Procedures for font comparison

Karel Píška
Institute of Physics of the ASCR, v. v. i.
CZ-182 21 Prague, Czech Republic
piska (at) fzu dot cz

Abstract

This contribution presents several programs: Linux standalone scripts for comparison and visualization of font elements. Simplified versions of the programs are restricted to use with the fonts already installed in a TEX system and show, for example, how to solve the following tasks:

- Comparison of two bitmapped or outline representations of a glyph pair at two different resolutions or from two related TEX fonts using a color mix technique in pdfTEX.
- Comparison of kerning pairs in two (or three) related TEX fonts or in two releases of one font.
- Comparison of a glyph pair in two versions or in two outline fonts, analyzing sequences of cubic curve segments in font programs.

The programs will be demonstrated by several examples.

1 Common characteristics

The purpose of the `tfcpr` package, containing procedures for TEX **f**ont **c**om**p**a**r**ison, is to provide tools for checking and comparison of the fonts used in a TEX environment. This paper is a complementary work to the article "Font verification and comparison in examples" [9]. The programs are written in the form of Linux scripts, written in the `bash` shell language. After download they are unzipped into a local working directory. The simplest method is to start the programs locally with the '`./`' prefix. They invoke various programs from Linux or its TEX subsystem, such as `gawk` [6], METAFONT, LATEX or pdfLATEX, `dvips`, etc., and Acrobat Reader is usually called in the final step. They read the font files: metrics (`tfm`), METAFONT sources, Type 1 (`pfb`) or OpenType (`otf`); generate intermediate font bitmaps (`pk`) and PostScript; and produce PDF documents. The `kpathsea` library is used to find font files. Uninstalled fonts, including unused versions or releases, should be copied for testing purposes also into the current local directory. Additionally, we have to change their file names to avoid name collisions if it is necessary to distinguish them from fonts already existing in our system. FontForge [8] and `t1utils` [7] are also used for preprocessing of data from outline fonts.

The real testing software is under permanent revision depending on a font family being processed at the moment, since it must often be modified for various encoding schemes and naming conventions; usually all glyphs and all character pairs for a given encoding should be processed. It has no general or common user interface and it would be too difficult and expensive to transform it to a transparent and computer independent form. The programs presented here are simplified excerpts restricted to processing of a single glyph or a character pair and extended with a user interface to call them via commands and produce the visual output in PDF. They can be used quickly. On the other hand, they do not work efficiently. In most examples, fonts from the Computer Modern [2, 3], CS fonts [4], and the Latin Modern [5] families are presented.

2 Glyph comparison for bitmapped fonts

The aim in comparing a METAFONT font and its Type 1 version at various resolutions is to check the correctness of the font design in METAFONT and its proper conversion into an outline font format.

We can compare glyph images for one METAFONT font at two different resolutions or the bitmapped representation with the Type 1 version of the font converted from its METAFONT sources.

The execution of the command

```
./cprpk cmr10 97 1200 2400
```

starts by generating bitmaps with METAFONT. The four parameters define the font name, the decimal (or octal, for example `\'141`) character code and

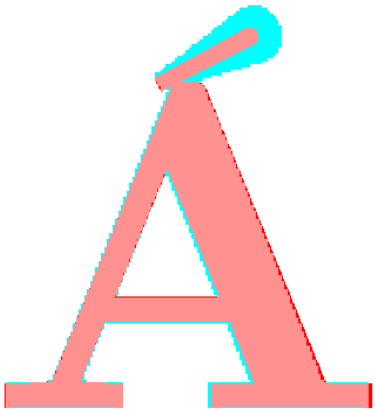

Figure 1: `./cprpk csbx10 193 1200 5333`

Figure 2: `./cprpk cmr10 97 1200 2400`

two resolutions. Then we produce two glyph instances for both resolutions distinguished by postfix "`a`" or "`b`" with pdfLaTeX — see the example document in Fig. 6. After that, again with pdfLaTeX we mix both components into one picture (Fig. 7). All these LaTeX sources have been generated automatically and dynamically according to the command. The final results are shown in Figure 1: the bitmapped `csbx10` "Á" at 1200 dpi and 5333 dpi; and in Fig. 2: the Computer Modern Roman `cmr10` "a" at 1200 dpi and 2400 dpi. The first resolution glyph image is filled with cyan (light), the second one or Type 1 is filled (stroked) with red (dark), and the common area is in pink (looks lightest). (For the printed issue, the figures have been manually converted to grayscale.) The Type 1 counterpart can substitute one bitmap (Fig. 3) or may be expressed in the contour mode "`1 Tr`" — see figures 4 and 5. The figure captions contain the commands generating the corresponding pictures.

