Automatic Braille Code Translation System

Hamid Reza Shahbazkia, Telmo Tavares Silva, and Rui Miguel Guerreiro

Universidade do Algarve,
Laboratório de Sistemas Funcionais Alternativos,
Campus de Gambelas, 8005-139 Faro-Portugal
{hshah, a17145, a14104}@ualg.pt

Abstract. This paper reports the results obtained in the implementation of an Optical Braille Recognizer (O.B.R.), as well as the construction of a keyboard for the Braille code. This project was developed with the objective of enabling teachers of blind people, who do not know the Braille code, to visualize the texts written by their students. An electronic keyboard, less noisy and less expensive than the traditional mechanical ones was built too. To achieve these objectives, the "Compendium of the Braille code for the Portuguese Language" was used. The final program translates plain text, mathematics and chemistry sheets written in Braille code. It's also possible to write plain text, mathematics or chemistry using the developed keyboard. The program is written in Java and the keyboard communicates with it through serial port.

1 Introduction

This project appeared as the result of the educational politic of mixing blind pupils in the normal school classrooms. The Optical Braille Recognizer is therefor intended for teachers, who don't understand Braille language, for visualization of texts written by their blind students in normal mechanical emboss machines. The keyboard is mainly intended for the blind students, as the common mechanical Braille keyboards are very noisy and expensive, so the creation of a new, low noise and low cost became crucial. To accomplish this objectives, the project has been split into several tasks, which are:

1. Locating the Braille points that form the text;
2. Segmenting the image into Braille text lines;
3. Processing of the points found to extract the text;
4. Creating the parsers representing the language;
5. Constructing the keyboard and connection to the parsers;
6. Integrating the text to speech system;
7. User interface

This project is still under development, although there are interesting results, which will be discussed from now on in this article.

In the beginning of the current project there were the following constraints:

M. Lazo and A. Sanfeliu (Eds.): CIARP 2005, LNCS 3773, pp. 233–241, 2005.

1. Translate single sided Braille sheets;
2. The system should be low cost;
3. The system should be platform independent;
4. Should run in a mid range computer;
5. Use of free software;
6. Simplicity of system circuits to allow a low skilled person to assemble the keyboard (possibly in an electronics class).

2 Location of the Points

In Braille code each character is represented by six points, 3 per column and 2 per row, having standard distances between them [1]. The distance between points is 2.5mm. The distance between characters is 3.5mm in the horizontal and 5mm in the vertical. These distances define the braille cells. The characters are formed by embossed dots in any position of the cell. These are the points to be located in the scanned image, as they represent the Braille text.

The location of the points is the first stage of the project. It is very important to have a good input image. The image can be scanned with any scanner, but it should be in high resolution. The input image is crucial for the success of the Optical Braille Recognition (O.B.R.), as in a bad image the points will be fuzzy and difficult to locate.

In Figure 1 it is possible to see distinct zones in the represented points, as this is a partial scanned test image. One zone is brilliant, above the point, and the other zone is darker, under the point [2]. This is due to the emboss of the points, illuminated by the oblique light source from the scanner. This discrepancy has to be enhanced to locate the points more precisely, reducing the probability to detect false points, introduced by eventual noise in the image. Thresholding [4] twice using the Otsu method, accomplished some results, which weren't very reliable with different background colors. A more accurately method was develop that calculates the threshold points based on a given percentage of

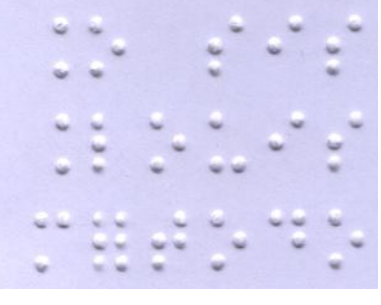


Fig. 1. Partial image of Braille Scanned sheet

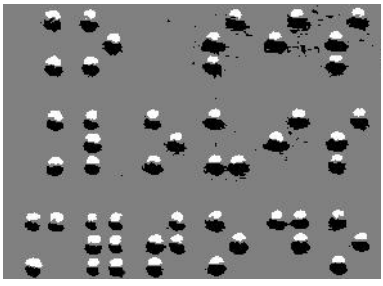


Fig. 2. Partial image after threshold process

the histogram peak[1]. This method was observed to be efficient has it permitted to work with different color sheets. Each zone, brilliant or dark, is isolated by one threshold.

After the threshold process, the image will be composed by three different color regions, as represented in the Figure 2. Now the points have to be located and marked to segment the image. A white zone, with a black zone under, is searched to locate the points. Grouping the regions this way, there is a small probability of detecting a false point. This process has some errors, as some points will not be detected because the white zone, or the black one, may not exist for all points. Although this may happen, it happens to few points, only the ones that are not well embossed by the Braille printer, or those that may be perforated.

