Experiences with micro-typographic extensions of pdfTEX in practice

Hàn Thế Thành
University of Education,
Ho Chi Minh City,
Vietnam

Februar 21, 2005

Abstract

pdfTEX provides two micro-typographic extensions: margin kerning (also known as character protrusion) and font expansion. Those extensions have been available for a while, however they are not used much yet, probably due to their complicated setup and not that visible benefits they bring. In this article I want to share some experiences, either good or bad, in using those extensions in practice, the tricky parts of them and how to get the best from what pdfTEX offers without having to know all the low-level details and messy font issues.

1 Introduction

Font expansion and margin kerning have been introduced several times in various articles, so I won't go to any detailed description here. From a practical point of view, what those extensions bring is pretty simple:

1. Margin kerning makes the margins of text *look* smooth, by moving certain characters out to the margins by a small amount. The most common case is to move the hyphen character or punctuation marks, but applying this technique to certain letters also improves the result.
2. Font expansion can help to improve line-breaking. Typically, a text typeset using font expansion has:
   (a) less hyphenations,
   (b) less overfull and underfull boxes,
   (c) more equal inter-word spacing ,
   (d) reduced occurrence of "rivers".

   These benefits are most visible in difficult cases, like narrow-column typesetting, disabled hyphenation, or simply in automated typesetting when manual work needed to correct problematic cases should be minimized or totally avoided.

Margin kerning and font expansion have been implemented in the *hz* program by Hermann Zapf and URW; that's why I called them together *hz* extensions of pdfTEX[1]. For more detailed information on margin kerning and font expansion, including background and related works on *hz* extensions in other systems, please refer to [Thành 2001]. From now, by *hz* extensions I mean margin kerning and font expansion in pdfTEX.

2 How to start using *hz* extensions

Similarly as in the case with TEX primitives, the *hz* extensions as provided by pdfTEX are not easy to use. They require the understanding how certain concepts in TEX work at the very low-level, as well as the ability to set up some complicated font-related stuff. Thus the best way to start using *hz* extensions is via some available

[1] The *hz* program is in fact a set of modules implementing certain micro-typographic improvements including margin kerning and font expansion. *hz* extensions of pdfTEX are only a small subset of the modules in the *hz* program

interface. An article about how to start using *hz* extensions in practice will be published in TUGboat soon so I don't go into the details here. In short, given that we have pdfTEX version at least 1.20a (1.21a is recommended at the time of writing this paper) and the LATEX package microtype installed, it's enough to say

```
\usepackage{microtype}
```

to activate both margin kerning and font expansion. A recommended next step is to read the microtype manual to learn more about the options the package offers, as well as advice for new users.

3 Practical experiences

hz extensions have been used rarely in practice, due to the lack of an easy interface and the necessity of complex font setup as mentioned above. Also, those features are not completely mature yet. For those reasons, a collection of practical issues would be helpful for those interested in using *hz* extensions. Some of the issues described here may sound very technical, or very weird to those who have not tried to use *hz* extensions in pdfTEX yet. If it is the case, simply skip the details you don't understand.

So far, the total number of people who have contacted me to discuss some issue concerning *hz* extensions is about ten. Some of them use *hz* for some occasional projects, some use *hz* for their regular work. Of course it doesn't have to mean that only ten people have been using *hz* extensions; it only means that at least ten people tried and had some trouble.

Recently, ConTEXt and LATEX (the pdfcprot and microtype package) make *hz* extensions easier to use and become more popular, especially margin kerning. The introduction of auto expansion is also a big step toward easy use of font expansion.

3.1 Using the auto expansion feature

To deploy font expansion, the expanded TFM fonts must be prepared ahead, using some utility as `fontinst`, afm2tfm or METAFONT. This is the most annoying part, even for experienced users. And usually this annoyance is doubled by the fact that most TFM's must be used with VF, hence the expanded VF's must be created as well.

From version 1.20a, pdfTEX supports a feature called auto expansion, which allows TFM and VF to be expanded automatically in memory at run time. That means expanded TFM and VF are no longer required, which is a great relief. In order to use font expansion now, it's enough to upgrade pdfTEX binaries, install the LATEX microtype package, and that's it. No need to deal with expanded TFM's and VF's, map files or whatever. This feature makes things really simple. The implementation however is not that simple and it took some time to evolve and mature.

