ADDIS ABABA UNIVERISTY

SCHOOL OF GRADUATE STUDIES

FACULTY OF COMPUTER AND MATHEMATICAL SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

RECOGNITION OF AMHARIC BRAILLE USING DIRECTION FIELD

By: Miftah Hassen

A THESIS SUBMITTED TO THE SCHOOL OF GRADUATE STUDIES OF THE ADDIS ABABA UNIVERSITY IN PARTIAL FULFILMENT FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE

June 2011

ADDIS ABABA UNIVERISTY

SCHOOL OF GRADUATE STUDIES

FACULTY OF COMPUTER AND MATHEMATICAL SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

RECOGNITION OF AMHARIC BRAILLE USING DIRECTION FIELD

By: Miftah Hassen

ADVISOR:

Dr. Yaregal Assabie

APPROVED BY

EXAMINING BOARD:

1. **Dr. Yaregal Assabie, Advisor _____**
2. **_____ _____**
3. **_____ _____**

To my mother

Acknowledgements

First and for most, I like to thank *Allah* for making this possible. I would like to thank my advisor Dr. Yaregal Assabie for his support in my work. I also like to thank my friends Tibebe Beshah and Zelalem Regassa for the invaluable ideas and comments they provided me.

I am grateful to employees of Misrach Center for their cooperation. I am also very indebted to Ato Sisay Tebeje, Ato Tatek Asmare, and Ato Meseret Abiye for unreservedly enlighten me about Amharic braille representation.

Finally yet importantly, I would like to thank Ebrahim Checkol for all the resources and ideas he shared.

Table of contents

LIST OF FIGURES ..... **III**

LIST OF TABLES ..... **IV**

LIST OF LISTS ..... **V**

ABSTRACT ..... **VI**

CHAPTER 1 ..... **1**

INTRODUCTION **1**

1.1 BACKGROUND 1
1.2 OBJECTIVES 4
1.2.1 General Objective 4
1.2.2 Specific Objectives 4
1.3 METHODS 4
1.3.1 Literature Review 4
1.3.2 Data Collection 4
1.3.3 Experimental Evaluation 5
1.4 APPLICATION OF RESULTS 5

CHAPTER 2 ..... **6**

LITERATURE REVIEW **6**

2.1 AMHARIC BRAILLE SYSTEM 6
2.1.1 Amharic Characters 6
2.1.2 Braille 7
2.1.3 Amharic Braille 9
2.2 AUTOMATIC BRAILLE RECOGNITION 13
2.2.1 Image Acquisition and Preprocessing 13
2.2.2 Dot Detection and Braille Cell Formation 15
2.2.3 Recognition 16
2.3 REVIEW OF AMHARIC BRAILLE RECOGNITION WORKS 17

CHAPTER 3 ..... **20**

BRAILLE RECOGNITION **20**

3.1 PREPROCESSING 20
3.1.1 Gaussian Filters and Derivatives of Gaussian 20
3.1.2 Gradient Field 21

3.1.3 Direction Field 22
3.1.4 Skewness Correction 24
3.2 Analysis of Important Braille Cell Values 24
3.3 Braille Character Line Identification 25
3.4 Half-character Detection 26
3.5 Half-character Recognition 28
3.6 Braille Cell Formulation 29
3.7 Translation 30
CHAPTER 4 32
IMPLEMENTATION AND EXPERIMENT 32
4.1 Preprocessing 32
4.1.1 Direction Field Image 32
4.1.2 Thresholding 34
4.1.3 Skewness Correction 35
4.2 Estimation of Important Braille Cell Values 36
4.3 Braille Character Line Detection 38
4.4 Half-character Detection 39
4.5 Half-character Recognition 41
4.6 Braille Cell Formulation 43
4.7 Translation 44
4.8 Performance Evaluation 46
CHAPTER 5 51
CONCLUSION AND RECOMMENDATION 51
5.1 Conclusion 51
5.2 Recommendation 52
REFERENCES 54
APPENDIX A. THE FIRST VERSION AMHARIC BRAILLE 57
APPENDIX B. THE SECOND VERSION AMHARIC BRAILLE 58
APPENDIX C. THE THIRD VERSION AMHARIC BRAILLE 59
APPENDIX D. SAMPLE TABLE LOOKUP OF LETTERS "ሀ" TO "ፘ" WITH THEIR 7 VARIANTS 60

List of Figures

Figure 2. 1. List of Amharic core characters and their seven variants 7

Figure 2. 2. A braille cell 8

Figure 2. 3 Braille code for print character ረ 12

Figure 2. 4. Double-sided gray color braille image 15

Figure 3. 1. Comparison of direction field and gradient field images. (a) the original image (b) the gradient field image (c) the direction field image 23

Figure 4. 1. Direction Field Image (a) represents the original image in gray level. The image labeled (b) shows the direction field image.) 33

Figure 4. 2. Thresholding direction field image (the picture labeled (a) shows the direction field image. The picture labeled (b) shows the logical image) 34

Figure 4. 3. Skewness correction ((a) 5 degree tilting towards the right hand side, (b) 5 degree tilting towards the left hand side, (c) the result of the skewness correction. 36

Figure 4. 4. Braille character line identification 39

Figure 4. 5. Half character identification 41

Figure 4. 6. Group of half characters with similar heights. 42

Figure 4. 7. Half character recogntion 43

Figure 4. 8. Braille cell formulation 44

Figure 4. 9. Comparison of good quality braille documents. ((a) original image. (b) Preprocessing result of a by the previous work [2]. (c) Preprocessing result of a by the current work) 48

Figure 4. 10. Comparison of poor quality braille documents. ((a) original image. (b) Preprocessing result of a by the previous work [2]. (c) Preprocessing result of a by the current work) 49

List of Tables

Table 2. 1. Fourth Version Amharic Braille Characters *11*

Table 2. 2. Fourth Version Amharic Braille Vowels *12*

Table 2. 3. Fourth Version Amharic Braille code for punctuation Marks *12*

Table 2. 4. Braille code for Ethiopic and Arabic numerals *13*

Table 3. 1. Half characters and their corresponding value *28*

Table 4. 1. Estimation of important braille cell values *38*

Table 4. 2. Statistics of braille documents collected for experiment *47*

List of Lists

List 4. 1. Pseudo code for skewness correction. *36*
List 4. 3. Pseudo code for identifying braille dot height. *37*
List 4. 4. Pseudo code for braille character line detection. *39*
List 4. 5. Pseudo code for separating merged dots. *40*
List 4. 6. Pseudo code for half character detection. *40*
List 4. 7. Pseudo code for half character recognition. *42*
List 4. 8. Pseudo code for braille cell formulation. *44*
List 4. 9. Pseudo code for braille to print translation. *46*

Abstract

Invented by Louis Braille in 1829, braille has been a widely used means of written communication for the blind people around the world. Since its introduction in Ethiopia, there have been piles of Amharic braille documents produced by the blind people for different purposes. Since these documents are usually read and understood only by the same group of people, the blind, who have written them, the invaluable knowledge and information found there is highly restricted from reaching the sighted society. Automatic recognition of braille documents is therefore very instrumental in bridging the communication gab that exists between the blind and the sighted people.

This thesis presents an approach for recognition of Amharic braille documents. We have used direction field tensor, which uses Gaussian filters and derivatives of Gaussians, for noise removal and isolation of braille dots from their background. We also designed a new skewness correction technique, which exploits the horizontal direction nature of braille dots. Attempts are made to determine braille dot sizes automatically so that the system works for different braille dot sizes. Braille character lines are constructed in order for subsequent operations to be performed only on the character lines. A half character detection method, which differentiates braille dots from noises is applied. Having recognized the half characters through analysis of their dot positions, the braille cells are formulated by way of examining horizontal distances between the half characters. We have finally used a lookup table for translating each of the formulated braille cells into their corresponding Amharic print characters.

Braille documents, especially those that were used by previous works for comparison, are collected for experiment purpose. Our system achieved an average accuracy of 98.5%. 99.9% accuracy is achieved for good quality braille documents, while the accuracy for poor quality braille documents is 96.5%. The small errors observed in the experiment are attributed to some stains and defects present on braille documents.

Chapter 1

Introduction

1.1 Background

According to the International Eye Foundation (IEF) report [6], there are 45 million blind people in the world and 90% of which is found in developing countries. The latest report [7] on the population of blind people in Ethiopia shows that there are about 1.2 million blind people in the country.

Since its invention in 1829 by Louis Braille, braille has been a widely used means of written communication for the blind people around the world. A braille character is a rectangular array of six points arranged in two columns of three. Each point is either raised or flat resulting in 64 (2^6) possible braille character combinations. Each braille character corresponds to an ordinary character (in grade 1 braille) or a group of characters (in grade 2 Braille). In languages like Amharic, however, up to three braille characters are used to represent a single character, as the 64 available braille character combinations are not large enough to represent the 310 characters that exist in Amharic language.

A standard braille page is 11 inches by 11.5 inches and typically has a maximum of 40 to 43 braille cells per line and 25 lines [1], which is why large volumes of braille documents are required to represent their equivalent ordinary text documents.

Despite its early invention in 1829, braille code was introduced in Ethiopia in 1934 and it has undergone different levels of improvements in 1954, 1956, and 1958 before it takes its current form [3]. Since its introduction in the country, there have been piles of Amharic braille documents produced by the blind people for different purposes. A significant number of braille documents are currently found in different libraries and educational institutions like, AAU Kennedy library braille documentation, Misrach Center, Sebeta visually impaired school, German Church visually impaired school, Ethiopian Association for Blind Society (EABS), Entoto blind people school and other centers that are found in different regions of the country [3]. Since these documents are usually read and understood by the same group of people, the blind, who have written

them, the invaluable knowledge and information found there is highly restricted from reaching the sighted society. Unless some form of organized means of knowledge transfer between the blind and the sighted people is created, the written communication gap that already exists between the blind and the sighted people will continue to widen. A research that focuses on bridging this gap is therefore believed to be indispensable.

Different researches have been undertaken for automatic recognition of braille documents in different languages such as, English [5], Arabic [1], Chinese [4], and so on. These research works follow some general basic steps in the recognition process such as *image acquisition*, *preprocessing*, *segmentation*, and *feature extraction and recognition*. Image acquisition refers to converting the braille document into a digital format while preprocessing is the process of modifying and rectifying some errors to prepare the digital image for subsequent operations. On the other hand, segmentation separates the braille dots from their background, and feature extraction and recognition refer to the process of changing the braille dots into braille cells and their final interpretation into the corresponding ordinary language text respectively.

As to the recognition of Amharic braille documents, two researches have been attempted by Alemu [3] and Chekol [2]. Alemu has adopted a global-threshold technique for image binarization and applied mesh grid construction for the purpose of both segmentation and feature extraction. For recognition, Alemu used neural-network classification technique. Alemu achieved an accuracy of 92.5% on clean or well-formed braille documents. Chekol‟s work is basically a continuation of Alemu‟s work though he mainly focused on the pre-processing part namely, noise detection and removal. Chekol achieved an accuracy of 95.5%, 95.5%, 90%, and 65% for clean, small-level noise, medium-level noise, and high-level noise braille documents respectively.

Though Alemu and Chekol are well appreciated for pioneering in the area of automatic Amharic braille recognition, their works are not without some problems. The problems are mainly found in the three activities of the whole recognition process namely; noise detection and removal, mesh-grid construction, which is used for segmentation and feature extraction and the neural network, which is used for classification purpose. As to noise detection and removal, Chekol applied Gaussian filter with morphological

operations. As shown in his work, his technique seems to result in deletion of valid braille dots that appear in a noisy area. Alemu notes that the braille dots that are identified in the course of binarization process are not of identical size. This severely affects the mesh-grid construction as some of the dots may detour from the horizontal and/or vertical lines of the mesh-grid failing to be identified as part of a braille cell. On the other hand, the neural network that is applied for classification purpose by both Alemu and Chekol exhibited a substitution problem, which simply means interpreting a braille cell as one character when the braille cell actually refers to another. It is, moreover, worth mentioning that since all the dot combinations designated to each and every character is fixed and known, instead of a neural network classifier, the use of a simple braille dictionary [4] would have sufficed.

The current study investigates different techniques to come up with a better solution for recognition of Amharic braille documents. To this end, direction field, which uses Gaussian filter and derivatives of Gaussians, is used as a tool for removal of noises and separating the braille dots from their background. Gaussian filter is frequently used as low-pass filter for noise suppression and Gaussian derivatives are used to detect and localize edges along with determining their orientation [24]. The direction field image is further analyzed and turned into a logical image using thresholding. A skewness correction is performed, in case of image tilting, followed by braille character line identification. The horizontal and vertical distances between dots within a character and between braille characters is specified by the Library of Congress [1] and are well known. Therefore, having automatically identified the braille dot height, horizontal analysis is employed for construction of the braille cells. This is believed to solve the problem observed in the previous works concerning valid braille dots exclusion from the braille character region. The constructed braille cells are then grouped in accordance with their representation of the Amharic characters and are recognized using a braille lookup table, which helps avoid the misinterpretation problem that Alemu and Chekol faced while using neural network classifier.

1.2 Objectives

The general and specific objectives of this study are presented as follows.

1.2.1 General Objective

The general objective of this research is to design an Amharic Braille Recognition System.

1.2.2 Specific Objectives

In order to achieve the general objective, the following specific objectives are addressed.

- Review of previous researches on automatic braille recognition.
- Collection of relevant Amharic braille documents.
- Analysis of the nature and characteristics of Amharic braille code.
- Designing an algorithm for the Amharic braille recognition system.
- Developing a prototype for the Amharic braille recognition system.
- Testing and evaluating the performance of the system using Amharic braille documents.
- Making conclusion and indicating the way forward for future works.

