OPTICAL BRAILLE TRANSLATOR FOR SINHALA BRAILLE SYSTEM: PAPER COMMUNICATION TOOL BETWEEN VISION IMPAIRED AND SIGHTED PERSONS

T. D. S. H. Perera, and W. K. I. L. Wanniarachchi*

Department of Physics, University of Sri Jayewardenepura, Nugegoda, Sri Lanka
*iwanni@sjp.ac.lk

ABSTRACT

In this paper we proposed a system; Optical Braille Translator (OBT), that identify Sinhala Braille characters in single sided Braille document and translates to Sinhala language. This system also capable of identifying Grade1 English Braille characters, numbers, capital letters and some words in Grade 2 English Braille system. Image processing techniques were used to developed the proposed system in MATLAB environment. The translated text displayed in a word application as the final outcome. Performance evaluation results reflect that the proposed method can recognize Braille characters and translated to user selected language either Sinhala or English efficiently, over 99% of accuracy.

KEYWORDS

Braille, Braille Recognition, Image Processing, Optical Recognition, Sinhala

1. INTRODUCTION

According to the thirteenth Census of Population and Housing survey which was conducted on 17th July, 2001 in Sri Lanka, the number of visually impaired people has reached 69,096 [1]. Out of that 35,419 were males and 33,677 were females. Among them 10,947 people out of the 69,096 were totally blind. Therefore, it is necessary to provide support those people with intelligent systems and technologies to improve communication and interaction with each other and with non-blind people. The major senses used by visual impaired people are hearing and touch feelings. The most famous communication system for blind people is the Braille system which depends on the sense of the touch of a fingertip. Braille is a system that allows visually impaired people to read through touch using a series of raised dots on special papers which can only be read using fingers. Braille is not a language and these Braille characters are used to specify character in any language [2].

Braille coding system made up of different type of characters which are also called “cells”. Each Braille character or a “cell” is made up of six dot positions arranged as two columns of three dots to form a rectangular shape. A dot may be raised at any of these six positions to form sixty-four combinations including the combination which no dots are raised. Positions of these dots are universally numbered 1 to 3 from top to bottom on the left, and 4 to 6 from top to bottom on the right [3]. The dimension of a Braille dot, distance between dots in a cell and distance between cells have been set according to the tactile resolution of a fingertip [3]. The horizontal and vertical distance between dots in a cell and distance between cells in a word and inter line distance also specified by the Library of Congress. Here, dot height is approximately 0.02 inches (0.5 mm), the horizontal and vertical spacing between dot centers within a Braille cell is

DOI: 10.5121/ijma.2018.10303

approximately 0.1 inches (2.5 mm), the blank space between dots on adjacent cells is approximately 0.15 inches (3.75 mm) horizontally and 0.2 inches (5.0 mm) vertically. A standard Braille page is 11 inches by 11.5 inches and typically has a maximum of 34 to 40 Braille cells per line and 25 lines per page [4]. The Braille has been adapted to write many different languages including Sinhala, also it is used for musical and mathematical notation. Sinhala and English Braille letters are read from left to write.

When complexity of Sinhala Braille is compared with the English Braille, Sinhala Braille system can be categorizing as a grade 01 Braille system. Because Sinhala Braille having one to one transcription with Sinhala letters. Rarely two Braille characters are used to represent single Sinhala letter such as "ඓ, ඖ, ඍ and ඎ". When the Sinhala Braille system is compared with the Sinhala language, some characters are missing in the Braille system. For an example, when the word "අම්මා" is written using Sinhala Braille "⠠⠁ ⠠⠍ ⠠⠍ ⠠⠜" which look likes "අම ් මා". Because character "ා" is not used in Sinhala Braille system. Therefore, pronunciation sound of the Sinhala word is used to write that word in the Sinhala Braille system.

However, most people in the society cannot understand Braille. Paper communication between the visually impaired people and non-blind people have become a problem that need to be addressed. Therefore, translating Braille into Sinhala or any other languages enable communication with people in society. In this research work, we developed a Braille translator system which can translate Braille characters into Sinhala language. The prosed system has been improved to identify grade1 English Braille characters, numbers, capital letters and some words in grade 2 English Braille system. An image of a single sided Braille paper is taken as the input to the system. Evaluation of the performance of the system showed that it can recognize Braille characters and translating to Sinhala/English language over 99% of accuracy. The developed system uses simple image processing techniques of low computational power and performs well over the other published work. In the paper we comprehensively discuss the image processing methods that we used to develop optical Braille translator (OBT).

2. PREVIOUS WORK

In the literature there are many researches have been carried out for Braille character recognition based on image processing techniques. In the paper "Smart Braille System Recognizer", authors claimed that the developed system can recognize characters in single side Braille document with 94.39% accuracy. Image acquisition stage, image pre-processing, modified image segmentation, feature extraction, and character recognition based on image processing techniques were the main staged of their work. At the image acquisition step authors used flat-bed scanner to obtain images of single side Braille documents [3]. On their work, J. Li *et.al.,* optical Braille recognition system used normal scanner to aquire the input Braille documents' images. Geometrical corrections were applied in preprosessing stages. Haar wavelet feature extraction and Support Vector Machine classification techniques were perforemed on croped sub images of Braille dots for identification. Identified Braille cells were converted to English language with aid of searching algorithm. Authors claims that the developed method is in acceptable level for Braille extraction [5]. M. Wajid *et. al.* developed Braille to Urdu language translation system based on image processing using MATLAB. After the pre-processing and segmentation steps,a 3×2 matrix was generated according to the dot pattermn available in a Braille cell. Here, they used a threshold value where the number of white pixels in selected region of the Braille cell greater than the threshold then that element in the matrix represent a Braille dot. Generation of this pattern matrix in terms of 0's and 1's was used to link corresponding letters in Urdu language [6]. L. Wong *et. al.* proposed a Braille recognition system based on image processing and probabilistic neural network. The statistics of performance evaluation of the system shows that the accuracy is 99% [7]. K.P.S.G. Sugirtha *et. al.* proposed a method of translating braille code into English language. In their work, simple

