21st International Conference on Pattern Recognition (ICPR 2012)
November 11-15, 2012. Tsukuba, Japan

On the Performance of Decapod's Digital Font Reconstruction

Hasan S. M. Al-Khaffaf[1], Faisal Shafait[2], Michael P. Cutter[3], Thomas M. Breuel[1]
[1]*IUPR Research Group, Technische Universität Kaiserslautern, Germany*
hasansm@rhrk.uni-kl.de, tmb@uni-kl.de

[2]*Multimedia Analysis and Data Mining Competence Center,*
German Research Center for Artificial Intelligence (DFKI)
faisal.shafait@dfki.de

[3]*Computer Vision Lab, Computer Engineering Department,*
University of California at Santa Cruz
mcutter@soe.ucsc.edu

Abstract

This paper presents the current status of Decapod's English font reconstruction. The Potrace algorithm and its parameters that affect glyph shape are examined. The visual fidelity of Decapod's font reconstruction is shown and compared to Adobe ClearScan™. The font reconstruction details of the two methods are presented. The experiment demonstrates the capabilities of the two methods in reconstructing the font for a synthetic book typeset each time with one of six English fonts, three serif and three sans-serif. For both typefaces, Decapod is able to create a reusable TTF font that is embedded in the generated PDF document.

1. Introduction

Documents are displayed today on a wide variety of devices, ranging from smart phones, e-book readers, to high resolution computer displays. While most of the electronically generated documents (e-mails, web pages, PDF/A etc.) intrinsically reflow themselves to fit the screen size of the display device, scanned documents remain static due to insufficient technological support to enable reflowing or rescaling them in an elegant way. Generation of vectorized fonts from a scanned document image is an important step towards filling this gap. E-reader devices would benefit from such a flexible reflowable document format.

The research in this paper is supported by Decapod project of the Andrew W. Mellon Foundation.

Font reconstruction is a process that analyzes character shapes of raster documents and creates a customized parametric font that is visually faithful to the original [8]. Font reconstruction will make old and rare documents available for the user in searchable and reflowable digital format. Optical font recognition (OFR) research problem has been in the literature since a decade. A method based on multivariate Bayesian classifier recognizes English fonts of high quality images with 97% accuracy [10]. Another method based on global texture analysis for font recognition achieves an average 99.1% in the detection of 65 popular English and Chinese fonts [9].

Only recently, the optical font reconstruction problem has been tackled by Cutter *et al.* [5, 6] within the framework of Decapod project. Decapod is a low cost open source document digitization solution for rare and scholarly material [2, 8]. It provides a complete solution for the digitization problem including both software and hardware parts. Decapod relies on OCRopus [3] to analyze page layout and detect the text. Then, the font reconstruction uses a binned nearest-neighbor classifier [5] to partition character images into tokens. Greedy graph segmentation [6] is then applied to partition tokens into candidate fonts. The next stage, font generation explained in detail in section 2, generates the font out of the input raster tokens. The font file is then embedded in the output PDF file. Another solution for font reconstruction is provided by Adobe in Acrobat X Pro® which includes a feature called ClearScan that creates custom fonts for scanned documents and embed them in PDF file along with the OCR text. Adobe's website [1] describes ClearScan as a feature that im-

proves text quality, reduces file size, and decreases print time. However, the internal details of the font customization algorithms are not shown.

In this paper we will evaluate the fidelity of font reconstruction step of the Decapod's pipeline. Visual resemblance of Decapod's font reconstruction is compared against the original fonts and Mean Square Error (MSE) is the objective measure used in the experiment. However, other typesetting parameters like left- and right- bearings and distance between characters are not included in the evaluation.

2. Font Generation in Decapod

The input to the font reconstruction stage is one bitmap for each character class identified in the image. This bitmap can be used directly to create what is called a bitmap font. However, such fonts are not desirable since they are vulnerable to scale operation leading to unpleasant viewing experience. The right option for the reconstructed font is to use outline fonts (e.g. TrueType) where each glyph is represented by its outline curves. The parametric nature of these curves enables scaling while keeping visual fidelity to the original character shape. In order to create outline fonts the character bitmaps need to be converted into vector form. The contour of the bitmaps need to be traced and converted into curves. The Potrace algorithm proposed by Selinger [7] is used for this purpose. The algorithm consists of four steps: (i) form a path on *n* consecutive contour points; (ii) approximate each path by a polygon of straight paths; (iii) the polygon is parametrized as a smooth curve; and (iv) the curve is optimized by merging consecutive branches, if possible.