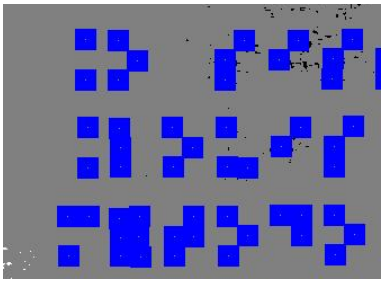


Fig. 3. Partial image after locating the points

The point's center is determined with a very simple approach, the center of mass of the point is computed. After finding the center, a blob is drawn representing the located point, as shown in Figure 3.

[1] The histogram peak represent the background of the image.

3 Isolation of the Lines of Text(Segmentation)

At this point, the text lines are isolated, so the text can be translated. The lines that do not have any blob represented are searched and marked, securing the segmentation in text lines. These will be discarded for a more efficient computing of the translations. Figure 4 represents the image after segmentation.

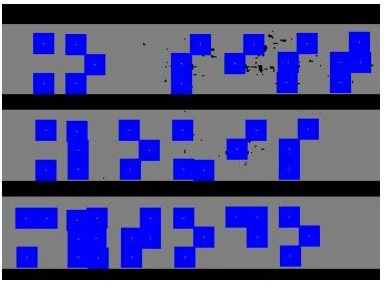


Fig. 4. Partial image after segmentation

4 Processing of the Points

With the points detected and the image segmented, the next task is to get the Braille characters. In this approach there is no need to segment the image vertically [3]. For each line of text, the first center is found. Having it's coordinates, it's position inside the cell is searched. This is done examining the distance between the current point and the next point center found. This gives the horizontal position, as the vertical one is found dividing the text line into three regions. The coordinates of the center and it's position are stored in a linked list. The rest of the line is computed, ending up with a linked list of centers and positions. The linked list is then parsed and, according to the information it holds, a mask is adjusted to each Braille character, ending up with a line of characters coded into 1's and 0's, representing respectively, position is a point and position is not a point.

This procedure is repeated for every line of text, ending up with the entire page coded.

5 Creation of the Parsers

The parsers are created to translate the text from the coded file. Three parsers were defined, one for text, one for mathematics and one for chemistry. These parsers were created for the Portuguese Braille code. There are many Braille Codes, and even inside one Braille Code there are characters that represent one character in plain text and another in e.g. mathematics. That's why three different parsers have been created.

The parsers were created using Javacc that allowed a more efficient integration with the developed application.

The parsers make use of the Unicode char set, and not the ASCII one. This solution was adopted for the use of the mathematical symbols, since most of these are not available in the ASCII set.

The input of these parsers is a string coded into 1's and 0's and the output is an HTML coded page. The use of HTML tables were very useful to enable printing of mathematical or chemical expressions.

6 Construction of the Keyboard

6.1 Motivations

The will to construct the electronic Braille keyboard came mainly from two reasons:

1. The high price of the mechanical Braille emboss machines;
2. The noisy environment created by students using mechanical machines;

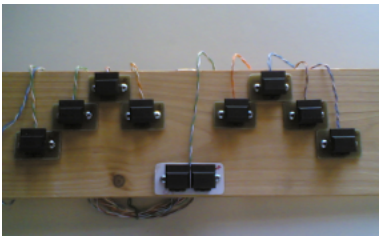
6.2 Hardware Overview

For a cost effective solution the existing school computers were thought to be used, if possible the most downgraded, that are also the most expendable ones. Having this into account the serial port was chosen to be the communication interface between the computer and the keyboard. This interface was widely used in the past and nowadays it is usually used in instrumentation and in several electronic devices, so cheap hardware and free software solutions could be created and used.

After some research [10,5] the PIC 16F628 [6] from Microchip® was chosen. This is a mid-range micro-controller, it has better specs than the widely used PIC 16F84 and the advantage that it is cheaper. The 16F PIC series is FLASH memory based and so can be erased and reprogrammed electrically. This feature allows a faster software development comparing to the UV-erasable micro-controllers. It is a RISC micro-controller with Harvard architecture, 14-bit wide instructions and 8-bit wide data word. It has 2048x14 FLASH program memory, a 224x8 RAM data memory and a 128x8 non-volatile EEPROM data memory. Contains an 8 bit ALU and working register. A total of 35 instructions are available, all are executed in one clock cycle except for program branches that take 2 cycles. The 16F628 architecture also contains some special features that can simplify the circuit system by eliminating the need of external components, reducing the cost and power consumption, and improving reliability. In the system two of these interesting specs were used: the 4MHz internal oscillator, and the USART for serial communication.

