model.

In conventional printing, on the other hand, we start out with white, or no ink. By adding inks, more and more light gets absorbed. The cmy (cyan, magenta, yellow) color model is the opposite of the rgb model: cyan absorbes red, magenta green and yellow blue. Cyan plus magenta plus yellow absorbes approximately everything, therefore is approximately black. In the cmyk model, black ink is added to get a better black than is possible with the other three colors alone, and because replacing equal amounts of cyan, magenta and yellow with black saves ink. Cmy and cmyk are subtractive color models.

Defining colors in TEX and LATEX

In LATEX, color support is standardized through the `color` package:

```
\usepackage[dvips]{color}
```

The color package supports various color models, which can be freely mixed within one document. We might use rgb and cmyk colors as follows:

```
\textcolor[rgb]{1,0.5,0.5}{pink}
{\color[cmyk]{0,0.5,0.5,0} pink}
```

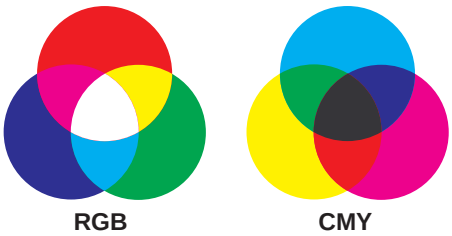


Figure 1. Mixing colors in rgb and cmyk

In the gray color model, 0 is black and 1 is white:

```
\definecolor{mygray}{gray}{0.5}
\textcolor{mygray}{gray}
```

The *named* color model allows one to use named colors:

```
{\color[named]{Apricot} Apricot}
```

In the case of dvips, a set of named colors, including 'Apricot', is predefined as cmyk colors.

A full description of the syntax for defining and using colors can be found in the LATEX Graphics Companion and

1. *Specials* are driver-dependent codes which TEX can place in a dvi file. Implementing color in TEX requires on the one hand macros which ask TEX to write specials to the dvi file, and on the other a dvi driver which interprets those specials. Dvi drivers are at liberty to ignore specials which they don't understand, guaranteeing that dvi files with specials still are device-independent in the sense that they can be processed by arbitrary dvi drivers.

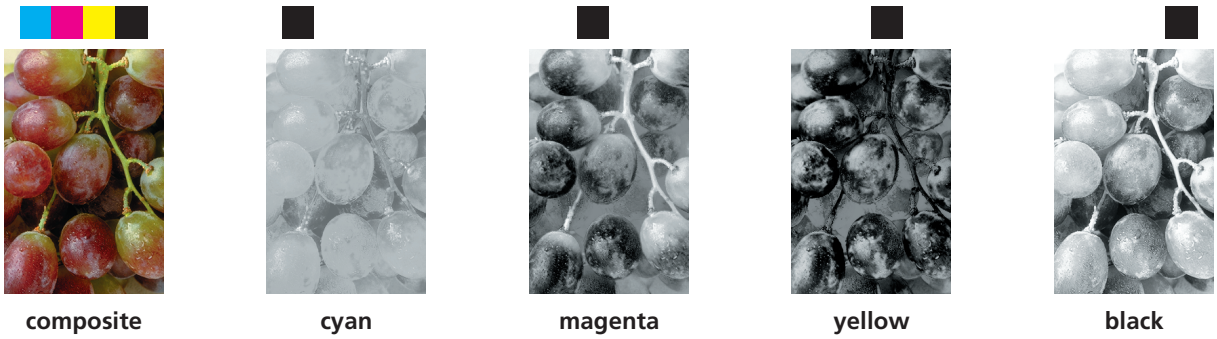


Figure 2. Separations

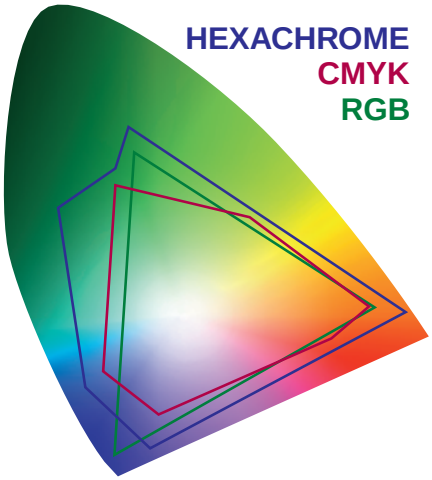


Figure 3. Color gamuts. The colored shape represents the gamut of visible colors; the inscribed polygons represent the gamuts of several device-based color models. The hexachrome model is a print- based model in which green and orange inks are added to slightly modified cmyk inks.

in the documentation accompanying the graphics and color packages.

