Addis Ababa University
College of Natural Sciences

Recognition of Double Sided Amharic Braille Documents

Hassen Seid Ali

A Thesis Submitted to the Department of Computer Science in Partial Fulfillment for the Degree of Master of Science in Computer Science

Addis Ababa, Ethiopia
November 2015

Addis Ababa University
College of Natural Sciences

Hassen Seid Ali

Advisor: *Yaregal Assabie (PhD)*

This is to certify that the thesis prepared by *Hassen Seid Ali*, titled *Recognition of Double Sided Amharic Braille Documents* and Submitted in partial fulfilment of the requirements for the Degree of Master of Science in Computer Science compiles with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by Examining Committee:

	Name	Signature	Date
Advisor:	*Yaregal Assabie (PhD)*	_____	_____
Examiner:	_____	_____	_____
Examiner:	_____	_____	_____

Abstract

Amharic language has large number of characters. As a result, Amharic Braille image recognition into print text is not an easy task. Amharic Braille cell formulation, encoding to a Braille code and translating the code to print text are different from Braille recognition systems of foreign languages' characters. Few researches have been conducted in recognition of Amharic Braille documents. However, recognition of double sided Amharic Braille documents, which needs segmentation and identification of recto and verso dots from the background, and separation of overlapping recto and verso dots, has not been conducted so far.

In this work, we propose a design for recognition of double sided Amharic Braille documents. The design has a preprocessing, segmentation, dot identification, page formulation, transformation, and recognition components. We used direction field tensor to preprocess and segment dots from the background. After segmentation, gradient field is used to identify a dot as recto or verso. Overlapping dots were further segmented and identified using Braille dot attributes (centroid, orientation, and area). The identified recto and verso dots are separated into two separate images or pages using the page formulation component. Then, we used Braille cell encoding algorithm in order to formulate identified recto or verso dots into a Braille code. Finally, the Braille code is translated to print text using Braille code translation algorithm. The designed algorithms encode and translate the dots starting from left-top corner of the first dot to the right downward over the page. In order to use the same Braille cell encoding and Braille code translation algorithms for both pages, dots on the recto page are mirrored about a vertical symmetric line. Moreover, we used rotation in reversing wrongly scanned documents automatically as long as the translation performance is less than some threshold value which notifies the system the page is wrongly scanned.

In order to test the proposed design, we developed a prototype using MATLAB and test performances of dot identification and translation of double sided Amharic Braille images to print texts. We achieved an average dot identification accuracy of 99.3% and average translation accuracy of 95.6%. This is remarkably motivating performance as it is the first achievement in recognition of double sided Amharic Braille documents.

Key Words: - *Braille Cell, Direction Field Tensor, Gradient Field, Recto Dot and Verso Dot.*

Dedicated to

My Father

and

My Mother

Acknowledgments

First and foremost, I would like to thank the almighty Allah, who gave me the determination, endurance and wisdom to bring this thesis to completion. Oh! Allah you are always paving my way.

Second, my gratitude goes to my advisor Dr. Yaregal Assabie for sharing his interesting ideas, experiences, and discussions with me throughout the course of writing this thesis.

I would like to thank Abebe Mola from Ethiopian Association for Blind Society for his technical support and helpful comments to effectively realize the study.

This thesis would not have been possible to write without the love and support of Sofi and my family. Lastly, I also thank my classmates and friends who shared me their ideas and helpful comments and suggestions. Thank you all!

Table of Contents

List of Tables

List of Figures

List of Algorithms

List of Acronyms

CIE	International Communication on Illumination
DFT	Discrete Fourier Transform
DPI	Dot Points per Inch
JPEG	Joint Photographic Experts Group
OCR	Optical Character Recognition
OBR	Optical Braille Recognition
RGB	Red Green Blue
TP	Translation Performance

Chapter One : Introduction

1.1 Background

According to a statistics reported by World Health Organization in 2012, there are 285 million visually impaired people in the world. Out of these people, 90% live in developing countries [1]. In Ethiopia, there are 1.2 million blind; 2.8 million with low vision people and 9 million children aged between 1-9 years are with active trachoma which leads to blindness [2].

Communication in written form is vital in daily life for many people. However, visually impaired people face problems to share their knowledge efficiently through written documents and communicate with most of the people in different areas such as Education, Office works, Research works and different publications. Visually impaired people use Braille as a writing convention, but this limits the communication among them only. Using new technologies, different researches have been undertaken for automatic recognition of Braille documents in different languages such as Amharic [3, 4, 5, 6], English [7], Arabic [8], Chinese [9] and Spanish [10].

Braille is a tactile format of written communication for people with low vision and blindness worldwide since its inception in 1829 [3] by Louis Braille. It is a system of writing that uses patterns of raised dots to inscribe characters on paper [8]. Therefore, it allows visually-impaired people to read and write using touch instead of vision. Also it is a way for blind people to participate in a literate culture.

The Braille system includes symbols using which, the blinds are able to review and study the written words. It provides a vehicle for literacy and gives a blind the ability to become familiar with spelling, punctuation, paragraphing, footnotes, bibliographies and other formatting considerations. Braille cell consists of 6 dots, 2 across and 3 down, which is considered as the basic unit for all Braille symbols. For easier identification, these dots are numbered downward as 1, 2 and 3 on the left, and 4, 5 and 6 on the right, as shown in Figure 1.1 [11]

.

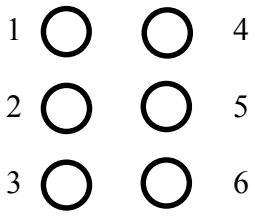


Figure 1.1: *Braille Cell*

As discussed in [12], a Braille document can either have dots embossed on one sides or on both sides of the document. The latter is also called inter-point Braille because the positions of the dots from one side lie between the positions of the dots on the other side. This is designed to reduce the bulk of the document and to save Braille paper. The dots on inter-point Braille are called recto and verso dots [11]. Recto dots are convex dots or protrusions directly sensed by fingers of visually impaired people while reading. Whereas, verso dots are concave dots or depressions which are sensed by fingers on the other side of a Braille document.

Modern education for visually impaired people in Ethiopia has started by the end of August 1924 [5]. Later, in 1934 Braille in Ethiopia was introduced by missionary and was designed by taking the 26 English characters pattern and 8 newly added patterns [13].

In Ethiopia, mainly in Addis Ababa, there are different education centers, such as AAU, Misrach Center, Entoto Blind people school, for visually impaired people at primary, secondary and tertiary levels. As a result, there have been massive educational Braille documents produced and used by visually impaired people. But there is a gap in communication through documents between vision and visually impaired people [3, 4, 5, 6].

Information in written form plays an undeniably important role in our daily life. From education and leisure to casual note taking and information exchange, recording and using information encoded in symbolic form is essential [14]. In order to address this need, the most widely adopted writing convention among visually impaired people is Braille.

Though the production of Braille documents is relatively easy now, the problem of converting Braille documents into a computer-readable form still exists. This is a significant problem for two main reasons [14] . First, there is a wealth of books and documents that only exist in Braille that are deteriorating and must be preserved (digitized). Second, there is an everyday need for duplicating (the equivalent of

photocopying) Braille documents and for translating Braille documents for use by non- Braille users. This minimizes the information gap between Braille users and non-Braille users.

Optical Braille Recognition (OBR) systems offer many benefits to Braille users and people who work with them. An OBR system is interesting due to the following reasons [15]:

- It is an excellent communication tool for sighted people with the blind writing.
- Braille writing is read using the finger which requires touching the document, for this reason the book after many readings possibly deteriorated. OBR systems enable to preserve such valuable documents.
- It is interesting to store a lot of documents of blind authors which were written in Braille and were never converted to digital information so that it can be used by the public at large.

Therefore, recognition of Braille documents automatically using OBR systems is an image processing work, thus it follows the following fundamental phases [11]: Image acquisition, preprocessing, segmentation, feature extraction and recognition.

1.2 Statement of the Problem

As Section 1.1 presents, in Ethiopia, there are 1.2 million visually impaired people. These individuals in their carriers and professions use Braille only to codify their knowledge as a written form. Since the introduction of Amharic Braille in 1934, there has been significant number of Braille documents that have been produced and found at different parts of the country: Addis Ababa University Kennedy Library, Misrach Center, Sebeta Visually Impaired School, German Church Visually Impaired School, and Ethiopian Association for Blind Society and Entoto Blind People School [3]. This knowledge, contributed by visually impaired people, is accessed only by those who can read and write Braille systems. According to Nebye Luel Yohannes, unless there is a smooth information flow from visually impaired people to sighted people and the reverse, people do not get the necessary information from visually impaired people [13]. As a result, the work of many visually impaired people would remain buried. Thus, it creates a wide gap between the visually impaired and sighted people.

Amharic language is the working language of the Federal Government of Ethiopia. The present writing system of the language is derived from Geez. The language consists of 34 core characters each

occurring in seven orders, representing syllable combination of a consonant and a vowel [16]. In Braille writing system, each syllable combination is uniquely represented by Braille code that uses one, two or three Braille cells. This makes recognition of Amharic Braille different from that of other languages such as English and Arabic Braille, which are represented by a Braille code that uses only one Braille cell. As a result, a number of attempts have been made to recognize Amharic Braille documents [3, 4, 5, 6]. However, they are not complete because none of them consider double sided Amharic Braille documents. In addition to the problems stated on one single sided Amharic Braille, recognition of double sided Amharic Braille needs the following:

- Identification of recto and verso dots.
- Recognition of Braille cells composed of recto and verso dots separately into the corresponding Amharic characters, numbers and punctuation marks.
- Separation of overlapped recto and verso dots.

Thus, research and development work on recognition of double sided Braille document is required to fully benefit from the state of the art of technology.

1.3 Objectives

General Objective

The general objective of this research work is to design a double sided Amharic Braille document recognition system and test its performance.

Specific Objectives

The specific objectives are:

- Perform an extensive review on previous research works focusing on design of Braille recognition systems for different languages with their own characters.
- Collect double sided Amharic Braille documents.
- Identify techniques or tools to segment dots from the background and identify the dots as recto and verso.
- Identify important features to identify a dot as recto or verso and design an algorithm.
- Identify important features to segment overlap of recto and verso dots and design an algorithm.
- Identify features and techniques to formulate a Braille cell so that the dots are encoded into a Braille code.
- Perform analysis on the nature of Amharic Braille cells and design a translation algorithm that can transcode Braille codes to Amharic print text.
- Develop a prototype so as to test performance of the designed algorithms.

1.4 Research Methodology

In order to conduct this research work, the methodologies mentioned below will be used to select and implement appropriate methods and techniques.

Literature Review

Different literatures like books, journals, proceedings, research papers, etc. will be reviewed to study different image acquisition, preprocessing, segmentation, feature extraction and recognition techniques. After that, appropriate tools for the recognition of Amharic Braille documents will be selected. Moreover, a review will be made in order to understand the domain knowledge: how Amharic characters, punctuation marks, numbers both Amharic and Arabic have been encoded in to a Braille code.

