FreeType_MF_Module: A module for using METAFONT directly inside the FreeType rasterizer

Jaeyoung Choi, Ammar Ul Hassan and Geunho Jeong

Abstract

METAFONT is a font description language which generates bitmap fonts for the use by TEX system, printer drivers, and related programs. One advantage of METAFONT over outline font is its capability to produce different font styles by changing various parameter values defined in its font specification file. Other major advantage of using METAFONT is that it produces various font styles like bold, italic, and bold-italic from one source file unlike outline fonts, which require development of a separate font file for each style in one font family. These advantages can be applied to design CJK (Chinese-Japanese-Korean) fonts which require significant time and cost because of the large number of characters used in Hangeul (Korean character) and Hanja (Chinese character). However, to use METAFONT in current font systems, users need to convert it into its corresponding outline font. Furthermore, font rendering engines like FreeType doesn't support METAFONT.

In this paper, *FreeType_MF_Module* for FreeType rasterizer is proposed. The proposed module enables a direct usage of METAFONT just like any other outline and bitmap font support in FreeType rasterizer. Users of METAFONT don't need to pre-convert METAFONT into its corresponding outline font as FreeType_MF_Module automatically performs this. Furthermore, FreeType_MF_Module allows the user to easily generate various font styles from one METAFONT source file by changing parameter values.

1 Introduction

As of today's, information society, the traditional pen and paper usage for communication between people are swiftly replaced by computers and mobile devices. Text has become an effective source for gathering information and means of communication between people. Although people use smart devices commonly these days with effective resources like media and sound, text plays the key role of interaction between user and device. Text is composed of characters, and these characters are physically build from specific font files in digital environment system.

Fonts are the graphical representation of text in a specific style and size. These fonts are mainly categorized in two types; outline fonts and bitmap fonts. Outline fonts are the most popular fonts for producing high-quality output used in digital environment system. However, for creating a new font style for outline font, font designers have to design a new font with extensive cost and time. This recreation of font files for each variant of font can be very hectic for font designers especially in case of CJK fonts which require designing of thousands individual letters one by one. Since CJK fonts have a lot more letters and complex shapes compared to Roman fonts, it often takes more than a year to design CJK fonts.

To overcome this disadvantage of outline fonts mentioned above a programmable font, METAFONT was developed. METAFONT is a programming language introduced by D. E. Knuth [1] that generates TeX-oriented bitmap fonts. It allows users to generate various font styles easily. METAFONT file is different from outline font file. It consists of functions for drawing characters and has parameters for different font styles. By changing the parameter values defined in its font specification file, various font styles can be easily generated. Therefore, a variety of font variants can be generated from one METAFONT source file.

However, users are unable to use METAFONT on their PCs because current font engines like FreeType [2] doesn't provide any direct support of METAFONT. Unlike standard bitmap and outline fonts, METAFONT is expressed as a source code that is compiled to generate fonts. To use METAFONT in a general font engine like FreeType rasterizer users have to convert METAFONT into its corresponding outline font. As it was developed in 1980s, the PC hardware was not fast enough to provide a real-time conversion of METAFONT into its corresponding outline font.

Current PC hardware, however, is fast enough to provide a real-time conversion of METAFONT into its corresponding outline font. If METAFONT is used directly in font engines like FreeType, then users can easily generate variety of font styles. METAFONT can also be used to produce various font styles like bold, italic, and bold-italic with one source file unlike outline fonts, which require development of a separate font file for each style in a font family. These advantages can also be applied to design CJK fonts to produce various high-quality font styles because, compared to alphabetic scripts, CJK characters are both complicated in shape and expressed by combinations of radicals [3].

In this paper, a FreeType_MF_Module for FreeType rasterizer is proposed. The proposed module enables a direct use of METAFONT in FreeType rasterizer just like any other default outline and bitmap font modules in it. With FreeType_MF_Module, users

don't need to pre-convert METAFONT into its corresponding outline font before using it with FreeType rasterizer as FreeType_MF_Module automatically performs this. It allows users to easily generate variants of font styles by applying different parameter values. This module is directly installed in FreeType rasterizer just like its default font modules so, there is no dependability and performance issues. We have tested our proposed module by generating different font styles with METAFONT and compared its performance with default modules of FreeType and other researches.

