Representation of Digitized Documents Using Document Specific Alphabets and Fonts

Stefan Pletschacher; Chemnitz University of Technology, Institute for Print and Media Technology; Chemnitz, Germany

Abstract

Today's digitization efforts lead to huge collections of scanned documents. However, the means for automatic preparation and further processing especially of ancient documents are still limited. In this paper, progress and implementation details of a framework for handling machine printed documents without traditional OCR-methods are shown. The approach is based on deriving any information needed for encoding directly from the original itself. This is achieved by extracting document specific alphabets and corresponding fonts. In particular, it is reported on how preprocessing, text segmentation, alphabet extraction, font generation, document encoding, as well as the repository work and interact. Moreover, the creation of ground truth data for evaluation and possible application scenarios for the system are shown.

Introduction

Encoding is an essential task for efficient storage and processing of digitized documents. This is especially true for originals containing predominantly textual contents. Usually the transition from bulky and not directly accessible bitmap images to encoded text is accomplished by means of automatic methods like OCR (Optical Character Recognition) or, in the worst case, through manual transcription. However, there are many scenarios where either of the before mentioned approaches are not appropriate. Huge collections of historical documents, for instance, can neither be dealt with manually nor converted by OCR due to special scripts and unknown type faces. Hence, it is desirable to have alternative means for automated processing of such documents without any restrictions on the contained alphabets, scripts or fonts. Moreover, for many academic disciplines, like arts, it is important to have access not only to contained text of source documents, but also to its original appearance. This includes the reproduction of original fonts and special symbols, but also broken characters or mistakes in writing.

In the following, an alternative approach and a purpose-built system for handling digitized machine printed documents which are not suitable for current OCR-methods are presented. In particular, it is shown how documents of that kind may be individually encoded and subsequently processed alike conventional text. Furthermore, a way for preserving the original appearance by generating document specific fonts is elaborated. Progress and first implementation details of the prototype system proposed in [1] are shown.

Background and Related Work

Up to now, there are mainly two ways of handling digitized documents. The straightforward and inexpensive manner is storing and outputting digital facsimiles in form of bitmap images without any analysis or preparation of contents. Unfortunately, the involved formats are very unhandy and offer only few possibilities for further processing. The more sophisticated way requires first of all recognition of contained text by means of OCR. Drawback of this procedure is, however, that for a broad spectrum of documents such methods are not applicable as outlined before.

To overcome these limitations, a new approach without true recognition is needed. The main idea at this point is not to rely on any prior knowledge or predefined alphabets for encoding but to extract all necessary information directly from the present document itself. For putting this into practice, a document specific alphabet has to be extracted from the original. This is achieved by clustering all occurrences of characters into classes and then representing each class by just one prototype [2]. Once this document specific alphabet has been determined, the original may be encoded by replacing all individual characters with a reference to their particular prototype. This follows the basic idea of symbolic compression [3].

Only a few systems are pursuing this general approach (e.g. [4]). However, the presented work goes one step further by generating a corresponding vector font along with the document specific alphabet. To obtain such a generic font, each prototype has to be transformed into a path description using a customized vectorization method that preserves important glyph details [5]. Result is a SVG-font (Scalable Vector Graphics) which can be directly integrated in XML-based (eXtensible Markup Language) descriptions of encoded documents [6, 7]. This representation, based on a document specific alphabet and font, may be easily processed, transformed and output on various platforms using standard XML-technologies.

System and Implemented Methods

According to the necessary processing steps for converting scanned documents into encoded representations the system is organized into distinct modules. Namely, these modules include preprocessing, text segmentation, alphabet extraction, font generation, document encoding, and a repository which also comprises means for transforming encoded documents into several output formats.

Each of them is working self-contained and provides all essential functionalities for the particular task. The modules are coupled via the repository to exchange data and results. This is transparent to the user since all components are integrated into a common user interface. The coupling makes it easy to substitute single system parts for testing of new implementations or if more specialized and better performing tools become available in the future.