To convert the output voltage levels of the micro-controller's USART to TIA/EIA-232 (serial port protocol) a MAX232 converter chip was used.

(a) Perkins Brailler.



(b) Keyboard layout.

Fig. 5. Keyboard layouts

Microchip® provides PIC programmers that transfer the developed code into the chip's memory, but these don't accomplish the low cost objective of the project and so a cheaper solution was used, the JDM PIC Programmer 2 (JDM). Jens Dyekjær Madsen [7] created this simple, efficient and low cost programmer. It is well documented on the World Wide Web by himself and many other hardware developers, including a small circuit alteration that allows compatibility with the PIC 16F6XX series. The communication interface of JDM programmer is also the serial port.

The layout of a Perkins Brailler mechanical embosser, like the one in Figure **??**, was followed maintaining the sequence but dislocating vertically the keys for a more ergonomic positioning, as can be seen in Figure 6.2. This way the Braille students would rapidly adapt to the keyboard.

6.3 Hardware Programming Software

For software development Microchip® provides an integrated development environment called MPLAB® IDE Software. It is MS Windows® based but there are Linux alternatives.

The programmer software used to load the code into the micro-controller was IC-Prog [8], a free MS Windows® based software developed by Bonny Gijzen. There are software alternatives for Linux.

6.4 Keyboard Assembly

The first stage was assembling the JDM programmer, as seen in Figure **??**. This was a very important step for the production of the keyboard, so after it was built it was tested with IC-Prog to assert it was a reliable programmer.

The keyboard circuit was mounted on a test board to allow the PIC's programming and testing.

The code was written in assembly language, using MPLAB® for writing and debugging the program. The main routine of the program needed to simultaneously debounce all keys, since a Braille character can be produced by simultaneously pressing until 6 keys. If a pooling approach had been chosen a

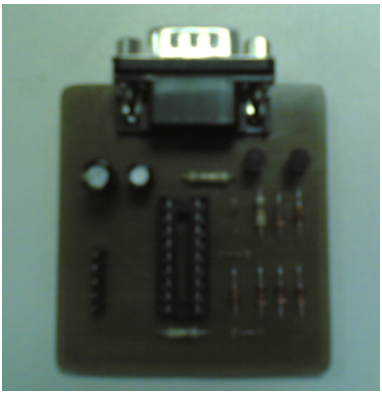
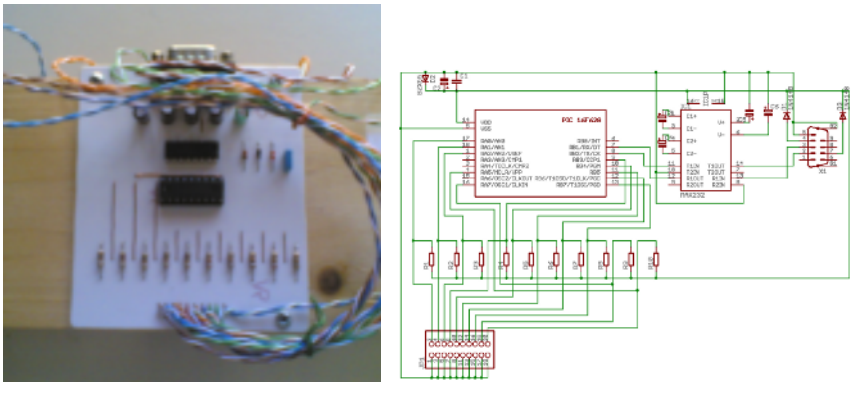


Fig. 6. JDM programmer circuit

very long, complicated and possibly ineffective program would be obtained. Instead, a routine by Scott Dattalo [9] that makes use of the notion of vertical counters was adapted. Using two 8-bit registers, grouping them in 2-bit counters in such way that e.g. the two LSBits form a counter, associating a key entry to each counter in such way that if the state of the key is maintained after 4 iterations the state is filtered. If it doesn't maintain the logical value, the 2 bit counter resets. One register accumulates key presses until no key is being pressed, in that moment the composed value of all valid key presses is sent via the PIC's USART.

After loading the final program to the PIC's memory all the components were mounted in a PCB circuit board, as can be seen in Figure 7, and the keyboard assembly was completed.

To finalize, the application developed for O.B.R. was linked to read from the serial port. This application needs to have an active parser mode, so a keyboard



(a) Final keyboard circuit. (b) Final keyboard schematic.