This paper is organized as follows. In Section 2, the related research regarding font modules and libraries are explained. The architecture of FreeType_MF_Module is explained in Section 3. The authors have tested the FreeType_MF_Module by demonstrating how FreeType can provide support for METAFONT directly in Section 4. FreeType_MF_Module's performance is also compared with FreeType default modules and other researches in this section. Section 5 gives concluding remarks.

2 Related research and their problems

MFCONFIG [4] is a plug-in module for Fontconfig [5]. It enables the use of METAFONT on Linux font system. Figure 1 shows the architecture of MFCONFIG module linked with Fontconfig.

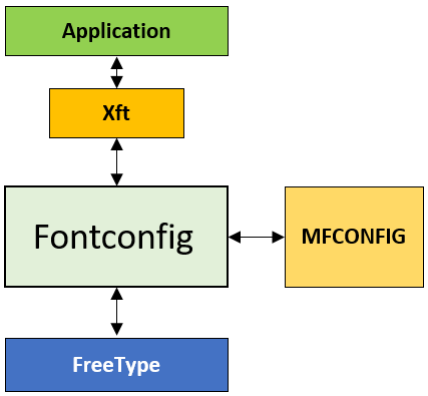


Figure 1: Basic architecture of MFCONFIG module

Although MFCONFIG supports METAFONT in Linux font system, it has performance and dependability problems. Since MFCONFIG is plugged into the high-level of font system i.e. Fontconfig, and not directly plugged inside FreeType, it makes its performance very slow compared to the font-specific driver modules supported by FreeType. Whenever the client application sends a METAFONT file request, Fontconfig communicates with MFCONFIG, performs operations, and then sends input to FreeType for rendering text. This whole process becomes slow because of the high-level operations before FreeType receives its input.

Other than the performance problem, MFCONFIG also has a dependability problem. As it is dependent on Fontconfig library, this means that if there is a font environment where there is no Fontconfig, then this module cannot be used. Fontconfig is mainly used in Linux font system for locating fonts and no other operating systems, so MFCONFIG cannot be supported in them.

VFlib [6], a virtual font library, is a font rasterizer developed for supporting multilingual fonts. VFlib can process fonts which are represented in different font formats and outputs glyphs as bitmap images from various font files. VFlib supports many font formats like TrueType, Type 1, GF, and PK fonts [7] etc. It provides a unified API for accessing different font formats. A new module can be added in this font library for adding support for METAFONT but this library has its own drawbacks; as it supports many different font formats and with a database support it can be very heavy font library for embedded systems. It is also dependent on additional font libraries like FreeType engine for TrueType font support, T1lib [8] for Type 1 font support so it has its dependency problems as well. Therefore, VFlib is not suitable to add support for METAFONT.

FreeType is a font rasterizer to render fonts and it can produce high quality output for mainly two kinds of font formats, vector and some bitmap font formats. FreeType mainly supports font formats such as TrueType, Type1, Windows, and OpenType fonts using same API irrespective of their font formats. Although FreeType supports many different font formats but it doesn't provide any support for METAFONT directly. If there is a module for FreeType that directly supports METAFONT then users can take the advantages of METAFONT by generating variants of font styles by just changing parameter values. MFCONFIG module problems can also be resolved using this module. FreeType can also be used for supporting TEX oriented fonts and not only outline or bitmap fonts.

The proposed FreeType_MF_Module in this paper uses the process for printing METAFONT from MFCONFIG module. FreeType_MF_Module intends to solve the two problems of MFCONFIG module. METAFONT can be used with any system having FreeType using the proposed module. As it is implemented just like any other default FreeType module, it can be easily installed or uninstalled.