The algorithm has many parameters that can affect the output. One parameter that can affect the shape of the created character glyph is the α parameter which is used in corner detection. Let's assume that a polygon vertices are $v_0 ... v_{k-1}$. The edges before and after a vertex, let's say v_1, need to be approximated by a corner or a curve. If α is set to a value ≤ 1, a curve is drawn. Otherwise, no curve will be drawn and the two edges are intersected at the vertex v_1. Figure 1-a shows the glyph of letter 'A' with two different values of α. Setting α to a small value creates a letter with sharp edges (Left part of Fig. 1-a) while setting α too high will create a letter that is curvy. A moderate value for the parameter ($\alpha = 1$) is appropriate for the characters we have which is created at 300 dots per inch (DPI) resolution.

Another parameter that affects shape of the glyph, although in less sensible way than α, is the curve optimization tolerance O. The algorithm merge many consecutive short Bézier curves into a single Bézier curve.

(a) L: $\alpha = 0.1$, R: $\alpha = 1.9$(b) L: $O = 0.1$, R: $O = 1.8$

Figure 1: The effect of Potrace parameters on character shape. (a) The α parameter. All polygons (left) .vs all curves (right). (b) The O parameter. Few merges (left) vs. many merges of Bézier curves (right)

On the other hand, consecutive line segments are not merged so as adjacent curves who disagree in convexity. This parameter sets the error threshold for the algorithm to merge curves. Figure 1-b shows an example with two different values of O. Note the curvy right limb in right part of Fig. 1-b created due to the relaxed tolerance resulting the merge of two curves.

3. Font Reconstruction: Decapod vs. Adobe X Pro

Both solutions are capable of creating an embedded font from a given raster image. However, there are differences in how this task is performed. Table 1 shows different features considered during font reconstruction.

Table 1: Features of Decapod and ClearScan font reconstruction

Method	Font	Reusable	Ligatures	Multi g/c[†]	Embeded
Decapod	TTF	Yes[‡]	No	No	Yes
ClearScan	CFF	No	Yes	Yes	Yes

[†]Multi glyph per character class

[‡]Fully reusable if all alphabet glyphs were created

Both solutions have different approaches when processing a document typeset with only one font. In Decapod, one TTF font is created with one single glyph per character class, because Decapod aims to create one font for a document typeset with one font type. On the other hand ClearScan creates multi CFF font files with possibly multi glyphs per character class. However, each glyph in ClearScan represents many samples of the same character class. One reason for Adobe X Pro to use CFF fonts might be to support a wide range of languages and also in the flexibility of CFF fonts in handling thousands of different glyphs [4]. The other difference between the two methods is in font reusability. The aim of Decapod is to create a font with high fidelity to the original. However, another target is to reconstruct the font of the document and to make it available to complete or typeset a new document. On the other hand Adobe X Pro seems to focus on creating a font that not only reflects the document visual appearance, but also improves its quality. The third difference

is in the use of ligatures. Decapod work is based on OCR using Ocropus. Ocropus detects each character individually. Hence, we are limited to using one glyph per character. Therefore Decapod can not produce ligature glyphs. In contrary, Adobe X Pro creates ligatures and stores each as a separate glyph in the CFF font file (Fig. 2). Finally, both solutions embed the cre-

(a) ff (b) ft (c) t: (d) rl

Figure 2: ClearScan reconstructed fonts include ligatures

ated font(s) in the generated PDF file. However, Decapod creates another external copy of the font file which can be used to typeset a new document assuming all required glyphs are recognized and exist in the original document.