Data collection

Amharic Braille documents embossed in both sides of the Braille will be collected, and preprocessed for testing from different libraries found in different institutes.

Tools

MATLAB will be used to implement the selected image processing techniques or tools in each image processing phase and finally used to develop a prototype.

Prototype Development

The performance of the designed system will be tested using a prototype developed on the collected data. The result will be analyzed and evaluated from which a conclusion and further works will be recommended.

1.5 Application of Results

From this research work, both visually impaired people and sighted people will benefit by using it as a communication channel in the following ways:

- It facilitates access of knowledge or artifacts contributed by visually impaired people for sighted people without knowledge of reading Braille.
- It facilitates access of Braille documents in audio format by integrating with text recognition system for both visually impaired and sighted people.
- It facilitates communication between visually impaired and sighted people.

1.6 Scope and Limitations

This study focuses on recognition of Amharic characters, punctuation marks, Amharic and Arabic numerals embossed on both sides of a Braille. The Braille we considered is a fourth version Grade 1 Braille documents. This study is delimited in recognition of Amharic Lablization characters and keep the structure of Braille documents.

1.7 Organization of the Rest of the Thesis

The remaining part of the thesis is organized as follows. Chapter Two covers literature review. Chapter Three covers related work which shows efforts of different researchers in recognition of Braille

documents for different languages including Amharic using different methods. The results of each work are presented with their pros and cons. The Fourth Chapter broadly explains and discusses the design of double sided Amharic Braille document recognition: Braille image acquisition and preprocessing, dot segmentation, dot identification, page formulation, transformation and recognition into Amharic characters. Chapter Five presents performance of the designed Braille recognition system using a prototype. Finally, Chapter Six summarizes our findings and presents future works.

Chapter Two : Literature Review

In this Chapter points related with Braille system in general, Amharic Braille system in particular, Braille recognition systems using image processing techniques, the tools and techniques implemented in each phases of the recognition system are presented.

2.1 Introduction

The Braille, invented by Luis Braille in 1829 is a writing system for visually impaired people [3]. It contains set of embossed or raised dots called Braille cells to represent characters of different languages. A single Braille cell is represented by 6 dots arranged in 3 rows and 2 columns numbered from 1 to 6 as shown in Figure 2.1 (a) [11]. The embossed dots in the cell are transcoded to the corresponding character depending on the language for which it is written. The 6 dots totally give $2^6 =$ 64 different possible combinations of Braille cells. Thus, a single Braille cell, also called Braille character, can represent a single character in English and Arabic languages [7, 8]. However, in Amharic a single character can use one, two, or three Braille cells. This is to accommodate all the characters, numbers and punctuation marks of Amharic language [17].

Coming to the standard Braille cell size parameters, a dot has a height of 0.5 mm, vertical and horizontal spacing between cells is 2.5 and 2.3 mm respectively as shown in Figure 2.1 (b) [11]. Standard Braille page has a size of 28 x 29 cm with 25 lines, where each line can entertain 40 to 43 Braille cells [11]. A Braille document can be embossed not only on the single side but also on either of the sides to overcome space consumption by single sided Braille documents. On double sided Braille, the embossing process is done with slight diagonal offset to prevent recto and verso dots interference as shown in Figure 2.1 (c) and Figure 2.1 (d) [11].

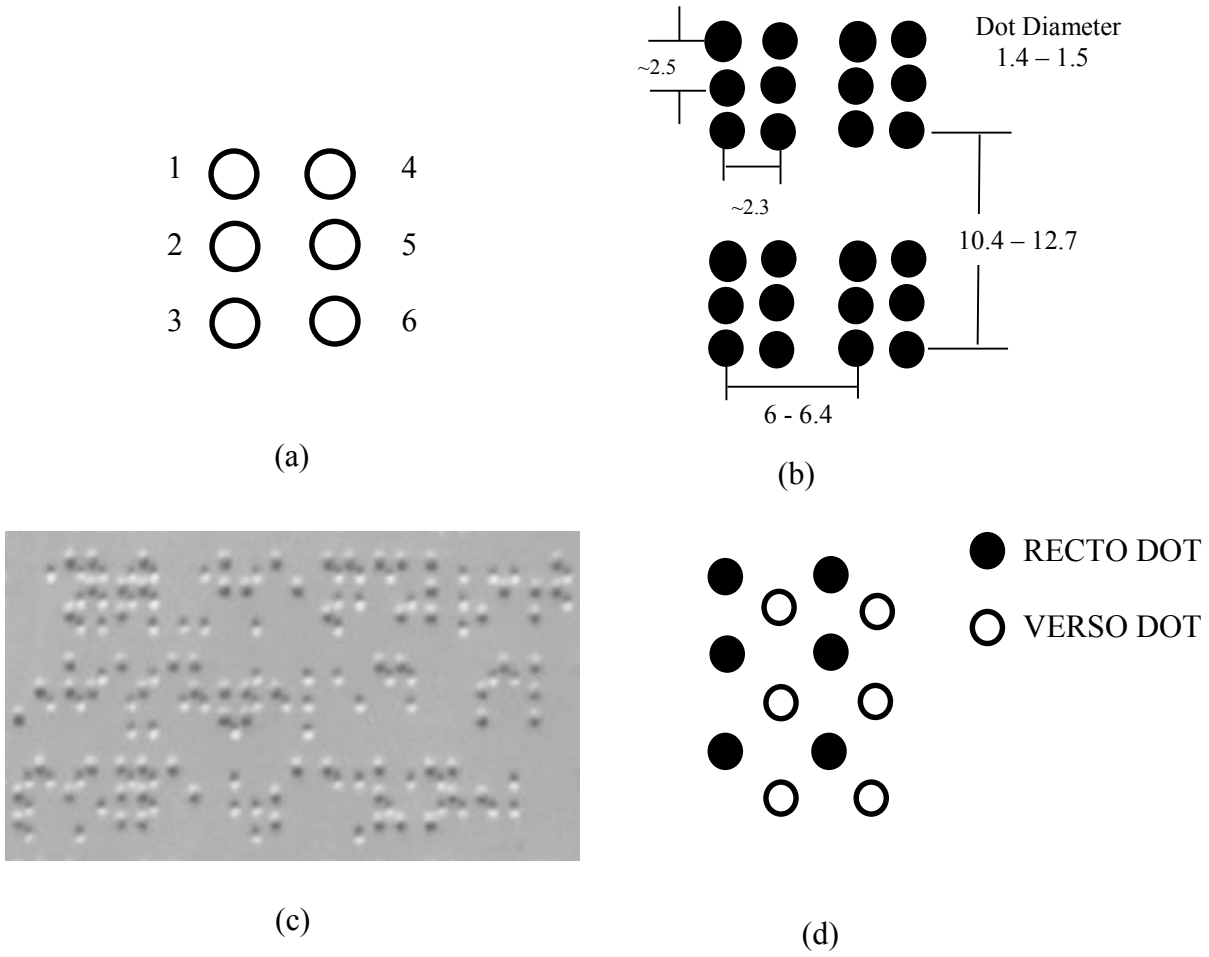


Figure 2.1: *(a) Braille Cell; (b) Braille Cell Dimension in Millimeters; (c) Double Sided Braille Image; (d) Inter Point Braille cell*

Even though most countries adopt and define their Braille code so as to fit their local language characters, Braille systems used in the world currently are categorized into two levels [8]: Grade 1 Braille and Grade 2 Braille.

Grade 1 Braille is a form of Braille in which print characters are represented by one Braille cell using a mode indicator character [8]. The mode indicator 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 or Braille cell, but with "shift" character as an indicator of others such as upper case, digits, and italics are represented [18].

A Grade 2 Braille is introduced as a result of the rigorous attempt to minimize the volume of Braille documents by contracting words so as to minimize the time required to read a Braille document as

compared to Grade 1 Braille document [8]. In Grade 2 Braille, context sensitive rules which are apparently language dependent and frequently used letter groups are used for the contraction of words. These rules determine the correspondence between one or more Braille cells and the print characters. For example, in standard English Braille, a Braille symbol may stand for 'dis' referring the word distance when it comes at the beginning of a word and 'dd' referring the word ladder when it comes at the middle of a word [18].

2.2 Amharic Braille System

Amharic language is the working language of the Federal Government of Ethiopia. The present writing system of the language is derived from Geez [16]. The language consists of 33 core characters and one additional character '፝'. Each occurs in seven orders, representing syllable combination of a consonant and a vowel (see Annex A). In addition to the 238 characters, Amharic language contains 44 labialization characters (see Annex B).

Amharic writing system uses both Ethiopic and Hindu Arabic numerals to represent numbers (see Annex C). The Ethiopic numeral doesn't have symbols to represent a zero, negative numbers, decimal points, and mathematical operators to perform mathematical operations. The writing system of Amharic also uses punctuation marks (see Annex D).

In Ethiopia, Amharic Braille system was first introduced in 1923 by American Missionary Schools to deliver education for visually impaired people. The Braille system now reaches its fourth version by incorporating different improvements for different reasons [19].

The first version was comprehensive and complete even though it did not work for Ge'ez characters [13]. This version was revised for the first time in 1952 for the following two amendments [19]:

- To eliminate the redundant Amharic characters like (ሀ፡ሐ፡ኀ), (አ፡ዐ) and (ሰ፡ሠ)
- To replace Amharic numerals with Arabic numerals.

In this version (Second Version), all forms of Amharic characters except the first (Ge'ez) and the sixth (Sadis) forms were represented using vowels. Four years later in 1956 a third version Amharic Braille system was introduced by including vowel for the first characters. However, it was finally revised in 2001 to give the fourth version Amharic Braille. Despite the introduction of the fourth version, the third version is still widely used by the blind community in Ethiopia.

The major issues of the Braille improvement committee established for the fourth version of Amharic Braille documents were 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 [19]. The committee decided the following [19]:

- The number of Braille codes was decided to be equal to the print characters.
- Modification of the punctuation marks was done mostly borrowed from the English language.
- Formation of Grade 2 Braille was deferred until Grade 1 Braille gets legal recognition.
- Yared musical symbols were decided to be discussed with professionals.
- It was also decided to use either of the American or the British system for Mathematical symbols.

Finally, the fourth version Amharic Braille was approved and distributed for use by different institutions that work with visually impaired people [19].

A single Braille cell represents a single character in English and Arabic languages that uses variants of characters not more than 2^6=64 characters. However, a single Amharic character needs one, two or three Braille cells as shown in Table 2.1 and Table 2.2 for consonants and vowels respectively [17]. For instance, the character '𝔘' is syllable combination of the consonant 'ሀ' and the vowel 'ኧ'. It is represented by two Braille cells combination '1:2:5' and '2:6' as shown in Figure 2.2, which are Braille codes of 'ሀ' and 'ኧ' respectively.