Figure 3: `./cprpkt1 cmr10 97 1200`

Figure 4: `./cprpkt1c cmr10 97 1200`

Figure 5: `./cprt1cpk cmmib5 55 2400`

```
\documentclass{article}\pagestyle{empty}\usepackage{graphicx}
\font\fa=cmr10a
\begin{document}\scalebox{40}{{\fa\char97}}\end{document}

\documentclass{article}\pagestyle{empty}\usepackage{graphicx}
\font\fb=cmr10b
\begin{document}\scalebox{40}{{\fb\char97}}\end{document}
```

Figure 6: Generating of glyph images for comparison.

```
\documentclass{article}
\usepackage{graphicx}
\def\Default{\pdfliteral{0 g 0 G}}
\pdfpageresources
{/ExtGState
 << /Luminosity << /Type /ExtGState /BM /Luminosity >> >>}
\def\Acolor{0 1 1 rg 0 1 1 RG}% cyan
\def\Bcolor{1 0 0 rg}% red
\begin{document}
\setlength{\fboxsep}{0pt}\setlength{\fboxrule}{0pt}
\begin{figure}\begin{center}
\makebox[0pt][l]{\pdfliteral{/Luminosity gs \Bcolor}%
\includegraphics[viewport=90 310 470 710]{cmr10_97-2400.pdf}}%
{{\pdfliteral{\Acolor}%
\includegraphics[viewport=90 310 470 710]{cmr10_97-1200.pdf}}}
\Default\label{cmr10_97}
\caption{cmr10: 97 {\pdfliteral{\Acolor} 1200pk}
{\pdfliteral{\Bcolor} 2400pk}}
\Default
\end{center}\end{figure}
\end{document}
```

Figure 7: A color mix of two glyph instances.

```
\documentclass{article}
\newlength{\bbox}\newlength{\cbox}%
\def\fboxsep{0pt}\def\fboxrule{0.1pt}
\usepackage{ifthen}
\def\krule#1{%
\ifthenelse{\lengthtest{#1>0pt}}{\rule{10#1}{1ex}{$>$}}{}%
\ifthenelse{\lengthtest{#1<0pt}}{{$<$}\rule{-10#1}{1ex}}{}}
\def\paira#1#2#3{{%
\font\fna=#1
\fna\settowidth{\bbox}{#2#3}%
\settowidth{\cbox}{\mbox{#2}\mbox{#3}}%
\addtolength{\bbox}{-\cbox}%
\fbox{#2}\kern-0.2pt\kern\bbox\fbox{#3}\fbox{#2#3}
#1 \krule{\bbox}
\\}}
\begin{document}
\noindent
\paira{cmr10}{k}{a}
\paira{csr10}{k}{a}
\paira{ec-lmr10}{k}{a}
\end{document}
```

Figure 8: Generating a test for three kerning pairs.

```
./prfkrna k a cmr10 csr10 ec-lmr10
./prfkrna K O cmr10 csr10 ec-lmr10
./prfkrna v a cmr10 csr10 ec-lmr10
./prfkrna A c cmr10 csr10 ec-lmr10
./prfkrna P , cmr10 csr10 ec-lmr10
./prfkrna T . cmr10 csr10 ec-lmr10
./prfkrna f ! cmr10 csr10 ec-lmr10
./cprkrna A C cmti10  ec-lmri10
./cprkrna i i cmcsc10 ec-lmcsc10
./cprkrna I I cmcsc10 ec-lmcsc10
```

Figure 9: Sample commands for kerning tests.

3 Comparison of kerning pairs

Our next aim is to verify compatibility between two or more related TEX fonts. Kerning data are taken from the corresponding metric files (`tfm`). The first example (Fig. 8), invoked by the command
`./prfkrna k a cmr10 csr10 ec-lmr10`
compares kerning of character pairs in three related fonts, extracting the first lines of the final output (Fig. 10). Figure 9 shows a sample command list.