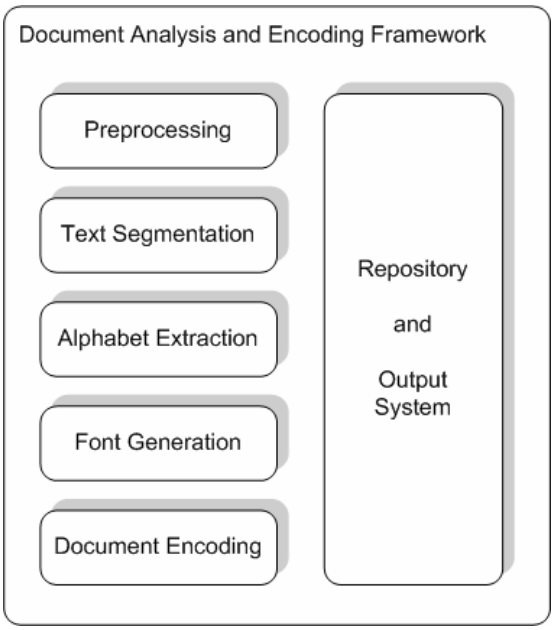

***Figure 1**. Modules of the implemented framework*

The information flow within the framework can be described as follows: Scanned documents which constitute the original work are fed into the repository and first registered. Afterwards, the single preparation tasks can be automatically triggered as a workflow or be manually performed. Since a main goal of the system is a high degree of automation each workflow step can be carried out using automatic estimation of appropriate parameters for the actual method. Nonetheless, it is possible for the operator to monitor each process step and if necessary go back and adjust used parameters.

Once all document pages are registered the preprocessing can be executed to obtain all relevant text blocks for further preparation. Other document parts like photographs or drawings are dealt with separately. Text segmentation detects the logical structure of lines, words, and characters and delivers the particular glyph images. These are to be further analyzed for ascertaining distinct classes and to create one prototype for each class. Subsequently a vector font is created for this document specific alphabet. Finally, the original document is encoded using this alphabet and font and stored in the repository for later use.

Preprocessing

The term preprocessing usually refers to more or less sophisticated image optimization methods. At this point, however, it stands for all preparation steps which are necessary to obtain image parts representing textual content.

First of all, common image enhancement procedures like deskewing and denoising are carried out. Next, each page is segmented into coherent regions. This is achieved using a modified run-length-smearing algorithm. Once separate regions were found, their type has to be determined in order to decide on how to treat them further. This means distinguishing between text parts and other elements like figures. The discrimination is handled by a specifically trained fuzzy-classifier which operates on features extracted from the single image parts. Important features are among others color gamut respectively grey level range (depending on the input format) and occurrence frequencies of specific textures. Eventually, the classifier delivers a confidence value whether the element in question represents textual content. In workflow mode, if a certain confidence threshold is not met this particular image part is treated as a non-text object and will not be further processed.

Recognized text objects are then binarized and routed to the text segmentation module. Up to now, the text analysis methods operate only on black and white images. Though, the system is currently extended to support grey level images as well.

Text Segmentation

Text segmentation is potentially a very complicated task, especially for scripts with touching characters. However, the presented system is mainly intended for processing machine printed documents containing ancient or newer European type faces from which characters and words can be extracted without extensive character recognition. Nevertheless, it is planned to integrate a feedback between prototype extraction and text segmentation to achieve more accurate results within a second segmentation pass. Due to the modular system design it is also possible to use specialized external tools, though the integrated methods perform quite well on a broad spectrum of type faces. Besides, it is always possible to correct a text segmentation result manually or to do it completely by hand (Figure 2.), which might be only an option for small documents and not for large collections.