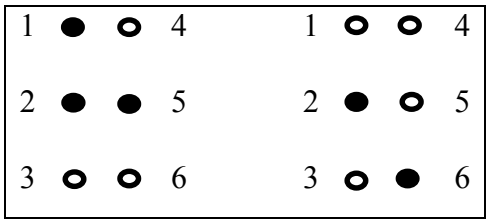


Figure 2.2: *Braille Code for Amharic Character '𝔘'*

Table 2.1: *Fourth Version Amharic Consonants Braille Code Representation*

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

Table 2.2: *Fourth Version Amharic Vowels Braille Code Representation*

Variants	Ge'ez	Ka'eb	Salis	Rab'e	Hamis	Sadis	Sab'e
Vowels	ኧ	ኡ	ኢ	ኣ	ኤ	እ	ኦ
Braille Code	2:6	1:3:6	2:4	1	1:5	-	1:3:5

Similar to the Amharic characters, in the fourth version Amharic Braille punctuation marks use one, two or three Braille cells as shown in Table 2.3. However, the Ge'ez and Arabic numerals are represented by the same Braille code as shown in Table 2.4. In order to differentiate which number system is represented by a Braille code, a mode indicator is used. The mode indicator is a Braille cell that tells Braille readers to recognize the next Braille cell is a Ge'ez or an Arabic number [14]. The mode indicators are represented by Braille codes of 3:4:5:6 and 1:2:3:4:5:6 for Arabic and Ge'ez number systems respectively [17].

Table 2.3: *Amharic Punctuations and Symbols Braille Code Representation*

Punctuations	Braille Code	Punctuations	Braille Code
.	3	።	2:5:6
፣	2	!	2:3:5
:-	2:5	…	3 , 3 and 3
?	2:3:6	()	2:3:5:6
፤	2:3	እና ወይም	3:4
/	5 and 2	-	3:6
'	4 and 1	"	2:3:6
*	3:5 and 3:5	$	4:5

Table 2.4: *Amharic and Arabic Numerals Braille Code Representation*

Ge'ez No	Arabic No	Braille Code
⠁	1	1
⠃	2	1:2
⠉	3	1:4
⠙	4	1:4:5
⠑	5	1:5
⠋	6	1:2:4
⠛	7	1:2:4:5
⠓	8	1:2:5
⠊	9	2:4
⠚	0	2:4:5

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 ways to write on a Braille paper that result in a tactile output [20]. Devices 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 [21].

A Braille embosser is also another device used for producing Braille documents. As in print documents, Braille character embossing or writing starts from the top of a document left to right down to the bottom of the Braille paper [17]. All dots on a Braille page should fall on an orthogonal grid. This is very important, when Braille characters are printed double sided as the grid of the inter-point text is shifted so that the dots fall in between the primary side dots [4].

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. Braille reading starts from the top of a document right to left down to the bottom of the Braille paper [17]. Due to this nature, 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 reading can typically read at a rate of 125 to 200 words per minute. On average,

eighth grade students can read 205 words per minute, and college students read 280 words per minute according to University of Buffalo [17].

2.3 Braille Recognition System

A Braille image recognition system consists of image acquisition and pre-processing, segmentation, feature extraction and interpretation [9]. Braille documents are of two types; single sided and double sided Braille [8]. In single sided Braille document embossing is created on one side, thus the recognition is easier as compared to double sided Braille document since in the latter case the embossing is done on both sides of the document with a small diagonal offset [8].

Image acquisition is the first step in any pattern recognition system. In Braille recognition systems, data is provided to the system in the form of images of Braille embossed pages. The process of acquiring these images digitally can be achieved by using a number of different equipment such as scanners or digital cameras, both of which have been used by developers and researchers [22].

Image preprocessing is an essential step to detect and eliminate noise, deformation, bad illumination or blurring. Image pre-processing can be used for image enhancement by reducing noise, sharpening images, or rotating a skewed page [9]. The preprocessing step has to deal with also binarization. Since analysis of a binary image is much simpler than that of gray-scale images [9].

In image segmentation stage, the regions in the image corresponding to Braille dots are identified. Each dot in a scanned Braille image is composed of light and dark areas separated by background. In a gray-level image a dot is represented by a combination of a dark and a light region. Braille dots manifest themselves in the image as white and black region pairs [8].

Feature extraction is a representational mechanism of the Braille image. The function of feature extraction is to extract the Braille dots from the image and group them into cells. Extracting features from sub images that come from segmentation stage helps to simplify the recognition process.

Interpretation converts the Braille cells into their corresponding language text. At this stage, an orthogonal, binary image without noise is obtained [9]. The work of this phase is to group the Braille dots into cells and converts them into their equivalent characters.

2.4 Image Acquisition and Preprocessing

Image acquisition is the first step in Braille recognition process. Devices like digital camera and flatbed scanner are means of acquiring a Braille document in a digital form. Attempts in [23, 24], predominantly older ones, have been to recognize Braille by capturing the Braille image using camera. In image acquisition using camera, the Braille document is illuminated under an oblique angle so that it reveals shadows from the Braille dots. However, it introduces some complexities such as, aberrations, irregular lightness, and relatively low resolution among dots [25]. This brings computationally expensive procedures to overcome these problems [12]. Hence, recent works [8, 25, 12] decided for a commercially available flatbed scanner to obtain Braille images digitally. They used flatbed scanner because it is quick, easy and cost effective. Researches use scanner for different scanner resolutions. However, a very high above 200dpi is not recommended as it may add too much detail into the image [26], which may tarnish the image and thus will have adverse effects on subsequent procedures.

Preprocessing is another important activity in the Braille recognition process. It refers to the process by which error correction is made that is crucial for the smoothing functions as a subsequent procedure. 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 [3, 8]. Some of the preprocessing tools used by researchers are as follows:

Median filter

Median Filter is a simple and non-linear filter which works based on order statistics. Median filtering is very widely used in digital image processing because it preserves edges while removing impulse noise that are caused by the amount of intensity variation between one pixel and the other pixel. Median filter gives better result in the presence of impulse noise on binary images [11].

Gaussian Filters and Their Derivatives

Efforts in [3, 9] applied low pass spatial Gaussian filter to attenuate the high spatial frequency noise from the image, and preserves the detailed edge information of the Braille dots using derivatives of the Gaussian filters. These preprocessing tools are selected due to the following properties [27] :

1. Directional isotropy: That is to say, in polar coordinates they depend on radius only.
2. Separable along x and y coordinates as g(x) and g(y) as shown in Equation 2 for performance optimization during convolution operation.
3. Simultaneous concentration in the spatial and the frequency domain.

The two dimensional Gaussian Filter g(x, y) is given by Equation 1.

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

where δ is the standard deviation.

The 2D Gaussian is more efficiently computed from convolution of two conscutive1D Gaussian filters, g(x) and g(y) shown in Equation 2.

$$g(x) = \frac{1}{\sqrt{2\pi}\delta} exp^{-(\frac{x^2)}{2\delta^2})} \quad and \quad g(y) = \frac{1}{\sqrt{2\pi}\delta} e^{-(\frac{y^2)}{2\delta^2})} \quad (2)$$

During the scanning process, Braille pages may not be aligned correctly. As a result, several researches dealt with correction of image tilting. Mennens *et al*. [26] used Discreet Fourier Transform to calculate the rotation angle in an attempt to de-skew tilted Braille pages. As stated in [25] a Hough transform[1] corrects tilted images by identifying the precise orientation of the rows, and rotate them accordingly.

2.5 Braille Dot Segmentation

Braille dot segmentation, which refers to separation of dots from the background, is vital in any Braille recognition system. As stated in [3], direction field tensor segmenting dots from the background better than gradient field. Direction field tensor computes the differences in the intensity of pixels by giving attention to linear structures [27]. However, using gradient field, it is not possible to analyze whether a pixel fits to a line structure or not [28].

Gradient field

It is a tool used for estimation of local direction of pixels in an image neighborhood by computing the change or gradient of intensity variation of the neighborhood pixels both in magnitude and direction [28]. However, during the computation of the gradient field, it is not possible to analyze whether a pixel fits to a line structure or not. Thus, the results obtained from the gradient field are not optimal solutions

[1] Hough transform is an algorithm used to detect lines in an image. Available at http://www.mathworks.com/help/images/ref/hough.html

to extract linear patterns over images. In addition, a linear structure will have two groups of opposite directions in the gradient field, averaging them over a window gives large deviations due to cancellation effects [28]. As a result, gradient field uses single angle representation 0-360^o to show the direction of the linear structures. From the International Communication on Illumination (CIE) color laboratory [29], yellow color and its derivative represent angle values ranging between 0 and 180^o, and blue color and its derivatives represent angle values ranging between 180^o and 360^o;

The gradient field of a pixel value f over an image at positions x and y, for local neighborhood f(x, y) is computed by using Gaussian derivative operators D_x and D_y as shown in Equation 3 [28].

$$\nabla f = \iint ((D_x + iD_y)f)dxdy \tag{3}$$

The integrals are implemented as convolutions with a Gaussian kernel. The complex partial derivative operator $D_x + iD_y$ is defined as shown in Equation 4.

$$D_x + iD_y = \frac{\partial}{\partial x} + i\frac{\partial}{\partial y} \tag{4}$$

Direction Field Tensor

Unlike a gradient field, direction field computes the differences in the intensity of pixels by giving attention to linear structures [27]. This suppresses noises in a noisy image. This is performed by computing the direction field tensor (S) by pixel wise complex squaring. This avoids the cancellation effect seen on gradient field. Direction field tensor uses single angle representation (0-180^o) to represent the direction of the direction field tensor. For a local neighborhood f(x, y) of an image f, the direction field tensor, also called the structure tensor S, is computed as a 2 x 2 symmetric matrix using derivative operators Dx and Dy as shown in Equation 5 [27, 28].

$$S = \begin{pmatrix} \iint (D_x f)^2 dxdy & \iint (D_x f)(D_y f)dxdy \\ \iint (D_x f)(D_y f)d\,xdy & \iint (D_y f)^2 dxdy \end{pmatrix} \tag{5}$$

Edges are places over an image where linear symmetry exists. This is because there is a gray intensity change in such areas [28]. Existence of linear symmetry in such areas can be estimated by eigenvalue analysis of the direction field tensor or equivalently by using complex moments of order (I_{mn}) given in

Equation 6 [28]. Among the complex moments two of them are I_{11} (magnitude of the direction field tensor) and I_{20} (direction of direction field tensor) computed as shown in Equations 7 and 8 respectively [27, 28].

$$I_{mn} = \iint ((D_x + iD_y)f)^m ((D_x - iD_y)f)^n dxdy \quad (6)$$

$$I_{11} = \iint |(D_x + iD_y)f|^2 dxdy \quad (7)$$

$$I_{20} = \iint \left((D_x + iD_y)f\right)^2 dxdy \quad (8)$$

2.6 Braille Cell Formulation

Once the Braille dots are segmented from the background, the dots are formulated in to a cell. A cell is a region that contains the dots, from which a Braille code is derived as discussed in Section 1.1. Different researchers used different methods to formulate Braille cells.