flat-bet scanner used to acquire image of the braille documents. Then image pre-processing steps including expel alignment, gray scaling, thresholding and dilation were performed on the subjected image. Author claimed that in the segmentation step of their work they considered the Euclidean distance to evolve new technique to recognize the characters [8]. The research work carried out by the E. Jacinto Gómez *et. al*. has clamied thet they have used very unique method to identify braille characters. In their work they used circle hough transform method to identify dots in image of the braille document [9]. In the paper "Braille Character Recognition Using Associtive Memory" authors S. H. Khaled *et. al.* has translated braille document into English language and voice. Their resech consisting two main stages, preprocessing stage and recognition stage. In the second stage Modify Multy-Connect Architecture (MMCA) and Modify Bidirectional Associative Memory (MBAM) algorithms were implemented. MMCA algorithm has achived araerage accuracy for correct letter is 98.26%, average correct word was 95.11% and average processing time around 11.5 seconds per page. MBAM algorithm achived aearage accuracy for correct letters is 91.87%, average accuracy for correct word was 51.26% and average processing time around 3.4 seconds per page [10].There are very few work previously done for recognizing Sinhala Braille letters. The research work carried out translating Sinhala text document into Braille by Soma Chatterjee [11]; the proposed system can convert MS word-based Unicode Sinhala document to Braille. Recently in 2016, N.M.T De Silva *et. al.* proposed a system to convert Braille to Sinhala characters. On their work, K-nearest neighbor classification method was used for identification of Braille characters. The outcome can implement Unicode mapping for 52 Sinhala characters and 10 numbers. The test results reflected that the developed system could translate Braille characters to Sinhala language with 91.4% accuracy [12].

3. METHOD

This research work is based on image processing techniques where computer algorithms implemented on MATLAB environment. Optical Braille Translator (OBT) is a system that identify the Braille characters in an image of a one side Braille document and translate them into corresponding natural language. Image can be a scanned image of a Braille document or color image taken by a camera. Finally, each and every extracted Braille character is translated into corresponding letter in the Sinhala language. Also, OBT has the ability to identify Grade 01 English Braille characters and some words (e.g. and, for, of, the, with…etc.) in English grade 2 Braille [13]–[15]. Furthermore, the OBT system is capable of finding numbers in both Sinhala and English Braille document and capital letters in grade 1 English Braille document. Final output is written to a Microsoft Word document[16]. A simple graphical user interface has been designed for user interaction. The main steps in the developed OBT system is shown in the figure 1. Each of the step according to the system flow chart in figure 1 is discussed comprehensively in the following sections.

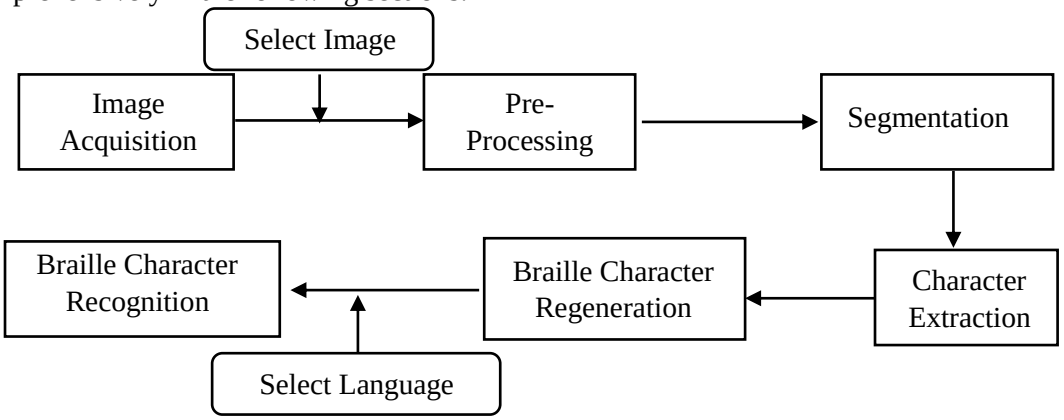
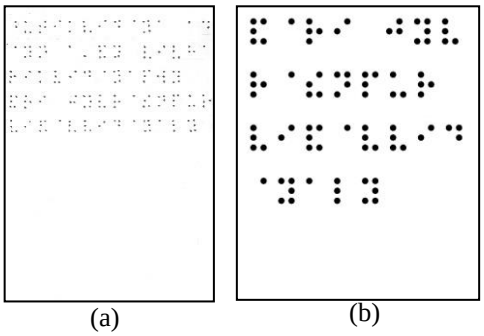


Figure 1: Flow chart of the developed OBT system

3.1 Image Acquisition

Image acquisition is the manual step of this system and the accuracy and time taken to translate the Braille document into a natural language mainly depend on the quality of the acquired image. In this research, two different types of methods were used to get an image of a Braille



(a) (b)

Figure 2: (a) Scanned image of a hand-written braille document (b) Computer generated braille document

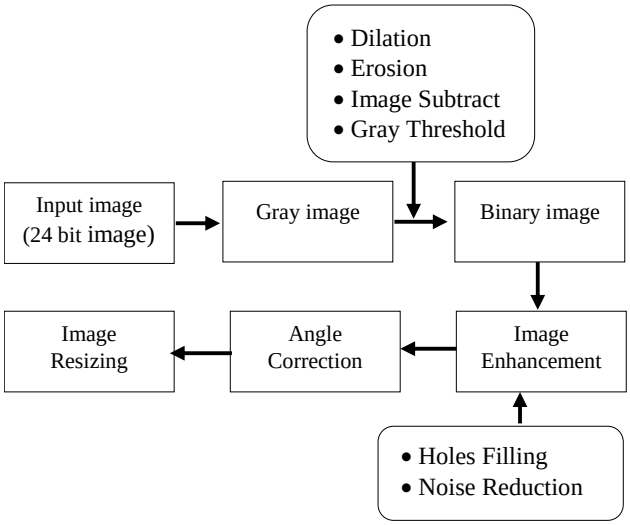


Figure 3: Sub steps of the image pre-processing

document: A hand written Braille document (a Braille document where Braille dots were written by pen) was scanned by a normal scanner, and an image of a Braille document was generated by a computer. Figures 2 (a), and 2(b) show the selected images for the further steps in this system. Here, the input image is a true color RGB image (24-bit image).

