Printing colour pictures

Friedhelm Sowa
Heinrich-Heine-University, Computing Centre, Universitätsstraße 1, D 40225 Düsseldorf, Germany
sowa@convex.rz.uni-duesseldorf.de

Abstract

Printing colour pictures in a TEX document needs a driver program that is able to exploit the capabilities of a colour device. The driver must separate the colours of the picture into the basic colours used by the colour model supported by the output device. This was the purpose to develop the dvidjc-drivers for the Hewlett Packard inkjet printers and to upgrade BM2FONT to version 3.0.

The solution described in this article proposes a device independent approach to printing coloured TEX-documents, not only on expensive PostScript devices but also on cheap colour printers.

How it started

Good reasons. During the last few years the hardware industry has supplied the market with different kinds of colour printers. In particular ink jet printers with increasing quality and decreasing prices appeared on the market. So not surprisingly, more and more TEX users demanded dvi driver programs, that exploit the colour ability of these printers.

The answer to this question usually was the recommendation to use the colour package by Jim Hafner and Tom Rokicki and to print the formatted document via dvips using GhostScript. This was a good recommendation, but the supplementary megabytes, necessary for a GhostScript installation, could be a problem. Another problem is the quality of the output produced by GhostScript, which is way below what users expect of TEX output. Moreover the procedure is rather complicated and slow.

Yet it was not only the availability of good and cheap colour printers that brought the colour problem into the foreground. There were also the discussions and decisions on colour support in TEX by the LATEX3 and $\mathcal{NTS}$ groups that made it clear that an interface will be designed similar to the known implementation of the dvips driver.

For all these reasons it was decided to write a driver program for the HP DeskJet family, hoping that it could be an example for other devices. It is even to be expected that it could be adapted to the final design of a graphics and colour interface, which is to be developed in the future.

An easy solution. The starting point of the project was a driver program for a dot matrix device and a program which could separate colours of a picture. The driver program was dvidot, written by Wolfgang R. Müller, which had to be extended in a way, that it could use the different inks of the HP DeskJet printers, support the \special-commands for coloured text and, the most important and difficult point, produce mixed colours from the primary colours cyan, magenta, yellow and black.

Colour separation could be done by BM2FONT, so that no special interface for the driver program had to be written. The four colour separations of the picture had to be stored in fonts, differentiated by their names, and then included into a document by overprinting each other.

This plan seemed to maintain the compatibility to dvips and the output it produces on paper, assuming that coloured text and pictures are handled similarly by PostScript. At the end of this paper it will be explained why this assumption was wrong.

Colour models

Before describing the dvidjc driver and the new features of BM2FONT version 3.0, some remarks have to be made on the different colour models we have to deal with in the real electronic world. The most important and mostly used models are based either on the primary colours red, green and blue, or on cyan, magenta, yellow and black.

The RGB-model. The colours red, green and blue are used for phosphorescent surfaces to produce mixed colours, a technique used in colour monitors. The following diagram shows what colours result from overlapping areas of fully saturated primary colours.

Digitized pictures mostly use the RGB model by storing the colour of the pixels in three bytes, each representing the intensities of the primary colours

in the range between 0 and 255, where 255 means a full saturation of the colour. Some economical picture formats use up to 256 colours by using a pixel index to a palette of such colour triplets like PCX or GIF. Others like TIFF or TIGA use 24 bits for each pixel and can reproduce up to 16,7 million possible colours.

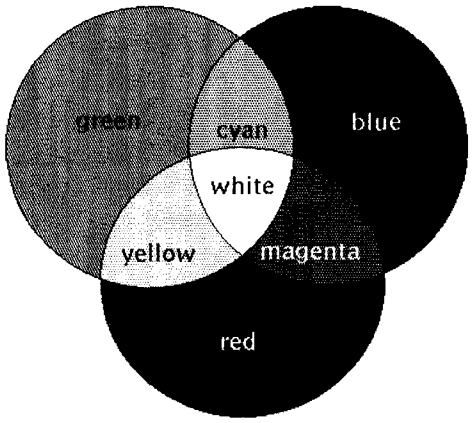

Figure 1: additive colour mix

The CMY-model. The primary colours cyan, magenta and yellow are used for reflecting surfaces like paper. Depending on the technique of a printer it is more or less difficult to position spots of primary colours on paper within a limited area to achieve a good quality picture.

