Conversion of TEX fonts into Type 1 format

PÉTER SZABÓ[*]

ABSTRACT. This paper analyses the problem of converting TEX fonts to Type 1 fonts, describes TEXTRACE, a new free conversion program, and compares it to other possible methods and existing utilities. TEXTRACE works by rendering the font in high resolution and then tracing (vectorizing) it.

KEYWORDS: PDF, font conversion, Type1, METAFONT, vector, outline, raster, bitmap, pdfTEX

THERE ARE two major ways to describe the shape of the glyphs[1] of a font: using *bitmap images* or *vector outlines.* A bitmap image is a rectangular array of dots; each dot can be painted white or black independently from the others. At a high resolution of around 600 DPIthe human eye cannot distinguish individual pixels and sees the glyph as if it was continuous. On the other hand, vector outlines specify the black regions by the curves (lines, arcs, ellipses and others) which delimit them. The most important advantage of this representation is that glyph outlines can be transformed (rotated, scaled etc.) without quality loss.

Most fonts used in today's desktop publishing systems are distributed in some kind of vector outline format. (Such formats are: Type 1, TrueType, OpenType, Speedo, METAFONT source.) The process of converting an outline font to its bitmap equivalent is called *rendering* or *rasterization.* The rendering must surely happen before actually printing the document onto paper or displaying it on screen, because today's printers and screens display everything using a rectangular layout of dots. So the difference between document preparation systems lies not in the presence of the rendering, but the moment when it occurs.

When printing a document onto paper, the exact time instant of the rendering is not important ~ the author wants the final output be beautiful, and all other inner parts of the printing process are of minor importance. However, the case is different when the document is about to be distributed in electronic form. Since readers might use different software for viewing and/or printing, the author should ensure that the document looks and prints nice in most environments, i.e. the document should be

[*]Many thanks to Ferenc Wettl and Gyöngyi Bujdosó

[1] *glyph:* a specific, visible representation of a character

created to be portable. This involves selecting a font format that works well in most reader software, and using this font format exclusively in electronically published versions of the document.

In our specific case, the file format used for document publishing is PDF, and one of the best font formats for embedding is *Type 1,* a traditional vector outline format. (These will be presented later in more detail.) Today's TEX-to-PDF generating software support Type 1 fonts, but unfortunately most TEX fonts are currently not available in Type 1 format. So they have to be converted.

This paper focuses on a quite difficult font conversion problem. The most common source format, METAFONT is rather complicated (in fact, it is an almost full-featured programming language), and the destination format, Type 1 has too many restrictions. Probably this is the reason why there have been no good and robust programs to do the conversion. TEXTRACE, described later in this document, overcomes the first problem by not even trying to understand the METAFONT files, but letting the original METAFONT program render the glyphs, and then calling the AutoTrace program which traces the bitmaps (approximates them with vector outlines). Fortunately AutoTrace's output needs only minor modifications to comply the restrictions of the Type 1 format. The main disadvantage of this "blind" conversion approach is a minor quality loss caused by both the rendering and the tracing process.

The availability of the fonts in a vector outline format is of primary importance when publishing the document as PDF, because Acrobat Reader, the most widely used PDF viewer displays bitmap fonts slowly and ugly on screen.

Motivation

Why PDF?

The PDF file format is very popular nowadays among authors to distribute their work electronically. PDF has become popular because it

- represents letters, spaces, line breaks and images etc. accurately: documents retain their structure and original visual appearance between PDF viewers (as opposed to markup languages, such as SGML and HTML).
- follows the industry standard PostScript page drawing model, PostScript files can be converted to PDF without quality loss (and vice versa).
- is portable and device-independent. Printing PDF files and rendering them on screen is often easier and more straightforward than doing the same for PostScript files. It is possible to create a portable PDF document, but one can never be sure about PostScript portability.
- has widespread freely available viewers (e.g. Acrobat Reader: either stand-alone or in the web browser, Ghostscript, xpdf, PDF e-Book readers) for most platforms. Most systems have some sort of PDF viewer installed.
- has hyperlink, form submit and other web-related capability. (This article ignores these facilities and deals only with static PDF files.)

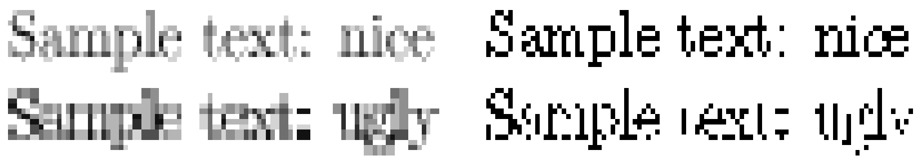

FIGURE 1: THE DIFFERENCE BETWEEN VECTOR (NICE) AND BITMAP FONTS IN ACROBAT READER

- allows relatively small file sizes, which is comparable to DVI files.
- is well suited for both short and long documents, including scientific articles, reports, books and software documentation.

