# A Braille Conversion Service Using GPU and Human Interaction by Computer Vision *

Roman Graf
Digital Memory Engineering
Safety & Security Department
AIT Austrian Institute of Technology GmbH
Vienna, Austria
roman.graf@ait.ac.at

Reinhold Huber-Mörk
High Performance Image Processing
Safety & Security Department
AIT Austrian Institute of Technology GmbH
Seibersdorf, Austria
reinhold.huber@ait.ac.at

## ABSTRACT

Scalable systems and services for preserving digital content became important technologies with increasing volumes of digitized data. This paper presents a new Braille converter service that is a sample implementation of scalable service for preserving digital content. The converter service facilitates complex conversion problems regarding Braille code. Braille code is a method which allows visually impaired people to read and write tactile text. Using a GPU with the CUDA architecture allows the creation of a parallel processing service with enhanced scalability. The Braille converter is a web service that provides automatic conversion from the older BRF to the newer PEF Braille format. This service can manage a large number of objects. Speedups on the order of magnitude of 5000 to 6900 (depending on the size of the object) were achieved using a GPU (GTX460 graphics card) with respect to a CPU implementation. An extension involving an image processing system is used for human interaction. Optical pattern recognition allows Braille code creation using Braille patterns. No special input device and skills are needed, only familiarity with Braille code is required.

## Categories and Subject Descriptors

H.3.7 [**Digital Libraries**]: System issues; I.5.5 [**Pattern Recognition**]: Interactive systems; H.3.5 [**Online Information Services**]: Web-based services

## General Terms

Algorithms

*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.



## Keywords

digital preservation, CUDA, performance measurement, information retrieval, image processing, haptic I/O

## 1. INTRODUCTION

We describe a scalable Braille conversion web service using CUDA GPU parallel processing. CUDA is a technology developed by NVIDIA [9] and could be used by a conversion service to increase performance and enhance scalability. CUDA programming requires a specific hardware. In this work, a GTX460 graphics card was used.

Scalability plays an important role with increasing volumes of digitized data. One of the goals of the SCAPE project is to create scalable services for digital preservation of large amounts of data. The SCAPE platform will include a storage infrastructure and an execution environment for performing sustainable, data-intensive and scalable digital preservation activities through parallel processing.

The Braille code [4] 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 [6]. Automatic migration of Braille files in BRF file format [3] to the newer and more flexible PEF format 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 [10]. Customized workflows support the definition of service parameters that manage workflow execution.

Braille [4] is a system which enables visually impaired persons to read text from tactile patterns by touch. The same system is also suitable for writing tactile text. The Braille code is a set of tactile patterns combined from raised dots. 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.

Interactive and automatic Braille recognition from images was successfully performed with different acquisition setups, various algorithms and different output format and media [2], [8]. We will describe a system with a fixed camera, edge based segmentation and recognition of Braille characters.

This paper is organized as follows. Section 2 describes the challenges associated with Braille encodings. Section 3 introduces a web-service for conversion of Braille between

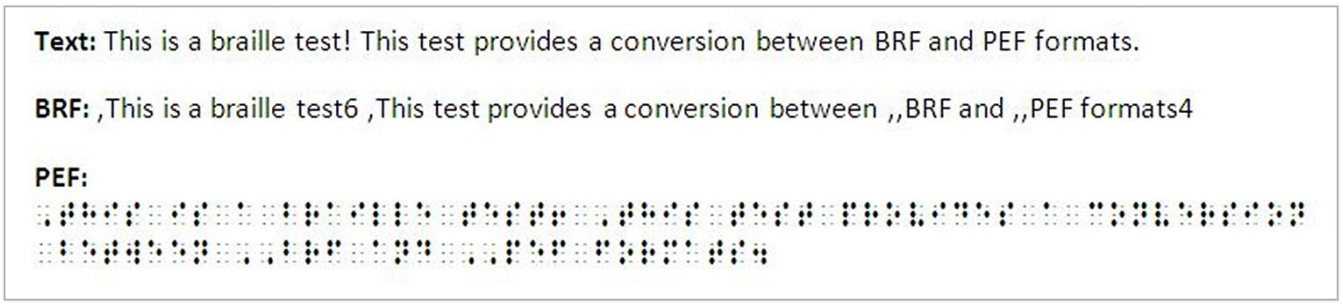


**Figure 1: Representation of a sample text in BRF and PEF format.**

BRF and PEF encodings and Sec. 4 demonstrates scalability based on GPU processing. Section 5 describes image based Braille pattern recognition. Section 6 concludes the paper.