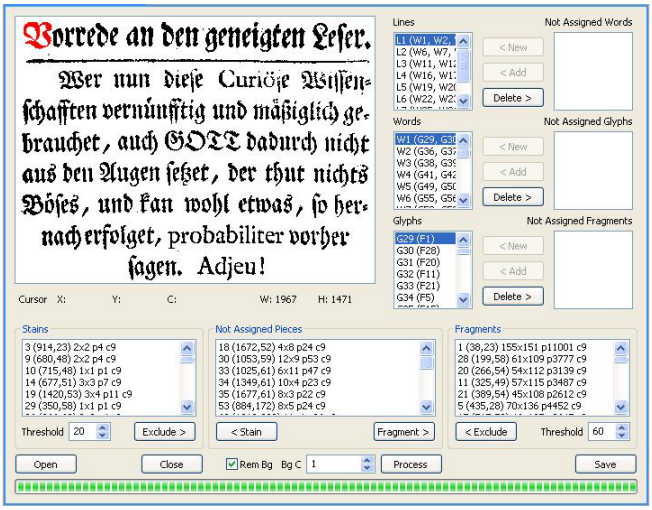

***Figure 2**. Text segmentation module*

The first step is to ascertain so called fragments by means of a region growing algorithm. Fragments can be whole glyphs for single characters, glyph parts (e.g. diacritical marks) but also multiple touching characters or ligatures. Following, the mapping of fragments into glyphs, glyphs into words and words into lines needs to be determined. This is done by a rule based method evaluating geometric properties and relative positions to decide whether elements have to be merged or split.

Text segmentation also serves as a basic structure and layout analysis. Positions and sizes of elements are stored for later document encoding

It is worthy of mention that the over-all system is quite robust to segmentation errors. This is due to the fact that an inadequately segmented glyph would rather result in a new prototype during alphabet extraction than being wrongly assigned.

Alphabet Extraction

Segmented glyph images constitute the input for the alphabet extraction module which is the crucial component of the document analysis framework. Its task is to cluster all elements into distinguishable classes of similar glyphs. There are especially two characteristics which demand a particular kind of clustering algorithm. The first is the potentially high amount of glyph instances. Millions of glyphs may arise when whole books are analyzed. Therefore, an iterative processing is required. The second is the number of true classes being unknown at first. Hence, methods which need an estimate of the cluster count in advance can be ruled out. These restrictions led to the implementation of a customized clustering algorithm based on vector quantization.

The algorithm is running over all glyphs and creates a new cluster each time the maximum similarity of the current glyph compared to all existing clusters drops below a certain threshold. Otherwise, the present glyph is merged into the cluster with the highest similarity value. The similarity is calculated by means of pattern matching. To consider not only the total amount of differing pixels but also details of the glyph's shapes a special weighted method has been implemented.

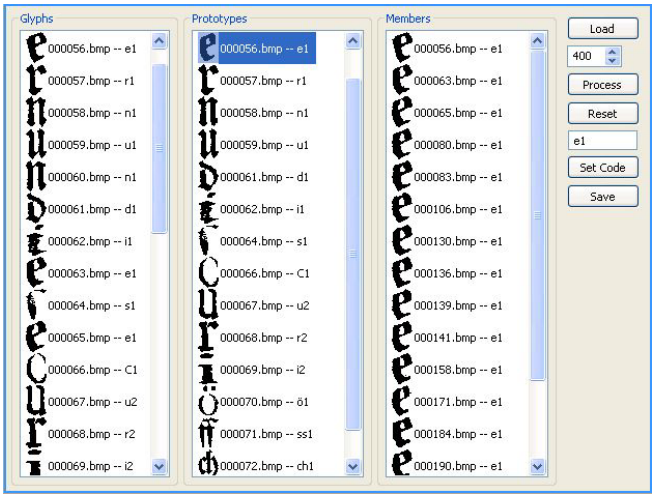

***Figure 3**. Alphabet extraction module: Left – incoming glyphs, middle – extracted prototype alphabet, right – glyphs represented by the selected prototype*

Every time a new glyph is merged into an existing cluster the particular prototype as representative for this class needs to be updated. This causes prototypes to slightly change with a growing number of represented glyphs. The actual prototype bitmap is determined based on the density of each pixel contributed by all represented glyphs. Hence, distortions and noise occurring only in few glyphs will be effectively suppressed.

In order to enable automated workflows, the clustering was extended towards an adaptive method which is capable of calculating the appropriate threshold depending on the given input documents. The key to this functionality lies in observing the growth of the maximum cluster distances. This will be elaborated in further publications.