1.3 Methods

The strategy adopted for the research is described as follows.

1.3.1 Literature Review

Extensive literature review is conducted on automatic braille recognition in order to obtain an in-depth understanding of the area and to find useful approaches for the Amharic language braille recognition.

1.3.2 Data Collection

Braille documents and information on Amharic braille code are collected from different libraries and institutions.

1.3.3 Experimental Evaluation

For the purpose of evaluating the system in a fair and logical manner, different Amharic braille documents of different quality level and different contents (alphabets, numbers, and punctuation marks) are given to the developed prototype. Accordingly, the system is evaluated by comparing its output against the actual content of the braille documents on character by character basis.

1.4 Application of results

The outcome of the study is believed to be instrumental in bridging the written communication gap that exists between the sighted and the visually impaired people in education, business and other areas. Moreover, different libraries where such documents reside and non-governmental organizations that closely work with the blind people will be highly beneficial. Most importantly, it will be useful in terms of enabling the society to benefit from the knowledge and experience of the visually impaired society in a well-organized and dynamic way.

Chapter 2

Literature Review

In this chapter, different points that are related to the current work are discussed. A brief introduction of the Amharic language and Ethiopic characters is made in section 2.1.1. An introduction of braille in general and Amharic braille in particular is provided in sections 2.1.2 and 2.1.3 respectively. In light of the major steps in automatic braille recognition, different related researches are discussed under section 2.2. Section 2.3 looks into the two researches done on automatic recognition of Amharic braille and provides the ground for the current work.

2.1 Amharic Braille System

This section briefly discusses the characters used in Amharic language followed by an introduction about braille writing and reading with special focus on Amharic language.

2.1.1 Amharic Characters

Used as an official language of Ethiopia since the 14th century, Amharic language is the second most widely spoken Semitic language in the world next to Arabic [8]. Amharic is mainly spoken in Ethiopia and Eritrea. The Ethiopic script, which is used in Amharic language for writing is composed of 33 „core" and 1 „special" characters each of which has 7 variants forming a total of 238 characters (See Figure 2.1). In addition to the 238 characters, there are over forty characters used in Amharic language for representing labialization, punctuations, and numbers.

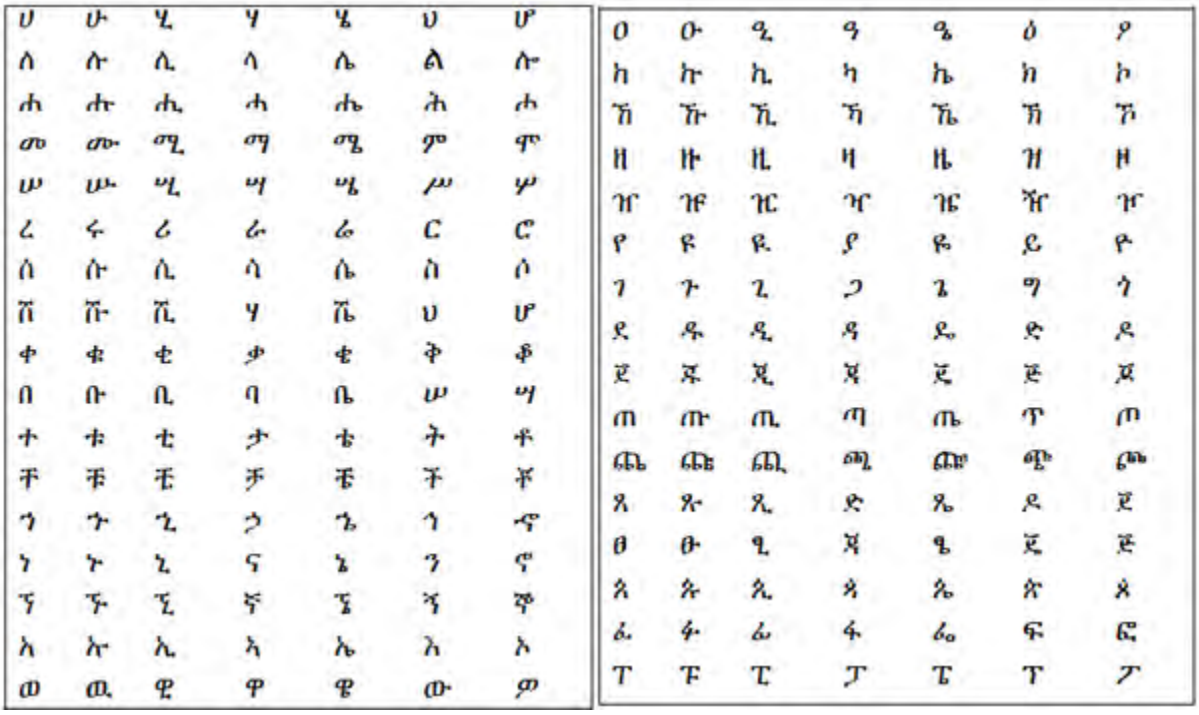


Figure 2. 1. List of Amharic core characters and their seven variants

2.1.2 Braille

Braille was invented in 1829 by a French man named Louis Braille. It is a system of writing that enables the visually-impaired people to read and write by way of touching. Braille consists of a system of six combination of dots that are arranged in a two by three form, which is called a braille cell (see Figure 2.2). Each of the six dot position can be either raised or flat resulting in 64 possible combinations. The horizontal and vertical distance between dots in a braille cell, the distance between cells representing a word and the inter-line distance are specified by the Library of Congress [1]. Height of a braille dot is approximately 0.02 inches (0.5 mm); the horizontal and vertical spacing between dot centers within a braille cell is 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 [1]. Due to the fact that a typical braille page is 11 by 11.5 inches and can only have a maximum of 40 to 43 braille cells per line and 25 lines per page, the information density of braille is low [1, 9]. This is why it takes volumes of braille documents to represent their print text counterparts. The use of two-sided braille

document is one of the attempts aimed at circumventing this inherent huge size problem of braille documents.

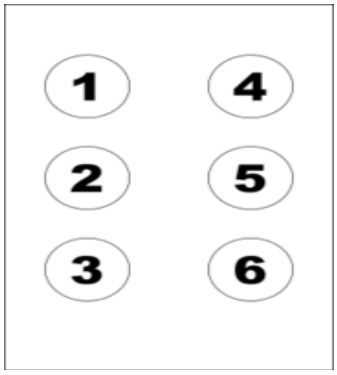


Figure 2. 2. A braille cell

Braille writing and reading

Ever since its invention, braille has been the main system of written communication for the majority of blind people who read and write using tactile means. There are a number of means of braille writing that result in a tactile output [10]. The devices that are used for braille writing range from the most low-tech method of writing by means of stylus and slate to the most advanced one that uses a braille printer as peripheral to a computer. A typewriter like device called brailler and braille embosser are also other devices used for producing braille documents.

As in print documents, braille character embossing starts from the top of a document and works its way down to the bottom. All dots on a braille page should fall on an orthogonal grid. This is especially very important when braille characters are printed double sided (Inter-point) as the grid of the inter-point text is shifted so that the dots fall in between the primary side dots [2]. The type of paper used for braille is usually thick and often has white or buff color though the color does not convey any information as braille is read by means of touching

Braille is read by moving one‟s hand on the embossed dots of the braille document. Due to this nature of the way a braille document is read, braille reading is by far slower than that of print text reading. According to American Council of the Blind, people who are fluent in braille can typically read at a rate of 125 to 200 words per minute. On average,

eighth graders read at a rate of 205 words per minute, and college students read at 280 words per minute according to University of Buffalo [11].

Text Representation in Braille

Text to braille cell translation is hardly one to one due to the limited number of representations (64) that we get out of the six positions of a braille cell. Different languages have character sets that are well beyond 64. In order to systematically represent the characters that a language has into a braille cell, two forms of braille representations have been employed; Grade 1 braille and Grade 2 braille (contracted Braille).

Grade 1 braille is a form of braille in which print characters are represented by one braille cell using a mode indicator character, which determines how the character is to be read. For instance, in the English braille the lower case letters A-Z and the major punctuation symbols are represented by a single braille character, where as "shift" character is used to indicate other information such as upper case, digits, and italics [12]. The other type of braille called Grade 2 braille or contracted braille is introduced as a result of the rigorous attempt to minimize the volume of braille documents and speed up the time with which braille is read among others. In this form of braille, context sensitive rules, which are apparently language dependent, are used for the contraction of words and frequently used letter groups. These rules determine the correspondence between one or more braille cells and the print, so, for example, in standard English braille, a braille symbol may stand for: "dis" when used at the start of a word (distance); "dd" when used in the middle of a word (ladder); or a period when used at the end of a word [12].

2.1.3 Amharic Braille

Amharic braille is embossed in a tactile form in the same way other language braille documents are embossed. As already stated previously, the number of dot positions used in a braille cell is six (though there is also an eight dot braille that is not as popular as the six dot one). This limits the possible number of characters that can be represented by a braille cell on a one-to-one basis to 64 (2^6). This is, in fact, the reason why different types of indicator braille cells are used in grade 1 braille to interpret a braille cell as one or

another. In the case of Amharic language, the problem gets even worse as the language has over 300 characters and therefore is very difficult to represent the characters using only mode indicator braille cells. As a result, a different form of representation is used whereby a single Amharic print character may be represented by one, very often by two or at some rare cases by three braille cells.

Evolution of Amharic braille

Braille was introduced to Ethiopia in 1934. The first Amharic braille was comprehensive that contains all Amharic characters from "ሀ" to "ፐ" including characters that even have similar sounds (such as, the three "ሀ" sound characters- ሀ, ሐ, and ኀ; the two "ሰ" sound characters- ሰ and ሠ ; the two "ጸ" sound characters- ፀ and ጸ; and the two "አ" sound characters-አ and ዐ), punctuation marks, Amharic numerals, and musical symbols taken from other languages, and also mathematical symbols [2].

The first Amharic braille was revised in 1945 E.C. The revision mainly focused on eliminating the redundant Amharic characters with similar sounds and replacing the Ethiopian numerals with English numerals [13]. In this version of the Amharic braille, all forms of a character except the first and the sixth forms were represented using vowels.

Just four years later in 1949, another revision on the Amharic braille was undertaken to form what is known to be the third version Amharic braille. In this version, a vowel was added to the first variant of the Amharic character [13]. This version lasted comparatively longer than the first two versions. However, it was finally revised in 1995 E.C. to give way to the fourth version Amharic braille. In terms of usage, the third version is still the one that is being widely used by the blind community in the country despite the introduction of a newer version.

Creating equal number of braille codes that match with print characters, finding means of substituting similar sound characters, preparing grade to Amharic braille, incorporating Geez numbers, use of Yared musical symbols and punctuation marks were the major issues that were considered for discussion by the committee that was formed for the fourth version Amharic braille [13]. As a consequence, the number of braille codes was decided to be equal to the print characters. Modification of the punctuation marks mostly

by borrowing them from the English language was deemed necessary. Formation of grade 2 braille was deferred at least until the grade 1 braille gets legal recognition. The case of Yared musical symbols was decided to be discussed with professionals. It was also decided to use either of the American or the British system for mathematical symbols [13]. Finally, the fourth version Amharic braille was approved and distributed for use by different institutions that work with visually impaired people.

Amharic braille characters

To find a way out of the limited number of representations in braille system, indicator braille cells, for instance in Grade 1 braille, are used. As to Amharic language, however, this was not enough since the language has over 300 characters. This has necessitated the use of more than one braille cells to represent a single Amharic print character (see *Table 2.1*).

Table 2. 1. Fourth Version Amharic Braille Characters

1st form (character)			6th form (character)		1st form (character)			6th form (character)	
		Vowel					vowel		
ሀ	1:2:5	2:6	ሀ	1:2:5	ኸ	2:3:6	2:6	ኽ	2:3:6
ለ	1:2:3	2:6	ል	1:2:3	ወ	2:4:5:6	2:6	ው	2:4:5:6
ሐ	*1:2:6	2:6	ሐ	*1:2:6	ዐ	*1:2:5:6	2:6	ዕ	*1:2:5:6
መ	1:3:4	2:6	ም	1:3:4	ዘ	1:3:5:6	2:6	ዝ	1:3:5:6
ሠ	2:3:4	2:6	ሥ	2:3:4	ዠ	3:5:6	2:6	ዥ	3:5:6
ረ	1:2:3:5	2:6	ር	1:2:3:5	የ	1:3:4:5:6	2:6	ይ	1:3:4:5:6
ሰ	*1:4:5:6	2:6	ስ	*1:4:5:6	ደ	1:4:5	2:6	ድ	1:4:5
ሸ	1:4:6	2:6	ሽ	1:4:6	ጀ	2:4:5	2:6	ጅ	2:4:5
ቀ	1:2:3:4:5	2:6	ቅ	1:2:3:4:5	ገ	1:2:4:5	2:6	ግ	1:2:4:5
በ	1:2	2:6	ብ	1:2	ጠ	2:3:4:5:6	2:6	ጥ	2:3:4:5:6
ተ	2:3:4:5	2:6	ት	2:3:4:5	ፈኽ	1:4	2:6	ፍ	1:4
ቸ	1:6	2:6	ች	1:6	ጸ	2:3:5	2:6	ጽ	2:3:5
ኀ	*1:5:6	2:6	ኅ	*1:5:6	ፀ	1:2:3:4:6	2:6	ፅ	1:2:3:4:6
ነ	1:3:4:5	2:6	ን	1:3:4:5	ጸ	*2:3:4:6	2:6	ጽ	*2:3:4:6
ኘ	3:4:6	2:6	ኝ	3:4:6	ፈ	1:2:4	2:6	፝	1:2:4
አ	1:2:3:5:6	2:6	ኧ	1:2:3:5:6	ፐ	1:2:3:4	2:6	ፕ	1:2:3:4
ከ	1:3	2:6	ክ	1:3	ቨ	*1:2:3:6	2:6	ቭ	*1:2:3:6

* Symbol indicates Braille code change made on this new version.