3.2 Pre-Processing

Pre-processing step consists of several sub steps such as gray image processing, binary image generation, angle correction and image resizing functions which are performed on the input image. The figure 3 shows the sub steps of the pre-processing step. The acquired image in the initial step is input to the image pre-processing routing. Then, the color image is converted into grayscale image. Image dilation, erosion, and image subtraction were performed before converting gray image into a binary image[17]. In order to obtain the edges of the image, image dilation and erosion operations were carried out. Here, morphological disk of radius 1 structural element [0 1 0;1 1 1; 0 1 0] was used for dilation and erosion operation[18]. Finally, edges of the foreground objects were successfully obtained by subtracting dilated image from the eroded image. Then the global threshold was considered to obtain the binary image. For the image enhancement, holes filling and noise reduction techniques were used. Figure 4. (a) shows the binary image obtained according to the global threshold which consists of noises whereas figure 4. (b) shows the hole filled binary image. In order to reduce the noises from the image, MATLAB function “bwareaopen” was used. Here, we removed objects that have fewer than 10 connected pixels and obtained the noise removed binary image (figure 4. (c)). Before proceed to the segmentation process, correct alignment of the Braille characters was processed. Figure 5 shows results of the angle correction obtained from the Radon transformation [19]. In order to decrease the computation time for further the processing, the correct alignment image is resized to lower resolution which consists of 480 rows while keeping original image aspect ratio.

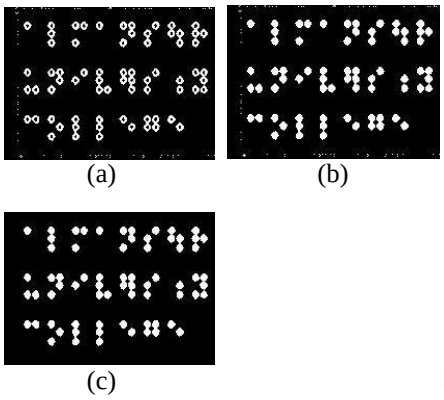


Figure 4: Image enhancement (a) binary image, (b) Hole filled binary image (c) noise reduced binary image

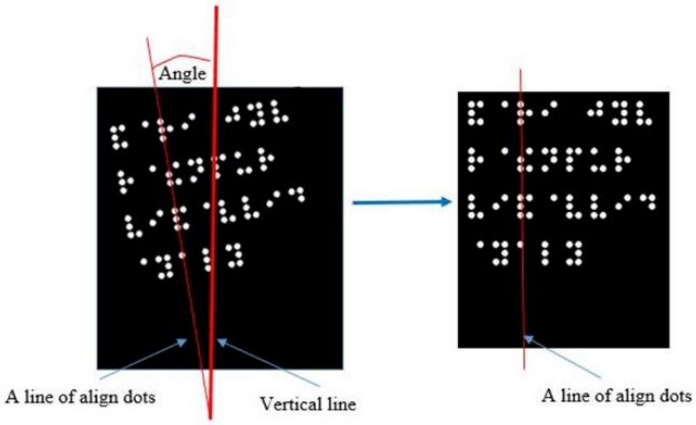
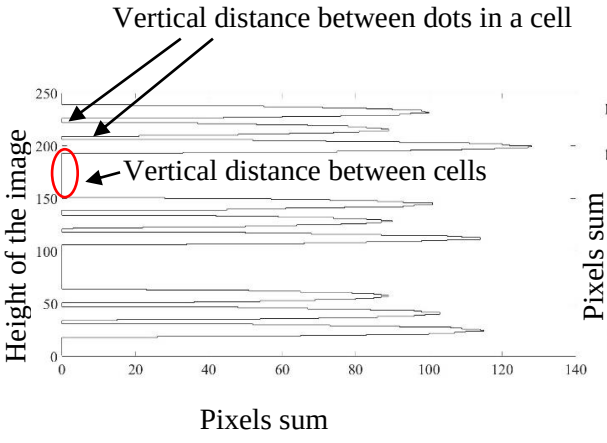


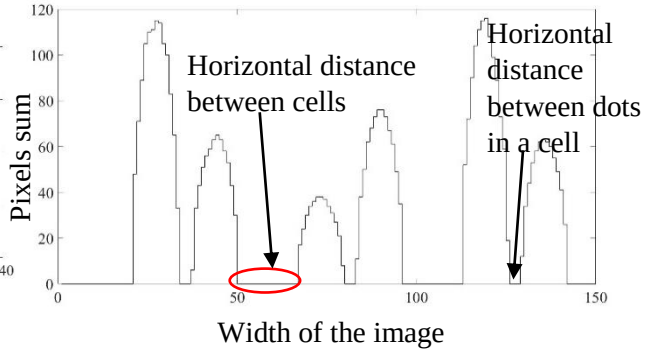
Figure 5: Angle correction using Radon transformation. Initial image (left) and angle corrected resized image (right)

3.3 Braille Character Segmentation

Summation of pixels' value took along the rows and columns of the noise removed binary image was taken into account in order to identify Braille character cells. Figure 6. (a) and figure 6. (b) show row sum and the column sum of an input binary image. Row summation (Figure 6. (a)) was used to identify the average vertical distance between two rows. An example is highlighted in red circle and average vertical distance between two Braille dots in a cell (shown in arrows). According to the row summation shown in figure 6 (a), the smaller zero count vertical gaps reflect the separation of two Braille dots in a given cell while the larger zero count vertical gaps reflect the separation of adjacent character rows. Similarly, in column sum (figure 6. (b)) larger horizontal zero count gaps represent the separation between Braille character columns while smaller zero count gaps reflect the Braille dot separation in a given cell. Accordingly, a computer algorithm was developed to find the approximate vertical and horizontal position of characters automatically (see figure 7). At the end of the segmentation step, there are two sets of data points, one represents the segmentation along the vertical direction while other represents the segmentation points along the horizontal direction.



(a) Sum of pixels' value along rows



(b) Sum of pixels' value along columns

Figure 6: Stair graph of row sum and column sum

$$y_i = \sum_{j=1}^{l_x} n_j \qquad \left(n_j = \{^{0\ (black\ pixel)}_{1\ (white\ pixel)}\right) \qquad (1)$$

l_x = length of the image by pixels (number of columns)
y_i = horizontal projection of the pixels (summation along the rows of pixels)

$$x_j = \sum_{i=1}^{l_y} n_i \qquad \left(n_i = \{^{0\ (black\ pixel)}_{1\ (white\ pixel)}\right) \qquad (2)$$

l_y = height of the image by pixels (number of rows)
x_i = vertical projection of the pixels (summation along the columns of pixels)

(i,j = vertical and horizontal coordination of the pixels)

3.4 Character Extraction

In the Braille character segmentation step, the upper and lower row separation positions and the upper and lower column separation positions were obtained. In the character extraction, first rows were separated from the noise removed binary image according to the row separation positions. Then the Braille characters were extracted by cropping the row character images as per the values obtained by the column separation position. The extracted characters were resized to 21×16 matrix binary images. At the end of this process, the Braille character cells were successfully extracted from the initial RGB input image.