Among the methods half character[2] detection is used to determine the location of the dot, this is selected because for English characters 16 of them don't use the two columns of a Braille cell and dealing with single column is easier than two columns [7]. The half characters are transcoded into a Braille codes using probabilistic neural network [7]. However, instead of applying a model like neural network, which makes predictions, a simple analysis of the dot gives better result [8]. This is mainly because the meaning of one Braille cell code can easily be converted to the corresponding print character by checking presence or absence of dots in a Braille dot position [8].

Mennens [22] used a method that constructs horizontal and vertical grid lines. The intersections of these lines are probable positions for Braille dots. The Braille cells are then formulated out of the constructed grid. This technique, however, does not tolerate a slight deviation of the grid from the dot position because the grid is fixed. Antonacopoulos and Bridson [25] used the same technique, but the grid is designed in a relatively flexible manner. This does not require alignment of Braille dots with the grid, because they use two frequency histograms that show distribution of the horizontal and vertical lines of the grid. Each histogram has two pick values, for instance, the pick values of a histogram for the horizontal lines shows vertical distance between dots in the same Braille cell (inter distance) and the

[2] Half Character refers detection or recognition of a single column (only right or left column) of a Braille cell.

vertical distance between Braille cells (intra distance). However, this has its own problem when the Braille document has large skew which can be corrected using Hough transform [25].

A window of size equals to the Braille cell is used to formulate a Braille cell [8, 12]. The window has six compartments arranged in three rows and two columns. The presence of a dot is checked in the compartments. A binary value of 1 (presence of a dot) or 0 (absence of a dot) is set for each compartments [8]. Finally, the binary string is formulated for each Braille cell.

2.7 Braille Cell Translation

Braille cell translation refers to the step in which the Braille codes are analyzed and translated into a print character. Most of the researches tend to use a Braille dictionary or lookups to do the translation into text after they examine the dot positions of the Braille cells.

According to [12, 30], the binary string formulated during Braille cell formulation is translated into a print text using appropriate look up tables. In a similar manner, the authors in [9] checked the dot position of a Braille cell against a Braille dictionary or lookup tables to translate into the appropriate character, which is grouped into words. Researchers in [8] use a binary string that shows presence or absence of a dot, but the binary string that represents a single Braille cell is further converted into a decimal code using Equation 9. Finally, the decimal code is further converted in to the corresponding character using lookup tables [8].

$$\boldsymbol{Decimal\ Code = b_1 + b_2 * 2^1 + b_3 * 2^2 + b_4 * 2^3 + b_5 * 2^4 + b_6 * 2^5} \qquad (9)$$

where b_i = binary value of the ith dot

The technique mentioned above works only for translation of Grade1 Braille documents. In Grade 2 Braille documents, the correspondence between Braille cells and print characters is not one to one. As a result Blenkhorn [18] uses the finite state system to hold the current context. This is because in Grade 2 Braille, different Braille codes have different meanings depending on the context.

2.8 Summary

Braille is a writing system for visually impaired people. It is used to represent characters of different languages. Depending on the number of characters used by different languages, a single character may use one, two or three Braille cells. For instance, most of the characters (syllable combination of vowel and consonant) in Amharic language used two Braille cells. This makes Amharic Braille system

different from Braille systems of other languages. Thus, Amharic Braille system needs its own recognition system.

A Braille recognition system is an automated system that translates Braille documents into print text. A Braille recognition system is an image processing work which passes through the following phases: Braille image acquisition and preprocessing, Braille dot segmentation, Braille cell formulation and Braille cell translation. In each phase, different researchers use different tools and techniques in conducting their research work.

We have seen from the literature that image acquisition with resolution power more than 200 dpi is not recommended. Segmentation of dots from the background is best using direction field tensor because it removes noise by considering linear structure and Braille cell formulation using sliding window is better than any other technique in order to optimize the recognition performance.

Chapter Three : Related Work

3.1 Introduction

Researchers have been trying to design and develop Braille document recognizer which is called OBR system for different languages. This is due to its importance for visually impaired people to communicate with sighted people or with each other. A Braille document recognition is made using different tools and techniques for different purpose. Researchers select different tools and techniques depending on the problem they attempted, because Braille documents vary from single to double sided, and from clean to noisy documents.

Different researchers have been investigating a design for Amharic Braille document recognition since 2009. From that onwards, the researchers designed an OBR system that can recognize Amharic Braille documents. Their attempt differs in the type of Braille document they select (clean and noisy single sided Braille documents) and the methods used to solve problems. The common problems of all researchers is that they are unable to address recognition of double sided Amharic Braille documents. In this Chapter, we presented the different approaches, tools and techniques used by the researchers. In addition, we covered recognition of double sided Braille documents conducted by researchers for Arabic, English and Bangla characters. Finally, we tried to summarize the gap of each work in a summary section.

3.2 Braille Recognition Systems for Foreign Languages

Braille recognition systems presented in this section are designed using image processing techniques. Ultimately, the techniques are applied in order to differentiate recto and verso dots based on the features observed on the original Braille documents. On the original image, there are three classes of useful information (shadows, light area, and background) [8, 25]. The order of appearance of these features from top to bottom is used to differentiate the dots as recto and verso [8]. The following works briefly explain how these features are used to identify the dots.

Arabic Braille Recognition

A research conducted by Abdul Malik, *et al,* [8] on double sided Arabic Braille recognition. In this work, Braille image is scanned using flatbed scanner. The scanned image is then converted to a gray scale image in order to decrease its complexity which can be colored. In order to correct image tilting,

a binary search algorithm is designed, which tolerates a maximum tilting angle of 4 degrees both on the left and right directions. An image thresholding algorithm has been developed to examine each pixel value in the image so as to classify a pixel into one of the three classes (dark, light and background) based on variation of intensity level.

The thresholding algorithm has two threshold values, low(L) and high (H), calculated based on the following Equations 10 - 13 after removing edge effect of the gray scaled Braille image , I.

$$a = \frac{1}{2}(mean(\max(I)) + avg) \quad (10)$$

$$b = \frac{1}{2}(mean(\min(I)) + avg) \quad (11)$$

$$L = avg - \frac{1}{3}(avg - \min\_avg) \quad (12)$$

$$H = avg + \frac{1}{3}(\max\_avg - avg) \quad (13)$$

where max (I) and min (I) represent the highest and lowest values respectively in each column of the array I, avg represents average gray level for the whole image and min_avg represents the average of all values less than b and max_avg represents the average of all values higher than a.

Using these threshold values, dot part detection algorithm is used to classify a dot as recto (contains bright region at the top and dark at the bottom) or verso (contains dark region at the top and bright at the bottom). After identifying recto and verso dots, the researchers designed a whole dot detection algorithm that represent dots (recto or verso) into dots of size 4x4 at their position based on the features extracted; dark followed by bright as recto and bright followed by dark as verso from top to bottom. Finally, they designed an algorithm to recognize a Braille cell into its corresponding Arabic character. The system is tested for a variety of A4 scanned Arabic Braille documents and achieved an accuracy of 99 % in detecting and identifying a dot as recto or verso.

English Braille Recognition

An effort in [31], focuses on segmentation of recto and verso dots after converting scanned Braille image into a gray scale image I. They designed a thresholding algorithm with single threshold value V computed by Equation 14.

$$V = \max(I) - 10 \tag{14}$$

Where max (I) is the maximum pixel intensity value of the gray scale image.

The gray scaled Braille image is converted into a binary image using this threshold value. Pixels of the image whose intensity value is less than the threshold value (V) and greater than the threshold value are assigned as 0 (Background) and 1 (dots). Median filter is applied to remove impulse noises created during thresholding. From the binarized image, the area and position of each dot (Recto and Verso) had been determined, then the centroids of the dots are marked on the original gray scale image. A mask of size 13x10 is designed and placed at the centroid of each dot on the gray scale image. The mask is used to count the number of pixels with intensity value of less than 200. They have found recto dots have less than 20 pixels and verso dots have more than 20 pixels with intensity value less than 200. However, the experiment is tested on a controlled environment. The Braille document used contains recto and verso dots in separate region as shown in Figure 3.1 [31]. Moreover, their work doesn't include Braille cell formulation and Translation of the Braille code into print text.

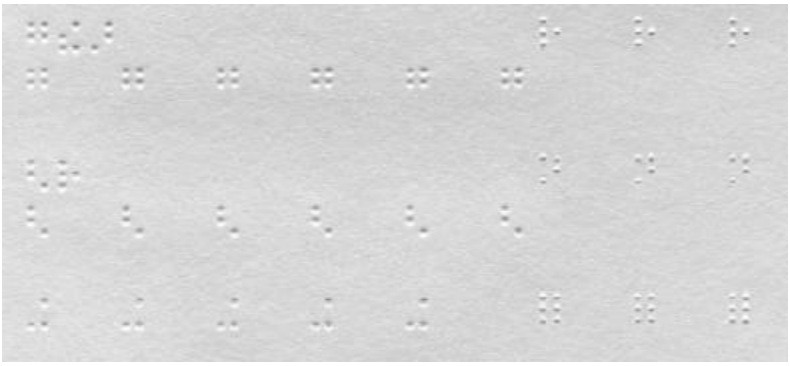


Figure 3.1: *Controlled Double Sided Braille Image*

A research work conducted in [25] uses a thresholding algorithm to extract features of a dot. The algorithm is applied in order to categorize pixel values over the image into three categories only. Black region referring the shadow, white region referring the highlights and mid gray referring the background.

After segmenting the dots from the background, the researchers look for pair of white and dark region and the order of their appearance. According to their work, if white region comes next to dark region, the dot is categorized as recto dot. However, if the reverse exists the dot is categorized as verso dot. After knowing the dots, a grid is formulated for each set of the dots based on inter character distance, inter line distance, character height and character width. The intersection points of the grid tells the location of the dots.

In each of the two grid, dot positions are examined for each Braille character or Braille cell. If a dot exists, a bit (1) corresponding to its existence is switched on. Otherwise, a bit (0) is used. The bit values of a Braille character is converted to the corresponding English character. For instance, in this work, the character 'A' is represented with bit value of (000001) in six bits Braille character or (00000001) in eight bit Braille character.

This work has a character validation component which enhances the recognition performance. This component corrects characters by analyzing the whole translated print text at word level. The whole translated text is split into words. Existence of each word is looked up in the dictionary. Words that exist in the dictionary are left unchanged. However, words that do not exist are corrected by removing or introducing one or more dots into the Braille character until the appropriate word is found.

The performance of this work is tested for variety of documents. They achieved dot identification performance of 99% and translation performance of 98.7% for good quality documents.