Font Generation

Encoding and reproduction of originals is already possible using the created alphabet together with the particular prototype bitmaps. However, such a bitmap font would limit further use scenarios like reformatting and repurposing of contents. Therefore, the framework includes a component for glyph vectorization and thus generation of a corresponding vector font.

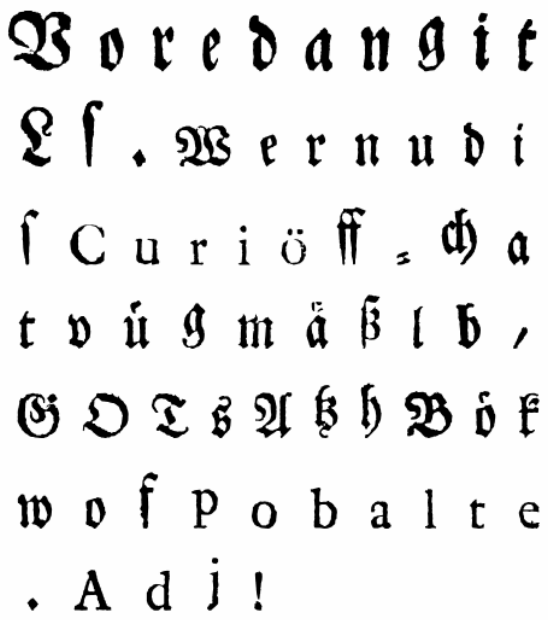

***Figure 4**. Rendered font for a document specific alphabet*

Core of the font generation module is a polygonal approximation method to achieve the raster to vector conversation. The algorithm searches for the simplest path description which still matches the glyph contour within a given maximum deviation.

The algorithm has been extended for automatically estimating the fitting deviation parameter depending on the complexity of the current glyph. This is done by an integrated quality assessment of vectorization results based on a ground truth comparison. As quality measure serves an error value determined by a weighted pattern matching which pays special attention to glyph details.

Encoding

Having the document specific alphabet and font available, the last preparation step is the creation of encoded facsimiles of the originals. Central format for this task is XML. The system uses a customized XML-Schema which is based on existing formats and defines all elements together with their structure.

Prerequisite is an individual code point being assigned to each prototype. For compatibility with standardized XML processing tools, Unicode [8] is used for this purpose. However, since this system does not conduct real character recognition, these code points have to be chosen in a manner that no predefined characters are overwritten. This is achieved by taking code points from the private use areas of the Unicode standard. References to the document specific alphabet can therefore coexist with any other Unicode text and be uniformly processed. The encoded document is assembled by putting the corresponding code point for each glyph into the structure obtained during text segmentation.

For later output the generated font needs to be preserved and is hence stored together with the encoded document. The best way for direct integration of font descriptions into XML-instances is SVG as a standardized syntax for vector graphics. The font generation module did already deliver the necessary path descriptions of all prototypes which are now used to set up the SVG-font within the definition part of the final XML-document.

Repository and Output System

The repository module serves as the central turntable for incoming documents, any kind of process data and intermediate results as well as for delivering prepared documents. It comprises the workflow engine which keeps track of jobs and triggers the single processes. The other components are coupled to the repository in order to receive their input data and to return results back to the system. As all interim results are stored along with originals and final documents it is easily feasible to make use of external tools for any of the delineated processing steps. The only requirement is that the program in question can be controlled by the workflow engine (e.g. by command line).

For utilizing prepared documents there are some output functionalities which are currently under development and will be prospectively moved to a separate module. Output transformations are performed using XSL (eXtensible Stylesheet Language) style sheets containing specific rules for different output channels like WWW, print and mobile devices. The benefits of encoded document representations are now becoming clearly evident as it is easily possible to use the content in various ways. New layouts with different text flows can be achieved using common XSL-transformations.

Discussion

The evaluation of the over-all system is in the early stage as parts of the framework are still under development and further optimized. Moreover, the production of ground truth data to measure performance values and characteristics of the implemented methods is a costly task. For significant tests huge amounts of samples are necessary. This concerns especially the alphabet extraction for which correctly segmented glyph images with known class membership are needed.