3 Implementation of FreeType_MF_Module in FreeType rasterizer

3.1 FreeType_MF_Module as an internal module of FreeType

FreeType can support various font formats. Processing a font file corresponding to its format is done by an internal module in FreeType. This internal module is called a font driver. FreeType contains a configuration list of all the driver modules installed in a specific order. When FreeType receives a request of font file from an application, it passes this request to the driver module mentioned on the top of the list for processing. This module performs some internal operations to check if this font format can be processed or not. If this driver module supports this request, it performs all other operations to process this font file request. Otherwise this font file request is sent to the second driver module mentioned in the list. This process continues until a font driver is selected for processing the font file request. If no font driver can process the request, an error message is sent to the client application. Similarly, FreeType_MF_Module is directly installed inside FreeType just like its other internal modules. When the client application sends a request of METAFONT file, FreeType_MF_Module receives this request and processes it. Figure 2 shows how FreeType will select a driver module for processing a METAFONT file request.

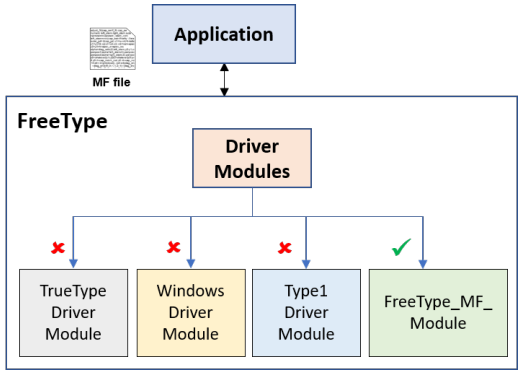


Figure 2: Process of selecting a module in FreeType

FreeType_MF_Module consists of three sub-modules: Linker module, Administrator module, and Transformation module.

3.2 Linker Module

Linker module is the starting point of FreeType_MF_Module. It is mainly responsible for linking FreeType internal modules with FreeType_MF_Module. It is divided into two parts: inner meta interface and outer meta interface. Inner meta interface receives font file request from internal modules and delivers it to Administrator module for processing. After processing by Administrator module, outer meta interface delivers the response to internal modules for further operations. The process of Linker module is shown in Figure 3.

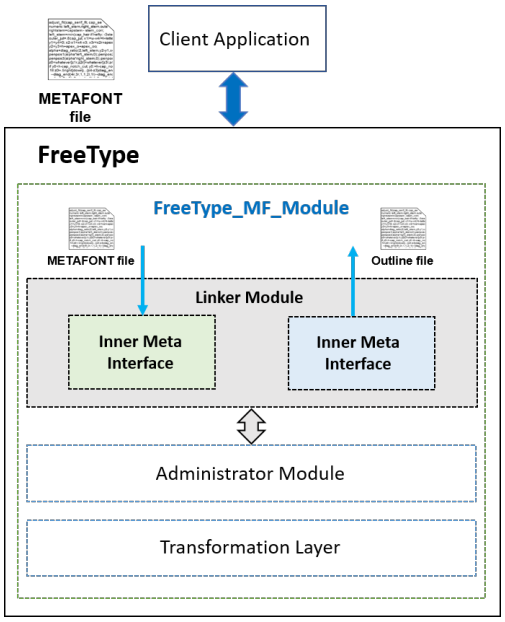


Figure 3: Linker module

3.3 Administrator Module

The core functionality of FreeType_MF_Module is performed in Administrator module. This module is divided into two layers; Search layer and Management layer.

Search layer is responsible for finding all the installed METAFONT fonts in a table. This table contains a list of all the METAFONT fonts installed and for fetching information related to them. Search layer is implemented in Meta scanner and Meta table.

Management layer mainly performs following tasks. (1) Checking whether the requested font file is METAFONT or not, (2) Checking the cache if the corresponding outline font for the METAFONT request is already stored. If yes, it directly sends the response from the cache. This functionality is implemented to achieve better performance and re-usability. (3) If the outline font is not prepared in the cache this request is sent to Transformation layer. (4) The outline font prepared by Transformation layer is stored in the cache (5) The response is sent back to FreeType internal modules by management layer. Management layer is implemented in Meta analyzer, Meta request, and Meta cache. Figure 4 shows the Administrator module and its sub layers.