Bangla Braille Recognition

According to a research conducted by Santanu *et al*. [32], Bangla language has letters (consonants and vowel), punctuations, and numbers. Some of the Bangla characters use two Braille cells (Braille characters). They adopt boundary detection technique of [9] in segmenting dots (recto and verso) from the background. They used fixed windows in formulating Braille cells. The use of fixed size window does not produce an anomaly because it is verified that the distance of inter-dot, inter-cell and line spacing are fixed as discussed in Section 2.1. They defined three windows. A window that contains Braille characters, cell window, and a window that fills space between Braille characters, inter cell window, and a window that fills the gap between character lines, line window. In the decoding phase, they designed an algorithm that decodes the Braille characters or cells into the corresponding binary code. Finally, the binary code is translated into the corresponding Bangla character.

3.3 Amharic Braille Recognition Systems

There were few attempts in recognition of Braille image for Amharic language. We discussed the different researches conducted so far in recognition of Amharic Braille documents using different approaches. The methodology, tools and results of their efforts are explained as follows.

As a pioneer research, Teshome Alemu [5] designed an OBR system. The researcher used a flatbed scanner whose resolution is adjusted to 200dpi to acquire Braille documents as an image. The work focused on recognition of single and clean Braille documents. The author applied global thresholding for binarization. That is to segment the Braille dots from the background, after segmenting Braille dots the author used mesh-grid technique for segmentation and feature extraction of Braille cells, and finally the author applied probabilistic neural network classification technique for recognition of the Braille cell into Amharic characters. The author trained the system with 267 Amharic Braille characters and tested it on features extracted from clean Braille documents only and attained a performance of 92.5 % accuracy.

As a continuation of the above work, Ebrahim Chekol [4] further explored the possibility of designing Amharic Braille recognizer that works with noisy Braille documents. The attempt focused only on the preprocessing phase of Braille image recognition. The author used Gaussian filters and morphological operators to remove noise. Finally, the author adopted Teshome Alemu's effort for segmentation feature extraction and recognition of the Braille documents into print text. The author tested the system with different level real life Braille documents and achieved a performance of 95.5 %, 95.5 %, 90 %, and 65 % for clean, small-level, medium-level, and high-level noise Braille documents respectively.

Miftah Hassen and Yaregal Assabie [3] explored recognition of Amharic Braille documents using direction field tensor in order to enhance performance of the recognition. The researchers used the low and high pass spatial Gaussian filters and the derivatives of the Gaussian filter combined with direction Field tensor for noise removal and edge detection respectively. The authors used skewness correction to prevent image tilting which has a great impact in the recognition process. The authors designed a

new algorithm using a global slope[3] and local slope[4] derived from global slope. This algorithm can tolerate a maximum tilting angle of 5 degrees.

Over the Gaussian filters and their derivatives the author applied direction field tensor for segmentation of dots from the background. Direction field tensor is selected over a gradient field. Because direction field removes noise by amplifying linear structures (dots) and suppressing nonlinear structures (noises). Normalization of intensity variation across the Braille document and thresholding combined with direction field tensor segment dots from the background.

Combination of half character recognition followed by half character detection is used in order to formulate and encode Braille cells into Braille codes. The half character detection algorithm works on one column of a Braille cell than the two columns. A half character detection algorithm has a single value for a single column as shown in Table 3.1 [3].

The values are recognized based on the total height of a column. However, there are different values with same height. For instance, columns with single dot (values of 1, 2 and 4), columns with values 3 and 6, and columns with values 5 and 7 have the same height. Thus, further analysis is performed to find dot position. Moreover, analysis on a distance between columns is performed before combining the half character values in order to formulate a Braille cell and encode into its corresponding Braille code.

Finally, a translation algorithm is designed to translate the Braille codes mapped to Unicode values of Amharic characters into print text using lookup tables. The performance of their work was tested for different quality levels of Braille documents and achieved an average accuracy value of 98.5%.

[3] Global slope is average of local slope.

[4] Local slope is slope of each Braille cell line calculated using the position (x, y) of the first and last dots found on the same line.

Table 3.1: *Half Character Values*

Half Character	Value
	1
	2
	3
	4
	5
	6
	7

From the result of the Braille cell formulation algorithms (half character detection and half character recognition) used in their attempt as shown in Figure 3.2 [3]. We inferred their translation algorithm translates from left to right. But, this contradicts with the Braille reading direction by Braille readers which is right to left down ward as discussed in Section 2.2 unless a transformation algorithm is implemented to turn the page or the Braille document should be scanned in its verso side as we have discussed in Section 4.7. In addition, as discussed in Section 2.6, half character detection algorithm is more effective for English language with 16 characters which don't use both columns of a Braille cell.

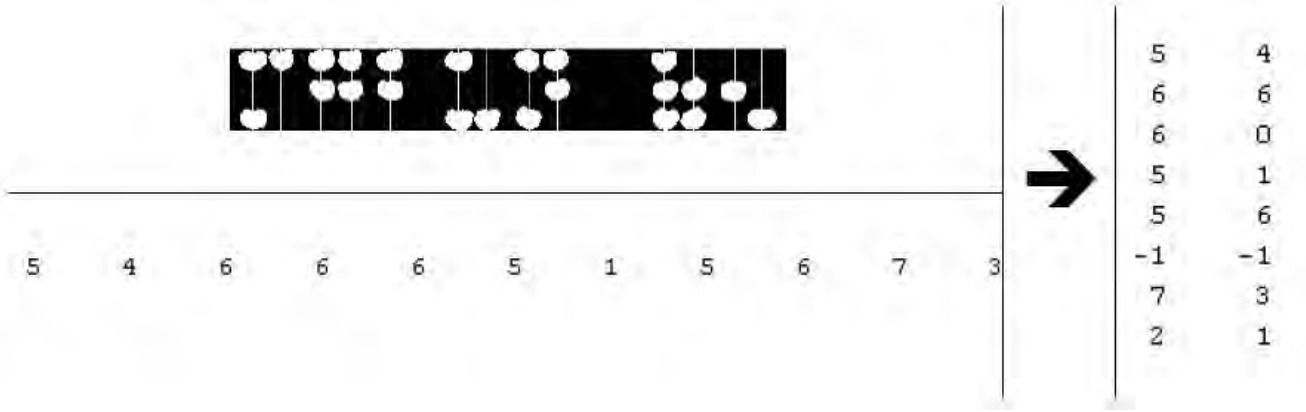


Figure 3.2: *Braille Cell Formulation*

3.4 Summary

Research works conducted in recognition of Amharic Braille documents focused only on single sided Braille documents. Recognition of double sided Braille documents designed for other languages cannot be applied directly to Amharic characters because the techniques and algorithms used to encode a Braille cell to a Braille code and translation of the code to a print text differs. From the related works, we observed that Amharic language has large number of characters; most of the characters use two Braille cells unlike that of English and Arabic characters which use one Braille cell only. Thus, encoding and translation process needs further work. That includes; how each character is represented using Braille code, and how the Braille codes should be translated into its equivalent print text. Moreover, none of them considered segmentation of overlapping dots into recto and verso dots. In this work, we conduct a research work that tries to fill the gaps mentioned by adopting the segmentation technique used in recognition of single sided Amharic Braille documents using direction field tensor.

Chapter Four : Designing Amharic Braille Recognition System

4.1 Introduction

As described in different research works, recognition of Braille images into the corresponding text using digital image processing passes through different phases: Braille image acquisition, preprocessing, segmentation, and recognition. In this Chapter, we propose a design for recognition of double sided Amharic Braille documents. The sections in this chapter present details of components of the proposed architecture. Section 4.2 presents the general overview of the proposed system architecture. Section 4.3 presents Braille image acquisition and preprocessing techniques. Section 4.4 explains segmentation of dots from the background. Section 4.5 briefly explains and discusses the tools and techniques used to identify recto and verso dots. Section 4.6 explains separation of recto and verso dots as two different images or pages. Section 4.7 explains the need for concept of transformation in double sided Braille image recognition. Section 4.8 explains the techniques applied in formulating Braille dots into the corresponding Braille code. After the dots are formulated to a Braille code, a design for the translation process is presented. Finally, a summary for the chapter is briefly presented in the last section.

4.2 The Proposed Amharic Braille Recognition System Architecture

The system architecture shown in Figure 4.1 is the proposed architecture for recognition of double sided Amharic Braille documents. It consists of six components working together: Braille image acquisition and preprocessing, Braille dot segmentation, Braille dot identification, page formulation, transformation and recognition. The shaded components on the architecture are adopted from an attempt of Miftah Hassen and Yaregal Assabie [3] with changes in algorithms used in the recognition component.

The first component acquires a Braille image using a scanner at 200dpi, and preprocesses the acquired image using Gaussian filters. The second component segments Braille dots from the background using direction field tensor. The third component identifies the segmented dots (isolated and overlapping dots) as recto and verso using gradient field and Braille dot attributes. The fourth component separates recto and verso dots into different images (recto and verso pages) containing recto and verso dots only. In the transformation component, dots on the recto page are mirrored using a vertical symmetric line, in order to use the same translation algorithm as verso dots. Moreover, in this component wrongly scanned Braille pages are rotated by 180^{o} automatically by the system when the translation performance is less than a threshold value. A translation performance of a Braille recognition system greater than the

threshold value, shows a Braille image is scanned in its correct direction, while if it is less than the threshold value, it shows the page is scanned in opposite direction. Therefore, the page should be reversed. However, for noisy documents, the translation performance can be less than the threshold value even if they are correctly scanned. Thus to prevent infinite loop, a Braille page (noisy or clean) should be rotated only once. In the recognition component, Braille cells are formulated using Braille cell attributes. Then after, each Braille cell is encoded into Braille codes. Finally, the Braille codes are translated into print text using lookup tables. The details of each component and how they are working are briefly explained in the following sections.