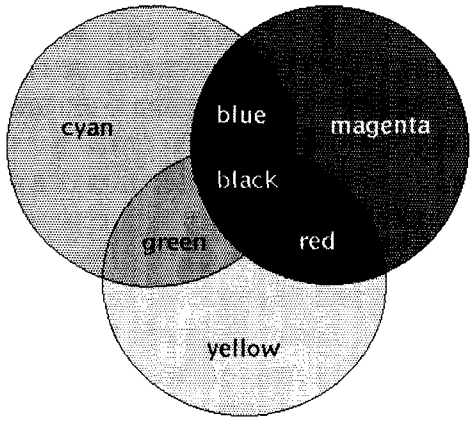

Figure 2: subtractive colour mix

The first problem is the saturation of the colour ink, because full saturation produces colours that are too intense. So usually a colour saturation of about 60% is used for printing each primary colour. The second problem is that areas of overlaying primary colours do not produce a solid black but a colour that looks like a mixture of dark brown and green. So in practice black ink, which extends the CMY-model to a CMYK-model, is used in addition for printing colour pictures.

The effect of mixing the primary colours of the CMY-model is demonstrated by a diagram similar to the one above, for the case of full saturation of cyan, magenta and yellow:

When producing the four different separations of a colour picture, screens of different angles should be used to avoid moiré effects. The common angles are 0° (yellow), 15° (magenta), 45° (black) and 55° (cyan).

Colour separation with BM2FONT

The way it works. To generate colour separations of a picture with BM2FONT, it was necessary to extend the program with the following features:

- converting RGB-colours to the CMYK colour model;
- correcting ink impurities;
- extracting the common black component of the colours.

The intensities of the complementary colours are calculated by subtracting the saturation of red, green and blue from 255, which means full saturation of this colour:

cyan = 255 – red
magenta = 255 – green
yellow = 255 – blue

This is done in BM2FONT3.0 for all bitmap formats, which come with a colour palette, and for TIFF files with RGB triplets, representing up to 2^{24} possible colours.

Correcting the impurity of the ink used is a very device dependent task. The process is known as *Undercolour Removal*. It removes a part of yellow intensity under magenta and partially magenta under cyan before finding out black:

cyan = cyan – $r_m$magenta
magenta = magenta – $r_y$yellow

BM2FONT uses 0.3 both for r_y and r_m, but further versions will read those values from the command line, because colour printers different from the HP DeskJet 5xxC might need other values to get good results.

The blackness of the pixel colour is the minimum value of the new CMY triplet (see figure 2). This blackness now has to be subtracted from the calculated primary colour intensities. The last step is not yet implemented in BM2FONT, because its effect depends on the generated screens, whose angles are different from those mentioned before. As the current solution gave good results on the HP

DeskJet printers, we released this version without implementing another routine, which still has to be tested for different devices.

All this is done in different runs, where the actual colour is given by the -k option on the command line. Each run produces one picture in the selected colour as one or more TEX fonts, which are to be overlaid in the document. Former versions of BM2FONT deleted the white space around the rectangle containing the picture, whose corners are determined by the first significant black pixel. This feature is not useful for colour pictures, so it should be switched off by using the -j option.

Input for TEX. The main input for TEX and the driver programs is created by BM2FONT in the form of the .tfm and .pk files, while the input files, generated in each run for the primary CMYK colours, construct the picture in the document. Unlike halftone pictures, where we deal with black and white pixels on paper, we now have to tell TEX, to write into the dvi file, which colour is to be used when printing the single parts of the colour picture.

This is done by a \special command, similar to the one which is defined in the colordvi-package, created by Jim Hafner and Tom Rokicki:

```
\special{color push #1}##1
\special{color pop}
```

Here we have in #1 the colour, for example "Apricot", and in ##1 the text that is to be printed in the desired colour. The dvips driver uses the file color.pro to tell the PostScript device how to produce this colour \Apricot:

```
/Apricot{0 0.32 0.52 0 setcmykcolor}
```

For efficiency reasons we did not implement inside the dvidjc driver a routine to read a device specific colour description from an ASCII file. Instead, the driver was supplied with a modified macro file colordjc.tex/sty, knowing that this was not the final solution. The difference is, that instead of the name of the colour the intensities of the primary colours cyan, magenta, yellow and black are written into the \special literal, while the new colour is defined as

```
\newColor Apricot {0 0.32 0.52 0 }
```

The current version 3.0 of BM2FONT provides no special TEX code to typeset colour pictures. This code is contained in the colordjc file. There a macro \loadcmykpic[#1,#2,#3,#4] is defined, which loads the descriptions of the coloured screens into the document. Then the complete picture is positioned within the text by using the command

\cmykpic[pic_c,pic_m,pic_y,pic_b]

A framed picture can be typeset by using the \fcmykpic command. Both commands expect in pic_n the names of the pictures, which are defined by the -f option in each of the four runs of BM2FONT.