See figure 2.3 to see how two braille cells are combined to represent one Amharic character.

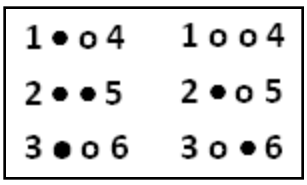


Figure 2. 3 Braille code for print character ረ

Table 2.1 has shown the first variant of the Amharic characters representation in braille. Table 2.2 below shows the braille cells that are used as vowels to form the seven variants.

Table 2. 2. Fourth Version Amharic Braille Vowels

Vowel	2:6	1:3:6	2:4	1	1:5	Not Applicable	1:3:5
Character Variant	1st	2nd	3rd	4th	5th	6^{th}	7th

As already stated, punctuation marks are among the representations included in the fourth version Amharic braille (see Table 2.3).

Table 2. 3. Fourth Version Amharic Braille code for punctuation Marks

Punctuation	**Braille Code**	**Punctuation**	**Braille Code**
.	3	::	2:5:6
:	6 and 3	!	2:3:5
:-	2:5	...	3, 3 and 3
/	5 and 2	()	2:3:5:6
„	4 and 1	[	6 and 2:3:5:6
-	3:6	]	2:3:5:6 and 3
፡	2	*	3:5 and 3:5
፤	2:3	_	4:6 and 4:6
“	2:3:6	→	2:4:6, 2:5 and 2:5
'	3:5:6 and 3	←	2:5, 2:5 and 1:3:5
?	2:3:6	↑	4:5:6 and 1:5
=//=	6 and 3	↓	4:5:6 and 3:5
X	1:3:4:6	እና ወይም	3:4
$	4:5		

In the fourth version Amharic braille, both Arabic and Ethiopic numerals are included in the representation. The braille cells 3:4:5:6 and 1:2:3:4:5:6 are used as mode indicators for Arabic and Ethiopic numerals respectively. Apart from the mode indicator cells, Arabic and Ethiopic numerals use the same braille codes (see Table 2.4).

Table 2. 4. Braille code for Ethiopic and Arabic numerals

⠚	⠁	⠉	⠙	⠑	⠋	⠛	⠓	⠊	⠚⠁
1	2	3	4	5	6	7	8	9	0
1	1:2	1:4	1:4:5	1:5	1:2:4	1:2:4:5	1:2:5	2:4	2:4:5

2.2 Automatic Braille Recognition

Any automatic braille recognition system basically consists of some major activities [1] such as; image acquisition, image preprocessing, dot localization and segmentation, and dot recognition and conversion. This section of the document discusses the whole recognition process in light of aforementioned activities.

2.2.1 Image Acquisition and Preprocessing

Image acquisition is the first step in the braille recognition process. Equipments like digital camera and flat bed scanner are means of acquiring a braille document in a digital form. Different researches [4, 14, 15, 16], predominantly older ones, attempted to recognize braille by capturing the image using camera. In order to capture braille using camera, the braille document is illuminated under an oblique angle so that it reveals shadows from the braille dots. The use of camera, however, introduces some complexities such as, aberrations, irregular lightness, and relatively low resolution among others [17]. Additional computationally expensive procedures have to be employed to overcome these problems [18]. Consequently, newer researches [1, 17, 18] on braille recognition opted for a commercially available flat bed scanner to obtain braille images digitally. In addition to being quick and easy to use, a scanner is cost effective as it can be used for so many other applications. The researches that used scanner for image acquisition used different scanner resolutions. However, a very high (most of the time above 200dpi) resolution is not recommended as it may add too much detail into the

image [9], which may tarnish the image and thus will have adverse effects on subsequent procedures.

Pre-processing is another important activity in the braille recognition process. It generally refers to the process by which error corrections that are crucial for the smooth functioning of subsequent procedures are undertaken. The errors that require correction in a braille image include, noise, bad illumination, image tilting and the like. Application of different image preprocessing techniques will help enhance the quality of the braille image for recognition by reducing noises, sharpening braille dots, and also appropriately rotating tilted images [1].

Subtraction of local average from the original image is the method employed by [9] to eliminate shadows that are introduced by the simple fact that braille pages are not perfectly flat. Ng and Lau [4] applied low-pass spatial Gaussian filter to attenuate the high-spatial frequency noise from the image while at the same time preserving the detailed edge information of the braille dots. They also applied edge enhancement technique to sharpen the fine details of the image. Several other researches [1, 17, 19, 20] applied thresholding to classify the braille image into background and dot (which is usually composed of white and dark regions) regions. However, naturally occurring reflections and shadows that are not part of the actual braille dots may not be easily eliminated by thresholding. These reflections and shadows, especially when they are located around braille cells, would lead to identification of non-braille dots as valid dots [18]

Correction of image tilting does not seem to get due attention by some researches on braille despite the fact that braille pages may not be correctly aligned during image capturing. Nevertheless, several researches dealt with correction of image tilting. Mennens et al. [9], for instance, used Discrete Fourier Transform (DFT) with further refinements to calculate the rotation angle in an attempt to de-skew tilted braille pages. A Hough transform is performed on the braille dots by Antonacopoulos and Bridson [17] to identify the precise orientation of the rows and rotate them accordingly. Tai et al. [21], however, argue that the Radon Transforms, which are able to transform two-dimensional images with lines into a domain of possible line parameters, performs better than Hough

transform. AlSalman et al. [1] on the other hand used a binary search algorithm, which tolerates a tilting of up to 4 degrees both on the left and right directions.

2.2.2 Dot Detection and Braille Cell Formation

Braille dot detection is an important element of braille recognition. Several researches [1, 4] attempted to exploit the specific color features of braille dots in a braille image. Braille dots tend to have a pair of dark and light regions, which are somewhat distinct from the background in a braille image. In a double-sided braille document, the order of dark and light regions differs for recto and verso dots, which represent depression and protrusion respectively (Figure 2.4).

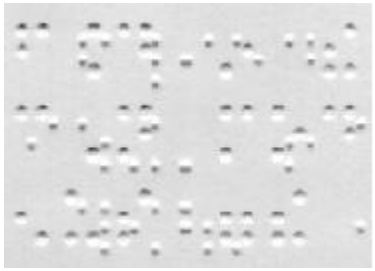


Figure 2. 4. Double-sided gray color braille image

Ng and Lau [4] used a template of the protrusions and depressions to separate dots (both front-faced and back-faced) from the background. However, this seems to have a problem in situations where the dark and light areas of a protrusion and a depression lie in an arrangement that gives rise to the detection of a non-existent depression or protrusion. Care should also be taken not to discard any black and white regions that have been used to describe a protrusion but that can also be used to describe a depression due to the white regions and the black regions of protrusions and depressions being merged [17]. Similarly, AlSalman et al. [1] applied a vertical search from top to bottom of a possible dot region to check the existence of a pair of dark and light regions, which if found in the correct order, will qualify as a valid braille dot.

Wong et al. [5] followed a different approach that identifies columns (half-characters), which may be consisting of one, two, or three dots, instead of dealing with individual dots. The half-characters are processed and classified as one of the seven possible arrangements that can be obtained out of three possible dot positions (ignoring the

"empty half character") using probabilistic neural network. However, instead of applying a model like neural network, which makes predictions, a simple analysis of the dot positions would give a better result [1]. This is mainly because the meaning of one braille character can easily be converted to another valid braille character by the simple presence or absence of dots in a braille dot position.

Mennens et al. [9] employed a method that constructs a horizontal and vertical grid lines the intersection of which is expected to be a possible braille dot position. The braille cells then will be segmented out of the constructed grid. This technique however does not tolerate a slight difference that may exist between positions of braille characters in different lines. Antonacopoulos and Bridson [17] used the same technique of grid construction in a different way in that the grid is designed in a relatively flexible manner that does not require the braille characters in different lines to be aligned.

Having identified all braille cell lines, Ritchings et al. [18] segmented the braille cells in each line based on the analysis of the horizontal spacing between columns with dots. They stress that care should be taken when two single columned characters are encountered as ambiguity may arise if they both have dots in the left column or in the right column.

Grouping of braille dots into braille cells was done by Wong et al. [5] through analysis of the distance between the centriod of a dot and the four possible neighbours.

2.2.3 Recognition

Recognition refers to the step in which the braille cells are analyzed and translated into any other code, specifically print character. Most of the researches tend to use a braille dictionary or something similar to do the translation into text after they examine the dot positions of the braille cells. For instance, the intersection of the two lines (that represents a dot position) of a braille cell that is constructed from a grid of horizontal and vertical lines is checked for the existence of a dot. A binary cell string is then formulated in which 1 is set in a position where dot exists and 0 is set otherwise. The binary cell string is finally translated into a specific code using an appropriate translation table [28]. In a similar manner, Ng and Lau [4] checked the dot patterns of a braille cell against a braille

dictionary to translate into the appropriate character, which is grouped into words. The word is then checked against an English Dictionary. If the word cannot be found, the word with the highest percentage of similarity will be selected and be highlighted for later edition.

After the position of each braille character is known, Ritchings et al. [18] divided the rectangular area in the image that corresponds to a segmented character into six equal compartments. The compartments are arranged in three rows and two columns and represent the bounds of the expected positions of dots in a braille character. If there is a dot in any of the compartments, a bit is set in a binary number at the position that corresponds to the number of the compartment. The binary number is then translated into a character.

In grade 2 braille, a one to one correspondence between braille cells and print characters is not possible as in grade 1. Therefore, the translation that is used for a grade 2 braille should be designed in a different form. Blenkhorn [12] uses the finite state system to hold the current context. This is important because different braille characters, in grade 2 braille, will have different meanings depending on the context. The system has also been designed in such a way that a wide range of options and data can be entered using a set of tables, including braille rules, which are presented in a clear manner.

2.3 Review of Amharic Braille Recognition Works

Even though several researches have been undertaken on automatic braille recognition for different languages like English, Arabic, and Chinese to mention but a few, as per our knowledge, only two works have been attempted so far on Amharic braille recognition by Alemu [3] and Chekol [2] at Addis Ababa University. Chekol‟s work is a simple continuation of Alemu‟s work, whose work is the first attempt ever on Amharic braille recognition.

Alemu used a flat bed scanner with 200dpi to capture braille images. He applied global-thresholding to separate the foreground (braille dots) of the braille document from the background. The average center point of dots both on the horizontal and vertical lines of the image is calculated to construct a mesh grid, which is used to detect braille dots. The same mesh grid is again used for feature extraction in which formulation of braille cells

and determination of context of a braille cell is undertaken. As a result of the context analysis, braille cells that represent one Amharic print character individually and those that together with other braille cells represent a single Amharic print character are identified.

Having identified the braille characters that correspond to Amharic print characters, Alemu used a probabilistic neural network to translate the braille characters into Amharic print characters. Alemu managed to achieve an accuracy of 92.5% on clean braille images.

Chekol mainly focused on the preprocessing part of automatic braille recognition and adopted Alemu‟s work for segmentation, feature extraction and recognition. Chekol accordingly adopted Gaussian filters and morphological operations in an attempt to remove noises from the braille image. Chekol achieved an accuracy of 95.5%, 95.5%, 90%, and 65% for clean, small-level noise, medium-level noise, and high-level noise braille documents respectively.

Alemu assumed braille documents to have uniform color intensity and accordingly adopted global-thresholding for binarization. However, researches suggest that let alone used and worn out braille documents, even well formed braille documents tend to have some kind of shadows due to bad illumination during image capturing. Chekol attempted to deal with such problems in the preprocessing step of his work though his preprocessing result still has problems especially with degraded braille documents. One of the important steps in image preprocessing is correction of skewed images, which none of the two works dealt with. A slight tilting of braille documents during image capturing will have a devastating impact on the result of the two works.

Chekol stated that the grid construction, which is used both for segmentation and feature extraction, is not flexible enough to tolerate different braille dot sizes. He also suggested that the neural network, which we argue was not necessary in the first place, has substitution errors in recognizing print characters. During feature extraction, In addition to formulating the braille cells, Alemu states, context analysis is performed to identify whether or not the formulated braille cell represents a single character in its own right or is part of the braille cells that together represent a single print character. This clearly

indicates that the braille cells are already recognized as to what they represent and a simple table (braille dictionary) lookup, as has been done in several researches, will suffice to translate them into their corresponding print character. However, both works adopted probabilistic neural network for recognition purpose, which as indicated, resulted in introducing more errors into the recognition process.

Moreover, the works are not complete with regards to the number of symbols they included for recognition. For example, some punctuation marks, special Amharic characters and Amharic numerals are excluded.

To sum up, though the two works are eye-openers in Amharic braille recognition research, they seem to lack some important elements in braille recognition and also adopted some techniques that are not suitable for the problem at hand. The current study, accordingly, proposes a system that would curb the aforementioned problems of the works of Alemu and Chekol. To this end, a direction field tensor, which uses Gaussian and derivatives of Gaussian, is used to isolate braille dots by suppressing non-dot regions of the braille document. Analysis on the slope of the braille dot lines is done and the braille document accordingly is rotated to avoid image tilting. Braille cell formulation is done in row wise manner in order to avoid problems that will result from non-aligned braille characters in different lines, which global mesh-grid construction suffers from. Moreover, recognition of braille characters is done by way of braille dictionary lookup. In the current work, all characters of the Amharic language are included.