## 2. 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 [3] is an ASCII file where the ASCII characters simply transliterate the Braille cells according to some convention (consider the the North American ASCII Braille BRF example in Fig. 1).

The more recent format for Braille files called PEF (Portable Embosser Format) [5] 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, and long term archive preservation safety. 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. The PEF is a document type that represents Braille pages in digital form, accurately and unambiguously; regardless of language, location, embosser settings, Braille code and computer environment. PEF uses Unicode Braille patterns which are widely accepted as a part of Unicode standard.

Braille files are Braille translations [3] of printed texts produced manually by experienced translators. Braille coding utilizes various combinations of contractions, markup, direct representation, and whitespace formatting. Each language has one or more different Braille codes 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. Furthermore, a Braille code is characterized by a context-sensitive grammar and, even if we know the correct specification for the used Braille system, it is impossible to regenerate the print text completely accurate. ASCII Braille represents Braille cells by ASCII characters instead of Unicode and has an advantage that it is easier for humans to use. The disadvantage of ASCII Braille is that the encoding has to be defined. The Unicode advantage is that it is an international standard.

## 3. 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 PEF format using a predefined conversion workflow and the Planets digital object model [10]. The functionality to manage Planets digital objects is provided through the Braille conversion web service written in Java and deployed with the JBoss application server. This service implements methods that allow the workflow engine to read data from a BRF file, to convert it and to write data to a PEF file.

The Braille conversion use-case describes the performed activities during the processing of the normalization strategy. The main goal of this service is the conversion of the source content from its original BRF data format into an open, preservation-friendly and compatible, PEF format. 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 BRF content evaluation. 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”, ”description”, etc.
5. Create error report.
6. Run the migration accordingly to the preservation plan.
7. Store migration results into the PEF file.
8. Generate report.

## 4. SCALABILITY ENHANCEMENT USING GPU

Scalability could be achieved by parallel processing, e.g. using OpenMP on multicore processors, distributed processing or using OpenCL or CUDA on general purpose graphics cards. In this work we describe a CUDA [9] implementation which utilizes GPU parallel processing. The implementation extends the Braille conversion service in order to enhance scalability. In an experimental setup BRF files of different sizes were converted to PEF files using a GPU processing application. In the experiment two implementations of the conversion service are compared. One implementation uses traditional CPU processing whereas the second implementation uses GPU parallel processing. Pure calculation processing time and total processing time were measured.