4. Evaluation of Font Generation

A PC running Ubuntu GNU/Linux is used to test Decapod's genpdf module while Windows Vista is the platform used to test Adobe Acrobat X Pro. The dataset is a novel of 60 pages and 170,000 letters (spaces not counted) typeset each time with one of six fonts namely Arial, Comic Sans MS, Times New Roman, Courier 10 Pitch, Liberation Serif, and Dejavu Sans Condensed. The books are then converted into PNG file format at 300DPI. The raster images are fed into the two software and the reconstructed PDF files with embedded fonts are generated. Figure 3 shows a sample output for the two methods from the book typeset with Times New Roman font. The PDF files are rasterized again and each page is segmented into lines with its corresponding detected text. Bounding boxes of all characters are also calculated. The line images, text, bounding boxes are stored in OCRopus book directory structure. Every character of the reconstructed document represented by its bitmap is compared to its corresponding rasterized font glyph to get a performance measure (MSE). Prior to the comparison, the reconstructed character image and the glyph image are normalized to same height and width. The MSE measures of all characters in the book are accumulated and each fonted book yields one final MSE score (Fig. 4). However, some lines are lost during the segmentation stage due to misclassification, preventing the calculation of MSE values for characters of missing lines. Hence, a penalty (Eq. 1) is added for any missing line.

$$P = L_{len} \times w \times [A_{char} \times E_{value}] \qquad (1)$$

Table 2: Penalty term values of Eq. 1. Where L_{len} is the average # characters/line and A_{char} is the approximate area/character (area in $pixel^2$ unit)

Font	L_{len}	A_{char}
Arial	60	48×54
Comic Sans MS	58	47×54
Times New Roman	60	42×47
Courier 10 Pitch	60	48×47
Liberation Serif	60	39×48
Dejavu Sans Condensed	61	42×58

where L_{len} is the average number of characters per text line and w is the weight of the penalty to consider for a single character. The approximate area of the character is A_{char} and the amount of error to consider for each pixel is E_{value}. The added penalty reflects the contribution of missing lines to the MSE. However, due to differences in character shapes between the different fonts, some term values of Eq. 1 differs between fonts, hence, the MSE becomes comparable between many methods on any single font, but not between different fonts. The term values of L_{len} and A_{char} of Eq. 1 used in the experiment is shown in Table 2. The other term values are as follows (w= 0.01 and E_{value}= 64^2). The final accumulated MSE's after adding the penalty are shown in Fig. 6. By using this penalty scheme, the normalized performance of the two systems becomes comparable.

5. Results and Discussion

The performance of the benchmarked methods are shown in Fig. 6 where low value indicates better visual match between original and reconstructed font. It is shown in Fig. 6 that the performance of genpdf is lower in quality than that of ClearScan for the six tested fonts. One reason for the high MSE values for genpdf compared with ClearScan is thicker glyphs in the former. An undesirable side effect of token compression is that the created token becomes a little thicker than the original character shape. The process of merging similar character images into a token is necessary to create a font that represents the document, however it has side effect of thickening the created glyph, hence raising the MSE score.

Another aspect of font reconstruction is the serif vs. sans-serif fonts. Figure 5 shows an example of three different fonts: one sans-serif (Arial) and two serifed fonts (Times New Roman and Courier 10 Pitch). Currently, Decapod is more capable of reconstructing sans-serif fonts than serif fonts due to difficulty in reconstructing the small short part of the serif.

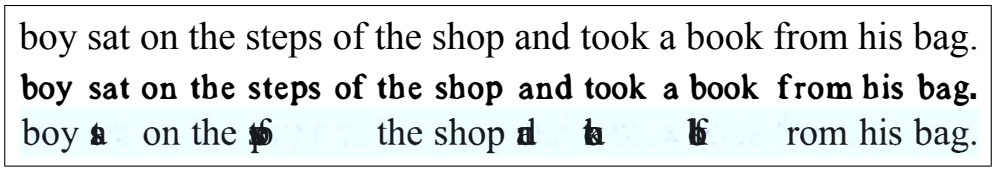

Figure 3: Sample line of Times font. Top- Original, Middle- Decapod, Bottom- ClearScan