The LATEX source in Fig. 11, generated by the command line
`./cprkrna i i cmcsc10 ec-lmcsc10`,
is connected with the last lines in figures 9 and 10 and, unlike the previous examples, we really do compare the kern values between two instances of the character pairs. The equal sign "=" in the CMCSC10 line (second from the bottom in Fig. 10) signals the identical kerns, while the rule in the EC-LMCSC10 line (bottom line) denotes the common kern value in both fonts. The varying widths of the rules relates to the different kerns and their relative ratio.

The table in Fig. 12 summarizes several selected differences in the kerning pairs for three font families expressed in the kern degree units (e.g., "`k#`").

For example, for Latin Modern in the T1/EC encoded metrics we have:

- kerning pairs absent in LM: `"k":"a"` and `"K":"O"`;
- LM compatible with CM: `"v":"a"` and `"P","T":".",",";`
- a new kerning pair in LM: `"A":"c"`;
- a kern doubled in LM: `"A":"C"` in italics;
- other changes in LM: `"i":"i"` in small caps or `"f":"!"`.

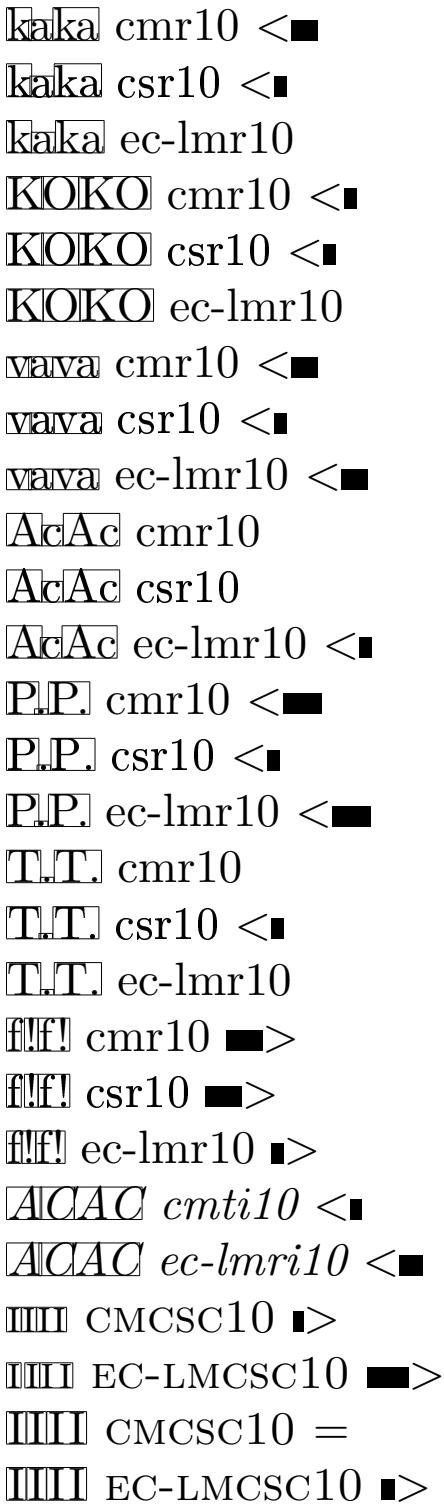

Figure 10: Several tests of kerning pairs.

```
\documentclass{article}\font\fna=cmcsc10\font\fnb=ec-lmcsc10
\newlength{\bbox}\newlength{\cbox}%
\newlength{\bxa}\newlength{\bxb}%
\def\fboxsep{0pt}\def\fboxrule{0.1pt}
\usepackage{ifthen}
\def\krule#1{%
\ifthenelse{\lengthtest{#1>0pt}}{\rule{10#1}{1ex}{$>$}}{}%
\ifthenelse{\lengthtest{#1<0pt}}{{$<$}\rule{-10#1}{1ex}}{}}
\def\pair#1#2#3#4{%
\fna\settowidth{\bxa}{#1#2}%
\settowidth{\cbox}{\mbox{#1}\mbox{#2}}%
\addtolength{\bxa}{-\cbox}%
\fnb\settowidth{\bxb}{#3#4}%
\settowidth{\cbox}{\mbox{#3}\mbox{#4}}%
\addtolength{\bxb}{-\cbox}%
\ifthenelse{\lengthtest{\bxa = \bxb}}%
{{\fna\fbox{#1}\kern-0.2pt\kern\bxa\fbox{#2}\fbox{#1#2}
cmcsc10 {$=$}
}\\
{\fnb\fbox{#3}\kern-0.2pt\kern\bxb\fbox{#4}\fbox{#3#4}
ec-lmcsc10 \krule{\bxb}
\\}}
{{\fna\fbox{#1}\kern-0.2pt\kern\bxa\fbox{#2}\fbox{#1#2}
cmcsc10 \krule{\bxa}
}\\
{\fnb\fbox{#3}\kern-0.2pt\kern\bxb\fbox{#4}\fbox{#3#4}
ec-lmcsc10 \krule{\bxb}
\\}}}
\begin{document}
\noindent
\pair{i}{i}{i}{i}
\end{document}
```