There is one catch in using virtual fonts with auto expansion: in virtual fonts accented characters are often drawn as composition of two glyphs, a base letter and an accent. Using auto expansion will cause the accent in such composed glyph to be misplaced by a small amount (0.01–0.1pt).

Now one may wonder whether auto expansion should be used in case all the expanded TFM's and VF's already exist (most likely because the user had to create them manually when auto expansion was not available yet). The answer is if DVI output is not considered (see the next issue) and only Type 1 fonts are being used, auto expansion should always be used. This is also the typical case.

3.2 Using *hz* extensions in DVI mode

hz extensions are available in both PDF and DVI mode. Using *hz* in DVI mode is much similar to PDF, although it requires some extra setup for dvips to process the expanded fonts. If only margin kerning is used, then there is no difference whether it is used in DVI or PDF mode.

The question is why one would want to use *hz* in DVI mode? There are some known reasons:

1. the output is smaller: in PDF mode pdfTEX embeds many instances for a single expanded font, while dvips embeds only one;
2. the document requires PS processing, for example PStricks;
3. the user just doesn't need or like PDF, but wants *hz*.

The following (typical) example demonstrates how to make dvips process DVI files with font expansion enabled. It only makes sense to people who can already make some setup to use font expansion in PDF mode without auto expansion, which means that you must be able to create the expanded TFM's and VF's using fontinst or afm2tfm or some similar tool. The detailed instructions on how to do that is however out of scope of this article.

Assume that we have activated font expansion for font cmr12 with stretch limit 20, shrink limit 20 and expansion step 5. To process the DVI file produced using this setup, we must update the map entry read by dvips. In my system, the entry for cmr12 is

```
cmr12     CMR12  <cmr12.pfb
```

Now it must be replaced by

```
cmr12     CMR12  <cmr12.pfb
cmr12+5   CMR12  "1.005 ExtendFont"  <cmr12.pfb
cmr12+10  CMR12  "1.010 ExtendFont"  <cmr12.pfb
cmr12+15  CMR12  "1.015 ExtendFont"  <cmr12.pfb
cmr12+20  CMR12  "1.020 ExtendFont"  <cmr12.pfb
cmr12-5   CMR12  ".995 ExtendFont"   <cmr12.pfb
cmr12-10  CMR12  ".990 ExtendFont"   <cmr12.pfb
cmr12-15  CMR12  ".985 ExtendFont"   <cmr12.pfb
cmr12-20  CMR12  ".980 ExtendFont"   <cmr12.pfb
```

Then dvips can process the DVI with expanded fonts just like any other DVI files.

The main disadvantage of DVI mode is, however, that the auto expansion feature of pdfTEX cannot be used. Or to be more precise, auto expansion can be activated in DVI mode and pdfTEX can create the DVI file with font expansion enabled. Such a DVI file however is pretty useless because DVI drivers cannot process that file, as they don't have access to the expanded TFM's and VF's (those exist in pdfTEX memory at run time only).

There have been requests to support auto expansion for use with dvips, by changing pdfTEX to write the expanded TFM's and VF's to disk as well as to update some map files. It is likely that this will never be implemented. The better way to do that is to change the script TEX (pdfTEX) calls to create missing TFM at run time (on web2c-based systems this is called mktextfm) to create all the required stuff on demand.

3.3 Margin kerning and non-character material

Margin kerning only works with characters. Sometime we need to protrude something that is not a character, for example some superscript (index), because such a thing would look bad when ending up at the margin without protrusion. The typical example is the index of footnote, which is typeset into an hbox.

There are two solutions to the above problem:

1. Make margin kerning work even with characters inside boxes. A patch has been made to check whether the ending element of a line is a box, and if so check the last element of that box, and so on, to ensure that the last character inside boxes will get protruded. This patch is still experimental at the time writing this article. Also beware that it is not enough just to have this patch and the right package loaded, because the default settings for margin kerning as provided by macro packages are suitable for normal text. Superscript text often has much smaller size than the normal text, so to get the right result the protrusion factors must be increased. Then it might cause conflicts in other places where the same font is used for normal text (for example in footnote text).

2. Append a "virtual character" immediately after the material we want to protrude into the right margin (or prepend in case of the left margin). Such a virtual character is not visible, has no dimension, but has non-zero protrusion factor so when ending up at the margin it will get protruded. Creating such a virtual character is quite easy using fontinst. There is also a script available to generate a virtual font with all blank characters for this purpose[2]. This approach has been used several times in practice and is reliable. The drawback is that it requires some extra work in creating those virtual characters and inserting them into the right places.