Users of plain TEX can make use of colordvi.tex, which offers color support for dvips and is distributed with this program.

Device-independent color

The rgb color model does not actually refer to the red, green and blue cones in our eyes, but to the red, green and blue components of devices that generate color images (monitors) or record them (film, scanners). The same rgb value on different devices does not necessarily correspond to the same color. In general, rgb devices are not able to represent all visible colors. The rgb *gamut* (for a typical rgb device) is smaller than the gamut of visible colors.

We already saw that on cmyk devices, i.e. printers, colors are generated in a radically different way than on monitors, so it is only natural that the cmyk gamut differs from the rgb gamut. The cmyk gamut falls even farther short of the gamut of visible colors.

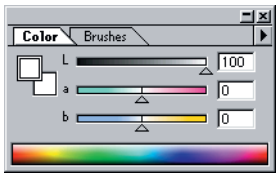


Figure 4. Specifying L*a*b colors in Photoshop

Even for someone not interested in color fidelity, the differences between rgb and cmyk are too gross to ignore. If you are used to choosing colors on screen, you'll find that vibrant colors turn dull and flat on paper; if on the other hand you select your colors from printed swatches you'll find that your screen just can't reproduce some cmyk print colors such as pure cyan.

You can specify color in vector graphics and within TEX as cmyk colors and you may never have to deal with color conversions. Photographic images, however, start out life in rgb mode, and if they are to be printed in conventional process color, the differences in rgb- and cmyk gamuts must be dealt with. The challenge here consists of contracting the source gamut within the target gamut without disturbing color relations within the image.

Note that we quietly ignore differences between devices with the same color model: not all monitors are the same, and the same cmyk color specification will look very different on newsprint and on glossy art paper. Professional designers, who really need accurate color, calibrate their monitors and carefully target their output for specific print setups. You might even have to pay money for color profiles of graphics devices.

To resolve these color conversion issues, the CIE (commission Internationale de l'Éclairage) created in 1931 a standard for color specification which allowed device-independent color and is based on color perception. The

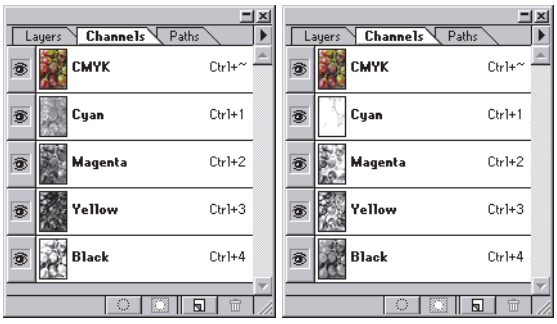


Figure 5. Converting rgb to cmyk: black generation. This illustration shows the Photoshop Channels palette for two cmyk versions of one rgb original (the same original as in the separation example). In Photoshop speak, the various color components of an image are channels. In the left palette, black generation is low, i.e. lesser amounts of cyan, magenta and black are replaced with black. In the other one, black generation is set to the maximum: equal amounts of cmy are replaced by the equivalent amount of black. This results in lighter cmy channels and a darker black channel. The advantage of a low black generation setting is more shadow detail but may result in more ink buildup than the paper can hold.

best-known CIE color model is L*a*b, whichcolor space has one lightness dimension *L* and two color dimensions *a* and *b*.

Ideally, images should be stored in a device-independent color model, and only at output time be converted to the appropriate device color space. This would facilitate designing simultaneously for different media, which is not unimportant in this age of the Internet.

Spot color

At the other end of the scale there is spot color, which is mostly relevant for printing. Pantone publishes books of color swatches which serve as a reference. If you want a second color for your publication then you cite the Pantone number or -name to your printer.

Black and one or two Pantone colors is cheaper than full-color: fewer print plates are required, and a spot color design usually doesn't require quite the same care in registration and color matching as a full-color job. However, spot color is not used exclusively for economy: a corporate identity might involve the use of spot color for the company logo and for letterheads, often in addition to the four process colors. A cmyk approximation of a Pantone color often isn't close enough, and fine details always suffer if they are rendered with a halftone screen or multiple inks.