Of course, it is possible to convert TEX output into PDF format either by calling pdfTEX [6], which directly creates PDF files instead of DVI files, or using a DVI-to-PDF or PostScript-to-PDF converter. Preparing a PDF version of a TEX document is somewhat harder than just printing the PostScript version, and requires a bit more time. To enjoy the benefits listed above, the author must use the proper version of the utilities, she must adjust the settings, and she must make sure that the proper versions of the fonts get embedded.

Font problems with PDF and TEX

The main problem with PDF files originating from TEX source is that characters of most TEX fonts show up slowly and look ugly in Acrobat Reader. This is because such fonts are generated by METAFONT, and are embedded into the PDF file as high resolution bitmap images; and Acrobat Reader displays bitmap images slowly and poorly on screen. This has been one of the famous Ugly Bits related to TeX for years. The problem doesn't occur when printing the document, because the resolution of the printer is most often high enough ($\geq$ 300 DPI) to eliminate rendering problems.

The solution to this problem is to embed the offending fonts in such a file format which Acrobat Reader can display quickly and beautifully. A good candidate for this is the classical, standard *Type 1* format, appeared in the late '80s. Fortunately pdfTEX and newer versions of `ps2pdf` (Ghostscript $\geq$ 6.50) fully support it, so the only problem remaining is that how to convert a font in METAFONT format to the Type 1 format. Many techniques have been proposed so far, and none of them is fully satisfactory.

The problem itself can be simply visualized by running `pdftex` on simple `.tex` file (see also Figure 1):

```
Sample text: nice\par \font\f=ecrm1000\f Sample text: ugly\end
```

Both the console output of `pdftex` and the `.log` file should contain the entries `ecrm1000.600pk>` and `<cmr10.pfb>` (the number 600 may vary across installations). These entries are the names of the two font files `pdftex` has embedded into the PDF file. The `pk` extension means that the glyphs of the font have been embedded as bitmap images (the resolution of 600 DPI is also indicated ~ this is of course printer dependent), and the `.pfb` extension means that the glyphs have been included as vector outlines in the Type 1 format.

Differences between METAFONT and Type 1

We'll focus only on how these formats describe the shapes of individual glyphs. The difference between how font-wide metrics, glyph metrics, kerning and ligatures are treated isn't much important, because this information can be converted easily and precisely – and this information will be acquired from the original TEX fonts (the `.tfm` files) anyway, so we don't have to convert them at all.

Taco Hoekwater gives a detailed, richly illustrated comparison of the two font formats, including many figures and a case study of converting some math and symbol fonts (e.g. `logo8`, `wasy5`, `stmary7`, `rsfs10`) in [3]. The curious but beginner reader is referred to that article.

The fundamental difference between the METAFONT and Type 1 is that METAFONT is a programming language, and Type 1 is just a compact file format describing glyph data[2]. The compiler/interpreter of the METAFONT language is the `mf` program, which reads the font program source from `.mf` files (written by humans). The METAFONT programming language contains – among others – line and curve drawing instructions. `mf` follows the instructions in the font program, and draws the glyphs into a bitmap in memory, and saves the bitmap into a file when ready. Thus the printable/embeddable version of a METAFONT font is available only in bitmap format – the rasterization cannot be delayed.

METAFONT programs describe lines and curves indirectly: they impose mathematical equations the shapes must fulfill, and `mf` determines the exact shapes by solving these equations. For example, the font designer is allowed to define a point as an intersection of two, previously defined lines, and to leave the calculation of the exact coordinates to `mf`.

On the other hand, Type 1 fonts define points and shapes calculated previously, in the time of font creation. This suggests that a Type 1 font is the final output of some font generation process, such as a compiler similar to METAFONT (see [4]), or – more commonly – the export facility of a graphical, WYSIWYG font editor[3].The process itself may be arbitrary as long as its output conforms to the syntax described in the Type 1 documentation [1]. The font contains the glyphs as vector outlines, in the form of curves consisting of straight lines and Bézier-curves. The rendering (rasterization) into the printer-specific resolution is deferred as much as possible; thus Type 1 fonts are scalable (resolution-independent). Type 1 doesn't define a specific rendering algorithm, so minor pixel differences might occur when printing the same document containing Type 1 fonts on different printers. The rendering algorithms are relatively simple (much simpler than an `mf` implementation), and they are fast enough for real-time operation: each instance of the glyph is rendered independently to the others by the printer[4]. The Type 1 file format relates closely to the PostScript programming language (the font file is a valid PostScript code in fact), but it is possible to render Type 1 fonts without knowing how to interpret PostScript.

