

# Towards a Faithful Visualization of Historical Books on E-book Readers

Simone Marinai  
DSI - Università di Firenze  
Firenze, Italy  
simone.marinai@unifi.it

Alessio Anzivino  
DSI - Università di Firenze  
Firenze, Italy  
alessio.anzivino@stud.unifi.it

Matteo Spampani  
DSI - Università di Firenze  
Firenze, Italy  
matteo.spampani@stud.unifi.it

## ABSTRACT

The faithful visualization of historical documents on e-book devices and tablet computers is addressed in this paper. To this purpose, digitized books should be converted to re-flowable formats where the characters are easily re-sized. This is accomplished by first analyzing the document to extract the characters that are then clustered and replaced by prototypes. The prototypes are represented as SVG objects and then arranged in the proper position in the converted document.

Among other applications, the proposed conversion can be used to allow visitors of archives and exhibitions to easily browse and consult historical documents on dedicated devices or on personal mobile devices that support standard re-flowable formats.

The system is quantitatively tested on the well known UW-I dataset by computing OCR errors on the original images and on the reconstructed ones. The visual rendering of historical documents is evaluated on a digitized book of the XIX-th Century.

## Categories and Subject Descriptors

I.7.4 [Document and Text Processing]: Electronic Publishing; H.3.1 [Content Analysis and Indexing]: Indexing Methods

## Keywords

E-book reader, historical documents, re-flowable format

## 1. INTRODUCTION

In the last few years, e-book readers are becoming increasingly used. The majority of these device adopt special E-ink displays that allow users to read documents with a paper-like feeling. The current technology is based on gray-level screens with a low refresh rate. Therefore the devices are more appropriate for static content such as books. Another important feature of these devices is that, apart from some

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

HIP '11 September 16 - September 17 2011, Beijing, China  
Copyright © 2011 ACM 978-1-4503-0916-5/11/09 ...\$10.00.

exceptions, the size is limited to 6-7 Inches. Larger screens are available as well, but in this case the portability of the device is limited.

Tablet computers are another category of devices that allow to read e-books. In this case the screens are larger, in color, and backlit. The latter feature is in general associated with a less comfortable reading experience and it is therefore more fatiguing to read whole books on these devices.

Besides this large plethora of devices there are a few file formats for e-books that are readable in most platforms. The two leading formats are the PDF and the ePub.

The PDF is a standard format, defined and maintained by Adobe, that allows a faithful rendering of documents on a broad range of devices including printers, personal computers, tablets, and mobile phones. Although evolving with new features, the PDF format is not designed to be easily re-sized; on the opposite, its principal aim is an accurate cross-platform rendering of the document contents and layout.

To address these limitations in the visualization on e-book readers, re-flowable formats, such as HTML, are the first choice. In recent years the support for many formats either proprietary or open has been included in e-book readers. After years of vagueness, the ePub open standard is now readable by most e-book readers (with the notable exception of Amazon Kindle that adopts a proprietary format) and tablet computers and it is therefore the de-facto standard.

The rest of the paper is organized as follows. In Section 2 we analyze the need for a conversion in ePub and we describe the main features of this format. Section 3 is focused on the description of the proposed system, that is experimentally evaluated in Section 4. Some conclusions and future work are addressed in Section 5.

## 2. FROM SCANNED BOOKS TO EPUB

E-book readers are convenient not only for the large number of books that can be stored on the devices, but also for additional features such as integrated dictionaries and, in some cases, simplified Internet browsing. One peculiarity of these devices is possibility of a dynamic modification of the font size with the subsequent rescaling of the pages so as to fit the screen.

As a consequence, fixed size formats, such as PDF are not best suited for these devices. This is a minor problem for contemporary documents both digitized and digital-born.

For digitized books, OCR tools can be used to recognize the textual content that can then be easily dynamically reformatted by suitable browsers embedded in the device.