Spot varnishes and metallic inks also require separate printing plates, often in addition to plates for process color.

Color separation

Color printing is still mostly done with conventional presses, with a separate plate for each ink; in other words, conventional color printing requires color separation.

In very simple cases this may be done manually: if a page contains black and red elements, one can make all red objects white and then print the black objects; for the red plate, one can make black objects white and red objects black.

This is in fact not so very different from what separation programs do: they would print such a page twice with appropriate redefinitions of the colors.

Professional graphics applications have this capability built in. TEX users can create cmyk separations with dvips using a command-line such as

```
dvips -h colorsep.pro -b 4 test.dvi
```

the parameter `-b 4` specifies that each page should be printed four times, and the parameter `-h colorsep.pro` loads a header file colorsep.pro which takes care of color redefinitions. It also adds cropmarks, which indicate the paper edge but also assist the printer in aligning the plates on press. With some tinkering it is not too difficult to modify this procedure for spot color separation. However, I am not aware of any ready-made solutions for using spot color *in addition to* process color. Another problem with colorsep is that bitmaps can't be separated this way, unless they contain cmyk data[2].

Another PostScript program, aurora.pro, does offer some support for spot color, but it recognizes a spot color by its cmyk composition instead of by its name. This program can handle more general color models in bitmapped images. A drawback of Aurora is that it generates separations per ink instead of per page. Film is subject to some stretch and shrink, therefore good registration requires that all films for one page are run at one go. Aurora does not add cropmarks.

When you use colorsep or aurora, you find yourself very much on the bleeding edge, so be prepared to run tests before committing yourself on a large job. Better, if at all possible you should leave separation to your printer or service provider. He may have postprocessing software to separate existing PostScript print files, or his imagesetter may have separation capabilities built-in.

Overprint, knockout and trap

Registration of print plates is never perfect. Even slight misregistration may be very noticeable. One remedy is to

2. The most economical way to turn a grayscale image into cmyk data from is to convert it to a black *monotone*, i.e. a duotone with just one color, which is black.

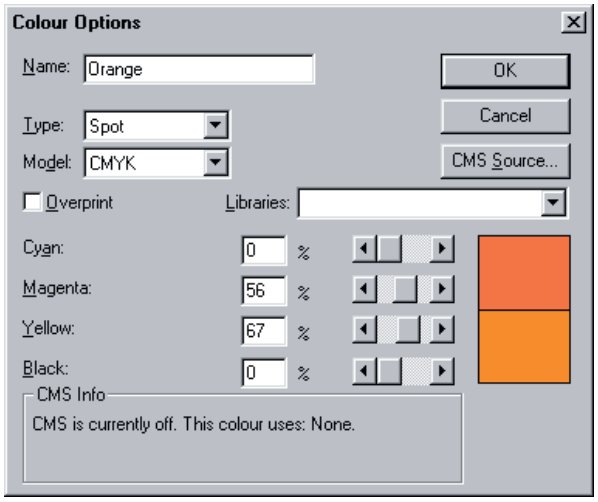


Figure 6. Specifying colors in PageMaker. Here, a spot color is being defined, with a cmyk equivalent. At separation time, one will have the choice whether to print 'Orange' on a plate of its own or whether to use the cmyk equivalent. Note the Overprint setting. If it had been checked, colors on other plates would not be knocked out underneath orange objects.

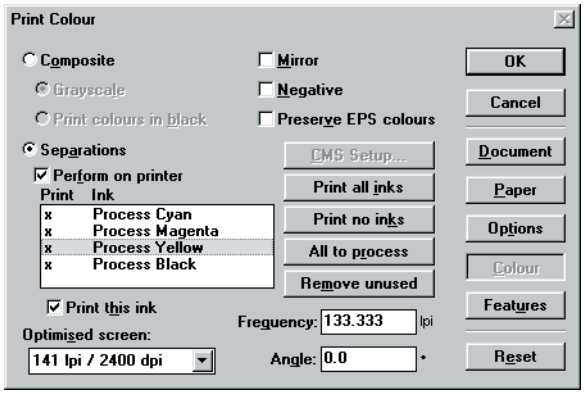
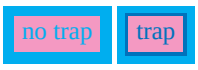