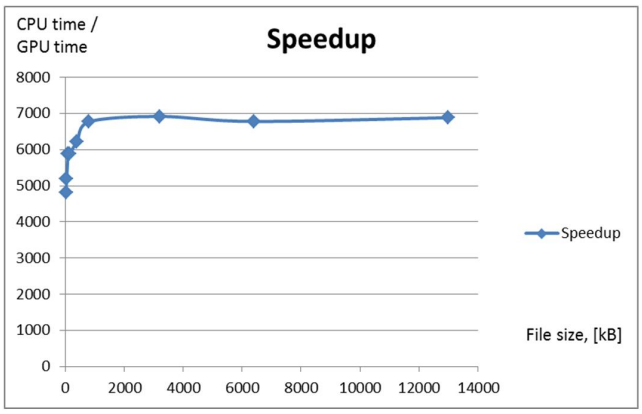


**Figure 2: Speedup achieved using GPU parallel processing.**

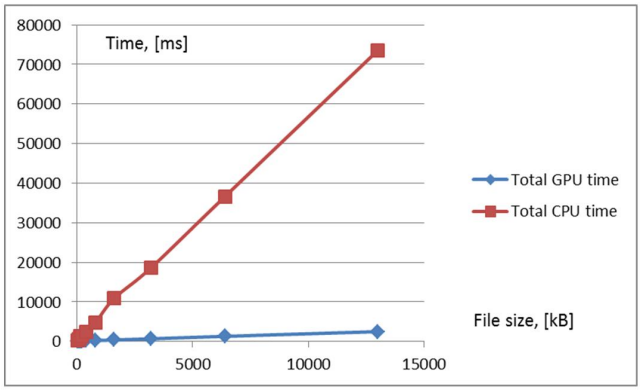


**Figure 3: Dependency between file size and execution time.**

In order to prove the advantages of GPU parallel processing the input data from the BRF file was divided into chunks and loaded to the device memory. In order to map algorithms to the GPU each chunk represents a BRF element that can be converted to PEF Unicode values using a conversion table. Then chunks are processed in parallel by the GPU kernels and results copied back to the host. Parallelization, i.e. thread and memory management, is provided by CUDA. Each natural language requires a special conversion algorithm. In the experiment a conversion algorithm for American English was implemented. The resulting output data is written to a PEF file. Performance measurements were computed to evaluate GPU and CPU processing times (used graphics card: GTX460). The relation of the pure GPU processing time in respect to pure CPU processing time, see Fig. 2, reveals the application processing time without taking in account memory management time. Figure 3 indicates the relation of the total processing times for both implementations including memory management from the start of the conversion service calculations to the completion of the content conversion process.

Files of larger sizes achieve higher speedups, where file size is measured in KB and time in milliseconds. Figure 3 depicts the dependency between the time needed for the BRF file content reading, converting and writing to PEF file using GPU and the time consumed by the same operations using

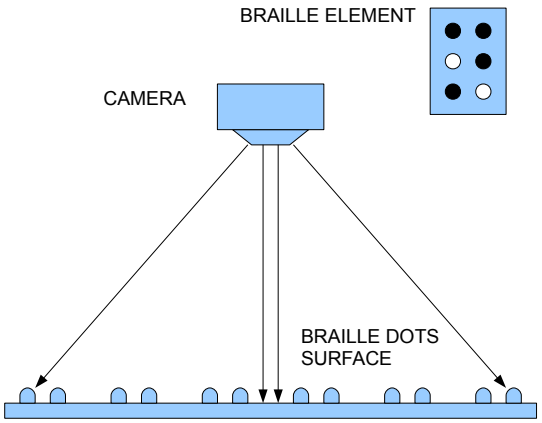


**Figure 4: Image acquisition of Braille elements.**

CPU. The memory management and file read/write operations have been taken into account. In order to evaluate the dependency of performance on file size a sample of eleven files having sizes between 25KB and 13MB were converted. The migration process on CPU mostly takes more time as the content conversion process on GPU. With the larger file size the CPU time consumption increases almost proportionally but the GPU time consumption remains approximately the same.

The gain from using the GPU starts with a file size of about 50KB because for smaller files, the overheads associated with allocating GPU memory dominate the computation time. While GPU parallel processing achieves its maximum speedup (6900 times) for Braille calculations for the file sizes starting from about 13MB. The parallel implementation can be improved by overlapping communication with computation. The reason for the significant speedup is that calculation task normally (with CPU) computed sequentially was broken down to the sub tasks and processed in parallel. The GPU comprises hundreds of cores (336 for GTX460 graphics card). The higher is the cores number the higher is the achieved speedup. Each core supports multiple threads that can be executed in parallel. Each thread has a subtask to execute. Therefore more resources are devoted to data processing rather than data caching and flow control. Web service and workflow engine overhead time consumption is independent from conversion time.

## 5. EXTRACTION OF BRAILLE CODE FROM IMAGES AND CONVERSION

This section describes an extension to the presented conversion service. Image processing is applied to Braille code extraction. Images are acquired using an area camera mounted at some fixed distance to the surface holding the Braille elements as shown in Fig. 4. Perspective distortions are avoided by the setup and geometric lense distortion is corrected. The camera provides gray-scale images of the surface holding physical Braille patterns.

The goal of this extension is to enable Braille coding for the users which are not familiar with Braille devices currently used as a computer interface. Currently, Braille code creation in digital form is only possible using specific Braille input devices. By interaction with a computer vision setup a user could arrange Braille pattern manually using physical

Braille elements, i.e. physical building blocks, representing Braille codes. The user identifies the meaning of each element scanning tactile pattern by moving the finger upon it. Visual impaired people are familiar with this technique and use it for Braille reading of printed books. Once Braille code creation is completed on a predefined surface the user starts imaging of the surface holding the Braille patterns. Subsequently, using a pattern recognition algorithm pins and semantics of Braille patterns are identified and delivered in ASCII code or Unicode standard. The output of pattern recognition and conversion provides an input for Braille conversion service described in Sec. 3.

