Glyph Extraction from Historic Document Images

Lothar Meyer-Lerbs
lml@tzi.de

Arne Schuldt
as@tzi.de

Björn Gottfried
bg@tzi.de

Center for Computing and Communication Technologies (TZI)
University of Bremen, Am Fallturm 1, 28359 Bremen, Germany

ABSTRACT

This paper is about the reproduction of ancient texts with vectorised fonts. While for OCR only recognition rates count, a reproduction process does not necessarily require the recognition of characters. Our system aims at extracting all characters from printed historic documents without the employment of knowledge of language, font, or writing system. It searches for the best prototypes and creates a document-specific font from these glyphs. To reach this goal, many common OCR preprocessing steps are no longer adequate. We describe the necessary changes of our system that deals particularly with documents typeset in Fraktur. On the one hand, algorithms are described that extract glyphs accurately for the purpose of precise reproduction. On the other hand, classification results of extracted Fraktur glyphs are presented for different shape descriptors.

Categories and Subject Descriptors

I.4.3 [**Image Processing and Computer Vision**]: Enhancement; I.5.3 [**Pattern Recognition**]: Clustering

General Terms

Algorithms, Experimentation

Keywords

Image enhancement, glyph extraction, document-specific font, glyph shape, glyph classification

1. INTRODUCTION

Old books display a huge variety of degradations, layouts, and typographic styles. They use nowadays unfamiliar ligatures, abbreviations, and special printer marks typeset in fonts that spur the field of paleography to decipher the ancient writing — we can hardly hope for OCR of these documents that, in addition, have all kinds of spelling variations using words that can not be found in current dictionaries. That is why we adopt the approach of [11], and focus on the extraction of glyphs from historic printed document images. These will be turned into vectorised fonts, using a document-specific encoding. The vectorisation will be based on grey-scale prototype glyphs, clustered to favor new prototypes over possible mixing of different glyphs. The best glyphs will be averaged into prototypes which replace all degraded versions. This results in improved compression [3] and sets it apart from OCR.

The paper is organised as follows. Section 2 discusses common preprocessing steps and describes our changes to extract better glyphs. Section 3 presents first classification results of the clustering of binarised glyphs to select suitable features. Conclusions are made in Section 4.

DocEng2010, September 21–24, 2010, Manchester, United Kingdom.

2. GLYPH EXTRACTION VS. OCR

We wish to extract document-specific fonts from historic document images. These fonts need size, style, and kerning information as well as subtle character details to be successful. In contrast to the basic goal of OCR — identification of all glyphs — which would allow the construction of fonts usable with current computer typesetting systems, our prototype system 'Venod' will assign Unicode codepoints to prototype glyphs from the Unicode 'private use area' and encode the generated fonts with unidentified glyphs. This allows the reflowing of text and high speed text searches from examples – in essence a fast form of word spotting.

Ledible and beautiful fonts require the extraction of many significant details from document images. Therefore, simple binarisation does not suffice. Following the binarisation introduced by [1] we compare and select appropriate preprocessing steps. Their binarisation proceeds from an input grey-scale source image via

1. a Wiener filter to a denoised grey-scale image I,
2. adaptive thresholding by Sauvola's local binarisation to a black and white initial segmentation image S,
3. interpolation of the non-text part to a grey-scale background surface B,
4. final thresholding by examining pixel contrast to a b/w image T, which might be upsampled during its creation and is called U here,
5. postprocessing by shrink and swell filtering to the final binarised (upsampled) image, called F here.

We follow these steps one by one and compare possibilities and changes on the way.

Figure 1: Noise filtering results after processing the original image, each row presents the results of a different noise filter and their window size.

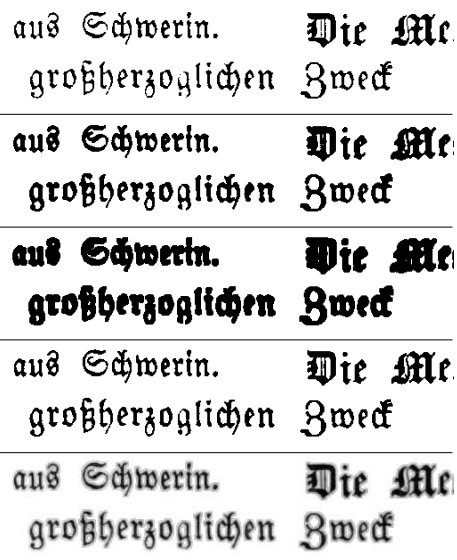

Figure 2: Binarised versions top four rows, bottom row: enlarged, up to distance 2, connected components of S cut from I – our intended glyph vectorisation source.