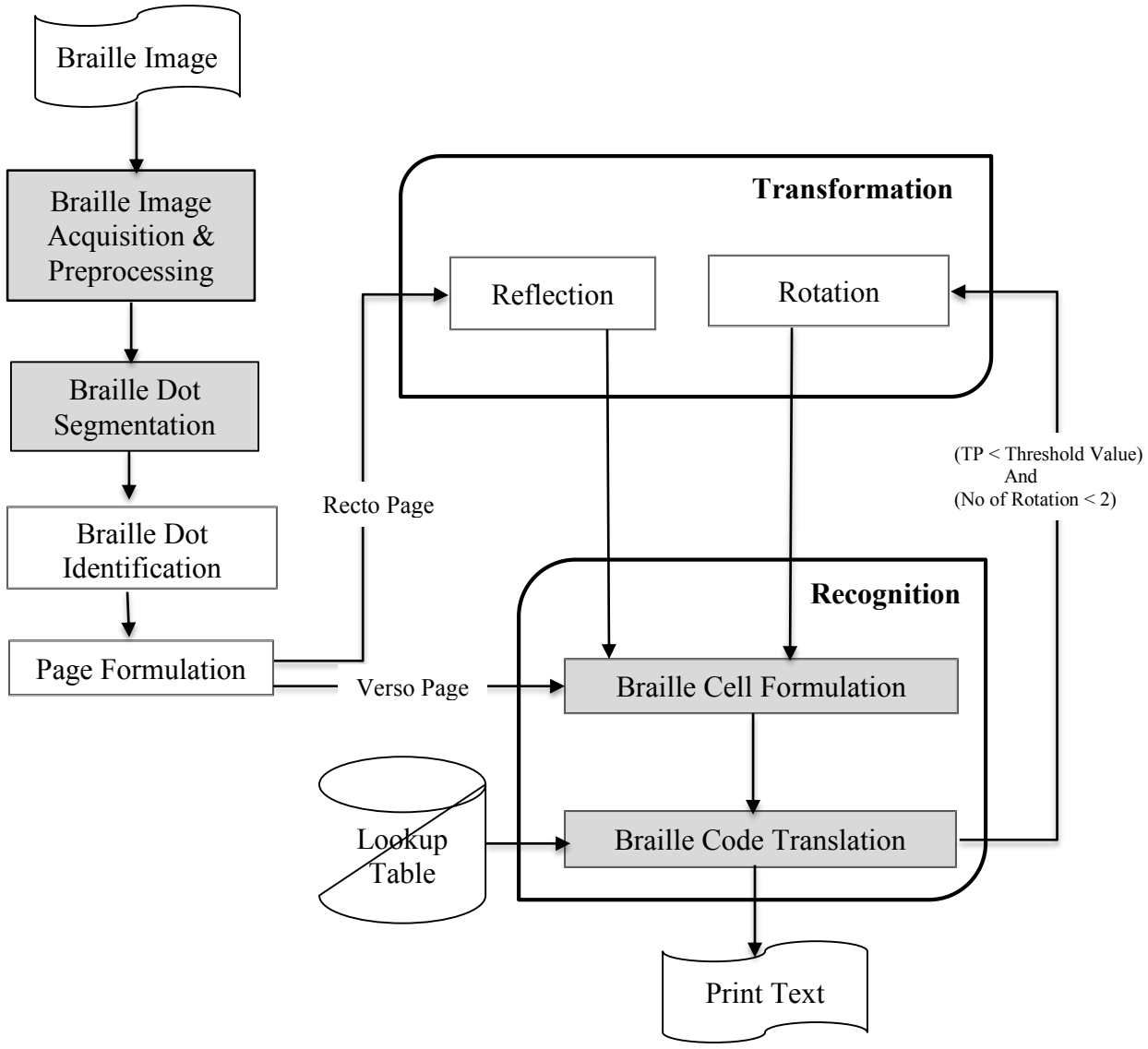


Figure 4.1: *Architecture of the Proposed Double Sided Braille Recognition System*

4.3 Image Acquisition and Preprocessing

This is the first step in digital image processing. For this system, double sided Braille image is scanned using HP scanner with image resolution of 200dpi in JPEG format.

In most low cost scanners, the document page is illuminated from an offset angle. The direct implication for Braille documents is that the illumination of a protrusions (Recto Dots) and depressions (Verso dots) in that page will not be even. The face of the protrusion or depression, which is angled towards the light source, will be more brightly lit and the face of the protrusion or depression angled away from the light source will be considerably less brightly lit. It is this property that can be exploited to enable the recognition of double sided Braille documents.

A scanned Braille page appears with a mid-gray background, and for each protrusion and depression a bright light and a shadow pair exist along the scanning direction. The order in which the shadow and the bright pair appearance for each dot depends upon the model of the scanner involved. Most models represent protrusions as bright light followed by shadow and depressions as shadow followed by bright light as shown in Figure 4.2, while some scanners produce the reverse. The scanner used with this system produces the former pattern and the possibility to reconfigure the system to work with other scanners is provided.

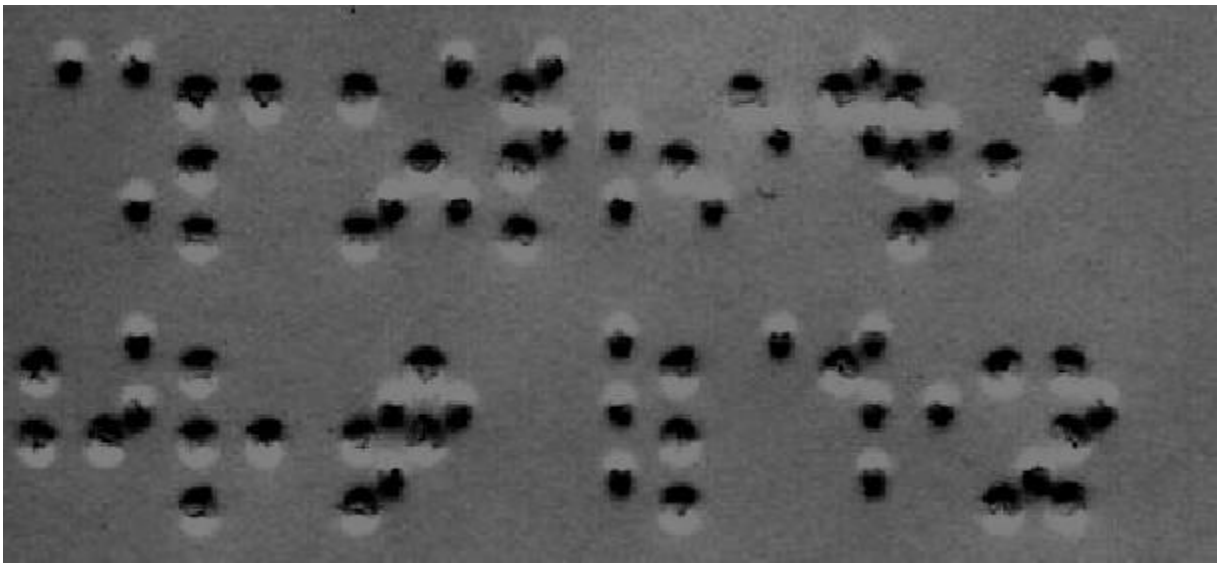


Figure 4.2: *Original Gray Scale Image*

After acquiring the Braille image, the image should be preprocessed for further investigation. The preprocessing step is vital in removing unnecessary parts called impulse noise over the Braille image. This design uses Gaussian filter and its derivatives in removing noise. The low pass spatial Gaussian

filter and its derivatives are selected and implemented successively. The low pass spatial filter attenuate the high spatial frequency noise from the image and the high pass filter preserves the detailed edge information of the Braille dots.

In the preprocessing phase, the original image which is RGB, is converted in to gray scale for performance optimization because dealing with gray scale image (two dimensional) is easier than RGB (three dimensional) image. In order to remove impulse noises and keep edges of the Braille dots, the image (I) is convolved with 1D Gaussian kernels (gx and gy) and 1D Gaussian derivative kernels (dx and dy) as shown in Equations 15 and 16 [3]:

$$dxI = gy * (dx * I) \quad (15)$$

$$dyI = gx * (dy * I) \quad (16)$$

4.4 Braille Dot Segmentation

In this stage, Braille dots are segmented from the background based on the features of the preprocessed image using direction field tensor. Direction field tensor computes the differences in the intensity of pixels giving attention to linear structures (dots) because it uses double angle representation (0-180^o) [3]. This avoids cancelation effects that exist in images computed by gradient field. Direction field tensor has an advantage of segmenting dots from the background by avoiding dominance of noises in noisy image [3].

In order to segment dots from the background using direction field, we compute the orientation tensor (I_{20}) by pixel wise complex squaring and its magnitude (I_{11}) using Equations 17 and 18 [3]. These eigenvalues are averaged by large Gaussian kernels in order to remove extreme values.

$$I_{20} = (dxI + i * dyI)^2 \quad (17)$$

$$I_{11} = abs\,(I_{20}) \quad (18)$$

where:

i is the imaginary unit number equals to the square root of -1,

$\hat{I}_{20}$ is angle of the direction field tensor along which the intensity changing and

$\hat{I}_{11}$ is magnitude of the direction field tensor.

.

From the above procedures, finally we get two eigenvalue analyses I_{11} and I_{20} from the direction field tensor. We use intensity strength or magnitude of the dots (I_{11}) of direction field tensor to significantly segment dots from the background as shown in Figure 4.3 by removing noise.

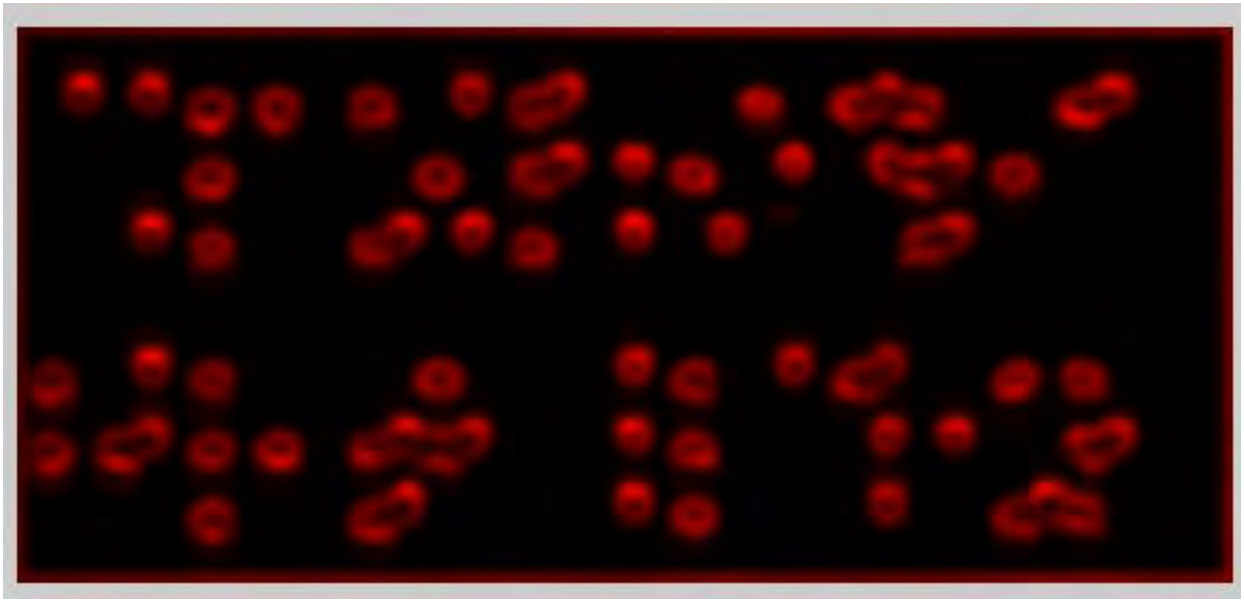


Figure 4.3: *Direction Field Image, I_{11}*

The I_{11} image shown in Figure 4.3 is then binarized with some threshold value generated automatically. But this image has holes on the dots and a white border line which are inside edges of the dots, and the image created on the I_{11} image. As a result, morphological operators are used to fill holes and remove border lines. Finally, we get the binary image show in Figure 4.4.