[2]The script was made by Hartmut Henkel, however having read the draft version of this paper he suggested to distribute the font itself; so the font will be available at pdftex.sarovar.org soon.

3.4 Margin kerning does not work in some cases

Sometimes it happens that margin kerning doesn't work as expected, some characters are not protruded. There are usually two reasons:

1. Margin kerning is blocked by some invisible material around the relevant character; usually there is a workaround using macros.
2. It is a bug in pdfTEX. If your pdfTEX is older than 1.20b then upgrading pdfTEX is the first thing to consider when encountering some problem with *hz* extensions. Margin kerning has been improved a lot from version 1.20b.

3.5 Using margin kerning and font expansion at the same time

Although this seems something evident, sometimes it doesn't work. The reason is most likely that you are using pdfTEX version 1.20a, which has this bug.

3.6 Reasonable settings for font expansion

Font expansion must be used with care. While it gives more room for line-breaking, it can also destroy the whole text if the effect of font expansion becomes visible. Then the question is when font expansion becomes visible? This question has no definitive answer, as to trained eyes font expansion can be easily detected and hence annoying, while to others it has no effect. For most people the safe limit is 2 % expansion, i. e. the stretch and shrink limit when expanding a font should not be more than 20 (see pdfTEX manual for explanation of stretch and shrink limit).

The expansion step is usually set to 5. A too small value leads to large output size (more fonts embedded), while a too big value can lead to some surprises like unexpected overfull or underfull boxes. This is not a bug but a limitation in the way pdfTEX implements font expansion.

4 Lessons learned

During the development of *hz* extensions, many decisions were made and not all of them were good. pdfTEX started as an experimental project. Most decisions in the beginning were made rather for the purpose to examine the effect of *hz* extensions than for practical purpose. Then *hz* extensions started to be used in practice and certain things had to be changed to make life easier. Here are some lessons I learned concerning *hz* extensions:

1. The way margin kerning can be used without changing existing fonts seems to be the right decision. Settings for margin kerning should be part of fonts just like kerning between pairs of characters, from the viewpoint of clean design. Doing so however would lead to two problems:

   (a) to use margin kerning, we need some kind of extended TFM;
   (b) the settings cannot be changed easily.

   These problems would make practical use of margin kerning impossible. Clean design is important but sometime backward compatibility and flexibility play a more significant role.

2. The concept of font expansion in pdfTEX was quite general and flexible, which was good for experiments as we needed to study the effect of font expansion in as many cases as possible. From the experimental viewpoint the way font expansion was implemented is not that bad, but from practical viewpoint it is too cumbersome and hard to deploy. Hence for practical use, a less flexible mechanism which is easier to use but offers 80% of what font expansion can give is more needed. That's the reason why auto expansion has been introduced.

   So was it a bad decision that pdfTEX needed expanded TFM for font expansion? For experimental purpose it was a good decision, however it would be a mistake not to change it after the experiments have been done.

3. The user interface is as important as the implementation to end users. Without LATEX or ConTEXt support, *hz* extensions are pretty useless to most users.

4. Feedback and help from the pdfTEX user community is vital for *hz* extensions to evolve and mature.

5 Recent changes

pdfTEX version 0.14h is the last version of pdfTEX released by me during my stay in Czech republic. Then I came back to Vietnam and didn't have time to work on pdfTEX for about two years. During that time pdfTEX was maintained by the pdfTEX team (lead by Hans Hagen and Martin Schröder). Since March 2004 I came back to work on pdfTEX development, and version 1.20a is the first release I participated in after the long break.

Here I would like to give a brief summary of some noticeable changes since version 1.20a, as I think that they might be of interest to pdfTEX users or simply to those who don't use pdfTEX but want to keep an eye on what is happening with pdfTEX. Apart from those, there have been many small bug fixes and improvements but they are too technical to mention here.

