Quality Assurance for Scalable Braille Web Service Using Human Interaction by Computer Vision System

Roman Graf[†], Reinhold Huber-Mörk[‡] and Alexander Schindler[‡]

[†]*AIT Austrian Institute of Technology GmbH, Digital Memory Engineering, 1220, Vienna, Austria*
roman.graf@ait.ac.at

[‡]*AIT Austrian Institute of Technology GmbH, High Performance Image Processing , 2444, Seibersdorf, Austria*
reinhold.huber@ait.ac.at, alexander.schindler.fl@ait.ac.at

Abstract: *This paper presents a Braille converter Web service that is a sample implementation of scalable service for preserving digital content. An image processing system is used for human interaction. Optical pattern recognition allows Braille code creation using Braille patterns. Quality assurance tools were implemented to simplify interaction with Braille service.*

Keywords: *digital preservation, image processing, quality assurance, haptic I/O*

DOI: *It would be provided by publication house.*

I. INTRODUCTION

Braille is a system which enables visually impaired persons to read text from tactile patterns by touch. The Braille code (BC) is a set of tactile patterns combined from raised dots (Jiménez *et al.*, 2009) . A physically tactile pattern is presented as a cell of six dots arranged in two columns and three rows. Letters, numerals and punctuation can be represented using different dots combinations.

We describe a scalable Braille conversion Web service. Scalability and quality assurance (QA) plays an important role with increasing volumes of digitized data. One of the goals of the SCAPE[1] project is to create quality-assured scalable services for digital preservation of large amounts of data.

The BC conversion service available through a web server is a sample implementation for such a scalable service. There is a lack of public Braille services and still no proper solution to the problem of automating Braille conversions (Manohar and Parthasarathy, 2009). Automatic migration of Braille files in Braille Ready Format (BRF) (Frees *et al.*, 2010) to the newer and more flexible Portable Embosser Format (PEF) is a practical implementation of a service that needs scalability. The output of the conversion service is a PEF file which can further be used for embossing or for presentation using standard Braille display. A demo workflow was implemented based on the Planets Workflow Engine (Schmidt *et al.*, 2010). Scalability is achieved using CUDA (Owens *et al.*, 2007) GPU parallel processing to increase performance and enhance scalability.

QA is required for preservation actions involving BC recognition to ensure high quality of BC images.The currently implemented QA tools enable feature extraction from images and comparison for image files in common file formats. The QA tools support image comparison and search operations. These tools enable visually impaired people to control BC creation process. The code created from acquired images by BC recognition system has to be checked for its quality, completeness and consistency.

Interactive and automatic Braille recognition from images was successfully performed with different acquisition setups, various algorithms and different output format and media (François and Calders, 1985), (Mihara *et al.*, 2005). We will describe a system with a fixed camera, edge based segmentation and recognition of Braille characters.

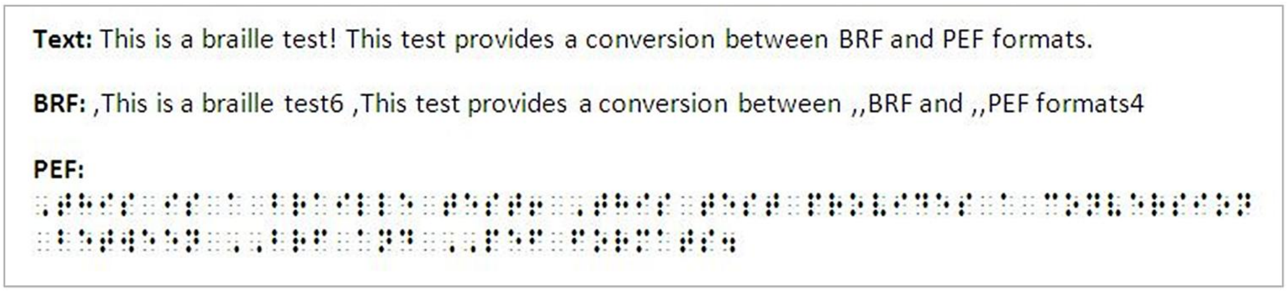
II. BRAILLE CODING CHALLENGES

There are a number of different formats for digital representation of Braille, two of which are considered in this section. A BRF file is an ASCII file where the ASCII characters simply transliterate the Braille cells according to some convention (consider the North American ASCII Braille BRF example in Fig. **??**).