In order to extract Braille elements in the image Braille and raster dots are initially segmented from the background [11]. Point and edge based features are regarded to be more robust against lighting variants [7]. In the suggested method the Canny edge detection algorithm was employed [1] in the segmentation step. The Braille pattern dots (see Fig. 1) 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 Braille pattern dots are depicted as smaller circles. The placeholder diameter is about 6 pixels. The placeholder dots are also important in Braille pattern calculation. The separation between Braille and placeholder dots is based on expected Braille dot height, width and maximal pixel count.

The Braille pattern recognition and conversion algorithm is summarized as follows:

1. Retrieve a Braille pattern image and apply geometric undistortion.
2. Segmentation of the retrieved image using the Canny edge detector. The output of segmentation is an array of detected points.
3. Verification of detected placeholder points using a Braille code 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 Braille patterns from detected Braille points. Braille patterns (for example 1-2-5) that are suitable for further processing are obtained.
8. Compute ASCII Braille code or Unicode values.

The resulting Braille encoding is used as input to the web conversion service described in Sec. 3.

# 6. CONCLUSION

A new scalable open Braille conversion web service for preserving digital content was created. The service provides conversion of Braille files into the new PEF format from the widely spread BRF format. Braille conversion service scalability is improved by application of CUDA GPU parallel processing. Measurements of conversion times for different BRF file sizes and comparison of results for GPU and CPU processing were given. Speedup enhancements up to 6900 times were achieved. The GPU parallel processing efficiency depends on the file size. For example 50KB file achieves speedup about 5000 times whereas 13MB file speedup is more than 6900 times. Apparently GPU processing is more efficient then CPU processing in terms of Braille conversion for large file sizes (50KB - 13MB) and enhances scalability of conversion service for large files collections.

The scalability improvement is that BRF files in the range between 50KB and 13Mb can be more efficient converted to the PEF files athrough GPU parallel processing and a web service. This acceleration is needed to efficiently migrate large amounts of currently widely used BRF file archives into PEF. Future work will include evaluation of scalability involving larger collections.

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 Braille patterns are acquired and automatically recognized and can also make use of the described web conversion service.

# 7. REFERENCES


[1] Canny, J.: A computational approach to edge detection. IEEE Trans. Pat. Anal. Mach. Intell. 8(6), 679–698 (1986)

[2] François, G., Calders, P.: The reproduction of Braille originals by means of optical pattern recognition. In: Proc. Int. Workshop on Computer Braille Production. pp. 119–122 (1985)

[3] Frees, B., Strobbe, C., Engelen, J.: Generating braille from Openoffice.org. In: Proc. Intl. Conf. Computers helping people with special needs. LNCS, vol. 6179, pp. 81–88 (2010)

[4] 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 54(1), 142–149 (2009)

[5] Leas, D., Persoon, E., Soiffer, N., Zacherle, M.: Daisy 3: A standard for accessible multimedia books. IEEE Multimedia 15(4), 28–37 (2008)

[6] Manohar, P., Parthasarathy, A.: An innovative Braille system keyboard for the visually impaired. In: Proc. of UKSim: Intl. Conf. on Comp. Modelling and Simulation. pp. 559–562 (2009)

[7] Marr, D., Hildreth, E.: Theory of edge detection. Proc. of the Royal Soc. London B-207, 187–217 (1980)

[8] Mihara, Y., Sugimoto, A., Shibayama, E., Takahashi, S.: An interactive Braille-recognition system for the visually impaired based on a portable camera. In: Proc. of CHI'05 extended abstracts on Human factors in comp. systems. pp. 1653–1656 (2005)

[9] 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 26(1), 80–113 (2007)

[10] Schmidt, R., King, R., Jackson, A.N., Wilson, C., Steeg, F., Melms, P.: A framework for distributed preservation workflows. Intl. J. of Digital Curation 5(1), 205–217 (2010)

[11] Sezgin, M., Sankur, B.: Survey over image thresholding techniques and quantitative performance evaluation. J. Electron. Imaging 13(1), 146–165 (2004)