Type 1 fonts can be easely converted to METAFONT format (see [4]). This conversion

[2]a similar difference exists between TEX and HTML; one cannot write a loop that prints n stars in pure HTML, but it is possible in TEX

[3]such as the free and sophisticated PfaEdit program [8]

[4]this is not exactly true for modern printers that have a glyph cache

is merely syntactical: similar to converting plain text into a program that just prints the text into its standard output and quits: we just convert a glyph outline to METAFONT drawing instructions that draw the outline. However, this direction of conversion is quite useless, because TEX can already deal with Type 1 fonts well (even for PDF files), plus the inner structure of the glyphs wouldn't be revealed, so we'd lose the benefit of the geometrical (programmed) font generation philosophy of METAFONT.

The other way, converting `.mf` to Type 1 is expected to be harder, because the METAFONT language is much more complicated than the Type 1 syntax. So we cannot convert the METAFONT programs exactly without "understanding" them. Because current `.mf` files tend to be quite complicated, full of dirty tricks, and they are utilizing most features of the language, the only reliable way to understand these files is using a full METAFONT interpreter. There are two such interpreters available: `mf` itself and `mpost` (METAPOST). `mf` doesn't fit our needs, because it generates bitmap images and not vector outlines. On the other hand, `mpost` generates EPS (Encapsulated PostScript) output, containing vector outlines which are very similar to the Type 1 outlines. But, due to the restrictions of the Type 1 format, these outlines must be post-processed. This cleanup process is so complicated that currently *no* program exists that can do it reliably. Alternative ways of using of METAFONT or METAPOST for conversion will be discussed two sections later.

Of course -- as in the case of TEXTRACE -- one may go ahead without trying to understand the METAFONT language, and looking at only `mf`'s bitmap output. These bitmaps must somehow be converted (traced) into vector outlines that approximate the bitmap. Fortunately there exist automatic utilities for this.

A tour from CM to EC

TEX's default and most common font family is the Computer Modern (CM) family, designed by the author of TEX and METAFONT, Donald Knuth himself, between 1979 and 1985[5]. The fonts were available only in METAFONT format till 1992, when BlueSky converted them to Type 1. Now the Type 1 `.pfb` files are available from CTAN and they are part of most standard TEX distributions (such as teTEX and MikTEX).

pdfTEX is configured to use these `.pfb` files by default (`pdftex.cfg` includes `bsr.map` which contains all the available font names), so there should be no problem when creating PDF from those TEX documents which use the CM fonts.

The EC abbreviation stands for the European Computer Modern family, designed by Jörg Knappen between 1995 and 1997. *European* means that these fonts suit typesetting in some non-English European languages better than the CM fonts. Apart from this, the EC fonts are basically the same as the CM fonts with a couple of minor additions and adjustments. It is almost impossible to distinguish a font in the EC family from its CM counterpart[6]. Tha main difference between these two families is technical: the EC family contains the accented characters directly in the font, while the CM family contains the letters and the accents separately and TEX macros are used to position the accents. The EC fonts are thus larger, but they have an important

[5] for those who are still wondering: this paper uses the Computer Modern fonts almost exclusively.

[6] one of the visible differences are the accents; for example the CM acute accent is taller, and the CM Hungarian umlaut accent is more angled than its EC counterpart

benefit: text entered using those fonts can be automatically hyphenated by TeX (in the case of CM fonts the macros prevent automatic hyphenation).

Unfortunately the EC fonts are not available in Type 1 format yet, and most European TeX users are suffering from this problem. The primary reason why TeXTRACE was written is to do the conversion. However, several methods existed before TeX-TRACE to overcome this Ugly Bit. Some of them are:

- The AE fonts with the LaTeX package `ae` simulate the EC fonts from the available CM fonts with virtual fonts[7]. Both requirements are fulfilled: automatic hyphenation works because the virtual font gives TeX all the accented glyphs, and PDF files look nice because the virtual font points to the CM fonts, already available in Type 1 format from BlueSky.

  The most serious problem of the AE fonts is that only the most commonly used EC fonts are simulated, and they are available only in some of the EC design sizes (5–8 of 14). Of course some glyphs are missing because they were missing from the CM fonts (examples: eth ð, gulliemots « »). Another problem is that the metrics are a bit different from the original font, and this affects line and page breaks of the document. (This could be easily solved by overwriting the `.tfm` files from the good EC fonts.)