The more recent format for Braille files called PEF (Leas *et al.*, 2008) was developed in 2005 by the Swedish Braille library. PEF is not-yet widely adopted but has some advantages when compared to the BRF file format because it contains information about file content, proper Braille publishing standard, file sharing ability, long term archive preservation safety and uses Unicode Braille pattern (BP). PEF files have a header which references the print source and other important metadata like title, author and so on. Customized metadata are also possible.

Braille coding utilizes various combinations of contractions, markup, direct representation, and whitespace formatting. Each language has one or more different BCs for converting text to Braille (literature, technical material, music, computer and so on). Currently there is no way to uniquely identify which Braille system has been used to produce the Braille file. ASCII

[1]This work was supported in part by the EU FP7 Project SCAPE (GA#270137) `www.scape-project.eu`. We would like to thank Susan Jolly from `www.dotlessbraille.org` for providing useful information regarding Braille formats.


Text: This is a braille test! This test provides a conversion between BRF and PEF formats.

BRF: ,This is a braille test6 ,This test provides a conversion between ,,BRF and ,,PEF formats4

PEF:

⠠⠞⠓⠊⠎⠀⠊⠎⠀⠁⠀⠃⠗⠁⠊⠇⠇⠑⠀⠞⠑⠎⠞⠖⠀⠠⠞⠓⠊⠎⠀⠞⠑⠎⠞⠀⠏⠗⠕⠧⠊⠙⠑⠎⠀⠁⠀⠀⠉⠕⠝⠧⠑⠗⠎⠊⠕⠝
⠀⠃⠑⠞⠺⠑⠑⠝⠀⠠⠠⠃⠗⠋⠀⠁⠝⠙⠀⠠⠠⠏⠑⠋⠀⠋⠕⠗⠍⠁⠞⠎⠲

Figure 1: Representation of a sample text in BRF and PEF format.

Braille represents Braille cells by ASCII characters and has the advantage of being easier to use for humans. The disadvantage of ASCII Braille is that the encoding has to be defined.

III. WEB SERVICE FOR CONVERSION BETWEEN BRF AND PEF FORMATS

The workflow engine provides the functionality of Braille data conversion from the BRF format to the open and preservation-friendly PEF format using a predefined conversion workflow and the Planets digital object model (Schmidt *et al.*, 2010). The functionality to manage Planets digital objects is provided through the Braille conversion Web service written in Java and deployed on the JBoss application server.

The conversion service performs the following actions:

1. The use-case starts with a user call of the conversion service providing the Braille data of a collection in BRF.
2. The service generates a preservation plan for each item in the data collection.
3. The normalization strategy processing starts with evaluation of BRF content. Binary files of the processed item will be harvested based on their URL. Information is collected from content providers and integrated into a representation of the objects in the preservation tasks.
4. Metadata is evaluated in order to build a PEF file header. Expected header should contain following terms: "title", "date", "format", etc.
5. Run the migration accordingly to the preservation plan.
6. Store migration results into the PEF file.
7. Generate report.

Performance measurements were computed to evaluate GPU and CPU processing times (used graphics card: GTX460).

Scalability is obtained through GPU processing provided the overhead for memory allocation is not dominating computation time. In the suggested prototype implementation it was observed that computation time for file sizes less than 50KB are dominated by the overhead associated with allocating GPU memory. The maximum speedup of 6900 was observed for file sizes larger than 13MB. The parallel implementation can be improved by overlapping communication with computation.