Fig. 7. Final keyboard circuit and schematic

user must send a code composed by the space, the delete, and one letter keys simultaneously pressed. The letter codes are "m" for mathematics (matemática), "q" for Chemistry (química) and "t" for text (texto).

7 Text to Speech System

A Text to Speech System was linked to the developed O.B.R. software. This has been done to enable the user of the software to hear what is being typed in real time. This system had already been developed in a former work [11]. Originally developed for the Linux platform, the program code was changed to compile also under Microsoft Windows®. Some changes have been made so the program could function more accordingly to our objectives.

The software can only synthesize text. For a more complete system, the rules of the text to speech program should be altered. This may be difficult to implement since this system was originally developed for plain text.

8 User Interface

The user interface has been programmed in Java. It permits the user to open the scanned image, and select an area of the image to be converted, also selecting the text, mathematics or chemistry

parser. It permits to get the input from the developed keyboard, allowing the user to type and hear the text. The final user interface program can be seen in Figure 8.

9 Conclusion

Some good results have been obtained so far. The developed software can translate quite accurately the texts in the Braille sheets. However some errors may occur because of bad input image. Some points don't have the white zone or the black one, which will produce bad results in the detection. The input image is very important for the success of the translation, so a good color high resolution image should be obtained. It is also expected that some scanners performed better than others because of different illumination conditions.

A text to speech software system was adapted to the program. This solution is limited since it was developed for plain text, however is very useful, because the user can't feel the output like in a mechanical embosser.

An efficient low cost keyboard was successfully developed. A mechanical embosser would cost about 1000 Euros in Portugal, while the construction of the keyboard and the PIC programmer would only cost about 50 Euros.

At this moment, the software translates an entire page of plain text under 15 seconds in an AMD Athlon 1.41 GHz. It can translate successfully plain text, mathematics and chemistry. However further work should improve the parser's rules, possibly with the aid of Braille teachers and students.

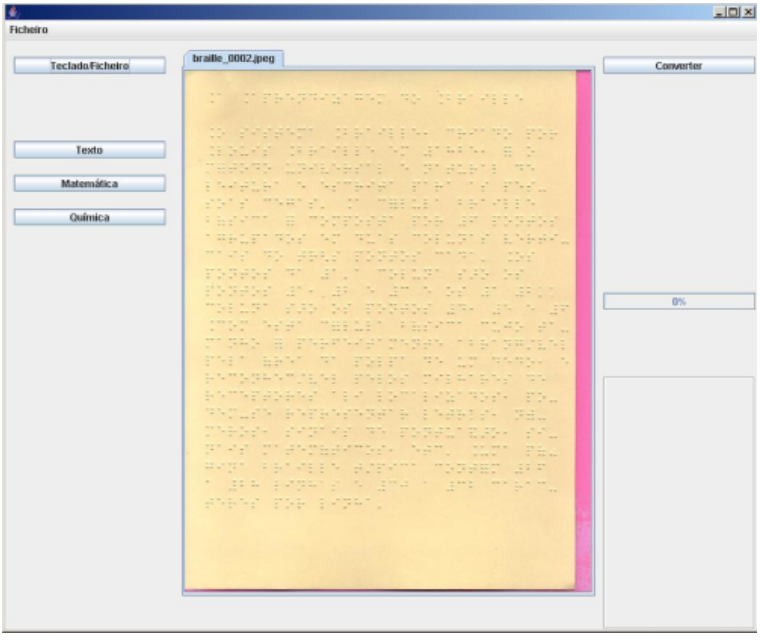


Fig. 8. User interface

References

1. Hermida, X. F., Rodriguez, A. C., Rodriguez, F.M.: Braille O.C.R. for Blind People. Proceedings of ICSPAT-96 (1996).
2. Ritchings, R.T., Antonacopoulos, A., Drakopoulos, D.:Analysis of Scanned Braille Documents. Document Analysis Systems (1995) 413-421.
3. Wong, L., Abdulla, W., Hussman, Stephan: A Software Algorithm Prototype for Optical Recognition of Embossed Braille. 17th International Conference on Pattern Recognition (2004)
4. Russ, J. C.: The Image Processing Handbook. 3rd edition (1998).
5. Microchip: www.microchip.com.
6. 16F628 PIC Datasheet: ww1.microchip.com/download/en/DeviceDoc/40300c.pdf.
7. Madsen, J. D.: www.jdm.homepage.dk.
8. IC-PROG: www.ic-prog.com.
9. Dattalo, S.: www.dattalo.com.
10. PicList: www.piclist.com.
11. Tomaz, F.: w3.ualg.pt/˜ftomaz.