To overcome this issue two approaches for creating test samples have been put into practice. The first is manual utilization of real document images, though, aided by the system. Classes are suggested by the alphabet extraction module and afterwards the operator has to go through all glyphs and check for correct affiliations. The advantage of this procedure is the output in form of real-world examples. Yet, the manual verification is a time consuming task. The second approach is capable of producing large amounts of test samples instantly. However, this is achieved by generating them synthetically. For this purpose existing texts or arbitrary character sequences with known codes are fed to a raster image processor which creates the actual glyph images. Noise and distortions can be added to simulate fluctuations of print processes, paper defects or scanner artifacts.

A comprehensive test set with ground truth data is currently under construction and used for an extensive evaluation of the system. This material can later be used for comparison with traditional OCR-methods as well.

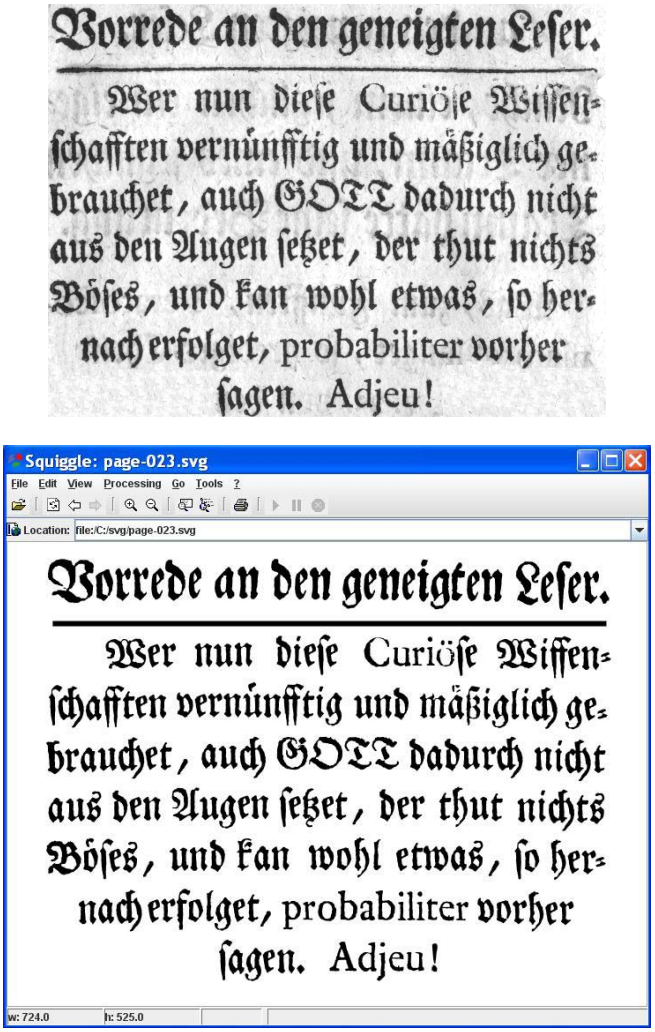

***Figure 5**. Original bitmap (top) and produced output of the encoded document (below)*

First results have already shown that the framework is well capable of handling digitized documents with complex scripts and ancient fonts. Figure 5. shows the reproduction of a text block out of a book from the 18th century using the document specific alphabet and font. It appears that similar glyphs from the original have been replaced by just one prototype. However, since the system has a bias towards accuracy rather than compression, it is also possible that several prototypes occur for semantically equal glyphs if they show a different style. This effect can be observed

comparing the black letter “a” and the roman-face “a”. In the document specific alphabet they form distinct characters. This makes sense as the different layout was used to express a special meaning, in the present case to distinguish between ordinary text and specialist terms.