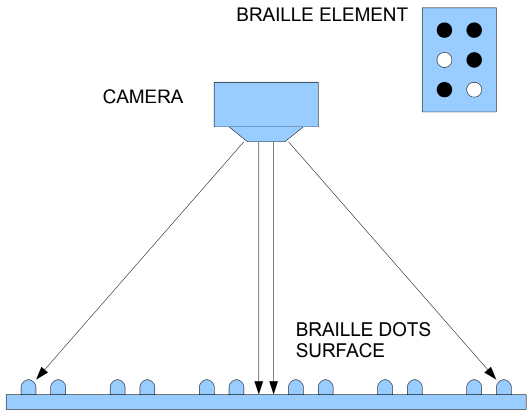


Figure 2: Image acquisition of Braille elements.

IV. EXTRACTION OF BRAILLE CODE FROM IMAGES AND CONVERSION

An extension involving image processing applied to BC extraction is suggested. Images are acquired using an area camera mounted at some fixed distance to the surface holding the Braille elements as shown in Fig. **??**. Perspective distortions are avoided by the setup and geometric lense distortion is corrected. The camera provides gray-scale images of the surface holding physical BPs.

Currently, BC creation in digital form is only possible using specific Braille input devices. The goal of the suggested extension is to provide an input device for users not familiar with common Braille input devices. By interaction with a computer vision setup a user could arrange BP manually using physical Braille elements, i.e. physical building blocks, representing BCs. Once BC creation is completed on a predefined surface the user starts image acquisition of the surface holding the BPs. Subsequently, using a pattern recognition algorithm pins and semantics of BPs are identified and delivered in ASCII code or Unicode standard.

In order to extract Braille elements in the image Braille and raster dots are initially segmented from the background (Sezgin and Sankur, 2004). Point and edge based features are regarded to be more robust against lighting variants (Marr and Hildreth, 1980). In the suggested method the Canny edge detection algorithm was employed (Canny, 1986) in the segmentation step.

The raster coordinates for the standard BP structure is computed using BC grid calculation algorithm

$$\begin{aligned} G &= \{P(x,y) | x \in f(x) \wedge y \in f(y)\} \qquad (1) \\ f(x) &= \begin{cases} x_o + \frac{n}{2} \cdot (x_d + x_p), & \text{if } n \text{ is even,} \\ x_o + x_d + \frac{n-1}{2} \cdot (x_d + x_p), & \text{if } n \text{ is odd,} \end{cases} \\ f(y) &= \begin{cases} y_o + \frac{m}{2} \cdot (y_d + y_p), & \text{if } m \text{ is even,} \\ y_o + y_d + \frac{m-1}{2} \cdot (y_d + y_p), & \text{if } m \text{ is odd.} \end{cases} \end{aligned}$$

Where G represents the standard Braille dots grid computed over the dimension 203 x 264 pixels, $P(x,y)$ represents the grid coordinate points set depending on $f(x)$ and $f(y)$ functions. The n and m represent the dot number for X and Y axis. Several constants were needed for accurately computation: x_o is an offset on X axis from the image edge, x_d is a distance between dots, x_p exposes a distance between two BPs on X axis and remaining constants stand for similar definitions on Y axis.

A pixel count around a grid point is computed as a sum over all evaluated pixels which are located in acceptable distance to the grid point

$$\begin{aligned} S_{n,m} &= \sum_{n,m} \delta_{n,m} \cdot d(P_{n,m}, G_{n,m}) \qquad (2) \\ \delta_{n,m} &= \begin{cases} 1 & \text{if } \sqrt{(G_x - P_x)^2 + (G_y - P_y)^2} < d_{max}, \\ 0 & \text{else.} \end{cases} \end{aligned}$$

Where $S_{n,m}$ represents the sum of matching detected pixels $P_{n,m}$ with coordinates P_x and P_y computed over the dimensions n and m around correspondent grid point $G_{n,m}$ with coordinates G_x and G_y. d represents the distance between a grid point and an evaluated current pixel point coordinates. d_{max} stands for maximal accepted distance where pixel is considered as a correct value. $\delta_{n,m}$ is a coefficient with value 1 for pixel range $< d_{max}$ or 0 otherwise.

