## Write-White Printing Engines and Tuning Fonts with METAFONT

Neenie Billawala

The wide variety of digital printers with different print characteristics presents the fact that the same font data will likely produce just as many results. There are differences in resolution, printer type (laser, photo-digital, dot matrix, screen), and in the characteristics of the marking device. Even within the range of same resolution and type of printer, e.g., 300 dpi laser printer, variations occur.

A most noticeable one occurs between those of the "write-white" and "write-black" variety.

The problem in getting a like result with write-white (ww) and write-black (wb) machines from the same pixel pattern lies in the fact that the effective size and/or shape of each pixel differs; the size is typically smaller in ww machines.

Theoretically, one pixel on a 300dpi printer has a width of 1/300 of an inch. On a ww machine, it's typically less; on a wb machine, it's usually a bit more. The gradations of thickness may look something like the following hypothetical data, actual values will vary.

| number of pixels | "ideal width" | write-white width | write-black width |
|---|---|---|---|
| 1 | 1 | .8 | 1.2 |
| 2 | 2 | 1.8 | 2.2 |
| 3 | 3 | 2.8 | 3.2 |
| 4 | 4 | 3.8 | 4.2 |
| 10 | 10 | 9.8 | 10.2 |

(width values are in 1/300 of an inch)

As you can see, the greater the number of pixels, the less important the .4 pixel difference becomes. The greatest discrepancies and problems occur with lines one pixel wide, where the percentage of difference is the greatest.

>> Try the following METAFONT example on different printers. Make an image which alternates a single pixel black vertical line with a single pixel "empty" or white vertical line and print the character on your machine. Do the same with alternating single pixel horizontal lines. Notice the shades of gray, the relative darknesses and the crispness of line. Try this on a write-white printer and then again on a write-black printer. There will probably be a marked difference. If you are more ambitious, repeat the same example with different thicknesses for the black lines and/or white lines. Change the length of your test lines; overlap the horizontal and vertical tests ... Notice also the corners and edges of the lines. You may have to limit the size of your test characters to keep METAFONT from running out of memory. <<

Larger pixel representations of characters are often shown as precise patterns with a clean square representing each pixel. Upon close inspection of an individual pixel, however, you see no precise square, but rather a globular figure, or a starburst pattern, the center being darker with the pattern becoming lighter away from the center. Individual pixels may vary in shape within a range.

The square represents a hypothetical pixel (fig 1a). The circles are the areas where blackness, e.g., toner, may be found. If the outermost edges of the circle fall totally inside the square, the pixel is light, and a series of these in a row will form an unconnected or broken line (fig 1b). On the other hand, if the square falls completely within the circle, the pixels will be relatively dark, and a line of them will be darker than the target pixel size due to overlap (fig 1c). The target pixel size is one that will produce a target line thickness, e.g., 1/300 of an inch, when several of these pixels are lined up. The "ideal" solution lies somewhere in between these two, similar to fig 1d.

Figure 2 demonstrates a simplification of the idea between "write-white" and "write-black" printers. With a wb printer, the circles are blackened, in a ww printer, the circles are white and the areas between them are dark.

At some point, digital font data contains information about which pixels or bits are black and which are white. If the same font data or pixel pattern is produced with printers, each having different output characteristics, the results will vary.

If the line thicknesses or other characteristics of two printers are not identical it will be impossible to produce exactly the same output given the same font data. However, the same font can often be made recognizable and comparable between printers.

The style of a font dictates how well it will hold up, how robust or how fragile it will be. Typically fonts with very thin lines, such as the "modern" fonts, tend to fall apart with ww printers. Many of the Computer Modern fonts fall into this category.

METAFONT offers the capability of changing parameters, modifying a font. Naturally, only those parameters which are included in the design of the font can be changed. It is important to leave the vertical and horizontal width information found in the "tfm" file intact, but it is possible to make slight alterations to characters to get back lines that drop out or fade due to a one pixel thickness.