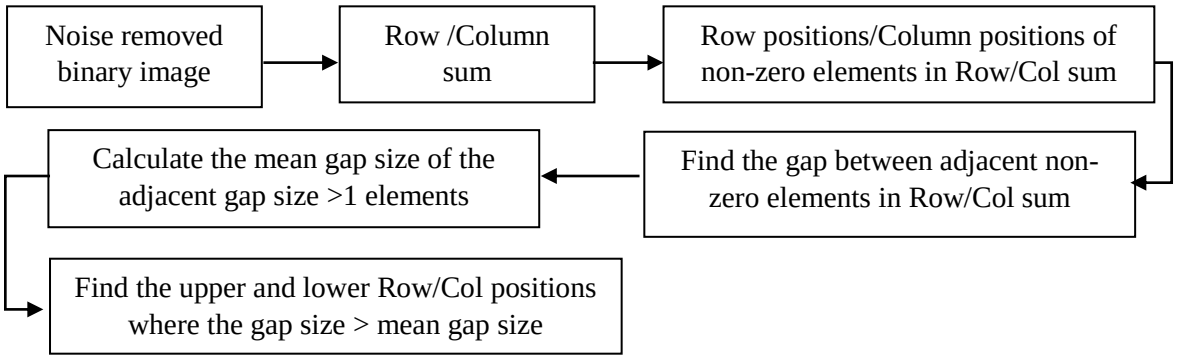


Figure 7: Main steps of the Braille character segmentation

3.5. Braille Character Regeneration

In order to increase the accuracy of Braille character recognition, we regenerated the Braille cell image according to the extracted Braille character images. Figure 8. (a) shows the extracted 21×16 matrix binary image of a Braille character. For the regeneration process, the column width was divided into three columns which have width of 5, 6 and 5 pixels while row height divided in to 5, 3, 5, 3 and 5 pixels. The figure 8. (b) graphically illustrates the separate region in the Braille character for the regeneration process. Here, a single Braille dot in a cell is represented by a 5×5 matrix element. Accordingly, there are six 5×5 matrix regions (region B) as shown in the figure 8. (c). In practical cases, we have observed that the Braille dots may not centered in these 5×5 matrix regions as shown in figure 8. (b). Hence, identification of Braille characters by a computer system may be time consuming without regenerating the Braille cell properly. The developed algorithm for regeneration of Braille characters is shown in figure 9. Here, the number of white

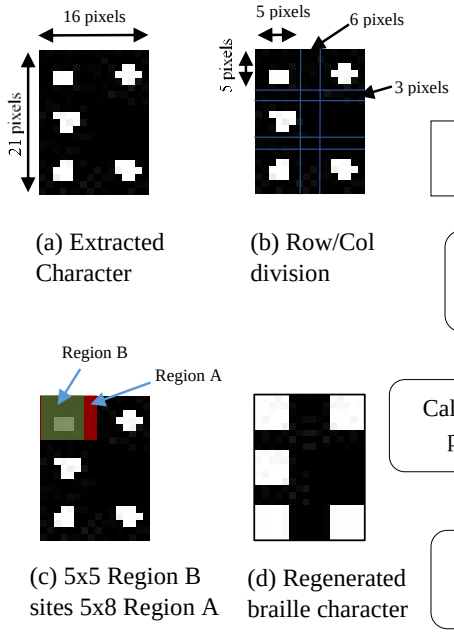


Figure 8: Braille character regeneration process

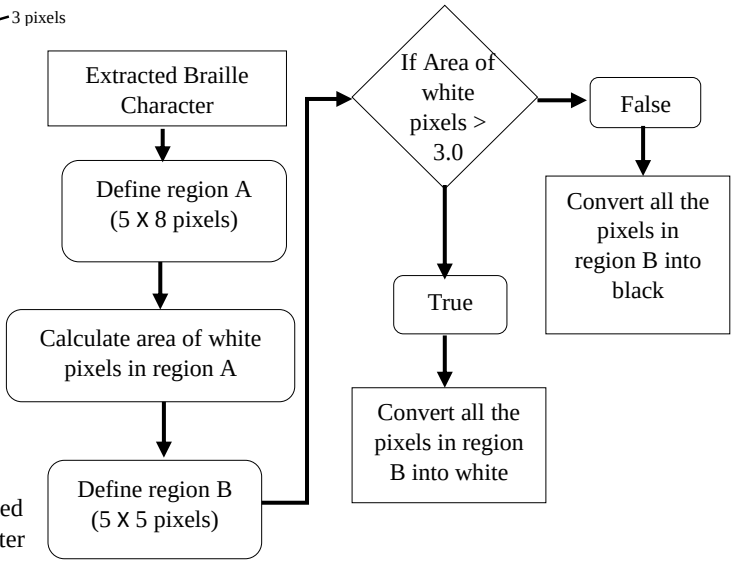


Figure 9: Braille character regeneration algorithm

pixels in 5×8 area (shown in the figure 8. (c) region A) which covers 5×5 matrix (region B), is taken in to account when deciding the Braille dot formation. If the number of white pixel in region A of any region B sites, then corresponding region B site converted to white color; else converted to black color. The final output of Braille character regeneration of a tested image is shown in figure 8 (d) which reflects the enhanced quality image for character identification.

3.6. Braille Character Recognition

In this step, binary to decimal equivalent number is incorporated for identification of each Braille character. As shown in figure 10., value of each midpoint of the Braille dot regions in the regenerated Braille cell were used to generate a 6-bit binary number. The binary equivalent decimal number can be obtained from $D5\times2^5+D4\times2^4+D3\times2^3+D2\times2^2+D1\times2^1+D0\times2^0$ arithmetic operation. The middle point of the Braille dot location one was taken as the least significant bit and the middle point of the Braille dot location six assigned to the most significant bit. The developed computer program takes the value of each midpoint located at (3,3), (11,3), (19,3), (3,14), (11,14) and (19,14) in the regenerated Braille character image to make digital representation of the Braille character where this digital representation 6-bit binary number identical to corresponding Braille character. The figure 11. indicates binary equivalent decimal numbers as a 2D array for a given input image consist of 32 (4×8) Braille characters. Hence, corresponding binary to decimal equivalent number can relate to characters in natural languages. The Sinhala and Grade I English letters and some of the Grade II English words with binary equivalent decimal number for corresponding Braille character are shown in the figure 12. Here, we used this database to decode the Braille characters in the proposed system.