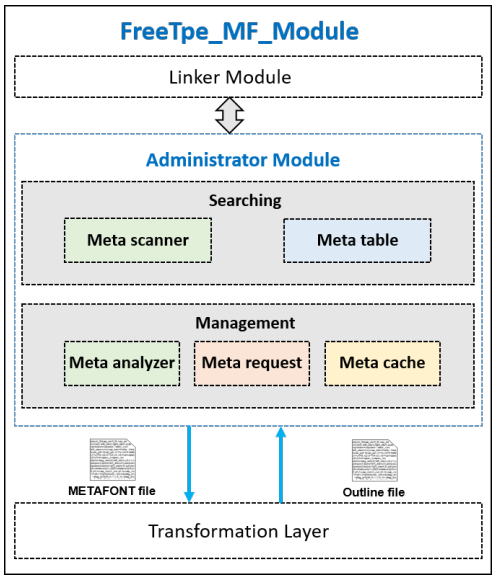


Figure 4: Administrator module

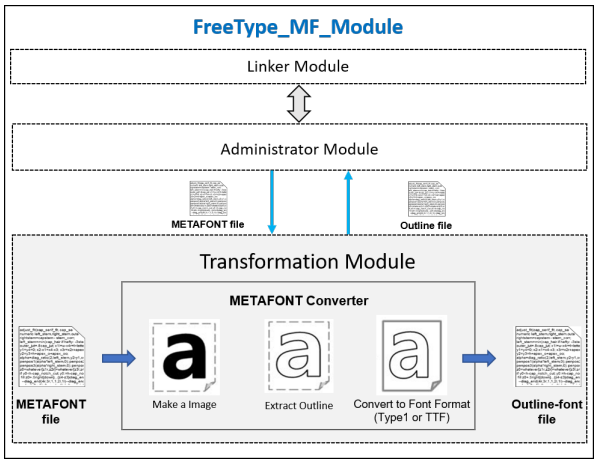


Figure 5: Transformation module

3.4 Transformation Module

Transformation module is mainly responsible for converting the METAFONT file into its corresponding outline font file. If the outline font file for a requested METAFONT file doesn't exist in the table then Administrator module sends the request to Transformation module. This module processes the request and returns the corresponding outline font file to Administrator module. Figure 5 shows how Transformation module converts METAFONT files into corresponding outline files.

3.5 METAFONT support in FreeType using FreeType_MF_Module

As shown in the Figure 6, FreeType_MF_Module is an internal module of FreeType which is responsible for processing METAFONT file request. First an application sends a font file to FreeType (step 1). If all other driver modules fail to process this font file request, this request is sent to FreeType_MF_Module through Linker module. Inner meta interface delivers this request to Administrator module (step 2). Meta request in Administrator module receives all information of this font file request and sends it Meta Analyzer to check if this font file is METAFONT or not (step 3). If this font file is not METAFONT this request is sent back to FreeType (step 3a). If this request is METAFONT file, Meta analyzer checks if this METAFONT file is installed or not by scanning Meta table. If this METAFONT information is not found in Meta table an error is sent back to FreeType internal modules(step 3b). There can be a scenario in which METAFONT is installed but its corresponding outline font is not stored in the cache. In this case, Meta cache is scanned to check if the corresponding outline file is stored in it (step 4). If it is already stored in the Meta cache with the same style parameters as requested, it is directly sent to FreeType (step 4a). If it is not stored in the Meta cache, the request is sent to Transformation layer (step 4b). Transformation layer converts the METAFONT file into its corresponding outline font by applying requested style parameters (step 5). Outline font is returned from Transformation module to administrator module where the Meta cache is updated for future re-usability (step 6). Outer Meta interface returns this outline font to core FreeType module for further processing (step 7). Lastly, FreeType renders this outline font that was made from the requested METAFONT with the styled parameter values.

The FreeType_MF_Module is perfectly compatible with the standard FreeType rasterizer. FreeType_MF_Module provides direct support of METAFONT in FreeType rasterizer just like its default Type1 driver module, TrueType driver module etc. The module manages METAFONT and its conversion to corresponding outline font. Client applications can request for any style parameters of METAFONT , FreeType_MF_Module processes them and the result can be easily displayed on the screen. As it is directly implemented inside FreeType rasterizer, it has no dependability problems as discussed in Section 2. FreeType_MF_Module can easily generate multiple font families like bold, italic, and bold-italic depending on the style parameter values passed to it.