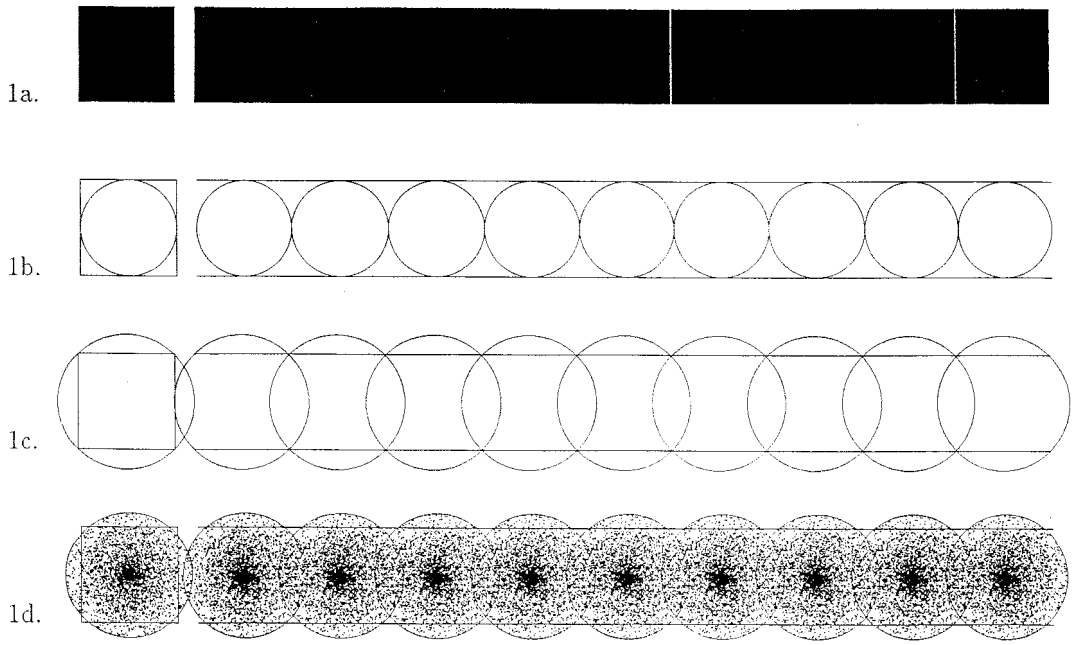



Figure 1. Area occupied by adjoining pixels

Editor's note: The small gaps in the "solid" bar of fig 1a appear to be an artifact of roundoff error; roundoff problems can sometimes be noticed in text output from laser printers, particularly when strings of monospaced type are interspersed with roman text, since the likelihood is small that the width of a monospaced letter is an integral number of pixels at all design sizes and magnifications.