The binary equivalent decimal numbers corresponding to Braille characters are shown in top of each cell in figure 12. The corresponding Sinhala and English letters (some words in Grade II English) are shown in black color. Green color letters (strings) represents the corresponding English keyboard characters in “AA Amali” font type which is discussed in section 3.7.

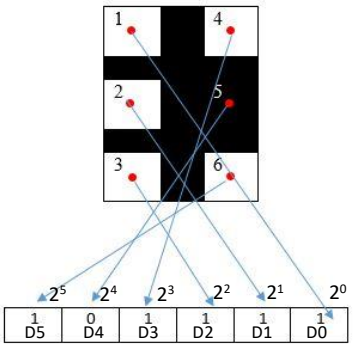
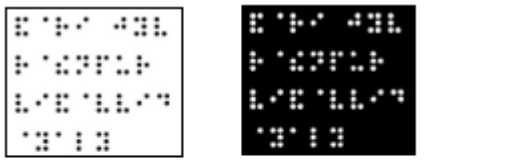
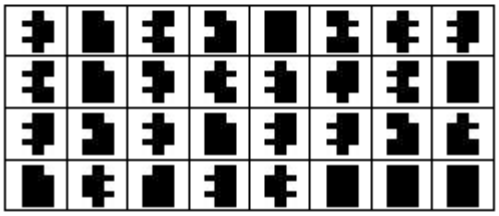


Figure 10: 6-bit binary number relevant to Braille character



(a) original image (b) noise removed image



(c) regenerated braille characters

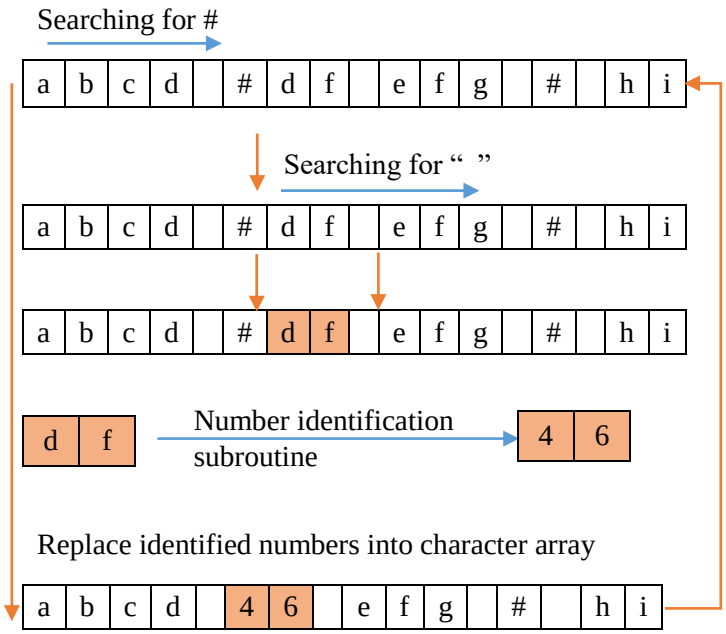
47	8	23	10	0	26	61	39
23	8	46	29	15	37	23	0
39	10	47	8	39	39	10	25
8	61	1	7	61	0	0	0

(d) binary equivalent decimal number array

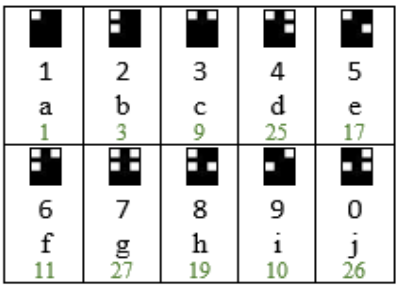
Figure 11: Binary equivalent decimal number array for a test image consists of 32 Braille characters

0 space	1 අ, a w	2 ,	3 බ, b n	4 ◌ං, ’ x	5 ක, k l	6 ;	7 ල, l ,	8 ් a	9 ච, c p	10 ඉ, i b	11 ෆ, f *	12 ඵ, / ft	13 ම, m u
14 ස, s i	15 ප, p m	16	17 එ, e t	18 ඤ, : [	19 හ, h y	20 ඊ, in B	21 ඕ, o Ts	22 !	23 ර, r r	24 භ N	25 ද, d o	26 ජ, j c	27 ග, g .
28 ආ, ar wd	29 න, n k	30 ත, t ;	31 ඥ, q {	32 @	33 ඡ, ch P	34 ඒ, en ta	35 ඝ, gh >	36 _	37 උ, u W	38 “	39 ව, v j	40 ඛ L	41 ෂ, sh I
42 ඖ,ow T!	43 ඩ, ed v	44 ඬ,ing X	45 ඔ, x T	46 ධ, the O	47 ශ,and Y	48	49 එ,wh M	50 .	51 ඌ,ou W!	52 ඪ, ” CO	53 ණ, z K	54 ( )	55 ඇ, of we
56 ළ <	57 ථ, th :	58 ඨ, w G	59 ඈ, er wE	60 #	61 ය, y h	62 ට,with g	63 ඹ, for V						

Figure 12: OBT Database for Decoding Braille Characters



(a) Searching for numbers



(b) 0-9 numbers in Braille characters and corresponding English letter with binary equivalent decimal number in green color

Figure 13: Number identification process in Braille to English translation

The proposed optical Braille translator system is also capable of identifying numbers. In Sinhala Braille system as well as in English Braille system "⠼" Braille character (binary equivalent decimal number is 60) is used to represent numbers. In this work, the corresponding character "#" was assigned to represent "⠼". If a space followed by #, then it represents the character #. If a character(s) followed by the #, then it represents a number. The proposed system first converts Braille characters in to user define language either Sinhala (see section 3.7 for more detail) or English. Then the developed algorithm searches for "#" character by scanning identified characters. When character "#" found, then searches for " " (space) in the array and all the characters in between # and " " converts to the relevant number. Then the identified numbers replaced to the correct position in the original character array. This process repeats until the end of array is reached. In the figure 14, steps for the number identification is visualized. In grade I English Braille uses " ⠠" Braille character (binary equivalent decimal number is 32) to represent capital letters. This Braille character is identified as "@" sign in our proposed system. Any character followed by "@" converted to capital letters and the capital letter searching function is developed similar to the number identification process shown in figure 13. This subroutine is applied for Braille to English translation only.