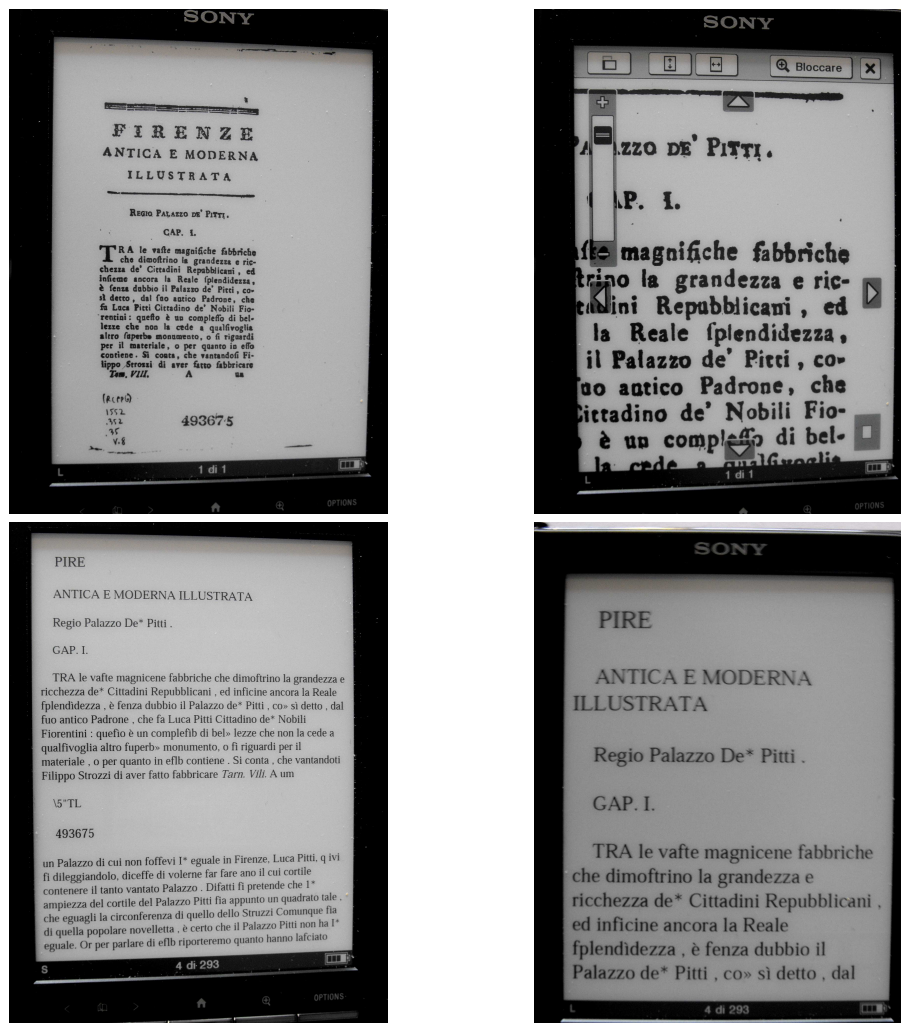


Figure 1: Visualization problems for historical books on e-book readers. Top: full page, zoom of a region. Bottom: OCR of the whole page, re-sized OCR text.

Problems are still open for more complex documents containing tables, equations, and complex layouts.

For digital-born documents, some techniques can be adopted for the conversion of PDF documents to re-flowable formats such as the ePub. When the various files used in the document production chain are still available, it is possible to use standard document processing techniques, such as latex or XSLT transformations to directly generate the desired format. When only the PDF files are available, it is possible to use content extraction tools that are designed to process digital-born PDF books. For instance, in [6] we described a tool for the automatic extraction of administrative metadata from PDF documents in Digital Libraries, while in [7] we described a system for the semi-automatic extraction of the Table of Contents (ToC) from PDF books. The latter system has then been extended in [8] in order to perform the conversion in ePub of digital born books.

Historical books are, however, more difficult to handle. In Figure 1 we show some options that are available to read one digitized book on an e-book reader. In the left part we have the digital image of the whole page. Even if the

page looks nice, reading the text is nearly impossible since the screen has a size of 6 Inches. In the second image we show the zoom of one region of the page. It is now easier to read the text, but moving the zoom area in the page is not the best way to access the whole document. The last two images contain the visualization of the text as recognized by an OCR engine. The text can now be easily re-sized and it is therefore possible to read the text as in the last image. However, old documents like this one are difficult to recognize by OCR engines. For instance the first word in the last image (“PIRE”) is an erroneous recognition of the title (“FIRENZE”).

## 2.1 The ePub format

In this section we summarize the main features of the ePub format. ePub [3] is an open standard based on XML that is designed for the publication of digital books.