When evaluating the results of the alphabet extraction, there are two different types of possible misclassifications which need to be considered. The first type arises when a glyph is merged into a wrong cluster. This is a fatal error and would cause falsification of the content. The second is less severe and emerges when a glyph is erroneously put into a new cluster instead of being merged into its already existing true class. Whereas the first is to be avoided by all means, the second causes only redundancy and can be tolerated up to a certain degree. Regardless, this would prevent the ideal compression of the encoded document. Ideal compression can only be reached if each true class of the original is represented by exactly one class in the extracted alphabet. Thus, there is a trade-off between the demanded accuracy and the achievable compression rate. Task of the adaptive clustering method is therefore to find the optimum working point which favors accuracy and still obtains a good compression.

The described framework can be employed in numerous use scenarios. It facilitates the implementation of infrastructures for efficiently storing and providing digitized documents as true to the original as possible. In this way, printed material of archives and libraries can be inexpensively integrated into recent information systems and used via internet. Furthermore, new output channels like mobile devices are becoming available as contents can be reformatted and presented in new ways.

Existing OCR-based systems could be extended to support the encoding of documents with unknown type faces. As a matter of fact, the delineated system can also be used for semi-automatic character recognition. The effort for transcribing texts is dramatically reduced as the alphabet extraction module delivers a condensed set of prototypes. Such a tool is already in use for the generation of test samples as described above.

Conclusion and Future Work

Today, a lot of effort is spent on digitizing huge archives and collections of printed originals. However, the means for further utilization of scanned documents are still limited while repositories are growing rapidly. This is especially true for the preparation of historical documents or originals with complex scripts and type faces which can not be processed using traditional OCR-methods.

The presented system makes a contribution to solve this problem by implementing a pragmatic approach. If the content of document images can not be automatically recognized, it can at least be encoded and treated alike text. This enables further application and reuse of digitized documents even if the true meaning is still unknown. Furthermore, documents can be stored in a compressed form, preserving the appearance as true to the original as possible.

The used approach is very robust and not prone to severe classification errors like substitutes in OCR-systems. This is achieved at the cost of a certain redundancy in the resulting document encoding. Still, the system tries to find the best valid solution by means of an adaptive alphabet extraction method. The over-all workflow can be automated as appropriate parameters for the particular algorithms are estimated by the system.

Aim of this paper was to give an overview of a concrete implementation serving the outlined purpose. The system structure and employed methods have been discussed briefly. More elaborate reports on particular algorithms and their performance will be subject of future publications. Planned features to be implemented and tested in coming versions of the system are feedback from alphabet extraction to text segmentation, means for editing and repurposing as well as search functions for document specific encoded documents.

References

[1] S. Pletschacher, OCR Alternatives for Electronic Publishing of Digitised Documents, Proc. International Conference on Electronic Publishing, Leuven, Belgium, 2005

[2] G. Kopec, M. Lomelin, Document-Specific Character Template Estimation, Proc. SPIE Vol. 2660, Document Recognition III, 1996

[3] R. N. Ascher, G. Nagy, A Means for Achieving a High Degree of Compaction on Scan-Digitized Printed Text, IEEE Transactions on Computers, Volume C-23, Issue 11, 1974

[4] T. M. Breuel, W. Janssen, K. Popat and H. Baird, Paper to PDA, Proc. International Conference on Pattern Recognition, Quebec, Canada, 2002

[5] S. Pletschacher, M. Eckert, A. C. Huebler, Vectorization of Glyphs and Their Representation in SVG for XML-Based Processing, Proc. International Conference on Electronic Publishing, Bansko, Bulgaria, 2006

[6] SVG - Scalable Vector Graphics, W3C, http://www.w3.org/Graphics/SVG/, last visited April 2008

[7] XML - Extensible Markup Language, W3C, http://www.w3.org/XML/, last visited April 2008

[8] Unicode Inc., Unicode Standard, http://www.unicode.org/standard/standard.html, last visited April 2008

Author Biography

Stefan Pletschacher studied Computer Science with special focus on Artificial Intelligence at Chemnitz University of Technology (Germany). In 2003 he obtained his masters degree “Diplom-Informatiker” from there. Since then he has been working as a research assistant at the Institute for Print and Media Technology at Chemnitz University of Technology. Currently he is pursuing his PhD in the field of Document Image Analysis.