3.7. Braille to Sinhala Translation

The OBT developed in this work capable of translating Sinhala Braille document to Sinhala letters in "AA Amali" font [20] in a Microsoft word document. After obtaining the binary equivalent decimal number array, corresponds to the input Braille document, identification of natural language character is proceeded. For the identification process, the matching character to the decimal number is selected as per the database shown in figure 12. But when converting to Sinhala language the relevant English keyboard character or string in "AA Amali" font type is used in MATLAB environment according to the database based on figure 12. As an example, if the obtained binary equivalent decimal number array is [28, 25, 23, 61], then the system stores the corresponding text array as [wd, o, r, h] in MATLAB environment. Here, we used English

```
wd = actxserver('Word.Application');
document = wd.Documents.Add;
selection = wd.Selection;
selection.Font.Size=12;
if strcmp(language,'Sinhala')
    selection.Font.Name='AA Amali';
else
    selection.Font.Name='Times New Roman'
end

selection.TypeText(text_array);
```

(a) MATLAB code for sending text_array to MS Word

Braille Character	⠭	⠧	⠗	⠓
Decimal Number	28	25	23	61
English representation in "AA Amali" (text_array)	wd	o	r	h
Sinhala Letter	ඳ	ෙ	ර	ය

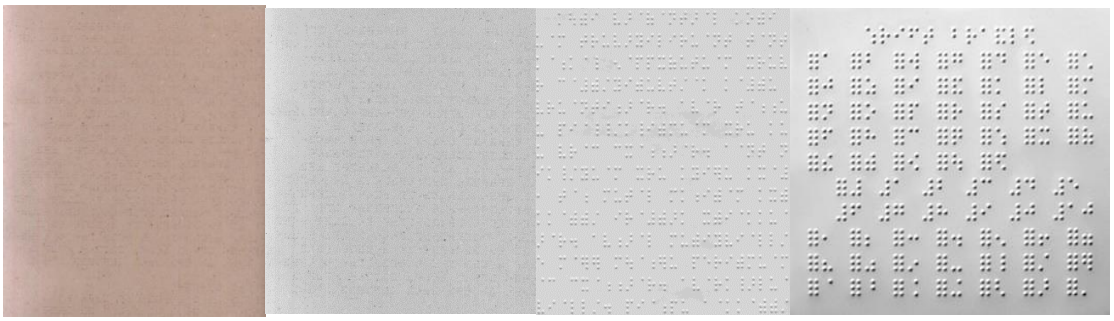
(b) An example of "text_array" corresponds to the Braille characters

Figure 14: Sinhala translated "text_array" in MATLAB Environment

keyboard letter/string representation of "AA Amali" font type when translating decimal number to Sinhala letter in MATLAB environment. Then the text array sent to word application where it displays the text array as "ඳොදරය" in selected font type "AA Amali". Before sending the text array into word application, number identification subroutine is called to identify the numbers in the text array. As illustrated in the figure 13. (a), number identification first searches for character "#". Any character followed by #, is converted to the relevant number. In this case, the text array consists of English keyboard letters/strings of "AA Amali" font type. Hence, the backward relation has considered to identify the numbers. In "AA Amali" font use following characters; [c, w, n, p, o, t, *, ., y, b] to represent 0-9 numbers respectively. The figure 14. (a) shows the MATLAB code used for sending text array from MATLAB environment to word application. Figure 14. (b) shows an example where the text array keeps corresponding English keyboard letters of "AA Amali" font type in MATLAB environment. After sending it to word application, the text array display in Sinhala letters (in AA Amali font type) by successfully completing the Braille to Sinhala translation.

4. RESULTS AND DISCUSSION

In this section, we present results obtained by the developed Optical Braille Translator system. Many researches claimed that their work on Braille character recognition using scanned Braille documents were success [4], [5], [21]–[23]. However, in our case it turned out that the scanned image of an available Braille documents cannot processed further due to less intensity differentiable of the Braille dots with respect to background.



(a) (b) (c)

Figure 15: Scanned images of Braille documents (a) Our work (b) Scan image in Ref [24] and (c) Scan image in Ref [25]

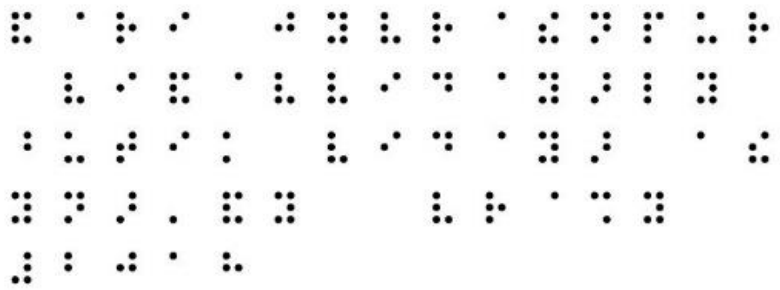
The first two images in figure 15(a) acquired by scanning available single sided Braille documents. As Shown above in our work we were unable to acquire detail images of the Braille documents by scanning method. Hence, we initiated with scanned Braille documents where Braille dots were written by pen and computer-generated Braille documents (screenshots of word documents) for the image acquisition (see figure 2). Then the developed method was tested with many scanned images found in internet (figure 15 (b) and (c)), where the OBT system successfully extracted the Braille characters. The developed OBT system worked over 99% accuracy by regenerating 319/322 and 220/220 Braille characters correctly in figure 15(b) and figure 15(c) respectively.

The performance of the developed OBT system was tested with different input images. Summary of the performance evaluation is presented in the table 01. Scanned handwritten and computer-generated Braille documents were tested in different resolution and different number of Braille characters. Most of the time, in both types the developed system successfully translated Braille characters to Sinhala or English language with 100% accuracy. The tests were executed on Intel Core i7-7500U CPU @ 2.70GHz – 2.90GHz, 8GB RAM, Windows based machine. Each tested image in the table 01 executed 10 times.