The ePub format is widely used and supported by several devices. An ePub document is basically one ZIP file that assembles some files containing both document metadata and the main book content. The latter is encoded

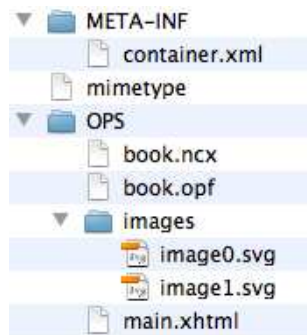
into one or more *XHTML* files corresponding to separated chapters. Images and CSS style-sheets can be included as well. In most books the ePub archive includes also an NCX file that describes the Table of Contents (*ToC*) and allows an easy navigation by pointing to the indexed parts of the book. The ePub standard is developed and maintained by the International Digital Publishing Forum (IDPF) [4], a no-profit organization that includes industries and research institutions.

The ePub standard relies on three main specifications:

- *Open Publication Structure (OPS)*: defines the features of the files involved in the ePub format and the subsequent formatting.
- *Open Packaging Format (OPF)*: defines the organization of the various components of an ePub file. For instance, the metadata, the sequential order of files in the chapters, the order of images, the Table of Contents.
- *OEBPS Container Format (OCF)*: defines the packaging of the ePub archive as a ZIP file.

Basically, the ePub is based on XHTML for the content and on CSS for layout. Moreover, XML is used for the *manifest*, the *Table Of Contents* (TOC), and the metadata.

All these files are organized in a specific folder structure and are compressed in a ZIP archive with extension `.epub`. The organization of the files in the ePub archive is sketched in Figure 2.



**Figure 2: Organization of an ePub archive.**

In the file structure there are two mandatory elements: **Mimetype** is one file containing one fixed string used to identify the ePub files. The **META-INF** folder contains the `container.xml` file that specifies the first file to open to read the book.

All the content files are stored in the **OPS** folder that in turn contains one file with extension `.OPF` describing the book holding metadata associated to the book. Some important fields are the name of the OPS publication (usually the book title) and metadata in Dublin Core format (e.g. title, authors, and language).

Additional files are described in the following. The *Manifest* contains the list of items belonging to the publication, with the MIME type associated. The *Spine* defines the reading order of the elements specified in the Manifest. The `.NCX` file contains the Table of Contents (TOC) allowing the e-book reader to generate a linked index to the book contents.

One or more files (in the example `main.xhtml`) contain the main book text (in most cases each chapter is contained in one separated XML file). Additional folders contain the images (in the example `image0.svg`) and style-sheets.

Most XHTML elements are allowed in the ePub standard. However, some features are not supported by e-book readers and are simply by-passed when rendering the book. For instance, some items not allowed are *forms* and sections with scripts. For the style-sheets a subset of CSS 2.1 is supported, but background images and some properties related to word and letter spacing are not allowed.

In 2011 the new standard *ePub3* is planned to be adopted by the International Digital Publishing Forum consortium. The new standard incorporates HTML5 and CSS3 and allows documents to include multimedia elements, such as video, and separated SVG objects.

### 3. THE PROPOSED APPROACH

The approach proposed in this paper is based on first identifying the textual parts of the document and then converting each character in a vectorial format. The latter format is based on Scalable Vector Graphics (SVG) that can be embedded in HTML and then re-sized by e-book readers that support embedded SVG objects.

One limitation of the current system is that it is possible to process only textual documents printed on a single column. Moreover, the documents should have a reduced skew and a left to right reading order.

The proposed method is based on the following main steps:

1. *Image preprocessing* that includes the image conversion to gray levels and the subsequent adaptive binarization.
2. *Location of connected components* to identify characters in the image.
3. *SOM-based clustering of connected components* in order to group together similar characters that will be replaced with one prototype. In so doing it is possible to reduce the number of symbols to be used in the final ePub file.
4. *Vectorization of prototypes* to allow an easy re-scaling of the text. Prototypes are described in SVG.
5. *Segmentation of lines and words* where for each connected component we identify the row and word it belongs to.
6. In the *ePub generation* we build the overall ePub structure.

In the rest of this section we describe with more details the above steps.

#### 3.1 Image preprocessing

In the current system, the clustering and vectorization are applied on a binary image. Therefore, we first convert color images in gray levels and subsequently binarize the latter images with an adaptive thresholding algorithm to deal with documents with a non-uniform background. The binarization adapts the threshold according to the average gray level in a region around the pixel of interest.