4 Experiment and performance evaluation of FreeType_MF_Module

For the experiment, that FreeType_MF_Module can

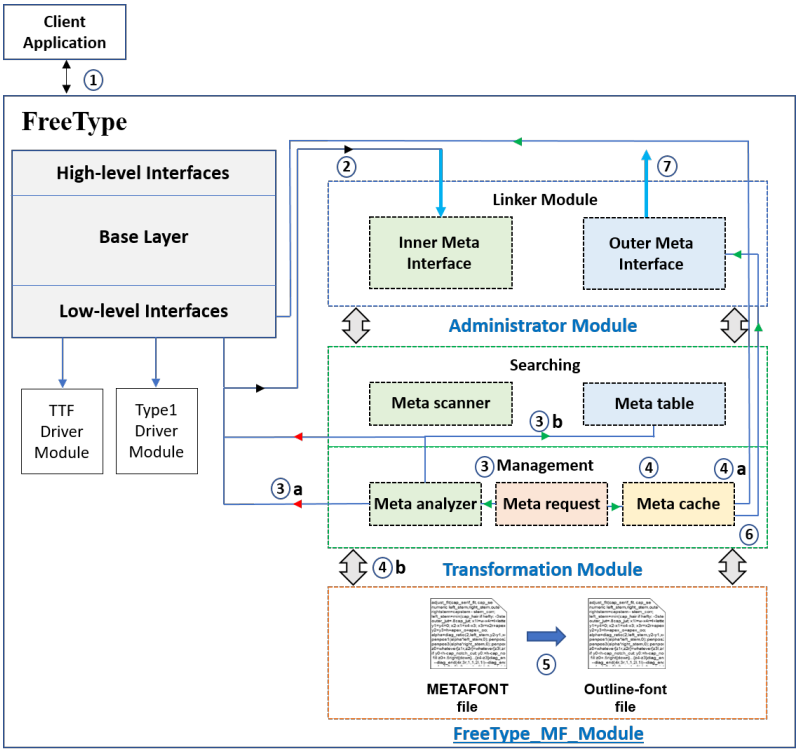


Figure 6: FreeType_MF_Module

Table 1: FreeSerif font family

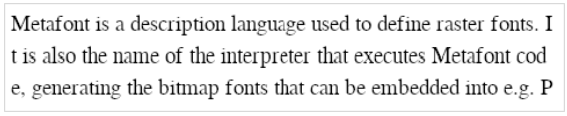
Style	Styled Output	Font files
Normal	FreeSerif	FreeSerif.ttf
Bold	**FreeSerifBold**	FreeSerifBold.ttf
Italic	*FreeSerifItalic*	FreeSerifItalic.ttf
Bold + Italic	***FreeSerifBoldItalic***	FreeSerifBoldItalic.ttf

Table 2: Various font styles with Computer Modern

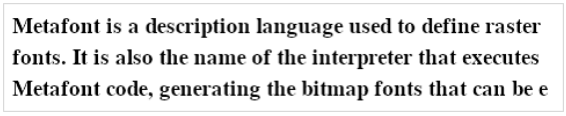
Style	Styled Output	Parameter values
Normal	ComputerModern	Default values of stem, hair, curve, slant
Bold	**ComputerModern**	stem+20, hair+20, curve+20, slant default
Italic	*ComputerModern*	Default values of stem, hair, curve, slant= 0.4
Bold + Italic	***ComputerModern***	stem+20, hair+20, curve+20, slant = 0.4

be used to generate different font styles from METAFONT the authors have used a font viewer application in Linux. This application directly uses FreeType to render fonts. It takes a font file and text as input and displays the styled text on the screen using X windows system. For testing, the authors have used all four styles of FreeSerif font family of TrueType font i.e. normal, bold, italic, bold-italic, and Computer Modern font of METAFONT.