Table 1: Performance of the Optical Braille Translator System

Image size Resolution Size (kB)	No. of Braille characters Handwritten /Computer generated	Language Selection	No. of characters correctly identified	Average time taken to identify characters (seconds)	Average time taken to write characters into word file (seconds)	Accuracy (%)
371×450 96dpi 21.5 kB	27 Com.gen.	Sinhala	27	7.16	1.06	100
594×302 96dpi 22.4 kB	50 Com.gen.	Sinhala	50	12.31	1.08	100
1160×1406 300dpi 215 kB	27 Com.gen.	Sinhala	27	17.18	1.07	100
3306×4676 96dpi 525 kB	63 Handwrit.	Sinhala	63	17.71	1.06	100
1856×944 300dpi 232 kB	50 Com.gen.	Sinhala	50	12.84	1.13	100
606×194 96dpi 15.9 kB	26 Com.gen.	English	26	7.35	1.10	100
1894 × 606 300dpi 114 kB	26 Com.gen.	English	26	8.05	1.13	100
2480×3507 300dpi 230 kB	46 Handwrit.	English	46	14.19	1.07	100

Next, Braille to Sinhala language translation process is discussed. The input Sinhala Braille image (consists of 66 Braille characters) is shown in figure 16(a). The last four Braille characters indicate a four-digit number as they follow "⠼" Braille character at the beginning of the last row. The proposed OBT system successfully extracted all 66 Braille characters and obtained the binary equivalent decimal number array which is shown in figure 16(b). In this work our main objective was to translate Sinhala Braille document to Sinhala language and display the identified Sinhala characters on a word application. Here, we used "AA Amali" Sinhala font to write text in word application. Hence, when relating obtained decimal values to Sinhala letters, English keyboard letters of "AA Amali" font was used in MATLAB environment. The figure 13 (section 3.7) shows database used for this research work where the English keyboard letters of "AA Amali" font displayed in green color.



(a) Input Sinhala Braille Image

47	8	23	10	0	26	61	39	23	8	46	29	15	37	23
0	39	10	47	8	39	39	10	25	8	61	28	7	61	0
24	37	30	10	5	0	39	10	25	8	61	28	0	1	46
61	29	28	4	47	61	0	0	39	23	8	41	61	0	0
60	3	26	1	19	0	0	0	0	0	0	0	0	0	0

(b) Corresponding decimal numbers

Y	a	r	b		c	h	j	r	a	O	k	m	W	r
	j	b	Y	a	j	j	b	o	a	h	wd	,	h	
N	W	;	b	l		j	b	o	a	h	wd		w	O
h	k	wd	x	Y	h			j	r	a	I	h		
#	n	c	w	y										

(c) Text array corresponding to decimal numbers in MATLAB environment

ශ	්	ර	ී		ජ	ය	ව	ර	්	ධ	න	ප	උ	ර
	ව	ී	ශ	්	ව	ව	ී	ද	්	ය	ආ	ල	ය	
භ	උ	න	ී	ක		ව	ී	ද	්	ය	ආ		අ	ධ
ය	න	ආ	ං	ශ	ය			ව	ර	්	ෂ	ය		
	2	0	1	8										

(d) Final output text array MATLAB environment

Figure 16: Sinhala Braille to Sinhala Language Translation in MATLAB

Accordingly, the figure 16(c) shows the text array corresponds to decimal values. Then the number identification subroutine was called to identify numbers in the text array. This subroutine converted the characters in between "#" and next immediate " " (space) to numbers. In "AA Amali" font use following characters; [c, w, n, p, o, t, *, ., y, b] to represent 0-9 numbers respectively. Accordingly, the output text array is shown in figure 17(d). Then this text array sent to word application by selecting "AA Amali" as the font type. The word application displayed the text array in Sinhala langue correctly as "ශ්‍රඉ ජයවර්ධනපුඋර වඉශ්වවඉද්‍යඇලය හඋතඉක වඉද්‍යඇ අධයනඇංශය වර්ෂය 2018". Since some characters not present in the Sinhala Braille system with respect to Sinhala language, pronunciation sound of the Sinhala word is used to write word in the Sinhala Braille system as in above case.

In order to interact with users, a simple graphical user interface (GUI) was developed for the proposed OBT system.

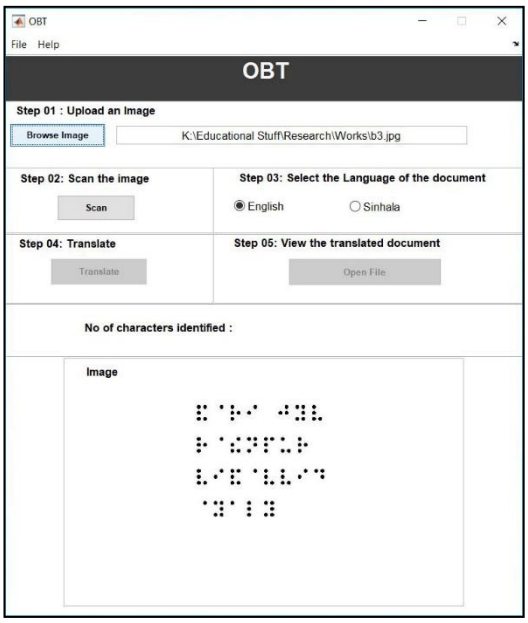


Figure 17: OBT GUI

Here, the user can select image of a Braille document in .jpg,.bmp and .png supporting formats. The selected image will be visualized on the GUI itself. The current version supports Sinhala application.

5. CONCLUSIONS

The developed Optical Braille Translator system is capable of translating Sinhala Braille to Sinhala language or Grade I English Braille to English language over 99% accurately. Further, OBT successfully recognized numbers in both Sinhala Braille and Grade I English Braille. Some characters/words in Grade II English Braille and capital letters in both Grade I and Grade II English Braille can be translated as well. In conclusion OBT system was successfully implemented to facilitate communication between visually impaired and sighted persons.

ACKNOWLEDGEMENT

The authors would like to thank the University of Sri Jayewardenepura (Grant No ASP/01/RE/SCI/2017/13) for the funding support and to Mr. Chanaka Gunarathna, Lecturer, Student Association of Blind, Faculty of Humanities and Social Sciences, University of Sri Jayewardenepura.