Figure 7. The Color page of PageMaker's print dialogue. Notice the 'Perform on printer' checkbox and the screen settings. Professional graphics applications consult PPD's or PostScript printer descriptions as to the capabilities of a printer. The PPD, which was selected in the first page of the print dialog, is for a PostScript level 2 imagesetter with built-in separation capabilities.

create a slight trap or overlap where colors touch. I hope that the printing of the MAPS won't be too perfect, otherwise the following illustration won't show my point (the trap is greatly exaggerated):



If one color is darker than the other, then the lighter color

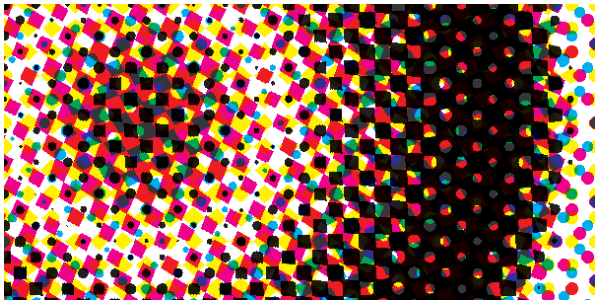


Figure 8. A greatly magnified process-color halftone. This example displays some moire.

should spread into the darker, in order that the integrity of shapes is maintained.

Trapping introduces device-dependence: the required amount of trap depends on press conditions. Trapping may be done by a postprocessing program or even by the imagesetter; ask your printer. Inexperienced users probably do well not to do their own trapping. Instead, avoid trapping by selecting adjacent colors with enough ink in common, so that misregistration won't be conspicuous, or have trapping done by the printer or service bureau, or even ignore the problem.

A simpler remedy against misregistration is overprinting. If e.g. a black shape sits on top of a magenta background, then with overprinting the magenta background continues underneath the black object:

	black plate	magenta plate	final printout
knockout	**text**	**text**	**text**
overprint	**text**	**text**	**text**

For small type and fine detail, this is the only reasonable solution. In TEX, this might be accomplished in the above example by adding magenta to the text color:

```
\colorbox{magenta{\color[cmyk]{0,1,0,1}text}}
```

This trick is also useful when printing to color printers, which ignore overprint specifications.

Professional draw- and desktop-publishing applications allow one to set a color simply to overprint instead of knockout. With TEX and dvips, one would have to insert raw PostScript. The workaround described above, viz. adding background color to the foreground object, breaks down if the background does not consist of a single color.

Screens

Tints of a color are achieved by printing dot patterns. To this end, the imageable area is divided into grid cells, with

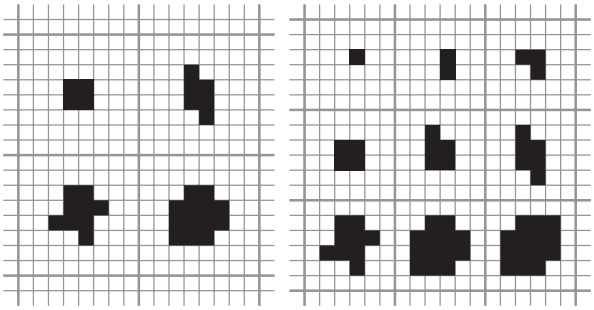


Figure 9. Left a 0° 8×8 grid with 65 graylevels, right a 0° 6×6 grid with 37 graylevels.

each cell being white, black or containing a cluster of black dots. Since the grid cells must consist of an integral number of dots, only a discrete series of screen frequencies is available for a given resolution.

Finer screens allow for rendering of finer detail, but because a grid cell consists of fewer printer dots, fewer gray levels are available. Also, an excessively fine screen might clog up on press or distort gray values, since dots tend to spread out somewhat on press. This latter phenomenon is termed *dot gain*.

With color printing, there is an additional twist: screen angles and -frequencies of the different plates must be coordinated with some care in order to avoid interference or *moire* patterns.

A recent development is FM, or frequency-modulated screening. This does away with moire problems, but aggravates dot gain, and is computationally very expensive.

This is yet another area where TEX offers no control except to PostScript hackers. It may not be smart to rely on default settings of an imagesetter, since these may be very conservative, resulting in unnecessarily coarse screens. If your printer is willing to handle screen settings for you, so much the better.

Color printers

For shorter print runs, digital printing deserves serious consideration. I am not going to quote a number, since the break-even point is moving up all the time, and quality is getting better too. These printers often employ custom screening technologies or *contone* or continuous tone, which is achieved through variable dot size. Contone technology won't help type or line art, but does offer very sharp and detailed rendering of photographic images at relatively low resolutions.