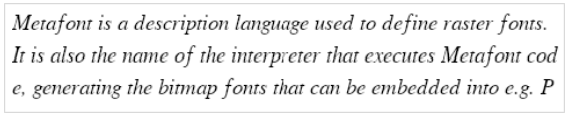
Table 1 shows the FreeSerif font family in four different styles. These styles are generated by using four different font files. Table 2 shows Computer Modern font in same four styles using different parameter values. These styles are made from one single METAFONT source file. The parameter values which are modified for generating these font styles are hair, stem, curve, and slant. The three parameters, hair, curve, and stem are related with the bold style. Incrementing their value, increases the boldness of text. These parameter values are different for lower-case and upper-case characters. The slant parameter is related with italic style. As shown in the Table 2, for normal style the default values of all these four parameters are used. For bold style, the values used are stem+20, hair+20, curve+20, and slant parameter default value. Default values of stem, hair, curve, and slant= 0.4 are used for italic style. Whereas, stem+20, hair+20, curve+20, slant = 0.4 values are used for bold-italic style. Similarly, many other font styles can be generated with this



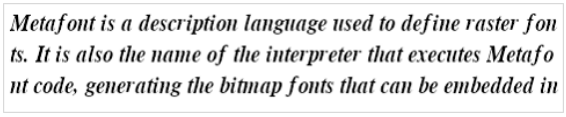
(a) FreeSerif normal style



(b) FreeSerif bold style

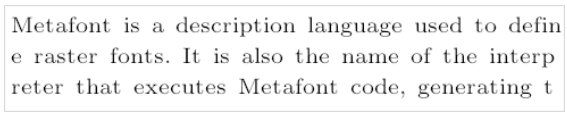


(c) FreeSerif italic style

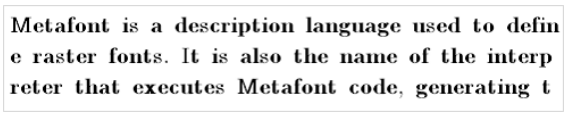


(d) FreeSerif bold-italic style

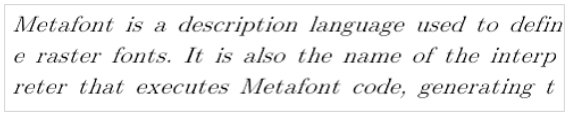
Figure 7: Dataset printed with FreeSerif font family



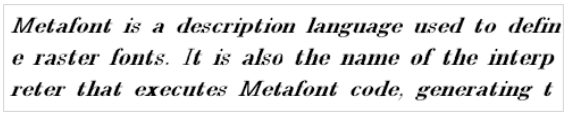
(a) default values of stem, hair, curve, slant



(b) stem+20, hair+20, curve+20, slant default



(c) default values of stem, hair, curve, slant= 0.4



(d) stem+20, hair+20, curve+20, slant = 0.4

Figure 8: Dataset printed with Computer Modern

single METAFONT source file by changing parameter values.

For the performance testing of FreeType_MF_Module compared with FreeType default driver modules and MFCONFIG module, an experiment was performed using the same font viewer application. All four font files of FreeSerif in Table 1 were used for testing TrueType driver module of FreeType, Computer Modern source file was used with four different parameter values to generate four different styles in Table 2. For the text input, a sample dataset was used, which comprised of 2,000 words and over 8000 characters, including space character. The average time per millisecond between the font style request from application and the successful display of styled text on the screen was computed and compared.

Figure 7 shows the result of printing four FreeSerif fonts and Figure 8 shows the result of four Computer Modern METAFONT fonts. Table 3 shows the average time to print the dataset with FreeSerif font by TrueType driver module, Computer Modern font by FreeType_MF_Module, and Computer Modern font with MFCONFIG module. The default TrueType driver module of FreeType takes 3 ms to 7 ms to print dataset with all four families of FreeSerif font. FreeType_MF_Module takes 4 ms to 10 ms to print this dataset with Computer Modern font. Whereas, MFCONFIG module took 50 ms to 120 ms to display similar size dataset.