2.1 Denoising

In their approach, [1] always use a Wiener filter [6, 8] as the first preprocessing step to denoise the input image. This will introduce Gaussian blur and diminish font details. Instead, edge preserving noise filters [5] like the Sigma filter by [7] or anisotropic diffusion filters introduced by [10] promise better results. Since diffusion filters introduce Gaussian blur at some iteration, our experiments lead us to use just four iterations with decreasing parameters $k = 32, 24, 16, 8$, corresponding to gradient differences influencing the filter.

The denoising results shown in Fig. 1 recommend the Sigma filter for our purpose. It keeps all glyph edges and eliminates the background noise.

2.2 Preliminary Binarisation

A first estimation of glyph positions is given by a rough binarisation. Therefore several algorithms and their parameterisation are compared; Fig. 2 shows some examples. The results indicate that global methods like Otsu's [9] have significant problems with noisy, unevenly illuminated images and need to be replaced by locally adaptive methods like the one by [13], which can be implemented to run fast with any window size [15]. The window size should include more than one glyph and was set to 40×40 in the experiments presented here.

2.3 Background Estimation

The approach by [1] interpolates the gaps left after removing the foreground pixels by averaging a rectangular background area, skipping foreground pixels along the way. This tends to leave a dark one-pixel halo around filled areas. We found out experimentally that by dilating the foreground image with a 3×3 black structure element, most of these artefacts can be removed (Fig. 4).

Since dilation will connect previously unconnected components and their later separation is prone to errors we devised an alternative: find all 8-connected components in S, label each one and create a Euclidean Distance Map (EDM) [12] around them. Thresholding this map at distance 2 allows us to enlarge every component by up to two pixels unless it overlaps other components, see the bottom row of Fig. 2. To test a different background filling method, we chose bilinear interpolation to fill the gaps. Our results indicate that the filling procedure is less important than halo removal (bottom rows of Fig. 4).

In [4], glyph edges are modeled with an exponential decay function that usually decays within two pixels from the edge. As our results show (Fig. 4 and bottom row of Fig. 2), all relevant boundary information is preserved. Using this enlarged component mask for every component we have also found grey-scale glyph components with sufficient many border pixels for grey-scale vectorisation.

2.4 Final Thresholding

Most grey-scale images will contain enough information for an upsampling in both directions by a factor of 2; see the top row of Fig. 3. Depending on the intended use, this step might improve later results or OCR. Since we intend to cluster the grey-scale glyphs, upsampling should be done after the prototypes have been found.

2.5 Postprocessing

A shrink filter to eliminate noise from the background is employed by [1]. With the suggested parameters, it would remove some isolated foreground pixels; but our experiments show no effect on the given test material. Then, a first swell filter, supposed to 'fill possible breaks, gaps or holes in the foreground', and a second one 'used to improve the quality of the character strokes' is applied by [1].

From our results (see the bottom rows of Fig. 3), we conclude that no swell filtering should be applied when glyphs need to be extracted. The entire postprocessing step is therefore superfluous for our application.

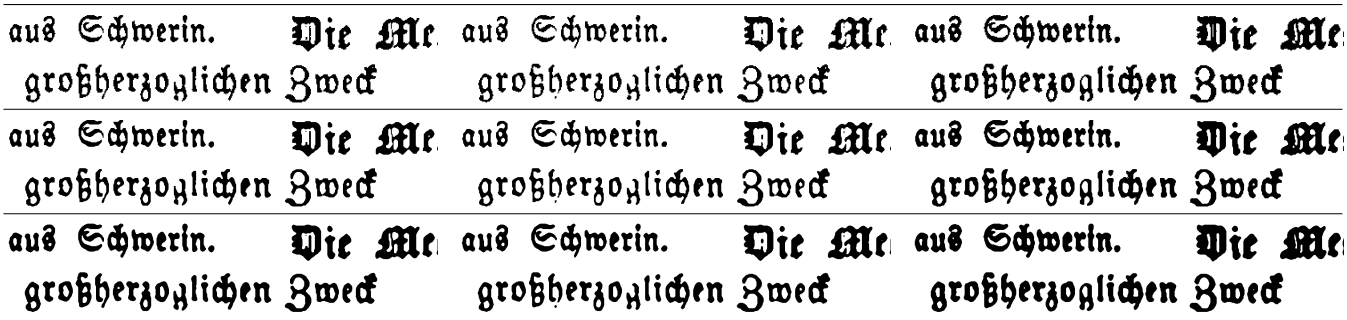

Figure 3: Binarised upsampled difference to interpolated background, and two swell filter applications for the final result. Top row: upsampled binarisation without postprocessing, U; middle row: first swell filter applied, bottom row: second swell filter applied, F. First column based on the original process, second column with differing Sauvola binarisation and the third column with a dilation added after the initial binarisation. The top right result shows the fewest broken characters.