Figure 11: Comparison of two instances of a kerning pair.

```
                cmr10    csr10    ec-lmr10
"k":"a" kern    2k=-u    k        0        CM<CS<EC=0
"K":"O" kern    k        k        0        CM<CS<EC=0
"v":"a" kern    2k=-u    k        2k=-u    EC=CM<CS=k
"A":"c" kern    0        0        k        EC<CM=CS=0
"P":"." kern    3k=kk    k        3k=kk    EC=CM<CS=k
"P":"," kern    3k=kk    k        3k=kk    EC=CM<CS=k
"T":"." kern    0        k        0        CS<CM=EC=0
"T":"," kern    0        k        0        CS<CM=EC=0
"f":"!" kern    -2.8k    -2.8k    -k       0<EC<CM=CS
                cmti10   csti10   ec-lmri10
"A":"C" kern    k        k        2k=-u    EC<CM=CS=k
                cmcsc10  cscsc10  ec-lmcsc10
"i":"i" kern    -0.76k   -0.76k   -3.05k   0<CS=CM<EC
```

Figure 12: Examples of differences between kerns.

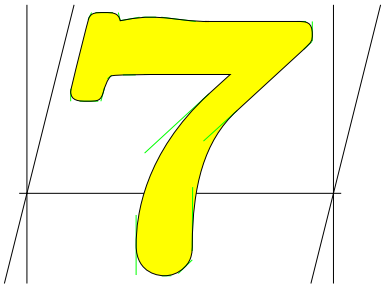

Figure 13: `./prfof cmmib5 sevenoldstyle`

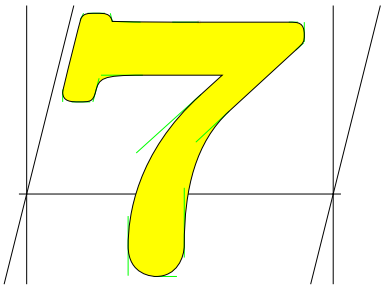

Figure 14: `./prfof lmmib5 seven.taboldstyle`

4 Proofs and comparison of outline fonts

The next tools are `prfof`, which produces proofsheets for single glyphs, and `cprof`, which compares glyphs from Type 1 or OpenType fonts. The font editor FontForge [8] is used to parse fonts, that is, for unpacking, extraction and export of a partial font and glyph data. Type 1 fonts are also processed with the `t1utils` package [7].

The examples in figures 13 and 14 demonstrate the "old style seven" in Computer Modern `cmmib5` and in Latin Modern `lmmib5` and its comparison in both representations (Fig. 15).

We must note that the Type 1 version converted by Blue Sky and Y&Y is older than the latest modifications of the CM sources. Please compare also with Fig. 5 where the METAFONT definitions have been corrected while the Type 1 outline curves are still wrong.

4.1 Comparison of different releases of the same font

An older font version can be unzipped and copied into the current directory; its file names must be renamed before starting comparison with the actual production font release in the installed TDS tree. Here is a sample approach:

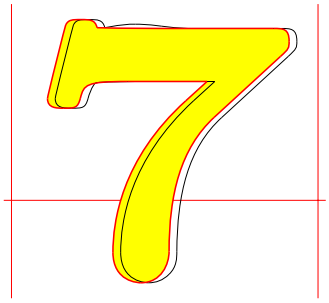

Figure 15: `./cprof cmmib5 lmmib5 sevenoldstyle seven.taboldstyle`

```
LM1="lm1.00bas.zip"
PFB="fonts/type1/public/lm"
unzip -j $LM1 $PFB/lmri10.pfb
mv lmri10.pfb lmri10a.pfb
./prfof lmri10a perthousand
./prfof lmri10 perthousand
./cprof lmri10a lmri10 perthousand
```