### 3.2 Connected component identification

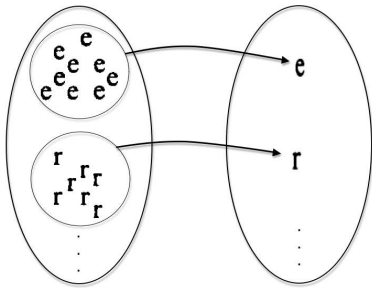
When dealing with clean documents the connected components mostly correspond to individual characters. On the other hand historical documents are often noisy and in this case contiguous characters can be merged in a single component. Although this fact can be problematic for the recognition of the text by OCR engines, it has a limited impact on the proposed technique.

For each connected component we store the size and position information that are used by subsequent steps.

### 3.3 SOM-based connected components clustering

The performance of the proposed approach are influenced by the clustering technique adopted. In particular, the clustering should allow to reduce the size of the generated ePub file while preserving as most as possible the readability of the text.

By increasing the number of prototypes we increase the file size and the fidelity in the visualization. The highest quality (and largest file size) is obtained when considering one different prototype for each character. The clustering is therefore very critical in the overall process. In Figure 3 we graphically depict the general idea of character prototyping.



**Figure 3: Connected components clustering and corresponding prototypes.**

In our case we use the *Self Organizing Map (SOM)* [5] as clustering algorithm. The SOM is a special kind of artificial neural network that has been widely used in several applications in document image analysis. In the SOM architecture, the neurons are in most cases organized in a bi-dimensional lattice (feature map). Each neuron receives information from the input and from the neighboring neurons in the map. When the training examples are represented by real valued vectors  $x = [x_1, x_2, \dots, x_n] \in \mathbb{R}^n$ , to each SOM unit it is associated one model vector  $m_i \in \mathbb{R}^n$  that can be regarded as the prototype for the objects in the corresponding cluster. During the SOM training the model vectors are updated so as to represent the objects in the cluster. In the case of an Euclidean space the average of patterns is considered as model vector.

One advantage of the SOM with respect to other clustering methods is that neighboring units in the map are more similar than farther ones. In this way the proximity relationships between patterns in the original space are preserved as most as possible in the map.

Each SOM map is characterized by:

- The *map size* that defines the number of units, organized in a two dimensional grid with a rectangular

topology.

- The *model vector size* that is the size of the vector associated to each unit. This dimension corresponds to the size of input patterns. In our application all the connected components that are handled by a given map are linearly scaled so that their dimension corresponds to the model vector size.

When dealing with scanned documents the dimension and shape of the bounding box of each connected component can significantly vary. In order to obtain a satisfactory clustering we split the connected components in three groups on the basis of their aspect-ratio. The latter is the ratio between width and height of the upright rectangle enclosing the connected component.

The three groups contain the connected components with an aspect ratio lower than 1, greater than 1, and approximately equal to 1. Each class is further split in three additional sub-classes on the basis of the length of the longest component side. In total, we build and train nine SOMs for each document to be converted.

It is important to notice that the ranges for the dimensions of the characters assigned to each SOM have been empirically fixed on the basis of some preliminary tests. A more robust and scalable system would require one statistical analysis of the documents to be processed to identify the most appropriate values for the following parameters:

- *Number of SOM maps.* When the range of dimensions of connected components in the pages is large, more maps are required to be able to provide a faithful reconstruction of the original characters.
- *Maps size.* When several characters (or more variations of characters in case of noisy/old documents) appear in the pages, more units in the SOM are required to assign different symbols to different prototypes.
- *Size of prototypes in the SOM.* This value defines the dimension of feature vectors in the SOM and is related to the average size of connected components.

Future improvements of the approach should consider automatic solutions to fix the above values on the basis of statistics computed from the connected components in the pages to be converted.

To provide a visual clue of the SOM maps generated it is possible to replace each SOM unit with a visual representation of the corresponding prototype. This is obtained by computing the average of the patterns belonging to its cluster. Two examples of SOMs are shown in Figure 4 where maps computed with two sizes of prototypes are compared. We can notice that some prototypes are more blurred, because the patterns in the corresponding cluster are less uniform.

In the next steps the prototypes are converted in a vectorial format to be included in the ePub document.