The BP dots (see Fig. ??) used in the experiment consists of black points on a white surface. In the experimental setup the dot diameter is about 10 pixels. The placeholders for empty BP dots are depicted as smaller circles. The placeholder diameter is about 6 pixels. The placeholder dots are also important in BP calculation. The separation between Braille and placeholder dots is based on expected Braille dot height, width and maximal pixel count.

The BP recognition and conversion algorithm is summarized as follows:

1. Retrieve a BP image and apply geometric undistorted.
2. Segmentation of the retrieved image using the Canny edge detector. The output of segmentation is an array of detected points.

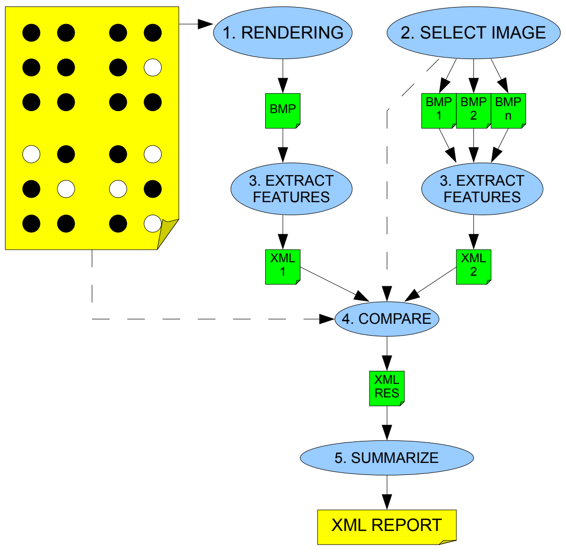


Figure 3: Braille code images comparison workflow.

3. Verification of detected placeholder points using a BC grid, as placeholder dots are more accurately localized than Braille dots. A suitable maximal pixel count value for dot discrimination using the described setup was found to be 14.
4. Remove false positive detections. Extracted detections that do not match the predefined grid are removed.
5. Merge spatially adjacent placeholder dots from step 3. This step rejects pixels that are part of already detected Braille dots.
6. Detect Braille dots using the grid derived from placeholder dots.
7. Compute BPs from detected Braille points. BPs (for example 1-2-5) that are suitable for further processing are obtained.
8. Compute ASCII BC or Unicode values.

The resulting Braille encoding is used as input to the web conversion service described in Sec. ??.

V. QUALITY ASSURANCE TOOLS

The objective of QA regarding BC recognition system is to verify quality and consistence of acquired images. Customized image feature extraction (FE) and comparison tools (CT) were developed. In order to verify BC quality a new image has to be acquired by the BC recognition system (BCRS) from assembled BP. This new image is passed as input parameter to the FE and resulting features are used by CT to compare with existing images. This approach enables missing Braille pages detection, broken pages detection, content error detection, search for particular BP set.

In Fig. ?? we illustrate the comparison workflow:

1. Request image rendering. The execution of workflow begins when BPs are assembled on the work surface and rendered by BCRS, resulting

in a image file in common raster image format, e.g. in BMP format.

2. Select image. User selects file to examine from a set of previously created BC images.
3. Extract features. This is a step where QA FET using created image file as an input is applied to each created BMP image file in order to extract histogram features from passed file. For the pixel wise comparison this step is not necessary and workflow proceeds directly with step 4.
4. Compare. In this step files containing information about pixels or histogram data from input image file are compared with correspondent features or image file using QA CT. The CT has two input parameters. These are two URLs to the pixel or histogram data files that comprise information about image features.
5. Summarize. In the last step comparison results are analyzed, summarized and reported to user.