The macros are written under the assumption, that a transparent imaging model is used, rather than an opaque imaging model. The opaque model replaces within the desired area on the page any colour, which was printed before, while the transparent model overprints the already coloured area. So it is possible to get mixed colours simply by overlaying the four planes of the picture in order to generate the final composite image. The original TEX does not know anything about imaging models, because it only expects black ink on white paper. But using colour for typesetting text as well as graphics requires the choice of an imaging model with respect to driver programs. Later on this problem will be discussed in connection with future developments.

The complete process for creating and typesetting a colour picture could look like the following, starting with the colour separation

```
bm2font picture.gif -jn -fpicb -kk
bm2font picture.gif -jn -fpicc -kc
bm2font picture.gif -jn -fpicm -km
bm2font picture.gif -jn -fpicy -ky
```

Inside the TEX document one should add the following commands

```
\loadcmykpic[picc,picm,picy,picb]
\fcmykpic[picc,picm,picy,picb]
```

It is important to mention that the colour macros used for typesetting colour pictures are the same that are used for typesetting colour text, when they are expanded by TEX. This means for the usage of a dvidjc driver, that we get the mixed colour of green in areas where two overlaying components of text in the primary colours cyan and yellow have coloured pixels at the same position.

The dvidjc-drivers

The old dvidot driver program, developed by Wolfgang R. Müller at the Computing Centre of the Heinrich-Heine-University, had to be extended by code to interpret the literals of the \special commands, which are written into the dvi file when using the macros of the colordjc style.

```
if (memcmp(comment,"color", 5)){
  fprintf(prot,"special »%s« ignored",comment);
  return;
  }
parm = comment+6;
if (!memcmp(parm, "pop", 3)){
```

```
  parm +=4;  colst-;
  if (colst < 0) {
    colst = 0;
    fprintf(prot," color stack underflow");
    }
  }
else {
if (!memcmp(parm, "push cmyk", 9)){
  parm +=10;  colst++;
  if (colst > smax-1) {
    colst = smax-1;
    fprintf(prot," color stack overflow");
    }
  }
if (sscanf(parm,"%lg%lg%lg%lg",
                &cy,&ma,&ye,&bl)<4){
  fprintf(prot," color: cmyk values <%s>
                incomplete ",comment);
  return;
  }
```

This code handles the most simple case, where some text has to be printed in a given mixed colour, which is made of CMYK intensities. Those intensities are defined by the `\newColor` macro. So the driver program knows how to handle the operation codes, enclosed by `color push cmyk` and `color pop`.

Colour production. Producing the desired mixed colour by the `dvidjc` driver is relevant only for printing coloured text, not for colour pictures. The screens made by BM2FONT already have the cyan. magenta, yellow or black pixels at those positions, where for example a blue sky or a red nose should be shown. Producing mixed colours for text is a task which has to be done by the program.

This is implemented by defining a 4 times 4 grid for each primary colour, which contains in certain positions a threshold value, that indicates whether a pixel is to be coloured or left white. This `maskmat` table is used to generate the bitmap for the current primary colour. The colour intensity is transformed to a value between 1 and 16, then a mask is generated with bits turned on in positions where the transformed intensity is less than the threshold value. The colour bitmap is then built up by switching on the bits which are black in both the mask and the originally black bitmap of the page. Color Example 11 shows the result of that process.

Different printers. The available `dvidjc` driver, released in January 1994, supports the colour HP DeskJet printers 500C and 550C and all monochrome printers with PCL support. When compiling the source it depends on a compiler definition, specifying the printer, to be supported by the generated program.

The monochrome version converts coloured areas within the bitmap of a page into different levels of grey. Colour versions distinguish between the additional availability of black ink or the primary colours cyan, magenta and yellow. In the latter case, black — or better a very dark colour — is produced by printing overlaying pixels in primary colours.

Of course good colour reproduction depends on the printer, its resolution, the purity of the ink, and on the kind of paper. All those factors influence some parts of the program:

- the generation of printer control sequences;
- the allocation of memory for colour bitmaps;
- the positioning of the threshold values for intensities of the primary colours.