### 3.4 Vectorization of prototypes

The character prototypes computed in the previous step are converted in SVG. In this way when the pages are resized by an e-book reader the characters are scaled as well. The vectorization is computed with the open source software *Potrace* [9]. A vectorial representation of a black and white

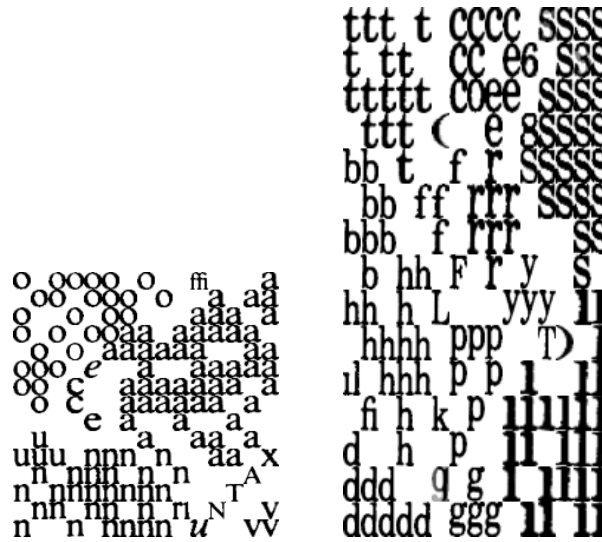


Figure 4: Examples of trained SOM. On the left the map size is  $15 \times 15$  and prototypes  $15 \times 15$ . On the right the map size is  $15 \times 15$  and prototypes  $24 \times 12$ .

image is based on a symbolic description of the contours, e.g. based on Bezier curves. Unlike bitmap images, the vectorial representation can be scaled at various sizes without quality losses. One example of one character prototype with the corresponding SVG representation is shown in Figure 5.



Figure 5: Conversion of one bitmap prototype (left) in SVG (right).

### 3.5 Row and word segmentation

At end of the SOM training and vectorization we have a library of prototypes that can be used to rebuild the original document by replacing each connected component with the closest prototype. In so doing it is possible to reduce the size of the generated file since only few prototypes for each symbol class are included in the file.

It is relatively easy to replace each connected component with the corresponding prototype. However, to build the ePub file we need to identify the relative position of each connected component with respect to the other components (characters) in the page.

Each image is therefore processed in order to:

1. identify the text-lines in the page;
2. sort the connected components in the text-lines according to the reading order;
3. identify the words in each text-line.

At the end of this analysis, each page is represented by an ordered sequence of words that are not constrained to stay on fixed physical coordinates.

#### 3.5.1 Text-line segmentation

The text-lines are identified considering horizontal projection profiles in the image. Peaks in the projection profile correspond to more dense areas in text-lines. By thresholding this histogram it is possible to identify the text-lines.

It is well known that one limit of this type of segmentation is related to the page skew. Therefore, this approach can provide satisfactory results only when the document skew is limited. Moreover, the method can fail when the page layout is more complex, the text is organized in multiple columns, or the document contains other objects such as tables and illustrations. In this case document segmentation algorithms should be considered, such as the recursive X-Y tree segmentation or bottom-up methods. The current prototype is limited to single-column documents with text-lines nearly horizontal.

#### 3.5.2 Connected component sorting in text-lines

The regions corresponding to text-lines identified in the previous step are used to identify and sort the connected components belonging to the lines. This is achieved by first identifying the connected components that intersect with each text-line and then sorting these connected components on the basis of the x value of the upper-left corner of the bounding box.

Organizing the connected components only on the basis of this x value can produce wrong results when dealing with characters split horizontally in two connected components such as the body and the dot of an 'i'. One example of this situation is shown in Figure 6.

To solve these problems we merge together in a single object all the connected components that belong to the same text-line and whose projections in the horizontal axis overlap. The merged components are then used to identify prototypes that are converted in SVG.

Figure 6: One word reconstructed in the wrong way by sorting overlapping connected components.

### 3.5.3 Word segmentation from text-lines

The word segmentation is required to insert an SVG symbol corresponding to an horizontal white space in the flow of graphical symbols. In this way the visualization device will not split a word between contiguous lines when resizing the document.

The word segmentation is obtained by first computing a statistical analysis of the horizontal distance between bounding boxes of connected components belonging to the same text-line and then estimating the expected inter-word distance. Between connected components that are separated by a larger space we insert the special SVG space.

Also in this case this is a simple algorithm that does not perform well with more complex objects such as equations. Another problem that will be addressed in future implementations are hyphenated words (ending with a dash at the end of the line) whose components are now treated as separate words.