  Correcting the accents and creating all design sizes would have been possible with a couple of weeks of work, but the missing glyphs imposed an unsolvable limitation. And the method worked only for the EC fonts, it couldn't cope with all TeX fonts.
- The ZE fonts are similar to the AE fonts, but they collect the missing glyphs from common Adobe fonts (such as /Times-Roman). Apart from this, they have the same fundamental problems as the AE fonts.
- Automatic hyphenation works well when typesetting an English (and other latin non-accented) document. The already available CM fonts can be used for this. In LaTeX, the only thing the author must ensure is that the document preamble *doesn't* contain `\usepackage{t1enc}`.
- There are language-specific conversion utilities which modify `.tex` source files and insert discretionary hyphens (`\-`) into the appropriate places. So the hyphenation problem is solved, the automatic hyphenation provided by TeX isn't used at all. These utilities are not part of TeX itself, and have different interface for different languages. They contradict the generic, consistent and accurate way as Babel works, and make typesetting with TeX more difficult and error-prone. The solution is font-independent, so it works with the CM fonts.

Thus, in the case of the Computer Modern family, one can use the tricks above to manage without the Type 1 versions of the EC fonts. But there are no such tricks for most other TeX fonts: they need conversion, and for the conversion, one needs a proper converter.

[7] a TeX virtual font builds its glyphs from glyphs of other fonts. Virtual fonts can be used for re-encoding and/or collecting glyphs from several fonts.

TEXTRACE

TEXTRACE is a collection of scripts for UNIX that convert any TEX font into a Type 1 `.pfb` outline font immediately suitable for use with DVIPS, pdfTEX, Acrobat Reader (and many other programs). The documents using these fonts cannot be visually distinguished from the documents using the original TEX fonts, moreover the PDF documents display quickly and nicely in Acrobat Reader.

The main goal for writing TEXTRACE was to convert the EC fonts to Type 1 format with it. This goal has been fulfilled, 372 of the 392 font files (28 fonts · 14 design sizes) are available as `.pfb` files with an average file size of 86 kb. The missing 20 files are the Typewriter fonts at 5pt and the experimental Fibonacci and Funny fonts at various sizes. They are missing because METAFONT failed to generate them. The conversion took 40 hours with a 333 MHz Celeron processor and 128 Mb memory.

The 392 TC (Text Companion) fonts and some others have been also converted. (With some failures, because of METAFONT overflow or font program error).

TEXTRACE is in beta status, the author is waiting bug reports from users.

How it works

TEXTRACE renders all the 256 glyphs of the TEX font in a very high resolution (≥ 7000 DPI), then it converts the resulting bitmaps to individual vector outlines, and finally it assembles these outlines to form a syntactically correct Type 1 font. The final result is a `.pfb` font file, which can be read by both DVIPS and pdfTEX to be included into documents.

The principle sounds simple, but the implementation is quite complicated because of the usual quirks of programs used, incompatible software and loose (or disobeyed) file format specifications.

TEXTRACE operates fully automatically. It doesn't require human intervention, it doesn't ask questions. The output of TEXTRACE is useful immediately, needs no further conversion, re-editing or adjustment in font editors. TEXTRACE can operate in batch mode; this means converting all the fonts listed in a `.map` file.

The operations of TEXTRACE for a single font:

1. A `.tex` file is generated, each page with a different glyph from the font.

   Not only the shape of the glyph is important, but the absolute coordinates of the character origin must also be specified: this guarantees that glyphs will line up to basepoint and follow each other the same distances in the original and the converted font. So the TEX primitive `\shipout` is used to output pages. This way page numbers and the various glues are completely avoided, so the glyph is typeset to a well-defined, exact place on the page.

   The font is requested at a relatively large size (120.45 pt for 600 DPI and fonts with design size of 10 pt), so that 1 *em* will be 1000 pixels long when rendered.

   TEX is instructed to output glyph metrics into a separate `.log` file.

2. `tex` is invoked to create the `.dvi` file from the `.tex` file. This is a quite simple job for TeX: no line or page breaks etc. occur. As a side effect, `tex` runs `mktextfm` (or whatever font generation mechanism is used in the TeX distribution) if the `.tfm` file was missing.
3. `dvips` is run, which converts the `.dvi` file to `.ps` and automatically embeds the font into the `.ps` file.

   DVIPS reads the font from the appropriate `.pk` file (e.g. `ecrm1000.7227pk`) most of the time. If the `.pk` doesn't already exist (which is probably true since the chances for a previous use of the font at 120.45 pt or so are negligible), it calls `mktexpk` to create it. `mktexpk` looks around and passes the job to `mf`.

   All this happens automatically, without TeXtrace having to know about TeX font generation and embedding internals.
4. `gs` (GNU Ghostscript) is called to render (rasterize) the `.ps` file.

   This is almost a syntactical conversion, because METAFONT fonts are already inserted into `.ps` file in rasterized bitmap `.pk` format by DVIPS[8]. Of course the output bitmaps of Ghostscript are not sent to the printer, but they are saved to individual image files (in the PBM format).