This has consequences for the resolution needed for photographic images: whereas for halftoned images a resolution of more than twice the screen frequency is pointless (say, 300 dpi for a 150 lpi screen), contone devices can make use of image resolutions up to the device resolution, which will be 400 and up.

The printer may use additional toners besides the traditional cmyk colors in order to improve color fidelity. Obviously, the printer must do some translating internally in order to use these toners, since it is likely to get cmyk or rgb instead of custom color data. If you have rgb images, it might be best to leave them in that format; ask your printer and/or arrange for some tests.

Pdf workflow

PostScript was meant to be device-independent but it turns out that we still need to know a lot about the output device. If your printer or service provider has facilities for postprocessing such as color separation, trapping and imposition, consider creating generic or device-independent PostScript and let him handle the device-specific part, including color model conversion.

Adobe's pdf (portable document) format should facilitate such a division of labor. Pdf has the same imaging model as PostScript. It should be a more tractable format for postprocessing programs since it does not support programming constructs, and page independence is built in. The current version 1.2 of the pdf format can hold the same prepress information as PostScript including CIE Lab and other CIE color specifications. PostScript 3 printers support pdf directly.

Adobe envisions a future in which documents travel the prepress workflow in pdf- instead of PostScript format, retaining device-independent color until a late stage, and in which device-specific processing such as trapping, color conversion and -separation, and imposition can be added at successive stages. TEX users will generally prefer to have most prepress information added late, i.e. by the printer, whereas many professional designers will prefer to retain control themselves, and add prepress settings early, i.e. from within their authoring software.

The prepress- and print industry seems quite enthusiastic about pdf. Pdf-based printers appear on the market, and software support is rapidly improving. One example is the Crackerjack plugin, which adds color separation with all its twists and turns to Acrobat Exchange. Soon, TEX users no longer need to be jealous of the prepress capabilities of QuarkXPress and PageMaker (not that we ever were, but we should have been), because these capabilities are increasingly becoming available outside authoring software.

Production note

The illustrations for this paper were all converted to or created with cmyk color. The publication itself will be converted by the editors from TEX to PostScript to pdf, and by the printer back to PostScript. If somebody fails to pay at-

tention, some colors will have traveled from rgb to cmyk back to rgb and again to cmyk and will be much the worse for wear ;-).

Further reading

For online documents, dates and exact locations are omitted, because these are subject to too much change.

Adobe Systems, PostScript Language Reference Manual, second edition, Addison-Wesley 1990.

Adobe Systems, Portable Document Format Reference Manual. Available online from `www.adobe.com`.

Adobe Systems, PDF for Prepress Workflow and Document Delivery. Available online from `www.adobe.com`. Various other documents of interest on pdf, PostScript Level 3 and color management can be found at this site.

David Carlisle and Sebastian Rahtz, Graphics package. Available from CTAN and included in most TEX/LATEX distributions. This is in docstrip format; just run `latex graphics.dtx` to get formatted documentation.

Michel Goossens, Sebastian Rahtz and Frank Mittelbach, The LATEX Graphics Companion, Addison Wesley, 1997.

Thomas Rokicky, Dvips manual. Included in the dvips distribution; available from CTAN and included in most TEX distributions.

Postprocessing software vendors, see e.g. `www.lantanarips.com` for Crackerjack, and `www.imation.com` for Trapwise and Presswise.

Expansion, what is that? (continued)

Although from the users point of view, TEX expands whatever macro it meets while reading the paragraph, in practice expansion can be postponed of prohibited. References for instance can only get their meaning when the content is placed on the page, and this can be many paragraphs later. Postponing is needed to keep the references in tune with the page numbers. When you hear macro writers talk of whatsits, they are probably talking about such postponed expansion and hidden things.

The story is still not completed. Hidden for the user, and depending on the macro package in use, TEX also inserts things like color directives, indentation, skips and when asked for all kind of frills, like paragraph numbers. This means that what you type is not per se what you will get, in fact, what you see is what TEX made of your input. Some things are quite complicated in TEX. Take for instance multi-column typesetting. Splitting columns is to be programmed and is not part of TEX. The same goes for adding line numbers. Such at first sight simple things ask for recalculating page breaks and/or reading back already typeset lines and post processing them.

Hans Hagen