## 3.6 ePub generation

In the last step we put together all the information previously extracted to build the file with the sequence of vectorized characters computed from the input image.

We first identify the prototypes occurring in the indexed pages so as to store in the ePub image folder only the prototypes required.

The ePub file is then generated by concatenating in the XHTML file the prototypes corresponding to the words in the original text. We build one block for each word. To nicely separate the words in the text-line we consider one *blank* symbol that corresponds to one space and is scaled together with the other characters when changing the font size in the e-book reader.

After segmenting contiguous words the last problem to solve is how to vertically align the characters in a word. For this alignment of prototypes we first compute the text-line baseline and then vertically arrange the symbol in the XHTML file according to the position of the corresponding connected component with respect to the baseline. In Figure 7 we show one text fragment before and after the correction of the vertical alignment. Characters with descenders (e.g. “g”, “p”) are placed in the right way only after correcting the vertical alignment (see e.g. the last word in the first line: “tempo” that before the correction is printed as “temPo”).

At the end of the ePub generation all the files are organized in the appropriate folder structure (Figure 2) and then compressed in a unique ZIP file (with extension ePub) that can be rendered on any device supporting SVG images embedded in XHTML.

Figure 7: Correction of the vertical alignment of the prototypes in the ePub file.

## 4. EXPERIMENTAL RESULTS

The experimental analysis of the proposed system is based on a numerical evaluation performed on a benchmark dataset and on a visual check made on an historical book.

The numerical tests are made considering the UW-I dataset produced by the University of Washington. Starting from the page layout ground-truth we extracted all the text regions with a total of 4472 regions with the associated textual ground truth. The quality of the documents encoded with the proposed method is indirectly measured on the basis of the text recognition by means of the open source OCR engine Tesseract [2].

In particular, we compare the recognition accuracy on the original images and on the images obtained by rendering the vectorial prototypes computed with the proposed method. In Table 1 we compare the recognition rates (percentage of correctly recognized words) in the two cases.

Comparison	Recognition rate
OCR on original documents	95,32 %
OCR on converted documents	92,73 %

Table 1: Comparison of OCR recognition on the original and on the reconstructed documents.

As we can see from the table, the difference in the recognition rate in the two cases is 2.59%.

To provide a visual comparison of the results obtained we show in Figure 8 one image with the corresponding page reconstructed by using the vectorial prototypes.

Another example is reported in Figure 9 where we show the page rendered with various font sizes on the SONY reader software that allows to read on Windows and Mac ePub documents with an interface similar to the SONY reader device [1].

The last experiments involved the conversion of a scanned historical book downloaded from Google Books. In the left of Figure 10 we show two pages of the book while the conversions with SVG prototypes are shown on the right. We can notice that several letters in the original image are touching. This feature is one of the reasons for the low recognition rate of OCR engines on these documents, as we shown already in the bottom of Figure 1.

From Figure 10 we can see that the quality of the converted image is relatively good and in some cases the removal of small connected components allows us to improve the overall readability.

Layered, garnet-bearing felsic gneiss occurs throughout the Bunger Hills area, but is rare in the Obruchev Hills. Almandine-pyrope garnet (2–15%), biotite (up to 6%), quartz (30–50%), perthite (up to 50%), plagioclase (commonly  $An_{25-40}$ ; up to 50%) and minor opaque minerals and zircon occur in most rocks, and small amounts of apatite, rutile, monazite, corundum, spinel, and sillimanite may also be present; orthopyroxene occurs in some layers. Much garnet-

Layered, garnet-bearing felsic gneiss occurs throughout the Bunger Hills area, but is rare in the Obruchev Hills. Almandine-pyrope garnet (2–15%), biotite (up to 6%), quartz (30–50%), perthite (up to 50%), plagioclase (commonly  $An_{25-40}$ ; up to 50%) and minor opaque minerals and zircon occur in most rocks, and small amounts of apatite, rutile, monazite, corundum, spinel, and sillimanite may also be present; orthopyroxene occurs in some layers. Much garnet-

Figure 8: Original image (top) and reconstruction with SVG prototypes (bottom).

## 5. CONCLUSIONS

In this paper we addressed the problems that come out when we aim at visualizing digitized historical books on e-book readers that support the ePub standard. In this preliminary work we focused on a first implementation of the system that has been quantitatively tested on contemporary scanned documents belonging to the UW-I dataset. Additional visual tests have been performed on one historical digitized book.