REFERENCES

[1] Department of Census and Statistics-Sri Lanka, "Characteristics of the disabled persons census of population and housing 2001," 2001. [Online]. Available: http://www.statistics.gov.lk/pophousat/des_chra.asp. [Accessed: 02-Mar-2016].

[2] A. M. S. Al-salman, A. El-zaart, Y. Al-suhaibani, K. Al-hokail, and A. Gumaei, "Designing Braille Copier Based on Image Processing Techniques," *Int. J. Soft Comput. Eng. ISSN*, vol. 4, no. 5, pp. 62–69, 2014.

[3] A. Mousa, H. Hiary, R. Alomari, and L. Alnemer, "Smart Braille System Recognizer," *IJCSI Int. J. Comput. Sci. Issues*, vol. 10, no. 6, pp. 52–60, 2013.

[4] S. D. Al-Shamma and S. Fathi, "Arabic braille recognition and transcription into text and voice," *2010 5th Cairo Int. Biomed. Eng. Conf. CIBEC 2010*, pp. 227–231, 2010.

[5] J. Li, X. Yan, and D. Zhang, "Optical Braille recognition with haar wavelet features and Support-Vector Machine," *2010 Int. Conf. Comput. Mechatronics, Control Electron. Eng. C. 2010*, vol. 5, pp. 64–67, 2010.

[6] M. Wajid, M. Waris Abdullah, and O. Farooq, "Imprinted Braille-character pattern recognition using image processing techniques," *2011 Int. Conf. Image Inf. Process.*, no. Iciip, pp. 1–5, 2011.

[7] L. Wong, W. Abdulla, and S. Hussmann, "A software algorithm prototype for optical recognition of embossed braille," *Proc. - Int. Conf. Pattern Recognit.*, vol. 2, pp. 586–589, 2004.

[8] K. P. S. G. Sugirtha and Dhanalakshmi.M, *Transliteration of Braille Code into Text in English Language*, vol. 2, no. 2. Springer Singapore, 2018.

[9] E. Jacinto Gómez, H. Montiel Ariza, and F. H. Martínez Sarmiento, "There are very few work previously done for recognizing Sinhala Braille letters.," *Eighth Int. Conf. Graph. Image Process. (ICGIP 2016)*, vol. 10225, no. Icgip 2016, p. 102250N, 2017.

[10] S. H. Khaled and H. S. Abbas, "Braille Character Recognition Using Associtive Memory," *Int. J. Eng. Res. Adv. Technol.*, no. 1, pp. 31–45, 2017.

[11] S. Chatterjee, "Creation of an IT Enabled Sinhala to Braille Conversion Engine," *Int. J. Comput. Appl. Eng. Sci.*, vol. IV, no. Ii, pp. 17–21, 2014.

[12] N. M. T. De Silva and S. R. Liyanage, "Sinhala Braille Translator," *Int. J. Trend Res. Dev.*, vol. 3, no. 4, pp. 380–384, 2016.

[13] BANA MEMBERS, *ENGLISH BRAILLE AMERICAN EDITION*. Louisville: American Printing House for the Blind., 1994.

[14] Braille Authority of North America, "Unified English Braille," 2016. [Online]. Available: http://www.brailleauthority.org/ueb.html. [Accessed: 21-Apr-2016].

[15] C. Simpson, *The Rules of Unified English Braille Edited by*, 2nd ed. California: International Council on English Braille, 2013.

[16] MathWorks, "Create COM server - MATLAB actxserver - MathWorks India," 2016. [Online]. Available: http://in.mathworks.com/help/matlab/ref/actxserver.html?searchHighlight=actxserver&s_tid=doc_srchtitle&requestedDomain=in.mathworks.com. [Accessed: 16-May-2016].

[17] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, *Digital Image Processing*. Delhi: Pearson Education (Singapore) Pte. Ltd, 2004.

[18] MathWorks, "Image Processing Toolbox Documentation - MathWorks India," 2016. [Online]. Available: http://in.mathworks.com/help/images/functionlist.html. [Accessed: 02-Mar-2016].

[19] MathWorks, "Radon transform - MATLAB radon - MathWorks India," 2016. [Online]. Available: https://in.mathworks.com/help/images/ref/radon.html. [Accessed: 02-Mar-2016].

[20] Sinhalese Font, "Aa Amali Font Download ? Free Sinhala Font," 2016. [Online]. Available: http://www.sinhalesefont.com/download.php?id=736725. [Accessed: 21-Apr-2016].

[21] C. N. R. Kumar and S. Srinath, "A novel and efficient algorithm to recognize any universally accepted braille characters: A case with kannada language," *Proc. - 2014 5th Int. Conf. Signal Image Process. ICSIP 2014*, pp. 292–296, 2014.

[22] G. Morgavi and M. Morando, "A neural network hybrid model for an optical braille recognitor," *Int. Conf. Signal, Speech …*, no. January 2002, 2002.

[23] A. Antonacopoulos and D. Bridson, "A robust Braille recognition system," *Doc. Anal. Syst. VI*, pp. 533–545, 2004.

[24] Wikipedia, "Bharati Braille," 2016. [Online]. Available: http://www.bpaindia.org/VIB Chapter-VI.pdf. [Accessed: 02-Mar-2016].

[25] Wiki 2, "Russian Braille chart - Russian Braille — Wikipedia Republished // WIKI 2," 2016. [Online]. Available: https://wiki2.org/en/Russian_Braille#/media/File:Russian_Braille_chart.jpg. [Accessed: 02-Mar-2016].

Authors

Dr. W. K. I. L. Wanniarachchi is working as a Senior Lecturer in the Department of Physics, Faculty of Applied Sciences, University of Sri Jayewardenepura, Sri Lanka. He is a graduate in Bachelor of Science (Physics). He received his PhD in Physics from the Wayne State University, MI, USA. His research interests are on Computer Vision and Image Processing, Embedded Systems and Electronic Structure. Email: iwanni@sjp.ac.lk

T. D. S. H. Perera is working as a teaching assistant in the Department of Physics, Faculty of Applied Sciences, University of Sri Jayewardenepura, Sri Lanka. He received B.Sc. degree in Physics from the University of Sri Jayewardenepura, Sri Lanka in 2017. His research interest includes Image Processing and Optics. Email: sankaharshana@gmail.com