4.2 Change list

A helpful idea is to assemble a change list: a list of the font and glyph names with known bugs or changes, and run it with a previous version and then with the current font release to confirm the changes have been done and the bugs have been fixed. An example of such a list is presented in Fig. 16.

```
lmri10 perthousand
lmri10 permyriad
lmu10 perthousand
lmu10 permyriad
lmu10 degree
lmdunh10 uring
lmduno10 uring
lmmib5 seven.taboldstyle
lmmib7 seven.taboldstyle
lmmi5 seven.taboldstyle
lmtcsc10 F
lmtcsc10 I
lmtcsc10 Iacute
lmtcso10 F
lmtcso10 I
lmtcso10 Iacute
```

Figure 16: A change list for LM.

5 Availability

The `tfcpr` package can be downloaded from `http://www-hep.fzu.cz/~piska/tfcpr.html`. The programs can be distributed as "public domain software", may be freely used (without warranty), corrected, modified, adapted or included in other packages.

References

[1] Donald E. Knuth. *The METAFONTbook.* Addison-Wesley, 1986. Volume C of *Computers and Typesetting.*

[2] Donald E. Knuth. *Computer Modern Typefaces.* Addison-Wesley, 1986. Volume E of *Computers and Typesetting.*

[3] Computer Modern fonts. `CTAN:/fonts/cm`.

[4] CS fonts. `ftp://math.feld.cvut.cz/pub/cstex/base/csfonts.tar.gz`.

[5] Latin Modern fonts. `CTAN:/fonts/lm`.

[6] Free Software Foundation. GNU awk, `http://www.gnu.org/software/gawk`.

[7] Eddie Kohler. `t1utils` (Type 1 tools), `http://freshmeat.net/projects/t1utils`.

[8] George Williams. Font creation with FontForge. *EuroTEX 2003 Proceedings*, *TUGboat*, 24(3):531–544, 2003; `http://fontforge.sourceforge.net`.

[9] Karel Píška. Font verification and comparison in examples, *EuroTEX 2006 Proceedings*, *TUGboat* 27(1):71–75, 2006.

A Appendix: tfcpr: Synopsis and examples

A.1 Glyph comparison for bitmapped fonts

```
cprpk font code res1 res2
cprpk cmr10 97 1200 2400
cprpk cmr10 \'141 1200 2400

cprpkt1 font code res
cprpkt1 cmr10 97 1200
cprpkt1 cmr10 \'141 1200

cprpkt1c font code res
cprpkt1c cmr10 97 1200
cprpkt1c cmr10 \'141 1200

cprt1cpk font code res
cprt1cpk cmr10 97 1200
cprt1cpk cmr10 \'141 1200

cprpkpk font1 font2 code1 code2 res1 res2
```

font a TEX font name, common to `.mf`, `.tfm`, `pk`
code may be decimal or octal (starts with `\'`)
Predefined resolutions: 300 600 1200 2400 2602 5333

A.2 Comparison of kerning pairs

```
prfkrn font code1 code2 [font code3 code4]...
prfkrn cmr10 65 99 ec-lmr10 65 99
prfkrn cmr10 \'101 \'143 ec-lmr10 \'101 \'143

prfkrna char1 char2 font [font]...
prfkrna A c cmr10 ec-lmr10

cprkrn font1 code1a code1b font2 code2a code2b
cprkrn cmr10 65 97 ec-lmr10 65 97
cprkrn cmr10 \'101 \'143 ec-lmr10 \'101 \'143

cprkrna char_a char_b font1 font2
cprkrna A c cmr10 ec-lmr10
```

font a TEX font name, i.e., `.tfm`
char an ASCII symbol recognized by TEX

A.3 Proofing and comparison for outline fonts

```
prfof font glyphname
prfof cmmib5 sevenoldstyle
cprof font1 font2 glyphname1 [glyphname2]
cprof cmmib5 lmmib5 sevenoldstyle \
                    seven.taboldstyle
```

font is *name*`[.pfb]` or *name*`.otf`, i.e.,
only `.pfb` may be omitted
glyphname is PostScript or OTF, according to context