   Special care must be taken for glyphs with negative sidebearing (i.e. glyphs extending to the left from their origin). For normal printing, this isn't a problem, because a sub-millimeter negative sidebearing is negligable with a left margin of 2 cm. In the case of TeXtrace, we must make sure that the glyph won't stick out of the page. This could be accomplished by moving a large (but constant) amount to the right and hoping that the left sidebearing is smaller than that amount. TeXtrace uses a more precise approach: it measures the glyph bounding box (with a careful trickery using the `bbox` device of Ghostscript), and moves the appropriate amount to the right.
5. The individual bitmaps are converted to vector outlines by the free external program AutoTrace (written by Martin Weber `<martweb@gmx.net>`).

   This converter supports several input and output file formats, but these are mostly equivalent. The EPS format is used for output, because it is very similar to how Type 1 represents glyphs. However, Type 1 imposes some restrictions on the glyphs, so a little modification of the original AutoTrace output routine was required to obey these restrictions:

   - Path directions (clockwise/counter-clockwise) had to be corrected, so that black regions would always be to the right-hand side from the path.
   - Multiple paths had to be converted into a single one.
   - The initial clearing of the image to white color had to be removed.
   - Curves didn't touch each other in the original output, so no modifications were necessary.

   The AutoTrace program has bugs: it errs out with some failed assertion error for 0.12% of the glyphs. The author of AutoTrace is aware of the bugs, but he currently has no fix for them. The bug is worked around empirically, by calling

[8] an exception: when DVIPS has embedded a font in other than `.pk` format.

FIGURE 2: AUTOTRACE COMPARED TO OTHER TRACING SOFTWARE

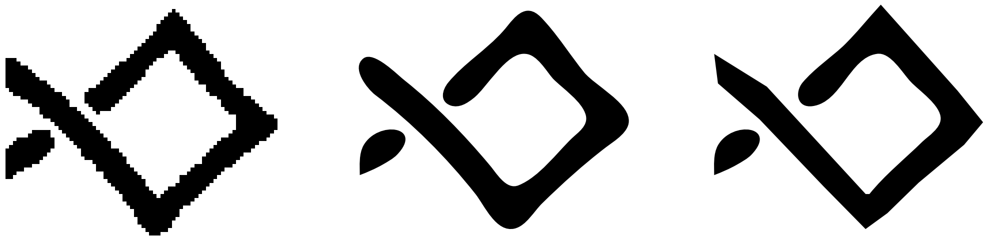

FIGURE 3: AUTOTRACE CANNOT FIND CORNERS AT LOW RESOLUTION

AutoTrace with different parameters until the bug stops to manifest.

AutoTrace – although it is free – isn't currently available from the web because of technical reasons. So the proper version of AutoTrace is included in the TEXTRACE distribution.

AutoTrace was chosen for the tracing because it is free, works automatically, and gives a good result (see Figure 2 for a comparison). Other examples of glyphs traced by AutoTrace are the ornaments throughout this document. AutoTrace does a quite good job at extremely high resolutions (such as the 7227 DPI used by TEXTRACE), but produces poor and useless output for low resolutions (example: when a glyph at 10pt is scanned at 600 DPI).

Figure 3 illustrates that the sharp corners produce the most problems for AutoTrace. Adjusting the command-line options doesn't help, and it might even do harm if different regions of the image require different options. Fortunately such distortions do not occur when using TEXTRACE, because fonts are rendered in a high enough resolution.

6. The vector outlines are cleaned up a bit to comply the Type 1 specification.

   The details of the cleanup have been explained in the previous item. The cleanup after AutoTrace is much easier than it would be after METAPOST (see [3]).

   The outlines are also properly scaled and shifted respect to the origin.

7. The outlines are merged into a Type 1 font file.

   `type1fix.pl` (a utility bundled with TEXTRACE) is run on the font file to fix quirks in common font editors and other font handling software (including pdfTEX and DVIPS).

   The output `.pfb` file will be highly portable and compatible with most font software. In an ideal world (with no Ugly Bits) there should be no font compatibility issues. However, in the real world, such issues are common since there is no standard way to read a Type 1 font apart from interpreting it as a PostScript program. (Of course that'd require a full PostScript interpreter in all font handling programs, which is too complicated and would make execution slow).

So most utilities (including those mentioned above) use a different, nonstandard Type 1 reader (invented by the "common sense" of their programmer). These readers tend to have specific lurking quirks: they are able to load most Type 1 fonts, but they might run into a problem with loading an "offending" fonts, which fully follow the Type 1 specification. Fortunately to output of `type1fix.pl` avoids all common quirks, so fonts generated by TeXtrace don't suffer from incompatibilities of third-party software.