Although working satisfactory on these tests there are some ways to extend the proposed system into a working one. First, a more complete layout analysis should be considered instead of the simple projection-based one addressed here. In so doing it will be possible to handle also documents with more complex layouts and with a larger skew. Second, the SOM features (number of maps and dimensions) are now fixed a priori and are not adapted to a given document. In a robust system these parameters should be adjusted on the basis of the documents to be converted. Third, the vectorization parameters should be tested to a larger extent in order to come to a compromise between fidelity of representation and size of the prototypes.

## 6. REFERENCES

- [1] *SONY eBook Reader emulator*. <http://ebookstore.sony.com/download/>.
- [2] *Tesseract OCR*. <http://code.google.com/p/tesseract-ocr/>.
- [3] H. Ainsworth. Epub format construction guide. 2010. <http://www.hxa.name/>.
- [4] IDPF. Epub3 international digital publishing forum, March 2011. <http://idpf.org/epub/30>.
- [5] T. Kohonen. *Self-organizing maps*. Springer Series in Information Sciences, 2001.
- [6] S. Marinai, E. Marino, and G. Soda. Metadata extraction from PDF papers for digital library ingest. In *10th Int.l Conf. on Document Analysis and Recognition*, pages 251–255, 2009.
- [7] S. Marinai, E. Marino, and G. Soda. Table of contents recognition for converting pdf documents in e-book

La persistenza è uno dei concetti fondamentali nello sviluppo di applicazioni sia web che desktop. L'approccio utilizzato nella gestione di dati persistenti rappresenta una decisione chiave che impatta radicalmente sui tempi di sviluppo, la portabilità e la manutenibilità dell'applicazione.

Quando si parla di persistenza con Java, si intende normalmente la memorizzazione di dati su un database relazionale con l'utilizzo del linguaggio SQL tramite un'insieme di interfacce standard. Se da un lato questo approccio permette di utilizzare convenientemente tutte le potenzialità offerte dai database relazionali, accedendo anche a quelle funzioni proprietarie che li rendono più performanti o più adatti al particolare caso d'uso, dall'altro introduce una serie di problematiche che vanno dall'obbligo di scrivere ogni query, persino per le operazioni più elementari CRUD (Create Read Update e Delete) col linguaggio SQL, la non portabilità su altri database relazionali dovuta ai vari dialetti di questi ultimi ed infine la differenza tra la rappresentazione di dati del mondo object-oriented e quella del mondo relazionale.

A proposito dell'ultimo problema Martin Fowler in [POEAA] propone un'insieme di pattern architetturali per il mapping tra i database relazionali e linguaggi object-oriented. In seguito Gavin King e Christian Bauer forniscono un'implementazione per Java di questi concetti, dando vita al progetto open source Hibernate ([HIA], [JPWH]).

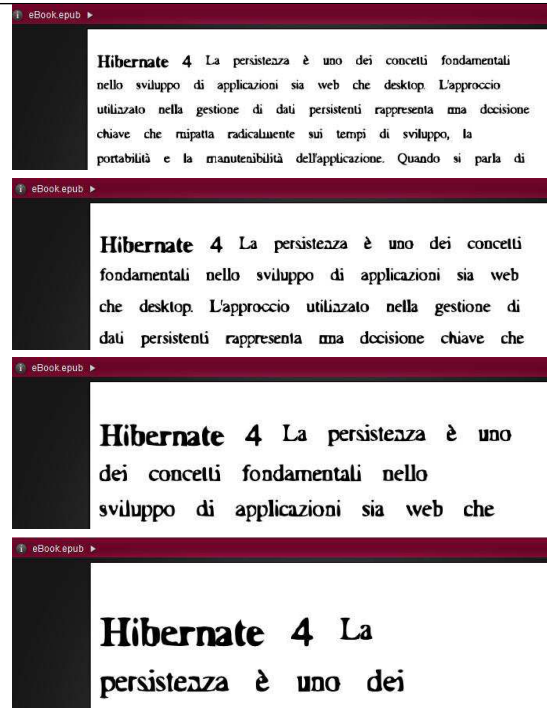


Figure 9: Document image (top) and the corresponding page on the SONY reader software shown with various font sizes (XS, S, M, L).

formats. In *Proc. 10th ACM symposium on Document engineering, DocEng '10*, pages 73–76, New York, NY, USA, 2010.