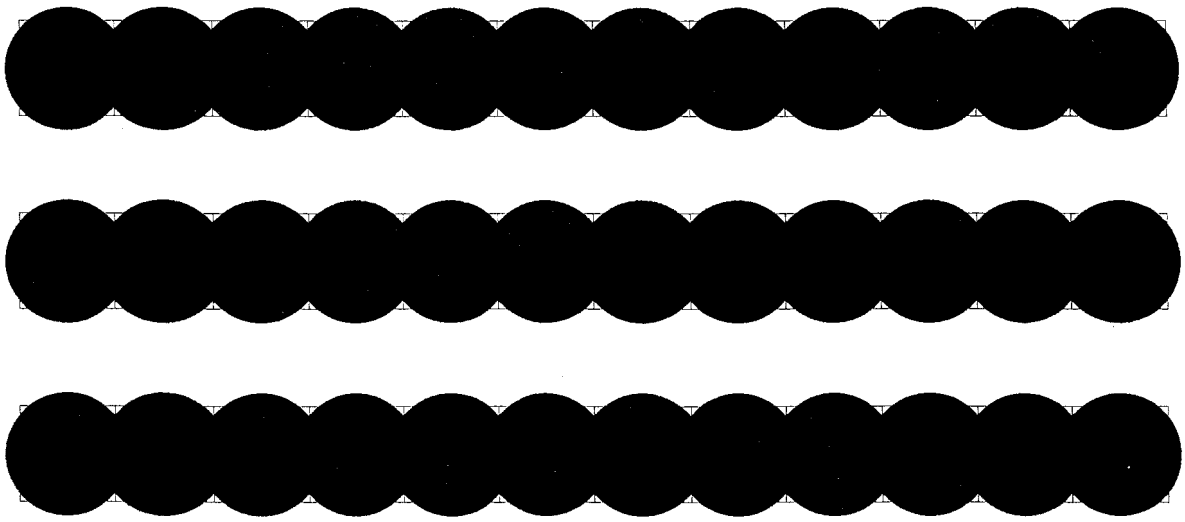


Figure 2. Alternating lines of written and unwritten pixels

At this point, one should consider a few things about the font you are trying to modify. Remember the printer characteristics, and the fact that two very different printers will give dissimilar results. Within this restriction, one generally looks for a representation that is recognizable and close to the "original" or "source" form. You should also consider

- For what purpose is the font being used? If it is for lower resolution previewing proofs for a higher resolution device, then it is essential to maintain "tfm" dimensions.
- How closely does it match its source?
- Is it important that it remain as true as possible to its source? Or is the darkness/lightness of the font more important?
- If darkness only is a consideration, think about changing to a darker font.
- How much effort do you want to put into the modification?

A while back, someone came by with the problem that the Computer Modern Roman fonts were falling apart on his 300 dpi write-white printer. Thin parts of the characters were so thin as to make unwanted breaks in the letters. Parts of characters where arches joined stems, as in the lowercase "n" and "u", disappeared. The overall page was quite light and a photocopy of it gave a result that was very difficult to read. As the pages of output were often to be photocopied, this was an important consideration.

We tried the following in order to improve the fonts:

(1) Some higher settings of "*blacker*" and changes to "*fillin*".
(2) Selectively adding "*blacker*" to some values.
(3) Setting a minimum line thickness, 2 pixels in this case.

The Computer Modern fonts use an amount called "*blacker*", which adds to or subtracts from certain line and pen thicknesses. Increasing the amount of "*blacker*" has the effect of adding an absolute pixel amount to certain values, e.g., *blacker* := .75 would always add .75 pixels to the stem, no matter the resolution. If a line has a value of 4.1 pixels before rounding, adding .75 would make that value 4.85 and round to 5 pixels. But if "*blacker*" were .25, then this line value would be 4.35 and round to 4; in this case "*blacker*" has no apparent effect on this value. Setting the value of blacker to 0 means that no adjustment is made. The value of "*blacker*" is given in the "*mode*" setting.

"*Fillin*" is used to compensate for extra heaviness that seems to appear in diagonal lines. A positive value means that the diagonal line will be thinner, a negative value would add thickness to that line.

Both (1) and (2) often created unwanted character distortions since the "tfm" widths wanted to be the same. When each line in a character was increased by one, the shapes were often distorted; counter (inside) shapes seemed to suffer the most. A first attempt with (3) showed promise, and though the result was not as dark as with the wb machine, the feeling of Computer Modern was retained.

In the first attempt, we did not manage to change all of the appropriate thicknesses to 2 pixels. Don Knuth then made the correction shown in figure 3, which keeps minimum thicknesses to 2 pixels. Looking at the pixel patterns, this fix appears to help quite a bit.

```
1) CMBASE.MF without the modifications
2) CMBASE.MF with the two modifications

1) min_Vround:=max(fine.breadth,crisp.breadth,tiny.breadth);
2) min_Vround:=max(fine.breadth,crisp.breadth,tiny.breadth,2);%"WRITE WHITE"

    >> The addition of the value 2

1) enddef;
2) forsuffixes $=thin_join,hair,curve,flare,dot_size,cap_hair,cap_curve,
2) vair,bar,slab,cap_bar,cap_band,stem',cap_stem',vair',fudged.hair,
2) fudged.stem,fudged.cap_stem: $:=max($,2); endfor  % "WRITE WHITE" ONLY!
2) enddef;

    >> The addition of these three lines before the "enddef"
```

Figure 3. Additions to *font_setup* in `cmbase.mf`

>> Add the two changes shown in figure 3 to the "*font_setup*" macro located in your `cmbase.mf` file. You may want to keep the old version around until you are sure that you have put in the fix. And it is a good idea to make a note of the changes made and why for future reference. Then run a test, using your original *mode_setup* specifications. If the test result looks unchanged, you may not actually have the fix in. It is useful to get a pixel representation to verify this.