Chapter 3

Braille Recognition

This chapter presents a detailed analysis of the proposed system. In section 3.1, a discussion is made regarding the preprocessing techniques used in the study namely the direction field tensor and skewness correction. Direction field tensor, which is adopted from Bigun et al. [25], is used as a tool for noise filtering and separation of braille dots from their background while skewness correction is used for avoiding image tilting. Section 3.2 discusses the different type of braille related values and how they are estimated in this study. Section 3.3 and 3.4 talk about half character detection and half character recognition respectively. Braille cell formulation is the subject discussed under section 3.5. Finally, in section 3.6, the mechanism that we have used for braille to print text translation is explained.

3.1 Preprocessing

Preprocessing is an important step in image processing where activities such as contrast enhancement, noise reduction or filtering are performed [29]. The preprocessing techniques used in our study are discussed below.

3.1.1 Gaussian Filters and Derivatives of Gaussian

During braille image acquisition, impulse noise is sometimes introduced into the image [4]. A low pass spatial Gaussian filter can be used to attenuate the high spatial frequency noise from the image while at the same time preserving the detailed edge information of the braille dots [23].

Being one of the most commonly used filters for image processing, Gaussian filters and derivatives of Gaussians are used as a low-pass and high-pass filters respectively for noise suppression and to detect and localize edges along with determining their orientation respectively [24]. The popularity of Gaussians as filters is due to their valuable properties including: 1) directional isotropy, i.e., in polar coordinates they depend on radius only, 2) separability in x and y coordinates, and 3) simultaneous concentration in the spatial and the frequency domain [25].

A 2D Gaussian Kernel is defined as:

$$g(x,y) = \frac{1}{2\pi\delta^2}\exp(-\frac{x^2+y^2}{2\delta^2}) \qquad (3.1)$$

where δ stands for standard deviation. For efficiency purpose, the 2D Gaussian is computed as convolution of two 1D Gaussians, $g(x)$ and $g(y)$, as defined below:

$$g(x) = \frac{1}{\sqrt{2\pi}\delta^2}\exp(-\frac{x^{2^2}}{2\delta^2}) \qquad (3.2)$$

$$g(y) = \frac{1}{\sqrt{2\pi}\delta^2}\exp(-\frac{y^2}{2\delta^2}) \qquad (3.3)$$

3.1.2 Gradient Field

Gradient field has been used as a traditional tool over many years in different image analysis problems [26]. Gradient is basically the change in gray level with direction, which can be calculated by taking the difference in value of neighboring pixels, producing a vector for each pixel. While magnitude of the vector of each pixel refers to the amount of changes in intensity, angle of the vector indicates the direction of intensity changes of pixels expressed in the range of [0-360] degrees. The gradient field ∇f for a local neighborhood *f(x, y)* is computed by using Gaussian derivative operators D_x and D_y [26].

$$\nabla f = \iint ((D_x + iD_y) f)dxdy \qquad (3.4)$$

The integrals are implemented as convolutions with a Gaussian kernel. The complex partial derivative operator $D_x + iD_y$ is defined as:

$$D_x + iD_y = \frac{\partial}{\partial x} + i\frac{\partial}{\partial y} \qquad (3.5)$$

Unlike direction field, which is discussed in the next section, gradient field simply computes the differences in the intensity of pixels with no due attention to linear structures. This has problems in suppression of noises in a noisy image.

3.1.3 Direction Field

A local neighborhood with ideal local direction is characterized by the fact that the gray value remains constant in one direction (along the direction of lines), and only changes in the orthogonal direction [27]. Since the directional features are observed along lines, the local direction is also called *Linear Symmetry (LS)* [26]. The linear symmetry (LS) property of an image can be estimated by analyzing the direction field tensor [25]. The direction tensor is a tensor with real valued triplet that represents the local direction of pixels. For a local neighborhood $f(x, y)$ of an image f, the direction tensor, also called the structure tensor S, is computed as a 2×2 symmetric matrix using Gaussian derivative operators Dx and Dy [26].

$$S = \begin{pmatrix} \iint (D_x f)^2 \, dxdy & \iint (D_x f)(D_y f) dxdy \\ \iint (D_x f)(D_y f) dxdy & \iint (D_x f)^2 \, dxdy \end{pmatrix} \qquad (3.6)$$

Edges are one of the places in an image where linear symmetry exists. This is because there is a gray level change in such areas. An evidence for the existence of linear symmetry in such areas can be estimated by eigenvalue analysis of the direction tensor or equivalently by using complex moments of order [26], two of which are defined as follows:

$$I_{mn} = \iint ((D_x + iD_y) f)^m ((D_x - iD_y) f)^n \, dxdy \qquad (3.7)$$

where m and n are non-negative integers. The orders of interest to us are I_{11} and I_{20}, which are derived as follows:

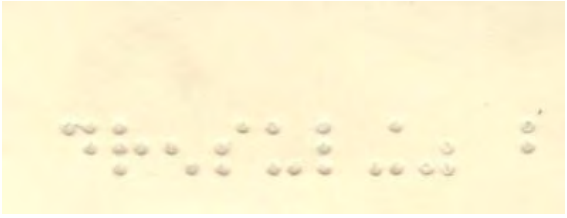
$$I_{20} = \iint ((D_x + iD_y) f)^2 \, dxdy \qquad (3.8)$$

$$I_{11} = \iint | (D_x + iD_y) f |^2 \, dxdy \qquad (3.9)$$

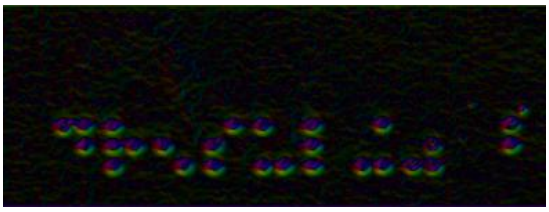
The complex number I_{20} has the local direction of pixels in double angle representation (the direction of major eigenvector) as its argument. The magnitude of I_{20} is a measure of the local linear symmetry strength (the difference of eigenvalues). On the other hand, the scalar I_{11} measures the amount of gray value changes in a local neighborhood of pixels (the sum of eigenvalues) [26].

In this study, direction field tensor is chosen (over gradient field) as a tool for removal of noises from the braille image and separation of the braille dots from their background for the following two major reasons:

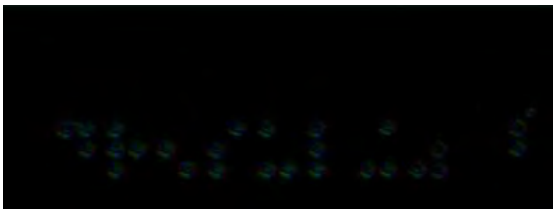
- The complex number I_{20}, unlike gradient field, is able to amplify linear structures and suppresses non-linear structures such as noises that are frequently encountered in digitized braille images (see Figure 3.1).
- The magnitude of gradient fields depends on variations of the intensity of features against the background. Where as, in direction field tensor, such variations can be normalized by using the magnitude of $\frac{I_{20}}{I_{11}}$.



(a)



(b)



(c)

Figure 3. 1. Comparison of direction field and gradient field images. (a) the original image (b) the gradient field image (c) the direction field image

3.1.4 Skewness Correction

One of the important image preprocessing activities in braille recognition is skewness correction. Skewness correction is very important in cases where there is image tilting that may happen as a result of pages getting askew while the braille is being written [9] and incorrect positioning of braille documents during image capturing. Therefore, correction of tilted braille images before proceeding to other steps is crucial. Several researches adopted different techniques for skewness correction. In this study, the horizontal direction nature of dots in a braille document is exploited for skewness correction.

Braille dots, regardless of whether or not the document is skewed, form a line in one direction. Having identified the center of a dot in a line, moving forward within a specific range of the dot"s center, the next dot and hence its center will be identified. This process continues up to the end of the line and the same process will be repeated for subsequent lines. Using the identified braille dot centers of each line, the slope of the horizontal line is determined. The angle θ, as shown in Equation (3.10), can be estimated from the slope.

$$\theta = \tan^{-1}(Slope) \tag{3.10}$$

Once the angle θ is obtained, skewness correction can be performed by rotating the original image by $-\theta$. This technique effectively corrects braille image tilting of up to 5 degrees from both directions.

3.2 Analysis of Important Braille Cell Values

One of the challenges in braille recognition is identification of important features like the size of the braille dot. Even though, the dot height and width, horizontal distance, vertical distance and other important features of a braille document are already specified for real braille documents, the digitized braille image may have different values for such kind of features depending on, for instance, the resolution of the scanner that is used during image capturing [22]. Chekol [2] states that attempts to recognize a braille document with a fixed dot size in mind is bound to result in some problems when a braille document with a different dot size is encountered. Thus, a system of automatic dot size detection

and appropriate estimation of the other values like the horizontal and vertical spaces between dots has a paramount importance for the recognition process.

We propose an algorithm that detects height of the braille dot and deduce other elements like inter-cell and intra-cell spacing from this value. The algorithm starts by searching for the top and bottom value of the left most dot of the first braille line. This gives us the height of the first dot. This dot alone, however, cannot be considered as a representative dot. The same task is, therefore, done for other braille dots in the braille document. Finally, the average dot height is calculated from all the dot heights identified. The average dot height is then used for estimation of important values that are used at later stages of the recognition process. The estimated values include; dot width, height of two dots (which is not a simple sum of two dot heights as there is a space between the two), height of three dots (or height of a braille character), inter-cell space, and intra-cell space.

In the process of determining braille dot height, it is important to deal with only one or two dots per line, especially in cases where there is image tilting. Though skewness correction is already dealt with in this study before identification of dot height, the algorithm proposed for dot height identification works well even in cases where there is a slight image titling. This is because the calculation of height of one or two dots (that are part of the same braille cell) in slightly tilted images will not have a significant deviation from the actual height. However, even better would be dealing with only one dot at a time to calculate the dot height. This option is not considered because separation of possible merging of adjacent dots that usually exists in braille images is not yet performed at this stage. Hence, the algorithm is designed to calculate the height of one or two consecutive dots at a time. Separation of merged adjacent dots is deferred since the process requires the value of a dot width, which is derived from the dot height.

3.3 Braille Character Line Identification

Characters in a braille document are ordered in the same arrangement as in print document. Therefore, there is apparently enough space between characters of different lines that enables us to distinguish one from the other (See Figure 3.2). Construction of the braille character lines is an important step forward in the process of identifying and

recognizing braille characters. The space difference just mentioned can be utilized to construct the braille character lines.

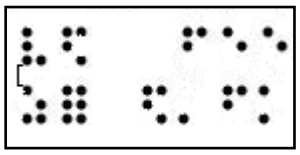


Figure 3. 2. Braille character line space

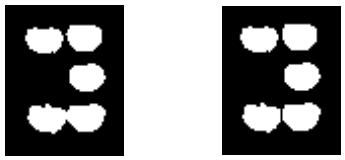
Braille character line points are identified by a simple horizontal scan across the braille document. The first braille dot‟s top point that would be obtained as a result of this process is taken as the top point of the character line. A successive analysis of subsequent lines of the braille image continues in order to identify bottom point of the character line. Bottom point is identified when the space distance between bottom of the immediate previous dot and top of the current dot becomes more than a specified value. The other way of identifying bottom of a character line is through analysis of the distance between top of the character line and bottom of the current braille dot. If the distance is approximately the same as the height of a braille character, which is estimated from dot height as discussed under section 3.3, the bottom of the current braille dot will be taken as the bottom point of the character line. Similar procedure is followed to identify top and bottom points of all the other character lines in the braille image until end of the braille image is reached. Each row‟s top and bottom points are then marked in order for them to be used by subsequent activities.

Determining character lines of the braille document before other steps like half-character detection will allow the other steps to be performed only within the braille character lines, instead of the entire document. This will have a positive effect on the performance of the system as only part of the image is processed.

3.4 Half-character Detection

Out of the 64 combinations that we obtain from six positions of a braille character, sixteen of them do not have any dot in one of the columns (half characters). In this step, all the half characters that do have at least one dot are detected.

The half character detection algorithm works correctly if merged adjacent dots are separated. For this reason, before the half character detection process begins, the merged dots are separated. Merged dots‟ separation process requires the knowledge of the actual braille dot width, which is derived from the braille dot height that we have already identified as discussed under section 3.2. Accordingly, those possible dots that have a width of more than one and half of the dot width value will be separated into two dots (See figure 3.3).



(a) (b)

Figure 3. 3. Separation of merged dots

As can be seen from Figure 3.3, the braille cell labeled (a) has its two dots at the bottom merged. Label (b) shows what the same braille cell looks like after the merged dots are separated.

Each braille character line is traversed and a check on the existence of a possible candidate half character is performed. The width of a dot is considered as a criterion to determine the validity of the candidate half character. If the candidate half character has a width that is greater than at least half of a braille dot width, it will be considered as a valid half character and will be marked accordingly. Otherwise, the candidate half character will be disregarded. This is very important to do away with some noises and artifacts that are detrimental to the recognition process.

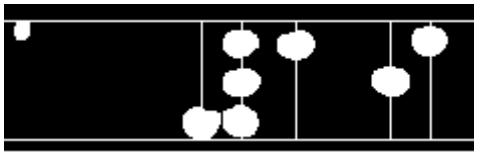


Figure 3. 4. Exclusion of noises from half character detection

As can be seen from Figure 3.4, the first candidate half character, which is a noise, is disregarded as its width does not fall in the range that is expected of valid half characters.