The Taverna workbench (Belhajjame *et al.*, 2008) provides functionality to address implemented QA tools as a Web service. WSDL Web services which wrap command line tools FET and CT were created using Taverna workflows. These tools concern different digital preservation criteria and provide analysis divided into three levels. In the first level retrieved image metadata is compared to detect obvious differences. The second level handles standard image processing features like histograms and profiles. The third level of abstraction means sophisticated features to detect similarities in images. For the QA analysis we apply all of these levels which provide pixel wise or image histogram and profile comparison. The difference between levels is that for the third level we compare images directly without implying of FET.

VI. CONCLUSION

A new scalable open Braille conversion Web service for preserving digital content was created. Braille conversion service scalability is improved by application of CUDA GPU parallel processing. Speedup enhancements up to 6900 times were achieved.

Image processing and pattern recognition can be used to enable Braille coding without Braille input device and to create input data for the Braille conversion service. Images of physical BPs are acquired and automatically recognized and can also make use of the described web conversion service. No special skills are needed, only familiarity with BC is required.

The image feature extraction and comparison tools support QA for BCRS. These tools provide error detection in created BC and support search functionality for particular BP set passed in image format. Only access to the created data is required for QA tools. User don't have to open files using standard Braille display and manually read it that is a time and human power consuming process. QA tools support high quality BC images creation for scalable conversion Web service.

REFERENCES

Belhajjame, K., Wolstencroft, K., Corcho, O., Oinn, T., Tanoh, F., William, A., Goble, C., "Metadata Management in the Taverna Workflow System," *Sch. of Comput. Sci.*, Lyon, 651 - 656 (2008).

Canny, J., "A computational approach to edge detection," *IEEE Trans. Pat. Anal. Mach. Intell.*, 679 - 698 (1986).

Elberzhager, F., Mnch, J. and Tran Ngoc Nha, V., "A systematic mapping study on the combination of static and dynamic quality assurance techniques," *Inf. and Soft. Tech.*, 1-15 (2011).

François, G., Calders, P., "The reproduction of Braille originals by means of optical pattern recognition," *Proc. Int. Workshop on Computer Braille Production*, 119 - 122 (1985).

Frees, B., Strobbe, C., Engelen, J., "Generationg braille from Openoffice.org," *Proc. Intl. Conf. Computers helping people with special needs. LNCS*, 81 - 88 (2010).

Jiménez, J., Olea, J., Torres, J., Alonso, I., Harder, D., Fischer, K., "Biography of Louis Braille and invention of the Braille alphabet," *Survey of Ophthalmology*, 142 - 149 (2009).

Leas, D., Person, E., Soiffer, N., Zacherle, M., "Daisy 3: A standard for accessible multimedia books," *IEEE Multimedia*, 28 - 37 (2008).

Manohar, P., Parthasarathy, A., "An innovative Braille system keyboard for the visually impaired," *Proc. of UKSim: Intl. Conf. on Comp. Modelling and Simulation*, 559 - 562 (2009).

Marr, D., Hildreth, E., "Theory of edge detection," *Proc. of the Royal Soc. London*, 187 - 217 (1980).

Mihara, Y., Sugimoto, A., Shibayama, E., Takahashi, S., "An interactive Braille-recognition system for the visually impaired based on a protable camera," *Proc. of CHI'05 extended abstracts on Human factors in comp. systems*, 1653 - 1656 (2005).

Owens, J.D., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A.E., Purcell, T., "A survey of general-purpose computation on graphics hardware," *Comput. Graph. Forum*, 80 - 113 (2007).

Schmidt, R., King, R., Jackson, A.N., Wilson, C., Steeg, F., Melms, P., "A framework for distributed preservation workflows," *Intl. J. of Digital Curation*, 205 - 217 (2010).

Sezgin, M., Sankur, B., "Survey over image thresholding techniques and quantitative performance evaluation," *J. Electron. Imaging*, 146 - 165 (2004).