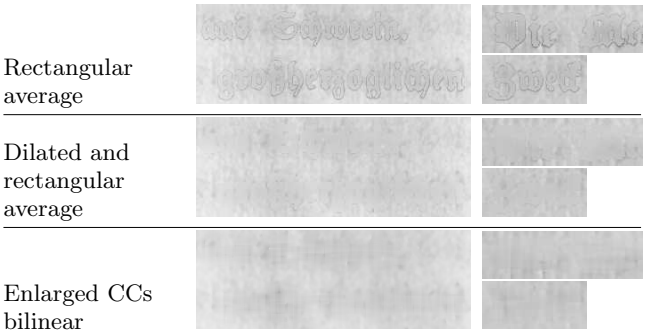

Figure 4: Interpolated background images B. Top row: I without S interpolated from a rectangular area leaving a halo; middle row: using a dilated version of S, eliminating the halo; bottom row: I without enlarged connected components from S after bilinear interpolation, also halofree.

3. GLYPH CLASSIFICATION

While many different classification methods exist, a fundamental constraint is efficiency. This is particularly important when thinking about mass digitisation workflows of which the presented classification would be an essential part of. Suitable features for classifying binarised glyphs should therefore enable fast comparisons. This is hardly possible by defining complex templates that describe shapes in a sophisticated and detailed way. By contrast, the most compact features characterise shapes by means of single numeric values. Textbook examples include the compactness of a glyph, its radius ratio, aspect ratio, and convexity. Comparisons based on such features stick to a constant runtime complexity, since they describe shapes independently from the number of components, which might either be contour points or all points contained within a shape. It is therefore worthwhile to investigate whether such features lead to sufficient classification precisions. In addition, another established and effective method are Hu moments [2] which are equally compact like the aforementioned features. What all those features have in common is that they are invariant with respect to translation, scale, and rotation.

3.1 The Scope Histogram

In order to improve the performance of those established features, further shape characteristics of the same runtime complexity are investigated that would complement the other features. This is guaranteed by introducing a new system of shape properties as follows. Instead of using contour points, shapes are approximated by straight segments which frequently represent a shape much more compactly, since many glyphs contain straight segments: just look at the letters of the present text as well as those contained in the figures.

Then, shapes are described with respect to single glyph segments: the shape of a glyph extends over a specific range of each segment. This range is referred to as a segment's *scope* that can be succinctly described as to be *left-of* a segment, *right-of* it, *on top*, *below*, and by some further directions which can be combined to assemble to many different scopes. Counting their frequencies for each segment of a glyph, the *scope histogram* [14] is obtained. It describes the shape of a glyph in a significantly different way than conventional methods; this is the reason why it should be analysed whether classification results can be improved when adding scope histograms to Hu moments and the other features.

3.2 Evaluation Experiment and Results

In order to systematically evaluate the different features, a set of twenty glyph classes has been compiled as a ground truth, each class containing twenty instances making a total of 400 glyphs. The examples are taken from one and the same document page for which a specific font would be generated; this is the reason for why this experiment is confined to twenty classes — not all characters are available on that page with a significant number of exemplars for the present evaluation; in the final application accordingly more classes would be generated for font generation, depending on the present characters. The glyphs differ in particular regarding specific indentations in their contours. Therefore, for each glyph the outer contour has been extracted, emphasising the curvature of the outer contour, neglecting interior shape information.

A recall-and-precision evaluation would explain how the precision degrades with regard to the correct recall. This provides fundamental information about the performance of features, and thus, being fundamental for learning how the

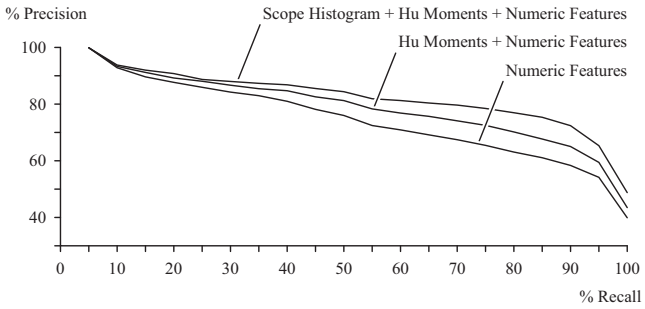

Figure 5: Recall-and-precision graphs for glyph retrieval with combinations of numeric features, Hu moments, and scope histogram.

features behave within the process of document individual font generation.