Whereas, the last five candidate half characters are considered as valid half characters because they satisfy the criteria.

3.5 Half-character Recognition

Half character detection is followed by half character recognition where each of the identified columns is analyzed to determine to which of the seven possible categories it belongs. The seven categories, as can be seen from Table 3.1, refer to the possible combinations that we get from the three positions of a half character (excluding of course the empty half character, which is yet to be discussed under section 3.6).

Analysis of the height of dot(s) in a column is the approach followed to determine the value of a specific half character. Initially, top point of the top dot in the column is determined and marked as top point of the column. The bottom point of the bottom dot in the column is determined and marked as bottom point of the column. Subtraction of top point of the column from the bottom point gives us the height of the column. According to this analysis, half characters whose values are 7 or 5; 3 or 6; 1, 2 or 4 will have the same value. Therefore, further analysis on the half characters is necessary in order to classify them accordingly.

Table 3. 1. Half characters and their corresponding value

Half character	**Value**
○ ○ ●	1
○ ● ○	2
○ ● ●	3
● ○ ○	4
● ○ ●	5
● ● ○	6
● ● ●	7

Half characters whose values are 7 or 5 will have the same height. This is because both types of half characters do have a dot at both the top and bottom positions of the column. The only thing that differentiates them is the dot that resides at the middle position. A half character whose value is 5 does not have a dot in the middle position of the column where as a half character whose value is 7 does (see Table 3.1). Therefore, having identified that the value of a column‟s height belongs to a half character with the value of either 5 or 7, we need only to check the existence of a dot in the middle position of the half character. If a dot is found, the half character is recognized as 7, otherwise it is recognized as 5.

6 and 3 are the values of the group of half characters that do have the same height. As can be seen in Table 3.1, the half characters with values 3 and 6 have only two consecutive dots in the column, which is why these two types of half characters have the same height. The feature that differentiates these two half characters is the positions at which the two dots are located in the column. The two dots of the half character with the value 3 are located at the second and third positions from top. The dots of the half character with the value 6, on the other hand, are located at the first and second positions of the column from top. This feature is used to recognize a half character with two consecutive dots as either 3 or 6.

The last group of half characters that have the same height are those with values 1, 2 and 4. The common feature of these half characters is that all have only one dot though at different positions of the column (see Table 3.1). The dots of the half characters with the value 4 and 1 are located at the top and bottom of the column respectively. Where as, the dot of the half character with the value 2 is located at the middle position of the column. This knowledge is used to recognize a half character that has only one dot as 1, 2 or 4.

3.6 Braille Cell Formulation

Half character detection and recognition deal with just half of the braille character. In order to have a complete picture of the braille document, the braille cells that are made of two half characters should be formulated.

Formulation of a braille cell is not as straight as simply combining the identified and recognized half characters. It is rather a lot more challenging due to the fact that some

braille cells have dots in only one of the two columns, which requires us to do some analysis to decide whether the other half (the column with no dots) is located to the right or to the left of the already identified half.

The inter-cell and intra-cell spaces of braille dots in a braille document is distinct. This knowledge is used by different researches [5, 18] to formulate braille cells. In the current study, as in many other values, the inter-cell and intra-cell space values are automatically derived from the dot height. It is worth noting that these values may not be necessarily the same to all kind of braille document scanned with different resolutions.

The horizontal distance analysis is used to determine if two consecutive columns are part of one braille cell or are simply part of two separate braille cells and the cell is formulated accordingly. In cases where a braille cell has dot(s) in only one of the two columns, the distance of the column from columns of the left and right side is determined to decide to which side the braille cell in question has the empty column. At the end of this step, all braille cells (those braille cells that have dots on both columns as well as those that have dots in either of the two columns) are formulated including the empty braille cell, which represents a space.

3.7 Translation

The final step in the braille recognition process is translation of the braille cells into their equivalent print characters. In grade 1 braille of languages like English, the translation from braille to print is almost one to one though some minor context analysis is required, for example, for digits. The translation in grade 2 braille, however, is a lot more complex as one braille cell may have different meanings depending on its position in a specific word.

As already mentioned in chapter two of this document, we still do not have grade 2 Amharic braille. Nevertheless, the translation of the Amharic grade 1 braille is not as direct and easy as the grade 1 braille of languages like English. This is due to the different form of braille representation that is used for Amharic characters. The different representation is necessitated, as discussed in chapter two, by the large number of symbols that exist in Amharic language.

In this representation one, two, or sometimes three braille characters may be used to represent one Amharic print character. In addition, as in other language braille representations, indicator braille characters are used to show whether the following character should be interpreted as digit or any other character. All these features are taken into consideration in the translation method employed in this study.

A translation table that contains the Amharic characters and their corresponding braille representation is prepared. Each braille cell is analyzed vis-à-vis its right side neighbours to determine what it actually represents. Having identified the corresponding print character of each braille cell, the print characters are written in a text file.

Chapter 4

Implementation and Experiment

This chapter presents implementation details of the proposed system in terms of narrative pseudo codes and some snap shots where deemed necessary. Accordingly, details regarding the preprocessing activities are discussed under section 4.1. Estimation of the different important braille related values that are used at later stages of the recognition process is explained in section 4.2. Braille character line detection that determines part of the braille image, which is the focus of subsequent activities, is the subject of section 4.3. Sections 4.4 and 4.5 talk about half character detection and recognition respectively. The horizontal space analysis that is used for braille cell formulation is discussed under section 4.6 while translation of the formulated braille cells into their print text representation is explained in section 4.7. Finally, the performance of the proposed system in comparison with the previous two works on Amharic braille is presented in section 4.8.

4.1 Preprocessing

The three preprocessing techniques used in our study namely, removal of noises and separation of dots from the background using direction field tensor as a tool, thresholding and skewness correction are discussed below.

4.1.1 Direction Field Image

Direction field tensor, which is a tensor field defined over local images for all points in the entire image, can be conveniently represented by the 2D complex I_{20} and 1D scalar I_{11}. The following procedure, as adopted from Begun et al. [25] is followed to compute the direction field (I_{20} and I_{11}):

1. Generate a 2D Gaussian kernel and a 2D Gaussian derivative kernel. For efficiency purpose, the two 1D Gaussian kernels (*gx* and *gy*) and two 1D Gaussian derivative kernels (*dx* and *dy*) can be used.
2. Apply convolutions on the original image *img* to generate *dxf* and *dyf* .

$$dxf = gy * (dx * img)$$
$$dyf = gy * (dy * img)$$

3. Compute:

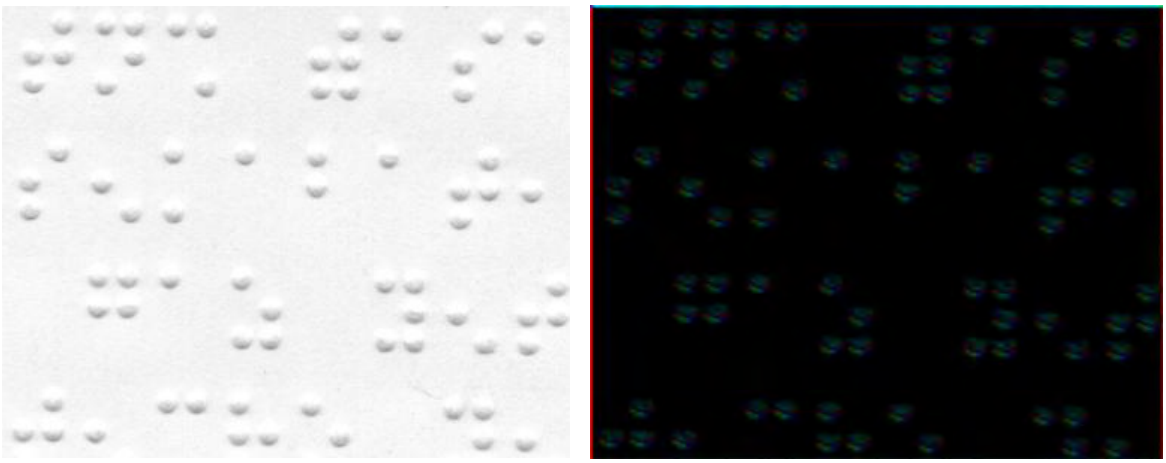
   a) $\hat{I}_{20}$ from dxf and dyf by pixel-wise complex squaring.

   $$\hat{I}_{20} = (dxf + i * dyf)^2 \text{ where } i = (-1)$$

   b) $\hat{I}_{11}$ from abs($\hat{I}_{20}$)

4. Compute the complex image I_{20} and the scalar image I_{11} from $\hat{I}_{20}$ and $\hat{I}_{11}$ respectively by averaging with a second set of Gaussian kernels.

After normalization, the direction field image looks something like as shown in Figure 4.1 (b).



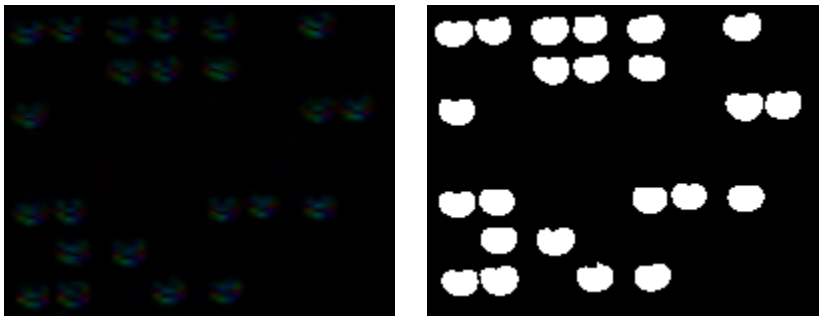
(a) (b)

Figure 4. 1. Direction Field Image (a) represents the original image in gray level. The image labeled (b) shows the direction field image.)

As can be seen in Figure 3.1., all pixel values except those of the braille dots, are set to the value zero.

4.1.2 Thresholding

Thresholding is a simple tool to separate objects from the background [30] the output of which is a binary/logical image. Working with a logical image reduces the complexity of the whole execution process as we deal with only two possible values (0 and 1). The direction field image obtained, as discussed under sub section 4.1.1, is converted into a logical image by setting all pixels that have a value different from zero into 1 and keeping the pixels with the value 0 as they are. This gives us another image where the braille dots are set to a white color while the background remains black (see Figure 4.2).



(a) (b)

Figure 4. 2. Thresholding direction field image (the picture labeled (a) shows the direction field image. The picture labeled (b) shows the logical image)

Additional operations are employed in order to remove some artifacts that might have remained in the braille image, especially when the braille document in question is old or worn out. The operations are used to fill connected objects in the image and then those objects with smaller sizes are removed from the image.

4.1.3 Skewness Correction

As already elaborated under sub section 3.2.3 of this document, we have taken advantage of the horizontal direction nature of dots in a braille document in an attempt to estimate the slope of the line that is constructed out of the braille dots.

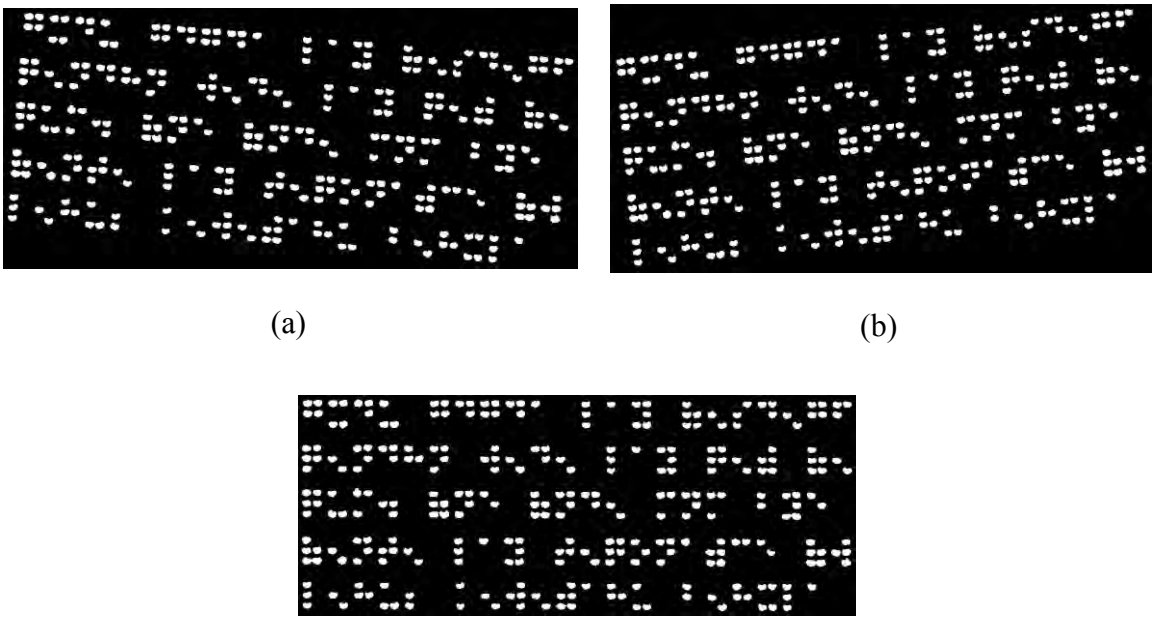
In order to estimate the slope, the first step to be taken is to identify the center values of braille dots in a line. The following algorithm shows the steps that we have followed for skewness correction.