>> To go about looking for the "right" settings, add the minimum 2-pixel fix; this number may want to be larger with a higher resolution printer. Then run some systematic tests with new "*mode*" definitions, typically changing the amount of "*blacker*" and the amount of "*fillin*" until you find an appropriate setting. It may be different for each printer. These modifications maintain "tfm" dimensions in the Computer Modern fonts. <<

"*Mode*" information tells a few specifics about each printer and you can make a new mode to suit your printer. The existing modes typically have information about the resolution, an "*o_correction*" for the amount a curve extends past a vertical limit, a value for "*blacker*" and a value for "*fillin*". There is also information about what will show up on the screen when running the METAFONT and such, but we'll only concern ourselves with the resolution, "*blacker*", and "*fillin*".

A typical mode definition, as found in the `waits.mf` file, looks like this:

```
% imagen mode: for the Imagen 8/300 (Canon engine)
mode_def imagen =
 proofing:=0; % no, we're not making proofs
 fontmaking:=1; % yes, we are making a font
 tracingtitles:=0; % don't show titles in the log
 pixels_per_inch:=300;
 blacker:=0; % Canon engine is black enough
 fillin:=.2; % and it tends to fill in diagonals
 o_correction:=.6; %
 enddef;
```

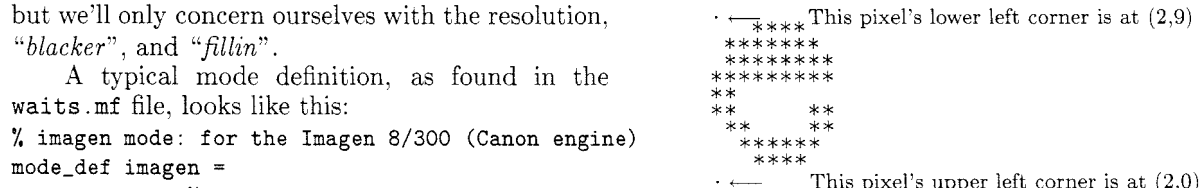
The Computer Modern fonts weren't tested on all possible print devices; you can imagine why. Some of the modes listed have conjectural values, as it wasn't possible to test extensively on these printers. When conjectural values are given, or a new machine is to be added, trying the following test method to help establish suitable settings for your printer.

(1) Check to see if a mode already exists with the same print engine or same or similar print characteristics; if so, use that mode.
(2) Look for a mode which has the same resolution as your printer. If no such mode exists, then make a mode following an existing pattern, changing the resolution to fit your printer.
(3) Set the value of "*blacker*" to 0.
(4) Set the value of "*fillin*" to 0.

Run a test with these settings; this will be your control. Then vary values of "*blacker*" and "*fillin*" systematically. For example, run 9 samples, setting "*blacker*" to 0, .5, and 1 while setting the values of "*fillin*" to $-.5$, 0, and .5. You will begin to see areas that are more successful than others. Look at the values in those ranges more closely. The next set may look like this: "*blacker*" $= 0, .2, .4$ and "*fillin*" $= -.5, 0$. Try this on a variety of the fonts you want to use on your printer to find an appropriate setting.

There are caveats to note though. The shapes may suffer, especially in the case of smaller or lower resolution fonts. In the case of a small lowercase "e", if the distance between the top of the curve of the "eye" and the bottom of the bar is 4 pixels or less, the "eye" will fill in and you will get a dark spot. Sometimes shapes produced with a single-pixel pen look better than those produced with the "minimum thickness" pen.

Here is "e" from CMR5 at a resolution of 300 dpi. Pixel locations are indicated in METAFONT coordinates.




(This example was sent by J. Sauter.)

There always was a question as to how well these fonts would work at low resolution, where often special compensations are made. In such low resolution cases, where the characters don't work well, it may be more economical to edit the character shapes directly. With some to a significant amount of extra work, you can also alter the source code to modify the fonts.

The previous suggestions are based on experiments that have already shown a degree of success within the limitations of the medium. I encourage everyone to experiment with these methods, to contemplate other possible solutions to this problem and to share their results.