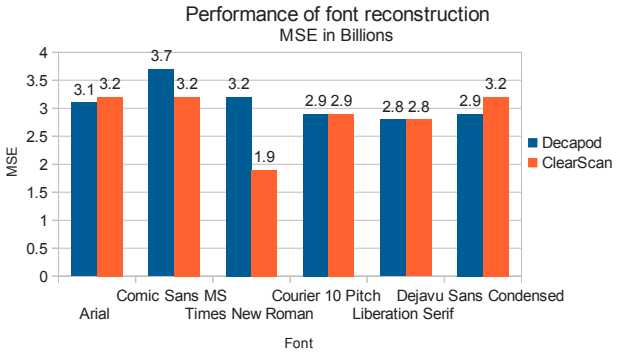

Figure 4: Font reconstruction performances on many fonts (before penalizing missing lines)

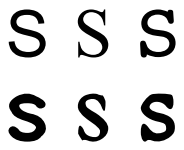

Figure 5: The letter S in Arial, Times New Roman, and Courier 10 Pitch fonts respectively. Top- Original fonts, Bottom- Decapod reconstructed fonts

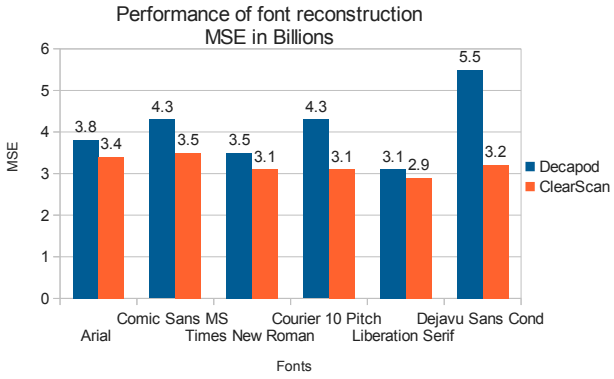

Figure 6: Font reconstruction performances on many fonts (after penalizing missing lines)

6. Conclusions

This paper presents the current status of the Decapod's font reconstruction which includes the use of Selinger Potrace method for tracing of character boundaries and the parameters affecting character shape. Decapod's font generation is also compared with Adobe ClearScan, a commercial digitization software. Our experiment on six fonts shows that the two solutions can reconstruct fonts in acceptable shape. ClearScan creates font(s) with high fidelity to the original. Decapod's current status produces fonts with readable text. However, considering that the test data is synthetic the reconstructed fonts from both software is still not perfect.

The future work for Decapod's font generation is to enhance glyph tracing by considering some morphological operations on the scaled-up bitmap of each glyph prior to tracing. A qualitative evaluation of font reconstruction is another planned future work to measure the user satisfaction with the reconstructed font.

References

[1] Adobe Blog: Better PDF OCR. ClearScan is smaller, looks better. http://blogs.adobe.com/acrolaw/2009/05/better_pdf_ocr_clearscan_is_smal/, accessed on Mar 2012.

[2] Decapod. http://sites.google.com/site/decapodproject/.

[3] The OCRopus(tm) open source document analysis and OCR system. http://code.google.com/p/ocropus/.

[4] PDF Reference, sixth edition, version 1.7, Nov 2006.

[5] M. P. Cutter, J. v. Beusekom, F. Shafait, and T. M. Breuel. Unsupervised font reconstruction based on token co-occurrence. In *Proceedings of the 10th ACM symposium on Document engineering*, DocEng'10, 143–150, New York, NY, USA, 2010. ACM.

[6] M. P. Cutter, J. van Beusekom, F. Shafait, and T. M. Breuel. Font group identification using reconstructed fonts. In G. Agam and C. Viard-Gaudin, editors, *DRR*, volume 7874 of *SPIE Proceedings*, 1–10. SPIE, 2011.

[7] P. Selinger. Potrace - transforming bitmaps into vector graphics. http://potrace.sourceforge.net/, accessed on Mar 2012.

[8] F. Shafait, M. P. Cutter, J. van Beusekom, S. S. Bukhari, and T. Breuel. Decapod: A flexible, low cost digitization solution for small and medium archives. In *4th International Workshop on Camera-Based Document Analysis and Recognition*, Lecture Notes in Computer Science. Springer, Sep 2011.

[9] Y. Zhu, T. Tan, and Y. Wang. Font recognition based on global texture analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(10):1192–1200, Oct 2001.

[10] A. Zramdini and R. Ingold. Optical font recognition using typographical features. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(8):877–882, Aug 1998.