Availability

TeXtrace is free software, licensed under the GNU General Public License. It can be downloaded from its home page:

`http://www.inf.bme.hu/~pts/textrace/`

TeXtrace requires a UNIX system with the teTeX distribution, GNU Ghostscript, Perl and GNU Bash to run. Successful runs have been reported on MacOS X and Win32, but those platforms are not supported. Of course, `.pfb` font files generated by TeXtrace are platform independent, and they can be used in any TeX environment.

Some fonts converted by TeXtrace are available for download from the same home page. These include all the EC and TC fonts, some missing AMS fonts and others.

Problems to be solved

- *Huge font files.* The `.pfb` files TeXtrace generates are around 3.15 times larger than the optimum. This is mostly because of AutoTrace cannot find the optimal Bézier-curves that describe the outline. To solve this, AutoTrace has to be partially rewritten or an other tracer must be used. (The author doesn't know of free tracers comparable to or better than AutoTrace.)
- *Corners get smoothed.* Finding the corners (sharp angles) of the outline is a weak point AutoTrace: sometimes it recognizes corners as small arcs. The author tried adjusting the tracing parameters to solve the problem, but hasn't succeeded.
- *Hinting information is missing.* Type 1 fonts use the so-called *hinting* technology to improve rendering on small resolutions (e.g. on screen, $\leq$ 100 DPI). This requires global and glyphwise additional information in the font, called *hints.* To achieve high quality, this information should be inserted by experts with great care, the process can be automated only with loss of quality. Fonts generated by TeXtrace completely lack hinting, so glyphs are expected to be badly readable in low resolutions. Fortunately, in practice, fonts generated by TeXtrace can be read quite well in Acrobat Reader unless one turns antialiasing off explicitly[9].
- *Conversion is slow.* With a 333 MHz Celeron processor and 128 Mb of memory, conversion of a single font (e.g. `ecit1728`) takes around six minutes. This is tolerable unless one wants to convert hundreds of fonts (such as all the 392 EC fonts). Although this might sound quite slow, currently TeXtrace is the fastest in its category. Moreover, the process is fully automatic, and the result is immediately

[9] with File→Preferences→General→Smooth. . .

usable without any further human intervention. Although doubled speed could be achieved with complete rewrite of the code, this wouldn't worth optimizing.

- *Limited portability.* TEXTRACE is requires a UNIX environment, but could be easily ported to other systems (with TEX, DVIPS and Ghostscript) in 1–2 days of work. TEXTRACE consists of programs written in Bash (shell script), Perl, PostScript and ANSI C. The C part cannot be avoided because the engine, AutoTrace is written in C. The PostScript part is OK, because Ghostscript is needed anyway. Shell scripts can be easily rewritten in Perl, and Perl scripts are possible to be rewritten in C, but it would require too much work.
- *Written for experts.* Although the documentation describes the conversion process for beginners, expert level knowledge is needed to enjoy TEXTRACE's benefits. The selection of source fonts, and the proper usage of the destination fonts in documents is left to the user. So is the recovery from errors: TEXTRACE is quite paranoid and stops on most errors, but the error message doesn't always indicate the real cause, and the user is left alone for figuring out and making corrections.
- *Fails for some `.mf` files.* METAFONT uses fixed point arithmetic to make rounding errors identical on all architectures. The bits representing the fractions are chopped from the integer part, so the range of numbers in METAFONT is small. The large rendering resolution asked by TEXTRACE requires large numbers. This results sometimes in overflow errors for some glyphs, and `mf` fails to generate those glyphs. TEXTRACE currently refuses to convert fonts with bad glyphs. The problem is hard to solve in the general case: the implementation of arithmetics in METAFONT should be rewritten from scratch.
- *No Unicode support.* TEXTRACE operates on normal TEX fonts with 256 or less glyphs. Although theoretically it wouldn't be hard to implement Omega and/or Unicode support, it is not available yet. TEXTRACE deals only with glyph indices, because encodings would be already resolved by the time fonts generated by TEXTRACE are needed.

PostScript font installation

Installing PostScript fonts for TEX in the general case is quite complicated. That's mostly because the font must be adapted to use a common TEX-specific encoding, accented glyphs should be composed by hand, kerning and ligature information must be adjusted etc. See the manual of DVIPS [7] or fontinst [5] for more information.

In the case of fonts generated by TEXTRACE the task is much easier. The metrics file (`.tfm`) requires no modifications, and moreover, it is already installed. Only the `.pfb` must be made available for TEX. TEXTRACE doesn't automate this process, the `.pfb` files must be installed manually. This consists of:

1. selecting the appropriate `.map` file to insert the font name and the filename. The file depends on the utility we'd like to use the font with:
   - *XDvi.* The file is `psfonts.map`, the location is probably `texmf/dvips/base`.
   - *DVIPS.* The file is any of the `.map` files mentioned in `config.ps`, most probably in `texmf/dvips/config`. `config.ps` almost always contains the entry for

psfonts.map.