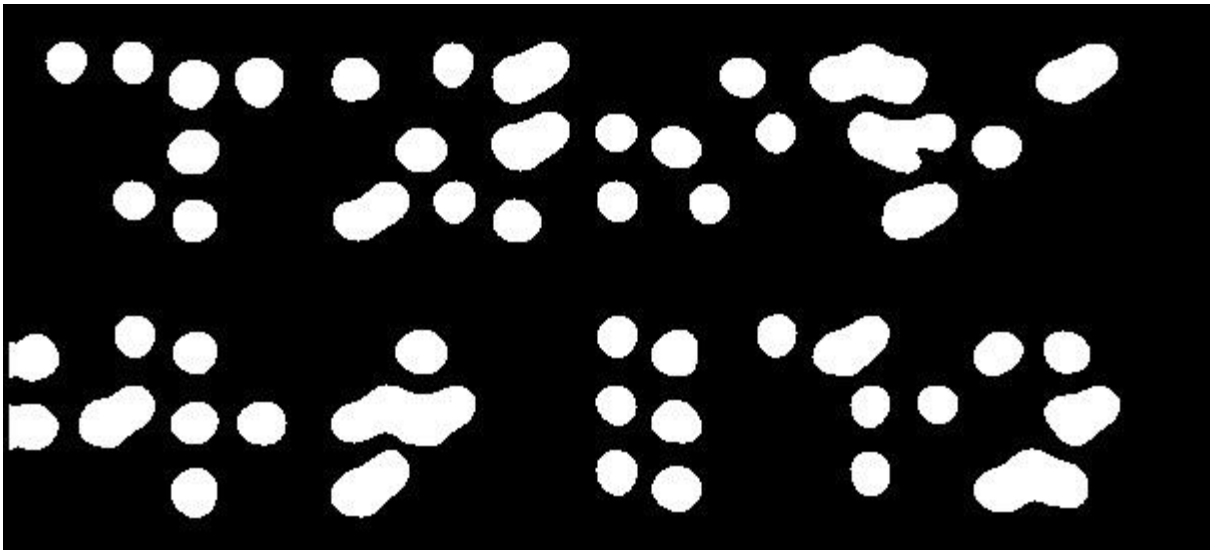


Figure 4.4: *Binary Equivalent of I_{11} Image*

4.5 Braille Dot Identification

This is the crucial stage at which the segmented Braille dots from the background in Section 4.4 are identified as recto or verso using gradient field and attributes of dots such as centroid, orientation and area. Image computation using gradient field has two groups of linear structures in opposite directions which results in a cancelation effect. As a result, gradient field uses a single angle representation (0-360^{o}). This angle representation shows to which direction the intensity is changing. We used this property of the gradient field to differentiate recto and verso dots which shows features of recto and verso dots in the preprocessed image shown in Figure 4.2. The gradient field of this image (I) and gradient angle of each pixel are computed by Equations 19 and 20 respectively [27].

$$\mathrm{I}_{10} = dxI + i * dyI \tag{19}$$

$$gAng = tan^{-1}(\frac{dyI}{dxI}) \tag{20}$$

where:

i is the imaginary unit number equals to square root of -1,

I_{10} is the gradient field image and

gAng is gradient angle at which the intensity is changing over the output G for each pixel.

From the above procedures, we get two eigenvalue analyses I_{10} and gAng. The gradient field output I_{10} is shown in Figure 4.5, from which the gradient angle (gAng) is computed.

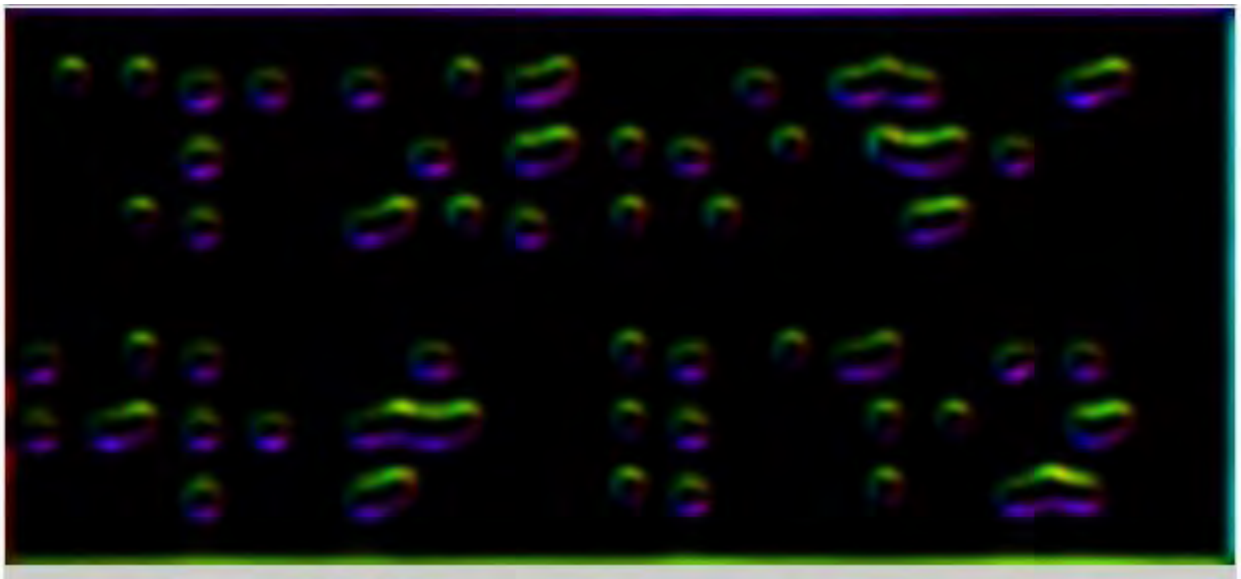
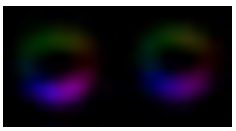


Figure 4.5: *Gradient Field Image, I_{10}*

From the gradient field image shown in Figure 4.5, we can see upper and lower parts of an isolated recto dot are represented with high intensity value of Yellow and low intensity value of Blue colors respectively. From the CIE color lab presented in Section 2.5, Yellow color and its derivatives represent angle values ranging between 0 and 180^o, and Blue color and its derivatives represent angle values ranging between 180^o and 360^o. Whereas, verso dots use the same color representation on the upper and lower part of the dot but low intensity level for Yellow and high intensity level for Blue as shown in Figure 4.6. This variation happens because of the following reason. For the recto dots the gray intensity level changes from bright to dark gray level from top to bottom. However, for the verso dots the gray intensity level changes from dark to bright gray level as shown in Figure 4.2.

This creates variation in number of pixels, angle of intensity level for each pixel changes in the direction between 0 and 180^o, and between 180^o and 360^o for a given dot. Our experiment shows recto dot has large number of pixels whose intensity level changes in angles ranging between 0 and 180^o and verso dot has large number of pixels whose intensity level changes in angles ranging between 180^o and 360^o.

(a) Recto Dots



(b) Verso Dots

Figure 4.6: *Features of Recto and Verso Dots from Gradient Field*

By incorporating the two features, gradient angle which is extracted from the gradient field image and pixel value of the binary image which is extracted from I_{11} image of the direction field, we extract the following features as depicted in Figure 4.7. A dot has gray (0.5) and white (1) regions or pixels. The gray region of the dot shows pixels with gradient angle less than 180^o, and the white region of the dot shows pixels with gradient angle greater than 180^o.

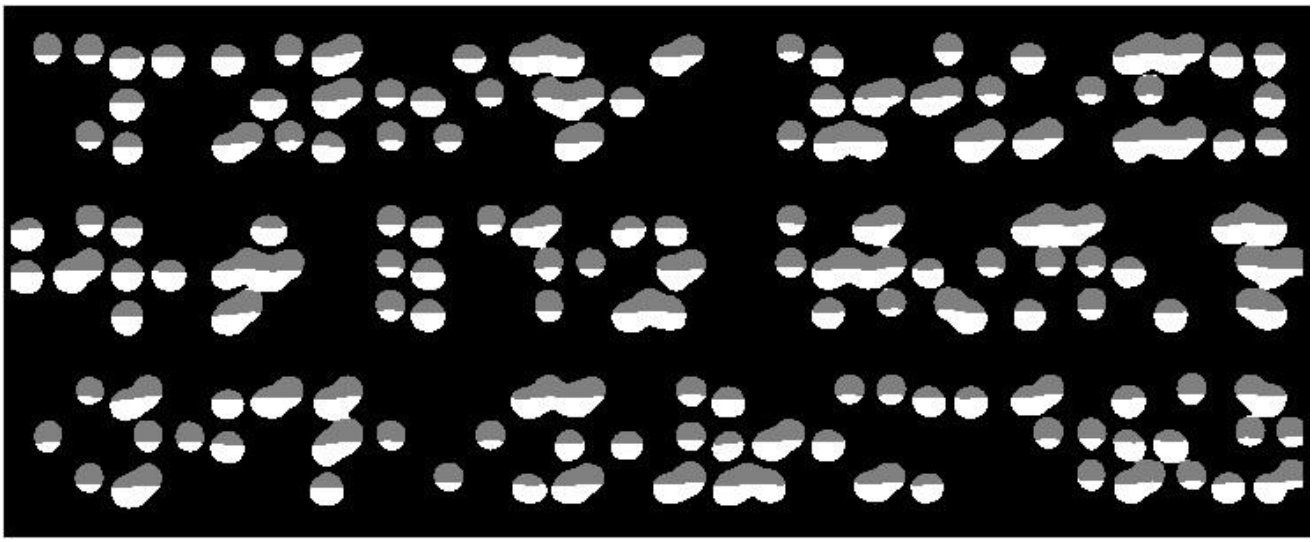


Figure 4.7: *Dot Features from Angle Variations*

Using these features (Figure 4.7) an isolated dot can be identified as recto (a dot with gray intensity domination), and verso (a dot with white intensity domination). The problem with these features is separating overlapping dots and identifying the separated dots as recto or verso. Figure 4.7 shows the overlapping exists in two ways: Vertical and horizontal overlapping of dots. The former case is easy to separate the dots, because at the junction of the overlapping, a pixel with gray value comes next to a pixel with white. Therefore, this feature is used to segment or separate overlapping of dots in the vertical direction using morphological operator. The operator is implemented at the junction pixels, and gives the image shown in Figure 4.8.