1. *hz* extensions have been significantly improved; some serious bugs have been fixed and margin kerning has been extended to handle some special cases.
2. Auto font expansion has been introduced: expanded TFM's are no longer required to use font expansion.
3. The font expansion mechanism has been simplified: the font expand factor (the last argument of `\pdffontexpand`) is no longer supported. This parameter was used to simulate font expansion using letter spacing. Experiments have shown that this technique is not the way to go: it didn't improve much the result while it caused many troubles.
4. Support for the configuration file `pdftex.cfg` is gone; all parameters are set via primitives. Their values can be dumped to the format file. The only exception is `\pdfmapfile`: its value cannot be dumped to the format file, so when pdfTEX starts the value of `\pdfmapfile` is always set to the default value `"pdftex.map"`.
5. pdfTEX uses the GNU libAVL library to speed up certain searchings.
6. Support for TrueType fonts has been improved, allowing refering to glyphs inside a TrueType font by their unicode index. ttf2afm also has been heavily revised.
7. There is a program called pdfxTEX, which is a variant of pdfeTEX that contains experimental features. Those features may be moved to pdfTEX when they seem to be useful and stable. At the moment the following extensions are avaiable:
   - `\pdflastximagecolordepth` returns the last color depth of a bitmapped image;
   - `\pdfximage` supports a keyword `colorspace` following an object number representing a PDF ColorSpace object;
   - `\pdfstrcmp` compares two strings;
   - `\pdfescapestring` and `\pdflastescapedstring` provide a means to escape strings;
   - `\pdffirstlineheight`, `\pdflastlinedepth`, `\pdfeachlineheight` and `\pdfeachlinedepth` allow fixing line dimensions during paragraph buiding;
   - various extensions from Taco Hoekwater:
     - support for dimension unit `px`;
     - `\tagcode` primitive allowing read and write access to a character's `char_tag` info.
     - `\quitvmode` primitive quits vertical mode;

6 Pending requests and future development

In this section I would like to mention shortly some issues that have been discussed and the future plan of pdfTEX.

PDF inclusion with annotations: when pdfTEX includes a PDF figure, all the annotations (the PDF term for hyperlinks and the like) from the figure are lost. There is a patch for pdfTEX version 1.10b by Andreas Matthias that copies annotations from included PDF figures. The patch was released during the period when I was not maintaining pdfTEX and didn't get attention from other pdfTEX maintainers either. I am aware of the patch but did not look into the code yet. If this becomes urgent or frequently asked then it should be re-considered.

Implement virtual fonts using Type 3 fonts: pdfTEX supports virtual fonts in the same way like other DVI drivers does, i. e. it interprets the DVI commands from virtual fonts to draw characters. It means that accented characters from virtual fonts are often unsearchable – in fact there are no such letters in the PDF output but sequences of PDF commands drawing the base letter and the accent. There have been requests to make letters from virtual fonts searchable. One possible solution is to implement virtual fonts as Type 3 fonts in PDF. This feature is something very handy to have, as it can allows other nice things as well. At the moment this feature is still being examined, and might be supported in the future if the effort required to implement it is not too much.

Support for subfont scheme for use with huge TrueType fonts: Subfont scheme is a trick to split huge TrueType fonts (usually used for Asian languages like Chinese, Japanese or Korean) into smaller pieces so that they can be used with 8-bit TEX. pdfTEX has some support for subfont scheme, but still very poor. At the moment this is being revised and should be improved in the near future.

Support for pdfsync: pdfsync is a package allowing synchronization between a PDF file created by pdfTEX and its LATEX source: the user clicks on some point in the PDF file and the editor "jumps" to the corresponding place in the source. The current implementation of pdfsync still has some unsolved issues, due to the lack of low-level support in pdfTEX. We had some discussions with the pdfsync author and I plan to provide some hooks to support pdfsync.

7 Acknowledgments

Too many people helped pdfTEX development in various ways, so it is impossible to thank all people here without missing someone. However, I would like to say thanks to a few people whose impact on pdfTEX is most vital in the last years:

1. Hans Hagen for his testing, discussions, feature requests, feedbacks and encouragement;
2. Hartmut Henkel for his important contributions on pdfTEX development and maintenance, especially in *hz* extensions;
3. Martin Schröder for his effort on keeping pdfTEX in sync with latest sources of libraries and other pieces of TEXLive and teTEX; he is also responsible for official pdfTEX releases;
4. and NTG and DANTE for financial support on this work.

References

[Thành 2001] Hàn Thế Thành, *Margin Kerning and Font Expansion with pdfTEX*, in: *TUGBoat*, vol. 22(2001), no. 3 – Proceedings of the 2001 Annual Meeting, pp. 146–148.
(Online at `http://www.tug.org/TUGboat/Articles/tb22-3/tb72thanh.pdf`)