- *pdfTEX.* The file is any of the .map files mentioned in pdftex.cfg (in texmf/pdftex/config). There is no default filename.

Note that the original .map files do not have to be overwritten, it is enough to place the new version into the current directory, thanks to Kpathsea.

2. appending a line describing the font to the .map file. This line is of the form:

   ⟨*TEX-font-name*⟩␣⟨*PostScript-FontName*⟩␣<⟨*PFB-file-name*⟩

   Example:

   ```
   ecrm1000 TeX-ecrm1000 <fcrm1000.pfb
   ```

   The ⟨*PostScript-FontName*⟩ can be read from the .pfb file after /FontName, but in the case of TEXTRACE it is simply the word TeX- prepended to ⟨*TEX-font-name*⟩. See the file contrib/ecpfb.map of the TEXTRACE distribution.

3. copying the .pfb file to its final, system-wide location. This step isn't mandatory, the font is also functional in the current directory, thanks to Kpathsea. The most common system-wide location for Type 1 fonts is texmf/fonts/type1/*.
4. testing whether the utility really finds the .pfb file by running it on a sample document and checking the console output whether it includes the .pfb file.

Other uses of TEXTRACE

Although TEXTRACE's primary goal is to convert METAFONT fonts to the Type 1 format automatically, the program is a bit more general: it can convert any font available to DVIPS (practically any TEX font) to Type 1, regardless of the original representation of the font.

The Type 1 format is standard since 1985, and TEXTRACE creates files that are compatible with all common font handling utilities (including many font editors, DTP programs, converters, printers, typesetters). Type 1 fonts can be easily converted to any of the font formats used nowadays. So TEXTRACE can be used as a first step (the TEX-specific step) to make a TEX font available in any publishing environment.

TEXTRACE converts only the glyph outlines, so kerning, ligature and other metrics information (all located in the .tfm file) must be converted separately. The documentation of TEXTRACE contains details about the automation of this process. Today's fonts tend to have more and more glyphs, exceeding TEX's limit of 256. It is common that a font is divided to smaller TEX fonts. These sub-fonts should be merged in a font editor after running TEXTRACE on all of them. There might be difficulties with kerning pairs and ligatures crossing font boundaries.

Alternatives

The method used by TEXTRACE isn't ultimate, there are many other possible solutions.

Respects of comparison

(F) *Final quality of the output.* This is the most important respect. Legibility and beauty must be considered in both low (screen) and high (printer) resolutions. The converted font must be accurately reflect the original, there should be no visual difference, not even in the smallest corners.

(C) *Compatibility.* The output must be compatible with both document preparation and reader software.

(S) *Size.* Although the capacity of hard disks increase rapidly nowadays, font files must be kept small because they might be loaded into printers very limited memory (even as few as 512 kb). The font sizes also affect download time of the PDF document.

(A) *Amount of human effort needed.* The more automated the process is, the better. Human effort is expected to be much for TEX fonts, because they are available in many (often 10 or more) design sizes and many styles.

Possible approaches

- Avoid conversion by using a virtual fonts to get a mixture of fonts that are already available in Type1 format. (See the subsection about EC fonts above.)

  The most serious limitation is that this approach is useless if no similar font is available in Type 1 format. (F), (C), (S): no problems. (A): it is not difficult to arrange that metrics and displacements in the virtual fonts are generated automatically.

  The AE, ZE and ZD fonts use this method.

- Design new fonts parallel in METAFONT and Type1. Develop a system that glues existing tools to aid parallel font development by allowing generation of both fonts from the same (new) source file (see [4]).

  The most serious problem is that existing fonts cannot be converted this way.

- Modify METAFONT somehow to output Type1 fonts (or vector outlines) instead of bitmaps.

  Modifying the `mf` engine itself would require too much effort. Another way is to modify only the output generation process of METAFONT by adding macros and `.log` file processing scripts. Such utilities already exist (for example: mf2ps by Daniel Berry and Shimon Yanai and the MF-PS package by Boguslav Jackowski), but they no longer work, cannot comply the restrictions of the Type 1 format or work only for `.mf` files specially written for using them.

- Interpret and execute `.mf` files directly, suppressing METAFONT altogether.

  This would be as hard as to completely rewrite METAFONT from scratch having Type 1 output capability in mind, because METAFONT is a full, complex programming language and most `.mf` files make use of this complexity. A faithful rewrite of METAFONT is really hard because there are too many (dirty) tricks in both the language and the current implementation.

- Post-process METAFONT's output.

  Effectively, post-processing means tracing. This is what TEXTRACE does. (F): font quality is far from the best, because of hinting information is missing. Traced fonts looks almost always ugly in low resolutions, unless (H) a hinting is inserted manually, using a great deal of human resources. (S): font size can be reduced with sophisticated tracing techniques, possibly exploiting the curve representation model of the renderer. AutoTrace isn't so sophisticated, so it generates somewhat large fonts.