Measuring the performance of all single numeric shape features together, namely of the compactness, radius ratio, aspect ratio, and shape convexity, the lower graph in Fig. 5 shows their recall-and-precision behaviour. The middle graph shows how the performance is improved when additionally considering the Hu moments. Moments obviously add information to the other features. This is not surprising when looking, for example, at the aspect ratio in comparison to how moments are defined for the entire set of points contained in a glyph.

A remarkable result is that the performance can still be improved by employing the scope histogram. Its definition takes into account how parts of the glyph surround other parts of that same glyph, computing more directly how the information contained in the outer contour contains relevant shape information. This is clearly different to what the Hu moments represent, explaining the results. To conclude from this experiment, glyphs can actually be categorised with high precision even with compact shape descriptors. Due to their low computational complexity, these features are thus well-suited for large-scale categorisation.

Further improvements are expected by considering the orientation variance of letters which is neglected by all of the features used in the current classification. Additionally, we aim at looking at interior contours of holes, entailing the consideration of more distinctive glyph properties which are, in our evaluation, solely taken into account by Hu's approach.

4. CONCLUSIONS

Our experiments show that, to extract accurate glyph prototypes from printed document images, even the binarisation proposed by [1] must be enhanced. Not-eight-connected binarised pixels cannot be discarded since they might represent broken parts of a glyph. Enlarging the binarised regions, up to a distance of two pixels, will include most information of character edges and might be used as the basis for better binarisation or to cut out grey-scale glyph areas ready for font clustering. First results of the latter indicate that shape features can be employed which enable fast comparisons for huge amounts of glyphs in voluminous documents. The scope histogram has been applied the first time to the domain of glyphs, showing that qualitative features do in fact improve conventional methods.

Our future work will include an interactive system to optimise character clusters, matching of existing glyphs to separate touching character groups and the production of usable fonts, which will result in OCR of most of the texts.

5. ACKNOWLEDGMENTS

This work was partially funded by the DFG (Deutsche Forschungsgemeinschaft), project Venod (HE 989/10-1). The Staats- und Universitätsbibliothek Bremen and the Bayerische Staatsbibliothek in Munich provided many fine examples of historic printed document images.

6. REFERENCES

[1] B. Gatos, I. Pratikakis, and S. J. Perantonis. Adaptive degraded document image binarization. *Pattern Recognition*, 39(3):317–327, 2006.

[2] M.-K. Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8(2):179–187, Feb. 1962.

[3] S. Inglis and I. H. Witten. Compression-based template matching. In *DCC 1994*, pages 106–115, Snowbird, UT, USA, 1994. IEEE Computer Society.

[4] T. Kanungo, R. M. Haralick, and I. Phillips. Global and local document degradation models. In *ICDAR 1993*, pages 730–734, Tsukuba, Japan, 1993. IEEE Computer Society.

[5] S. Kröner and G. Ramponi. Edge preserving noise smoothing with an optimized cubic filter. In *COST-254 Workshop*, pages 3–6, Ljubljana, Slovenia, 1998.

[6] J.-S. Lee. Digital image enhancement and noise filtering by use of local statistics. *IEEE PAMI*, 2(2):165–168, Mar. 1980.

[7] J.-S. Lee. Digital image smoothing and the sigma filter. *Computer Vision, Graphics, and Image Processing*, 24(2):255–269, 1983.

[8] J. S. Lim. *Two-Dimensional Signal and Image Processing*. Prentice Hall, Englewood Cliffs, NJ, USA, 1990.

[9] N. Otsu. A threshold selection method from gray-level histograms. *IEEE SMC*, 9(1):62–66, Jan. 1979.

[10] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE PAMI*, 12(7):629–639, July 1990.

[11] S. Pletschacher. A self-adaptive method for extraction of document-specific alphabets. In *ICDAR 2009*, pages 656–660, Barcelona, Spain, 2009. IEEE Computer Society.

[12] J. C. Russ. *The Image Processing Handbook*. CRC Press, Boca Raton, FL, USA, 2nd edition, 1995.

[13] J. Sauvola and M. Pietikäinen. Adaptive document image binarization. *Pattern Recognition*, 33(2):225–236, 2000.

[14] A. Schuldt, B. Gottfried, and O. Herzog. Towards the visualisation of shape features: The scope histogram. In *KI 2006*, pages 289–301, Bremen, Germany, 2006. Springer-Verlag.

[15] F. Shafait, D. Keysers, and T. M. Breuel. Efficient implementation of local adaptive thresholding techniques using integral images. In *DRR 2008*, San Jose, CA, USA, 2008. SPIE.