1. *The braille image is traversed horizontally starting from the top of the image and the first braille dot identified will be taken as the starting point. (And the first dot's bottom value will be buffered so that the search for the next line's dot starts just from below this point)*
2. *The center of the first dot is calculated and its row and column values are stored in two arrays that are prepared to store row and column values of the center of braille dots.*
3. *By taking the first dot's center value as a starting point, the search for the second dot will continue in ± 3 pixel ranges from the center. (Note that for subsequent dots, the range will have higher values from top or bottom and decreases from the opposite side in accordance with the direction of the line)*
4. *Having identified the center value of the second dot, the search for the other dots in the line continues in the same fashion.*
5. *The moment the center of the last dot in the line is identified, the center values of the first and the last dots are used to calculate the local slope. (local slope refers to the slope calculated for a single line)*
6. *Steps 1-5 are repeated for subsequent dot lines in the braille image.*
7. *Finally, the average of all local slopes is calculated to get the global slope.*
8. *MATLAB's* ***atand()*** *function is used to obtain the degree of the tilting from the global slope.*

9. *The negative of the degree obtained from the* ***atand()*** *function is used to rotate the image back to its proper position using MATLAB's* ***imrotate()*** *function.*

List 4. 1. Pseudo code for skewness correction.

Figure 4.3 portrays the result of the skewness correction from both directions using the above algorithm.



(a) (b)

(c)

Figure 4. 3. Skewness correction ((a) 5 degree tilting towards the right hand side, (b) 5 degree tilting towards the left hand side, (c) the result of the skewness correction.

As can be seen from Figure 4.3, the proposed algorithm works perfectly for image tilting of up to 5 degrees. The result is encouraging as it performs better than previous works like [1], where tilting of up to 4 degrees is achieved.

4.2 Estimation of Important Braille Cell Values

An automatic braille recognition system basically deals with important values of a braille cell such as dot height, dot width, inter-cell space, intra-cell space and the like. A flexible analysis and estimation of such values is important if the recognition system is to work effectively for different braille documents with apparently different values.

The braille dot height is what we found in this study to be the only value that can be directly obtained by a simple analysis of the braille document through exploiting the distinct spacing that we have between braille dots of different lines. The possible existence of horizontal braille dot merging makes it difficult to obtain values like braille width, inter-cell space and intra-cell space directly from the braille document. Therefore, we have identified the dot height from the braille document and derived all the other values from the dot height. The following algorithm explains how the braille dot height is obtained.

1. *The braille image is traversed horizontally from top to bottom in search of the top point of the first braille dot.*
2. *Having identified the top point of the first dot (of the first line), the search for the bottom point of the same dot continues.*
3. *The top and bottom points are then used to calculate the height of the braille dot and the value is stored in an array that is prepared to store braille dot heights.*
4. *Steps 1-3 are performed to subsequent lines in the braille image.*
5. *After dot height is calculated for all lines of the braille document, the average value is calculated and used for further processing*

List 4. 2. Pseudo code for identifying braille dot height.

Identification of the braille dot height helps us estimate the height of a braille character, which is further used for the estimation of values like inter-cell space, intra-cell space and dot width. In order to estimate the braille character height, the height of each half character is obtained. The height of the half-character is compared against 3 times the braille dot height. If it is greater, it will be considered as the height of a braille character. The values obtained from each braille character are finally summed up and their average is taken as the braille character height. The other values are estimated as shown in Table 4.1.

Table 4. 1. Estimation of important braille cell values

Value	**Derived from**
Two dot height	(2/3)*three dot height
One dot height	(1/3)*three dot height
Dot width	(Two dot height)/2
Inter-cell space	(one dot height)+((one dot height)*0.1)
Intra-cell space	((three dot height)/2)+((one dot height)*0.2)

4.3 Braille Character Line Detection

In this study, out of the entire braille image, analysis for identification and recognition of braille cells is done only on the braille character lines. This is advantageous from performance point of view as only part of the braille image is processed instead of the entire image. Analysis of the vertical distance between dots is the approach used to identify braille character lines.

The following algorithm shows the steps followed for identification of braille character lines.

1. *The braille image is horizontally traversed in search of the top point of the first braille character line. This is simply accomplished when the first pixel value of 1 (background pixel values are all zeros) is identified.*
2. *The validity of the braille dot the top point of which is to be considered as the top point of the character line is checked.*
3. *If it is not a valid braille dot (if it is a noise), it will be disregarded and the top point of the character line will be taken from other valid braille dots.*
4. *To identify bottom point, pixels of the image are horizontally analyzed and when a line with pixel values of all zeros is found, the immediate previous pixel line is marked as a candidate bottom point of the braille character line.*

5. *The candidate bottom point of a braille character line is marked as an actual bottom point if it is located at a distance of approximately three times the dot height away from the top point.*
6. *The candidate bottom point of a braille character line is also marked as an actual bottom point if the distance between its bottom point and the top point of the next dot line is greater than a braille dot height.*
7. *Once the top and bottom points of the first braille character line are identified, steps 1 to 6 are repeated to identify all the other braille character lines in the image.*

List 4. 3. Pseudo code for braille character line detection.

The top and bottom points of each character line of the braille image identified are shown in Figure 4.4.

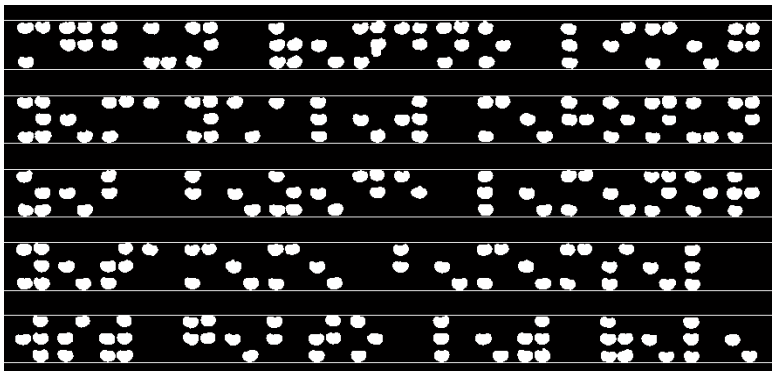


Figure 4. 4. Braille character line identification

4.4 Half-character Detection

Some braille cells have dots in both columns while others do have only in one of the two columns. Half character detection focuses on identifying columns of braille cells where there is at least one dot.

The half character detection algorithm simply traverses through each braille character line and identifies half character and marks them accordingly. In order for this process to go smoothly, possible merging of adjacent dots should be resolved first. Merged columns are therefore separated (see List 4.4) before the half character detection step begins.

1. *For each braille character line, the search for the left position of the first braille cell column is done.*
2. *Then the search for right position of the first braille cell column continues.*
3. *The width of the column is calculated from the right and left positions identified.*
4. *If the width is approximately greater than the known braille dot width, the column is separated into two braille cell columns.*
5. *Steps 1 to 4 are repeated to all braille cell columns of each braille character line.*

List 4. 4. Pseudo code for separating merged dots.

The half character detection algorithm is described below.

1. *For each braille character line, left position of the first braille cell column is searched.*
2. *Starting from the left position of the first braille cell column, the search for the right position continues.*
3. *The width of the braille cell column is calculated from the left and right positions just identified.*
4. *If the width is greater than half of the known braille dot width, the center of the column will be calculated from the left and right positions and is marked as a half character position.*
5. *If the width is not greater than half of the known braille dot width, the candidate half character is disregarded.*
6. *Steps 1 to 5 are repeated to all candidate columns of each braille character line.*

List 4. 5. Pseudo code for half character detection.

Figure 4.5 below shows the half characters identified in each of the braille character lines.

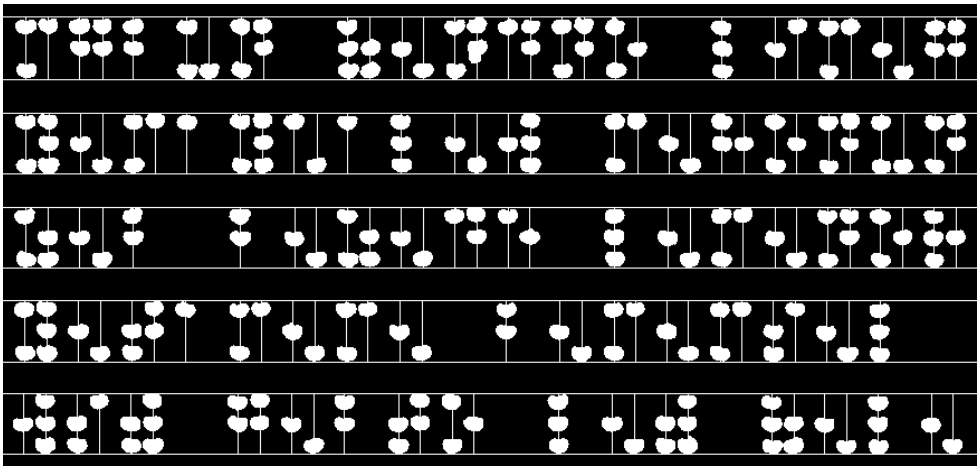


Figure 4. 5. Half character identification

4.5 Half-character Recognition

At this step, part of the recognition process where by the half characters are classified is performed. The half characters are made of three dot positions. Excluding the condition where all dot positions are flat, we have 7 different combinations. Based on the values designated to each type of half character, (see Table 3.1) the algorithm to classify half characters to their corresponding values is described in List 4.6 below.

1. *For each braille character line, the first dot's top point and last dot's bottom point of a half character is obtained. (Note that if there is only one dot in the half character the top and bottom points of the dot are taken as the top and bottom points of the half character).*
2. *The height of the half character is calculated from the top and bottom points obtained.*
3. *Based on the value of the height the half character will be forwarded to one of the three functions for further processing. (The three functions work on those group of half characters that have similar heights as shown in Figure 4.6)*
4. *If the height is approximately the same as the height of a three dot height, the function that processes half characters of values 5 and 7 will be called.*

5. *The function checks the middle position of the half character and if it is empty, it returns the value 5. Otherwise it returns the value 7.*
6. *If the height is approximately the same as the height of a two dot height (the height of two consecutive dots), the function that processes half characters of values 3 and 6 will be called.*
7. *The function checks whether the two consecutive dots are located at locations **1 and 2** or **2 and 3** from top. If the dots are located at **1 and 2,** the value 6 is returned. If the dots are located at **2 and 3**, the value 3 is returned.*
8. *If the height is approximately the same as the height of a one dot height, the function that processes half characters of values 1, 2, and 4 will be called.*
9. *The function checks if the dot is located in one of the three possible positions of the half character.*
10. *If the dot is found to be at the top position, the value 4 is returned. If the dot is found to be at the middle position, the value 2 is returned. Otherwise, if the dot is found to be at the bottom position, the value 1 is returned.*
11. *All the above steps are repeated to all half characters of all the braille character lines.*

List 4. 6. Pseudo code for half character recognition.

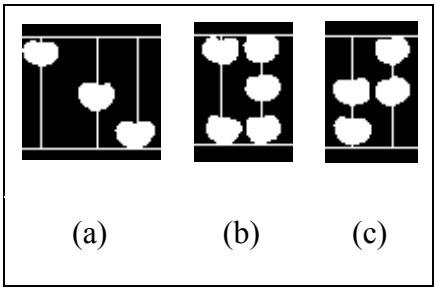


Figure 4. 6. Group of half characters with similar heights.

Figure 4.7 below shows the result of the half character recognition process for some set of half characters.

Figure 4. 7. Half character recogntion

4.6 Braille Cell Formulation

Half characters do not convey a complete meaning by themselves as they are just part of braille cells. In order for the half characters to have meaning, they should be put together to form a braille cell. Analysis of the horizontal distance between half characters is the approach used to formulate braille cells. Inter cell space and intra cell spaces are used to differentiate between half characters of the same braille cells from those that are of different cells. The algorithm described below in List 4.7 shows how braille cells are formulated.

1. *For each braille character line, the distance of the position of the first half character is compared with the position of the next half character.*
2. *If the distance is approximately the same as the distance of an inter-cell space, the two half characters are grouped together to form one braille cell.*
3. *In a similar fashion, for subsequent half characters, the distance between the position of a half character and the positions of the next and the previous half characters is computed.*
4. *If the distance is found to be approximately similar to the distance of* ***inter-cell + intra-cell distances*** *or* ***2*(inter-cell) + 2*(intra-cell distances)*** *(where there is a space between the two columns), the half character in question is combined with a half character with the value zero to form a complete braille cell. (The half character with the value zero may be put on the left or right hand side of the half character based on the analysis. In the case of the second situation, a space indicator will be inserted between the previous and the current braille cells. Space indicators are also inserted in other cases where there is an approximate*

distance of ***(inter-cell) + 2*(intra-cell distances)*** *is found between two braille cells with half characters of values different from zero).*

5. *Steps 1 to 4 are repeated to all half characters of all braille character lines.*

List 4. 7. Pseudo code for braille cell formulation.

Figure 4.8 below shows how the braille cells are formulated out of the half characters using the above algorithm.