- Use METAPOST instead of METAFONT to interpret MF files. Post-process METAPOST's output.

  METAPOST is a METAFONT-clone with PostScript output. The PostScript it generates is suitable for conversion to Type 1 outlines.

  The commercial software MetaFog does this. It also makes the outline comply restrictions in Type 1. Such a restriction is that all paths must be filled, not stroked, and overlaps must be removed. It is theoretically possible to achieve this, but currently no robust implementation exists, not even MetaFog.

  There are fundamental problems with METAPOST itself. First, its language is only a subset of METAFONT's, so most fonts (including the EC fonts) don't even compile. Second, it cannot output strokes of elliptical or more complicated pens (which is required by some fonts). So it is a matter of luck whether one can succeed with METAPOST for a specific font.

- Apply a mixture of these methods.

  For example, one may get most glyphs from existing Type 1 fonts, and trace only the missing glyphs.

For the PDF problem

Our original goal was to find out a method for embedding TEX fonts into the PDF file so that the document would look nice and display quickly in PDF viewers, especially in Acrobat Reader.

It turned up that this can be ensured by inserting the fonts in some vector outline format. The quality of the font (minimal file size, presence of good hinting) seemed to be subsidiary when embeding it into PDF: current PDF viewers tend display even medium-quality fonts nicely. The conversion target must be undoubtly a vector outline format, but not necessarily Type 1. Another alternative is TrueType, which differs from Type 1 in two main respects:

- Hinting mechanism is much better, fonts are eye-pleasing to read even on screen and without antialiasing.
- It uses second order curves instead of third order (Bézier-) curves. Second level curves approximate the outline less precisely, so more control points are needed to achieve the quality of Type 1. This results larger file sizes.

For our intentions, neither of these respects is really relevant; the two font formats provide an equally good solution for the problem. However, considering compatibility issues, Type 1 turns out to be more adequate: it is older than TrueType, and more TEX-related utilities support it. Moreover, Type 1 has no dialects, can be represented in plain text canonically, and is easier to assemble and disassemble.

Conclusion

For those dealing with computers, Ugly Bits are experienced in everyday life, not just in memories. European TeX users have suffered for years because of an Ugly Bit in font inclusion to PDF documents. With the appearance of TeXtrace, the problem became less severe (and documents became much less ugly).

Font conversion retaining the quality of the original is almost as tough as designing new fonts. For best results, both of them must be done manually. However, hand-converting hundreds of similar fonts is an enormous work, needs expert level knowledge and isn't really inspiring. Probably that's the reason why most of the TeX fonts weren't available in Type 1 format despite of the great need for it. (It is clear that the conversion in high quality is *possible*.)

TeXtrace is an automatic and generic converter. It doesn't provide the best quality theoretically achievable in today's technology and human resources, but its output is good enough for most purposes. The three most important benefits of TeXtrace are that it works for all TeX fonts (with a few technical exceptions), it runs completely automatically, without any human intervention, and it is free.

Due to the reasons mentioned above, professional quality Type 1 variants of all popular TeX fonts are not expected to appear for years. Until this happens, fonts generated by TeXtrace will be the best alternative.

References

[1] Adobe Systems Inc. *Adobe Type 1 Font Format.* Addison–Wesley Publishing Company, 1995.

[2] Alexander Berdnikov, Yury Yarmola, Olga Lapko and Andrew Janishewsky. *Some experience in converting LH Fonts from METAFONT to Type1 format.* Proceedings of the 2000 Annual Meeting, TUGboat, 21(3), 2000.

[3] Taco Hoekwater. Generating Type 1 Fonts from Metafont Sources. *Proceedings of the TUG conference*, TUGboat 16(3), 1995.

[4] Boguslaw Jackowski, Janusz M. Nowacki and Piotr Strzelczyk. *METATYPE1: a METAPOST-based engine for generating Type 1 fonts.* Proceedings of the EuroTeX conference, 2001.

[5] Alan Jeffrey and Rowland McDonnell. *fontinst – Font installation software for TeX, fontinst v1.8.* `teTeX:doc/fontinst/fontinst.dvi.gz`, 1998.

[6] Hàn Thế Thành, Sebastian Rahtz and Hans Hagen. *The pdfTeX user manual.* `teTeX:doc/pdftex/base/pdftexman.pdf.gz`, 1999.

[7] Tomas Rokicki. *Dvips: A DVI-to-PostScript Translator.* `teTeX:doc/programs/dvips.dvi.gz`, 1997.

[8] George Williams. *PfaEdit – A PostScript® Font Editor, User's manual.* `http://pfaedit.sourceforge.net/overview.html`, 2000.