- [8] S. Marinai, E. Marino, and G. Soda. Conversion of PDF books in ePub format. In *11th Int.l Conf. on Document Analysis and Recognition*, 2011.
- [9] P. Selinger. *Potrace: a polygon-based tracing algorithm*, 2003. Software available at <http://potrace.sourceforge.net/>.

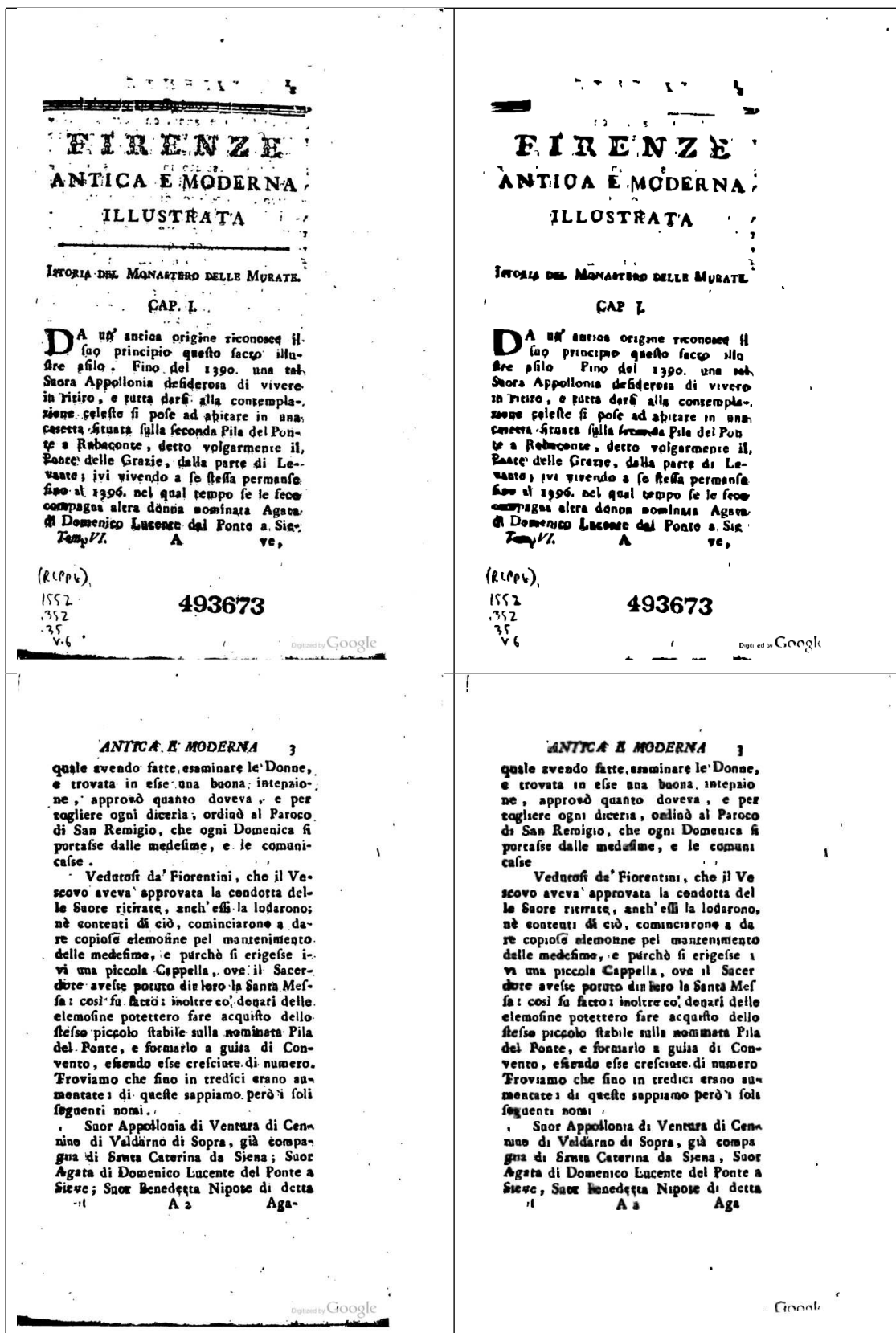


Figure 10: Two digitized pages downloaded from Google Books (left) and the corresponding pages obtained by replacing the characters with SVG prototypes (right).