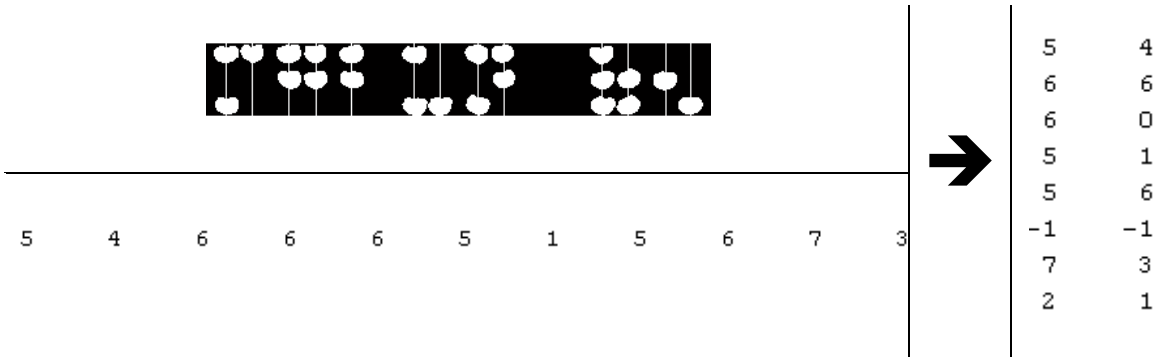


Figure 4. 8. Braille cell formulation

Note that the third braille cell, in Figure 4.8, has only one column on the left side with the value 6. The right hand side column is therefore deduced as 0 by distance analysis with neighboring cells, hence the braille cell (6, 0). On the other hand, the sixth braille cell has the value (-1, -1), which is used as a space indicator. This space can be clearly seen in the braille image of Figure 4.8.

4.7 Translation

In this step, the braille cells that are formulated are translated into their corresponding print text values. Four lookup tables for letters (and those labialized letters that are represented by two braille characters), numerals (both Arabic and Ethiopic), punctuation marks, and labialized letters (only those labialized letters that are represented by three braille characters) are prepared for the translation purpose. The lookup tables contain the braille cell values and the corresponding Unicode values of the respective character groups.

Some punctuation marks are found to have the same braille cell representation with the Amharic letters, which creates confusion in the translation process. To avoid this

confusion, a word or sentence level analysis, which is way beyond the scope of the current work, is required. Hence, those punctuation marks that are known to create such confusions are excluded from the lookup table.

In Amharic braille reading, since characters are represented by up to three braille cells, analysis of the neighboring braille cells is necessary in order to determine the print text value. Moreover, digit mode indicator cells also need to be taken into consideration in the translation process.

The algorithm that is used for translation purpose is described as follows in List 4.8.

1. *From the list of braille cells formulated, the value of the first braille cell is checked to see if it has the same value as the Arabic or Amharic digit mode indicators.*
2. *If the braille cell has the same value as either Arabic or Ethiopic digit mode indicator, the next braille cell is checked against the numerals lookup table and the Unicode value of the print character value is returned and stored in the array that is prepared to store print text values.*
3. *If the braille cell is different from digit mode indicators, the braille cell along with two more succeeding braille cells are check against the labialized lookup table. If a match is found the Unicode value of the print text is returned and stored in the array that is prepared to store print text values.*
4. *If the braille cell is different from the digit mode indicators and no match is found from the labialized lookup table, the next braille cell is checked to see if it is a braille cell that represents vowels.*
5. *If the next braille cell is not a vowel, the first braille cell's value is checked against the letters lookup table and the Unicode value of the print character is returned and stored in the array that is prepared to store print text values.*
6. *If the next braille cell is found to be a vowel, the first cell along with the second cell (the vowel) will be checked against the letter lookup table and the Unicode value of the print character is returned and stored in the array that is prepared to store print text values.*

7. *If no value is returned from above steps, the braille cell is expected to represent a punctuation mark. Accordingly, a check against on the punctuation mark lookup table is done for* ***the first, the second and the third braille cells*** *or* ***the first and the second braille cells*** *or* ***the first braille cell*** *only. The corresponding Unicode value of the print character is finally returned and stored in the array that is prepared to store print text values.*
8. *Each of the above steps is repeated to all other braille cells in the list.*
9. *Finally, each of the Unicode values that are stored in the array is written into a Rich Text Format (RTF) file.*

List 4. 8. Pseudo code for braille to print translation.

As can be learnt from the above algorithm, while a maximum of two braille cells are analyzed for Amharic letter translation, up to three braille cells are processed for the translation of Amharic punctuation marks and labialized letters. This is because the Amharic characters that are represented by three braille cells are those that fall in the category of punctuation marks and labialized letters.

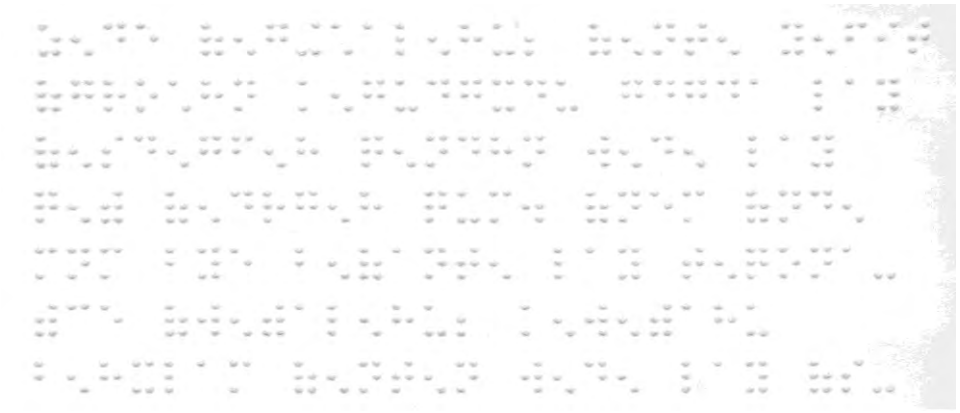
4.8 Performance Evaluation

In an attempt to determine the performance of the proposed system, the prototype we have designed has been experimented against different braille documents. The braille documents used for experiment purpose are mainly collected from Misrach Center. Attempts are done to use representative braille documents of all sorts: skewed, worn-out, and different contents. Table 4.2. shows the statistics of the braille documents that are used in the experiment. For comparison purpose, majority of the braille documents used in the experiment are those that were used by Chekol [2].

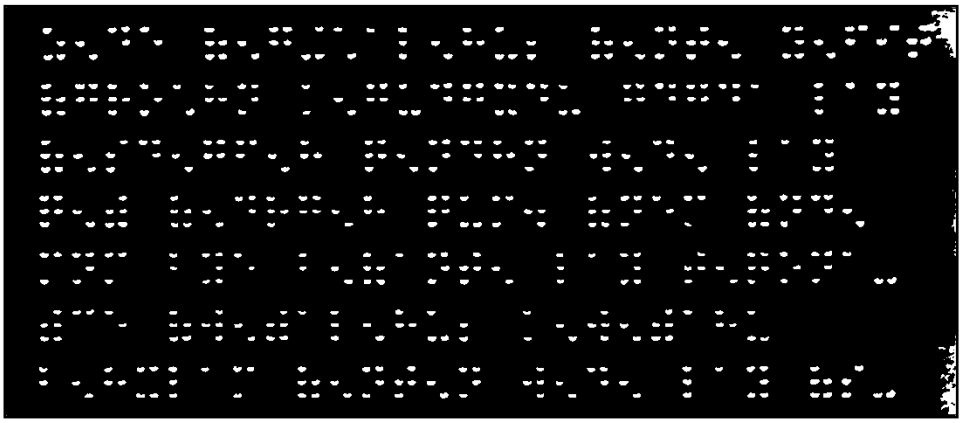
Table 4. 2. Statistics of braille documents collected for experiment

Braille Attribute	**Value**
Number of braille documents	7
Average number of characters per document	375
Color Type	True color
Resolution	200 dpi
Average image size	8.5MB
Braille Type	Single sided
Image format	Bmp
Document type	A4 size

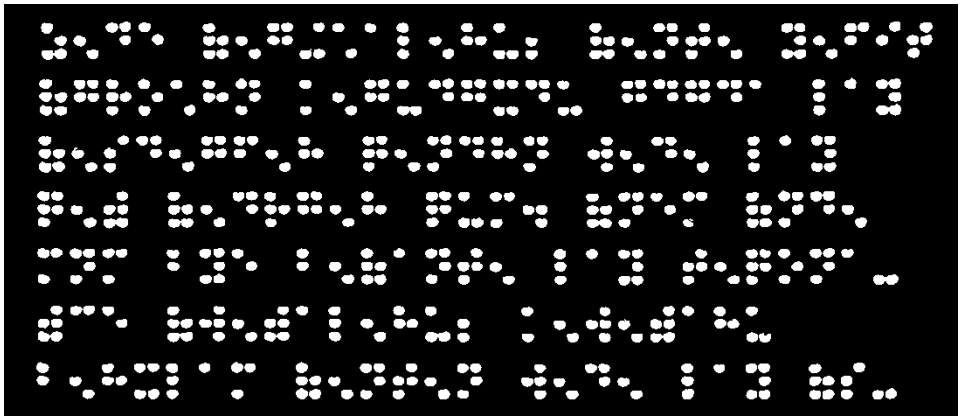
The correctness of each translated print text is checked against the original braille document and the accuracy of our system is calculated on a print character basis. Accordingly, the system has achieved an average accuracy of 98.5%. The accuracy for good quality braille documents is 99.9%. This is by far better than the previous two works [2, 3] on Amharic braille that achieved an accuracy of 92% and 96% respectively for good quality braille documents (See Figure 4.9). We have also achieved 96.5% accuracy for poor quality braille documents, which again is remarkable compared to the previous work of Chekol that achieved 65% for poor quality braille documents (See Figure 4.10). It is also worth mentioning that our system includes labialized letters, Amharic numerals, and more punctuation marks, which were not dealt with in the previous two works.



(a)



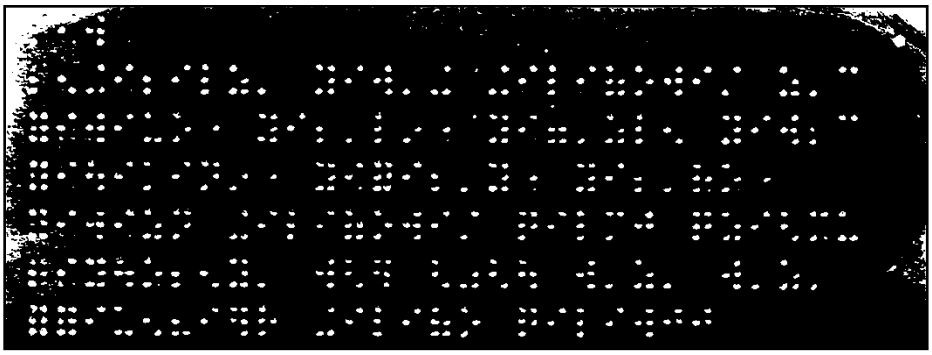
(b)



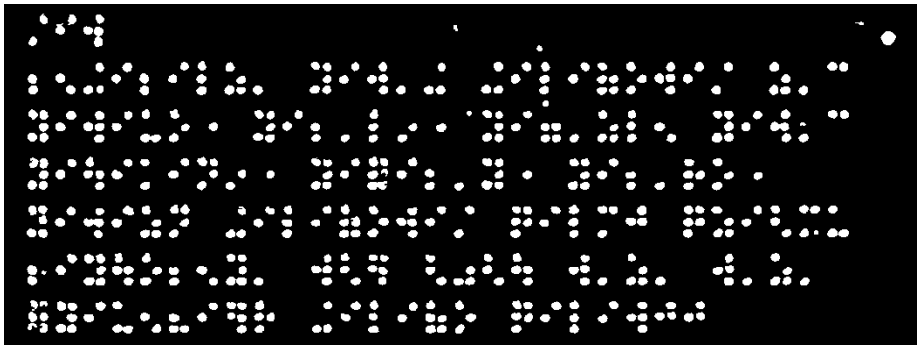
(c)

Figure 4. 9. Comparison of good quality braille documents. ((a) original image. (b) Preprocessing result of a by the previous work [2]. (c) Preprocessing result of a by the current work)

(a)



(b)



(c)

Figure 4. 10. Comparison of poor quality braille documents. ((a) original image. (b) Preprocessing result of a by the previous work [2]. (c) Preprocessing result of a by the current work)

The commendable result we have achieved is mainly due to the effective noise removal technique we have adopted and the different mechanisms we have employed for exclusion of non-braille dot artifacts during braille character line identification and half

character detection. The skewness correction we have designed has also played a significant role for the result.

The small errors observed in the experiment of our system are attributed to some stains and defects present on the surface of the braille documents that are caused by overuse and bad handling of the documents among others.

Chapter 5

Conclusion and Recommendation

5.1 Conclusion

In this study, an attempt has been made to design and implement an automatic braille recognition system for Amharic language. We have, to this end, adopted and designed different techniques for the different steps of the recognition process. Direction field image, which uses Gaussian filters and derivatives of Gaussians have been used as a tool for removing noises and isolating braille dots from their background. The possible image tilting that usually occurs during image capturing has been resolved by the new skewness correction algorithm we have designed, which helps us tolerate up to 5 degree of image tilting on both directions. We have exploited the horizontal direction nature of braille dots to design our skewness correction algorithm.

We have attempted to determine automatically the different values that we encounter in braille images such as braille dot height, dot width, inter-cell space, intra-cell space and the like. These values tend to be different depending on the resolution level used during image capturing. What we have proposed however determines these values automatically from the braille image and therefore works for braille images of possibly different dot sizes.

We have identified the braille character lines and limited the analysis of subsequent operations to focus only on this part of the braille image, which has benefits from performance point of view.

Braille cells are formulated from half characters by way of analysis of spaces between the half characters. The braille cells are translated into their corresponding print characters using table lookup, which contains list of braille cell values and the associated print characters.

The prototype we have designed for our system has been experimented against braille documents of different sorts in terms of quality and content. We have achieved an average accuracy of 98.5%. The small errors observed in our experiment are attributed to the different stains and defects present on the braille documents.