The necessary code is written in the `hardcopy` procedure or preceded by a `#ifdef DJCOLOUR`, `DJ500C` or `DJ550C`. We mention this to invite *everybody* to extend the `dvidjc.c` source to support printers other than the HP DeskJet.

In the future the information about the CMYK intensity of a defined colour like "Apricot" should be read from a device specific file instead of getting that information by a `\special` command when reading the `dvi` operation codes. So the `dvi` file contains a `color Apricot` and the description file specifies how the mixed colour "Apricot" is realized on that device.

Previewing. The `dvidjc` driver package contains a previewer `dvivgac` for MS-DOS, which supports colour output. The driver starts in halftone mode, showing in the left part of the graphics screen the first page selected in the document, while in the right screen information about the usage of the program is given. When pressing a cursor key a rectangle appears on the left, showing the area of the page, which is magnified on the right.

Color Example 12 shows the hardcopy of a screen corresponding to the titlepage of the `dvidjc` documentation.

The next step

One of the next jobs will be to make the colour driver more generic as mentioned before and to distinguish between different imaging models. The authors of the package can not do this on their own. The result of further efforts will be a summary of the work of other people with access to different colour printers, with programming experience and some enthusiasm for TEX.

The most important job for the future can not be done only by hacking code. Writing a colour

driver requires a complete recommendation how to include graphics, both monochrome and coloured, into a TEX document. In spite of the fact that the described method works, a standard is necessary, because there are some problems left.

Problem with PostScript. The PostScript colour model works under the default assumption that a coloured region on the page is printed after the colour in an overlapping part of another region is removed. This *knockout* mode is unknown to the HP DeskJet printers, and was the reason why the first attempt to check the compatibility between the `dvidjc` drivers and the combination of `dvips` and `GhostScript` was discouraging: the pictures looked ugly, cyan text within a yellow box was not green. But reading the PostScript manual helped.

Simulation of knockout mode can be done by TEX, to achieve compatibility. If, for example, a yellow coloured text is to be printed inside a cyan box, then the text simply has to be positioned by using the colour `\White` before using the yellow colour. This was the cheapest method for the `dvidjc` drivers to simulate an opaque imaging model.

This example should also make it clear that overprint mode is not useful when using colours with equal intensities of primary colours.

Anyway it is advisable not to mix the ink of a region containing text and the ink of a background area of the text. Overprinting that region would not produce the desired colour for the text. But what is correct for text is not necessarily correct for pictures. Here we need mixed colours, when the screens for the primary colours are overprinted.

A solution for that problem could be a `\special`, that tells the driver to switch into overprint mode or back into knockout mode. A probably better way could be to design a graphics interface, that enables driver programs to get information about the kind of picture included. Depending on the characteristics of the included graphic, the driver could switch to the appropriate mode automatically.

Problem with BM2FONT. The main disadvantage of BM2FONT is the usage of TEX fonts to include graphics into a document. This is the reason why the number of pictures printed in a document is limited. When printing colour pictures, up to four times more fonts than for a monochrome picture are generated.

The conversion of graphics into an easy to handle rectangular box consisting of characters, is an advantage compared to existing DVI drivers. But this method makes it impossible for colour supporting drivers to distinguish between text and graphics. A solution by marking that part of a page with enclosing `\special` commands would be contrary to the aim of BM2FONT to print included pictures with any device driver.

It looks like the expansion to colour support signals the beginning of the end of BM2FONT. We are confident that the long standing demand of a graphics standard and the problems connected with global colour support in TEX will lead to a solution that is similar to the one adopted by the driver family of the emTEX package and by `dvips`. External files will contain the graphics, and the typographic information like metric and colour will be derived from a description file.

Colour Pictures

Color Example 9 shows the locations of European home pages in the World Wide Web (WWW). The picture comes from `//s700.uminho.pt/europa.html`.

Color Example 10 is a picture of the campus of the University of Dortmund, Germany, while Color Example 11 shows the same picture separated in its four colour components cyan, magenta, yellow and black.

Color Example 12 is a screen dump of the `dvivgac` previewer in colour mode. The original picture was scanned from page 304, *The TEXbook*. Colours were added with the `Xpaint` program.

Bibliography

Clark, Adrian F. *Practical Halftoning with TEX*, *TUGboat* **12** (1), pages 157–165, 1991

Hilgefort, Ulrich, *Farbe aufs Papier*, c't **4**, pages 132–139, 1994.