The performance of FreeType_MF_Module is comparatively slower than default FreeType driver module, because it takes an extra time to convert a METAFONT into its corresponding outline font by applying styled parameters. Whereas, FreeType_MF_Module has a very good performance than MFCONFIG module, because it is directly implemented inside FreeType rasterizer just like any other internal module of FreeType and it is not dependent on any other font libraries like Fontconfig and Xft [9] etc. Hence, we can conclude that FreeType_MF_Module can be used in FreeType rasterizer for providing direct support of METAFONT in almost real time on a modern Linux PC.

FreeType_MF_Module is a suitable module to provide users with parameterized font support on the screen by applying style parameters directly to the METAFONT font. Users don't need to pre-convert METAFONT into its corresponding outline font before using with FreeType rasterizer, as FreeType_MF_Module automatically performs this. The users can use METAFONT easily just like TrueType fonts using FreeType. FreeType_MF_Module has also overcome the problems of MFCONFIG module. The performance can be further improved by optimizing the METAFONT converter in Transformation layer. Currently, the METAFONT converter works with mftrace and autotrace programs. Future work will consider

Style	TrueType driver Avg. Time	FreeType_MF_Module Avg. Time	MFCONFIG Avg. Time
Normal	4.5 ms (3-6)	6 ms (5-8)	70 ms (50-80)
Bold	4 ms (4-6)	7 ms (6-9)	85 ms (70-100)
Italic	4 ms (4-6)	6 ms (4-7)	105 ms (70-110)
Bold + Italic	5 ms (5-7)	8 ms (6-10)	100 ms (90-120)

Table 3: Average time to display dataset with TrueType driver, FreeType_MF_Module, and MFCONFIG

about proposed module optimization and direct usage of TEX bitmap fonts like GF and PK which are not supported by FreeType rasterizer.

5 Conclusion

In this paper, we have proposed a module named FreeType_MF_Module, which enables the direct support of METAFONT in FreeType rasterizer. Outline fonts such as TrueType and Type1 don't allow users to easily change font styles. For every different font style in outline fonts, a new font file is created which can be very time consuming and costly process for CJK fonts which consists of large number of characters. To overcome this disadvantage of the outline font production method, METAFONT font can be used.

FreeType supports many different font formats like TrueType, Type1, windows font etc. but doesn't provide any support for METAFONT . The proposed module, FreeType_MF_Module is installed directly inside FreeType and can be used just like other internal modules to support METAFONT font. The authors have demonstrated experiments which shows that variety of styled fonts can be generated from METAFONT by adjusting parameter values in single METAFONT source file with FreeType_MF_Module in almost real time.

Acknowledgement

This work was supported by Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korean government(MSIT) (No. R0117-17-0001, Technology Development Project for Information, Communication, and Broadcast).

References

[1] Donald E. Knuth, *Computers and Typesetting,* Volume C: The METAFONTbook. Addison-Wesley, 1996.

[2] David Turner, Robert Wilhelm, Werner Lemberg, *FreeType,* www.freetype.org.

[3] Kim Jinpyung, *et al. Basic Study of Hangeul Font.* Seoul: Korea Publishing, Research Institute, 1988.

[4] Choi Jaeyoung, *MFCONFIG: METAFONT plug-in module for Freetype rasterizer* TUG 2016 (TUGboat, 2016): 163170.

[5] K. Packard, Keith Packard, Behdad Esfahbod, *et al. Fontconfig.* www.fontconfig.org.

[6] H. Kakugawa, *VFlib - a general font library that supports multiple font formants,* EuroTEX conference, March 1998.

[7] Rokicki T, *GFtoPK version 2.3,17 April 2001* https://www.tug.org/svn/texlive/trunk/Master/texmf-dist/doc/generic/knuth/mfware-/gftopk.pdf?view=co.

[8] Menzner R, *A library for generating character bitmaps from Adobe Type 1 fonts.* www.fifi.org/doc/t1lib-dev/t1lib_doc.pdf.gz.

[9] Keith Packard, *The Xft font library:* Architecture and users guide. Proceedings of the 5th annual conference on Linux Showcase Conference, 2001. https://keithp.com/keithp/talks/xtc2001/xft.-pdf.