Compared to the previous two works on recognition of Amharic braille, our proposed systems has dealt with more number of Amharic characters and introduced more additional techniques for the recognition process such as skewness correction and automatic determination of braille dot size. We have also adopted appropriate noise removal tool and translation mechanism. As a consequence, we have achieved better results, especially for poor quality braille documents.

5.2 Recommendation

We have adopted and designed different techniques for recognition of Amharic braille documents and achieved an encouraging result. However, with the incorporation of the following points in future works, we believe a better result would be achieved.

- We have used the dot height and the dot width values to determine whether a candidate dot is valid or not during the braille character line and half character detection steps. Incorporating additional analysis, on top of this, like examination of the bright and dark regions that braille dots exhibit would help better differentiate braille dots from noises.
- Our current work aimed at single-sided braille documents. However, nowadays, double-sided braille documents are being widely used. A research work that focuses on both single-sided and double-sided braille documents would be more useful.
- Some of the errors we have encountered due to the stains and defects present on the braille documents would result in translation of incorrect words. A pos-processing activity that corrects such errors by changing the incorrect words with the closest correct word from a dictionary would be very appropriate.
- In this study, we have included all Amharic characters except some punctuation marks that have the same representations with other characters like some of the core Amharic letters. In order for these punctuation marks to be included and interpreted properly, a word or sentence level analysis of the braille characters, which is beyond our scope, is required. Future works can focus on this area so that such punctuation marks would be easily included in the translation.

- Keeping the format of braille documents is sometimes very important when, for instance, the content in the document is a poem. Keeping the format would also be very instrumental if the recognized content is required for reproduction of braille documents.

References

[1] A. AlSalman, Y. AlOhali, M. AlKanhal and A. AlRajih, "An Arabic Optical Braille Recognition System. ", ICTA'07, April 12-14, Hammamet, Tunisia

[2] E. Chekol, "Recognition of Amharic Braille Documents." (Masters Thesis), Faculty of Informatics, Addis Ababa University, 2010

[3] T. Alemu, "Recognition of Amharic Braille." (Masters Thesis), Faculty of Informatics, Addis Ababa University, 2009.

[4] C. Ng and V. Lau, "Regular feature extraction for recognition of Braille", In Proceedings of Third International Conference on Computational Intelligence and Multimedia Applications, ICCIMA'99, pp. 302-306, 1999.

[5] L. Wong, W. Abdulla and S. Hussmann, "A Software Algorithm Prototype for Optical Recognition of Embossed Braille", the 17th conference of the International Conference in Pattern Recognition, Cambridge, UK, pp. 23-26, August 2004.

[6] International Eye Foundation FY 2007-2008 Annual Report, available at: http://www.iefusa.org/download/IEF 2008 Annual Report final.pdf, accessed on 8/28/2010.

[7] National survey on blindness-low vision and trachoma in Ethiopia, available at: http://www.trachoma.org/tmatters/itin4.pdf, accessed on 8/28/2010.

[8] M. Million, "A Generalized Approach to character Recognition of Amharic Texts", (MSc. Thesis). Addis Ababa Univerity, School of information Studies for Africa, Addis Ababa, 2000.

[9] J. Mennens, L.P. Tichelen, G. Francois and J.J. Engelen, "Optical Recognition of Braille Writing Using Standard Equipment", IEEE transactions of rehabilitation engineering, 2(4): 207-212, 1994.

[10] http://www.dotlessbraille.org/braillewritingmethods.htm: accessed on April 27, 2011

[11] በልዩ ትምህርት ቡድን ሥልት/ዝ/ጥናትና ምርምር ኢንስቲትዩት:: የብሬል ሜትሜሪያ የመማህሩ መመሪያ አዲስ አበባ 1991::

[12] P. Blenkhorn, “A System for Converting Braille into Print”, IEEE Transactions on Rehabilitation Engineering, Vol. 3, No. 2, pp. 215-221, June 1995

[13] በኢትዮጵያ አይነስወራን ብሔራዊ ማህበር 12ኛ ዙር ስራ አስፈፃሚ ኮሜቴ የተቋቋመው የአሞኛ ብሬል አሻሻይ ኮሜቴ:: የአሞኛ ብሬል አሻሻይ ኮሜቴ ዘገባ:: ታህሳስ 19 እና 20 ቀን 1995 ዓ.ም. ለተጠራው አገር አቀፍ ጉባኤ የቀረበ:: 1995

[14] J.P. Dubus, M. Benjelloun, V. Devlaminck, F. Wauquier and P. Altmayer, “ Image Processing Techniques to Perform an Autonomous System to Translate Relief Braille into Black-Ink, Called: Lectobraille”, Proc. IEEE Engineering in Medicine and Biology Society 10th Annual Int. Conf., New Orleans (1988) 1584–1585

[15] G. François and P. Calders, “The Reproduction of Braille Originals by Means of Optical Pattern Recognition”, Proc. 5th Int. Workshop on Computer Braille Production, Heverlee (1985) 119–122

[16] T.W. Hentzschel, “An Optical Braille Reading System”, M.Sc. Dissertation, Faculty of Technology, University of Manchester (1992)

[17] A. Antonacopoulos and D. Bridson, ”A Robust Braille Recognition System”, Pattern Recognition and Image Analysis (PRImA) group, Department of Computer Science University of Liverpool, Liverpool, L69 3BX, United Kingdom

[18] R.T. Ritchings, A. Antonacopoulos and D. Drakopoulos, “Analysis of scanned Braille document”, Document Analysis system, A. Dengel and A.L. Spitz (eds.), World scientific Publishing co, 1995.

[19] X.F. Hermida, A.C. Rodríguez and F.M. Rodríguez, "A Braille OCR for Blind People. “, Proceedings of ICSPAT-96. Boston (U.S.A.).October, 1996.

[20] N. Falcon, C.M. Travieso, J.B. Alonso and M.A. Ferrer, “Image Processing Techniques for Braille Writing Recongition”, Computer Aided Systems Theory-ERUROCAST, Lecture Notes in Computer Sciences, 379-385, 2005.

[21] Z. Tai, S. Cheng and P. Verma, “Braille Document Parameters Estimation for Optical Character Recongition”, ISVC '08 Proceedings of the 4th International Symposium on Advances in Visual Computing, Part II, 2008.

[22] A. Al-Saleh, A. El-Zaart and A. AlSalman, “Dot Detection of Optical Braille Images for Braille Cells Recognition”, Computers Helping People with Special Needs Lecture Notes in Computer Science, 821-826, 2008.

[23] T. Agui, and T. Nagao, “Computer Image Processing and Recognition”, Tokyo: Shoho-do, 1994.

[24] M. Basu, “Gaussian derivative model for edge enhancement”, Pattern Recognition 27 (11), 1451-1461, 1994.

[25] J. Bigun, Tl. Bigun and K. Nilsson, “Recognition by symmetry derivatives and the generalized structure tensor”, IEEE TPAMI 26 (2), 1590-1605, 2004.

[26] Y. Assabie and J. Bigun, “Offline Handwritten Amharic Word Recognition”, Pattern Recognition Letters, 32 (2011), 1089-1099, 2011.

[27] L. Premaratne, Y. Assabie and J. Bigun, “Recognition of Modification-based Scripts Using Direction Tensors”, 4th Indian Conference on Computer Vision, Graphics and Image Processing.

[28] J. Li, X. Yan and D. Zhang, “Optical Braille Recognition with Haar Wavelet Features and Support-Vector Machine”, International Conference on Computer, Mechatronics, Control and Electric Engineering (CMCE), 2010.

[29] S. Bhattacharyya, “A Brief Survey of Color Image Preprocessing and Segmentation Techniques”, Journal of Pattern Recognition Research 1 (120-129), 2011.

[30] M. Sezgin, “Survey over Image Thresholding Techniques and Quantitative Performance Evaluation”, Journal of Electronic Imaging 13(1), 146-165, 2004.

Appendix A. The First Version Amharic Braille

List of first version basic Amharic braille characters

ሀ	1:2:4	ኸ	2:3:4
ለ	1:4:5	ወ	2:3:4:5
ሐ	1:2:4:5	ዐ	1:6
መ	1:2:5	ዘ	1:4:6
ሠ	2:4:5	ዠ	1:3:4:6
ረ	1:3	የ	1:4:5:6
ሰ	1:2:3	ደ	1:5:6
ሸ	1:2:3:6	ጀ	1:2:4;6
ቀ	1:3:4	ገ	1:2:4:5:6
በ	1:3:4:5	ጠ	1:2:5:6
ተ	1:3:5	ጨ	1:2:3:5:6
ቸ	1:3:5:6	ጰ	2:4;5;6
ኀ	1:2:3:4	ጸ	2:4:5:6
ነ	1:2:3:4:5	ፀ	1:3:6
ኘ	1:2:3:4:5:6	ፈ	1:3:4:5:6
አ	1:2:3:5	ፐ	2:3:4:5:6
ከ	1:2:6		

List of vowels for first version Amharic Braille

Vowels	1:4	2:5	3:6	1:5	2:4	2:6	2:6
Variant	1st	2nd	3rd	4th	5th	6th	7th

List of first version extended Amharic braille characters

ቈ	1:2:3:4:6 2:3:4
ኰ	1:2:3:4:6 1:3:4
ጎ	1:2:3:4:6 1:2:3:4
ጐ	1:2:3:4:6 1:2:4:5:6

Appendix B. The Second Version Amharic Braille

List of second version basic Amharic braille characters

1st		7th		1st		7th	
ሀ	1:2:5	ሀ	1:3:4:6	ኸ	2:3:6	ኸ	5 2:3:6
ለ	4:5:6	ል	1:2:3	ወ	2:4:5:6	ው	2:4:6
ሐ	Not apply			ዐ	Not apply		
መ	1:3:4	ም	2:3	ዘ	1:3:5:6	ዝ	2:3:4:6
ሠ	2:3:4	ሥ	5	ዠ	3:5:6	ዥ	5:6
ረ	1:2:3:5	ር	1:2:5	የ	1:3:4:5:6	ይ	1:4:5:6
ሰ	Not apply			ደ	1:4:5	ድ	1:4:5:6
ሸ	1:4:6	ሽ	1:5:6	ጀ	2:4:5	ጅ	1:2:6
ቀ	1:2:3:4:5	ቅ	4:6	ገ	1:2:4:5	ግ	2:3:5:6
በ	1:2	ብ	4:5	ጠ	2:3:4:5:6	ጥ	1:2:3:5:6
ተ	2:3:4:5	ት	123456	ጨ	1:4	ጭ	3:6
ቸ	1:6	ች	2:5	ጰ	2:3:5	ጵ	3:4:5:6
ኀ	Not apply			ፀ	1:2:3:4:6	ፅ	3:4:5:6
ነ	1:3:4:5	ን	1:2:4:6	ጸ	Not apply		
ኘ	3:4:6	ኝ	2:6	ፈ	1:2:4	ፍ	5 1:2:4
አ	3	እ	3:4	ፐ			
ከ	1:3	ክ	3:5				

List of vowels for second version Amharic braille

Vowel	None	1:3:6	2:4	1	1:5	none	1:3:5
Variant	1st	2nd	3rd	4th	5th	6th	7th

Appendix C. The Third Version Amharic Braille

List of third version basic Amharic braille characters

6th		6th	
ሀ	1:2:5	¦	2:3:6
ል	1:2:3	¨<	2:4:5:6
Q	Not apply	ዕ	Not apply
U	1:3:4	´	1:3:5:6
Y	2:3:4	»	3:5:6
`	1:2:3:5	Ã	1:3:4:5:6
ስ	Not apply	É	1:4:5
ሽ	1:4:6	Ï	2:4:5
ቦ	1:2:3:4:5	Ó	1:2:4:5
w	1:2	Ø	2:3:4:5:6
ት	2:3:4:5	ß	1:4
ፈ	1:6	ጵ	2:3:5
ጎ	Not apply	ፅ	1:2:3:4:6
''	1:3:4:5	ጽ	Not apply
~	3:4:6	ፍ	1:2:4
□	*1:2:3:5:6	ý	1:2:3:4
¡	1:3		

**** Symbol indicates Braille code change made on the new version.***

List of vowels for third version Amharic braille

Vowel	2:6	1:3:6	2:4	1	1:5	Independent	1:3:5
Character Variant	1st	2nd	3rd	4th	5th	6th	7th

List of third version extended Amharic braille character

Amharic irregular character		
ቈ	1:3	2:4:5:6
ኰ	1:2:3:4:5	2:4:5
ጎ	1:2:5	2:4:5:6
ጐ	1:2:4:5	2:4:5:6

Appendix D. Sample table lookup of letters "ဗ" to "ၐ" with their 7 variants

Consonant	Vowel	Unicode value of
62	21	4608
62	51	4609
62	24	4610
62	40	4611
62	42	4612
62	0	4613
62	52	4614
70	21	4616
70	51	4617
70	24	4618
70	40	4619
70	42	4620
70	0	4621
70	52	4622
70	55	4623
61	21	4624
61	51	4625
61	24	4626
61	40	4627
61	42	4628
61	0	4629
61	52	4630
54	21	4632
54	51	4633
54	24	4634

54	40	4635
54	42	4636
54	0	4637
54	52	4638
54	55	4639
34	21	4640
34	51	4641
34	24	4642
34	40	4643
34	42	4644
34	0	4645
34	52	4646

Declaration

I, the undersigned, declare that this thesis is my original work and has not been presented for a degree in any other university, and that all sources of materials for the thesis have been duly acknowledged.

_____

Miftah Hassen Jemal

This thesis has been submitted for examination with my approval as an advisor.

_____

Dr. Yaregal Assabie

Addis Ababa, Ethiopia

June 2011