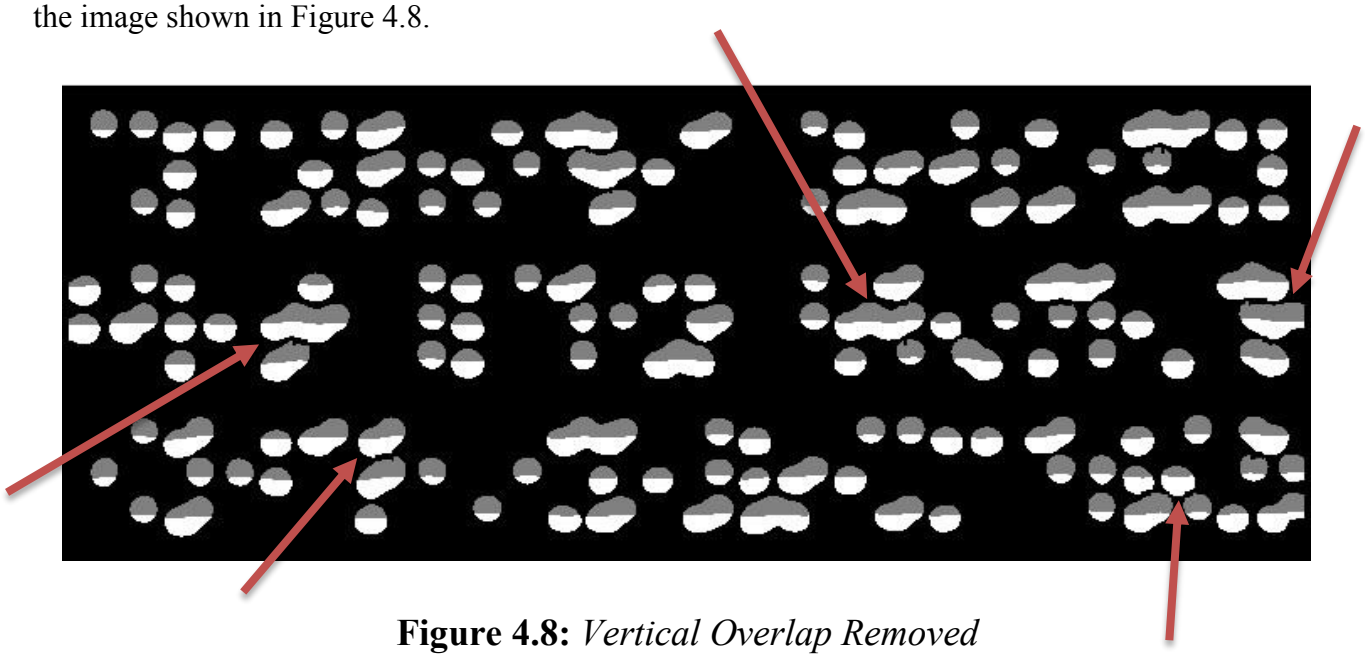


Figure 4.8: *Vertical Overlap Removed*

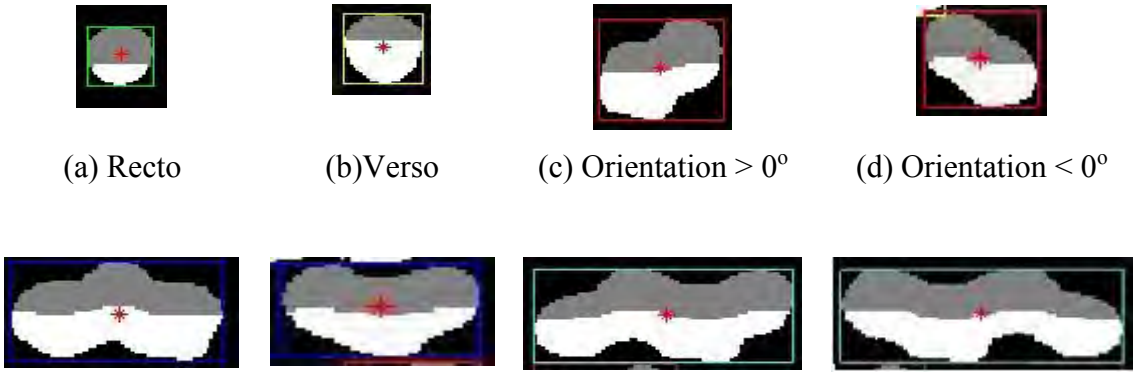
Thus, once we remove vertically overlapping dots, we are left with horizontally overlapping dots. In practice, there are maximum of four horizontal dots (two dots from recto Braille cell and two from verso Braille cell). Thus, the maximum number of dots which produces horizontal overlapping is four. As a result, we managed horizontal overlapping of dots in three cases for different orientations as shown in the Figure 4.9 (c) to Figure 4.9 (h).

Once we knew features of isolated recto and verso dots, we minimized the complexity of the overlapping into horizontal type. We used the following isolated and overlapping dot attributes to identify any type of dot as recto or verso.

1. **Centroid**: This is a point over a Braille dot that tells intensity distribution over a Braille dot. It locates the center of the intensity distribution over a dot like locating center of mass distribution. We get from our experiment that the centroid of a recto dot lies on the gray region of the dot as shown in Figure 4.9 (a). This is because the gray pixel value dominates the white pixel value. In other words, number of pixels having angle values less than 180^{o} dominate than number of pixels whose angle values are greater than 180^{o}. On the other hand, the white region dominates the gray region for verso dots. Thus, the centroid of a verso dot lies on the white region of the dot as shown in Figure 4.9 (b). Table 4.1 summarizes pixel value of dots at the centroid for different cases: For isolated recto and verso dots, overlap of two, three or four dots as shown in Figure 4.9 (a-h).

Table 4.1: *Pixel Value of Isolated and Overlapping Dots at their Centroid*

Dot Variants	Pixel Value at the Centroid	Dot Type
Isolated Dot	Gray=0.5	Recto
	White=1	Verso
Overlap of two dots	Gray or white	Indefinable
Overlap of three dots	White =1	Middle dot is Recto
	Gray = 0.5	Middle dot is Verso
Overlap of Four dots	Gray or white	Indefinable



(a) Recto (b)Verso (c) Orientation > 0° (d) Orientation < 0°

(e) Recto at the middle (f) Verso at the middle (g) Orientation > 0° (h) Orientation < 0°

Figure 4.9: *Identified Dots and their Variations*

2. **Area**: It is the area covered by an object over an image. In our experiment, area refers to the area of an isolated dot or area of overlapping dots. We inferred ranges of area for varieties of dots (Table 4.2) from the graph showing area distribution of dots (Figure 4.10) found in our experiment. We used area to differentiate a segmented object as isolated or overlapping dots. The overlap can be overlap of two, three or four dots.

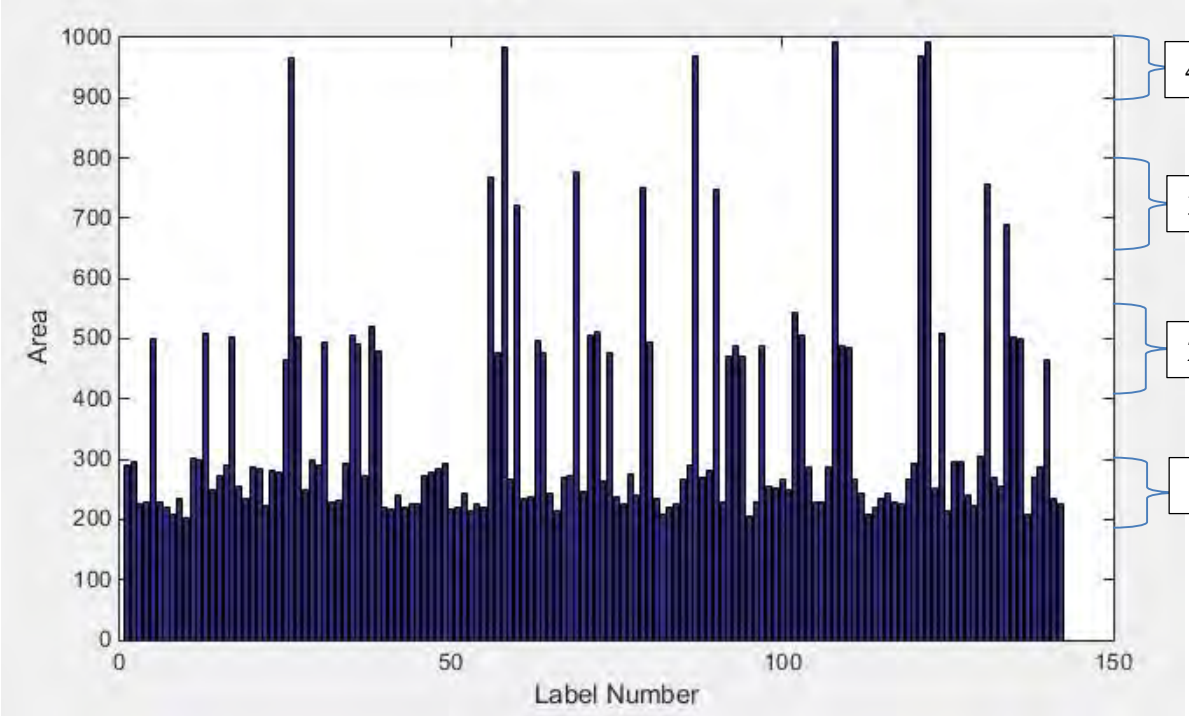


Figure 4.10: *Area Distribution for Recto, Verso and Overlapping Dots*

Table 4.2: Range of *Area for Varity of Dots*

Range Number	Range of Area	Number of Dots in the Overlapping
1	200-300	One (Recto or Verso)
2	400-550	Two overlapping Dots
3	650-800	Three overlapping Dots
4	900-1000	Four overlapping Dots

3. **Orientation:** This is an angle from the horizontal that measures to which direction the dots make the overlapping. In Braille writing system, to prevent overlapping of dots (Recto and Verso), Braille embossers or Braille writer makes a diagonal offset downward whenever they need to use the two sides of a Braille. However, in practice the overlapping exists. The diagonal offset downward creates a remarkable orientation angle. We used this property to separate overlapping dots. The orientation angle shows the locations of the overlapping dots relative to the centroid. For instance, for two overlapping dots whose orientation angle is greater than 0^o, the first dot is found left below the centroid and the second dot is found right above the centroid as shown in Figure 4.9 (c). However, for two overlapping dots whose orientation angle is less than 0^o, the first dot is found left above the centroid and the second dot is found right below the centroid as shown in Figure 4.9 (d). Thus, once we know the position of the dots, we can easily determine the centroids of the dots which make the overlapping and identify the dots as recto or verso by looking at pixel values at the centroids (Algorithm 4.2). The same principle works for overlapping of four dots, except the method we used to determine the centroids of the four dots making the overlapping is different (Algorithm 4.4).

Using the parameters (intensity value and angle of a pixel on the dot, Area and Centroid of a dot) we identify a dot as recto, verso or overlapping dots (Algorithm 4.1) as shown in Figure 4.11.

Input: - preprocessed Braille Image
Output: - an image whose dots are identified as Recto, Verso or Overlapping

```
Compute the direction and gradient field of the input image.
Compute I11 from direction filled image.
Compute gradient angle (gAng) from gradient filled image, I10.
Find binary equivalent of I11 image, B.
Create a zero image, V having the same size as B.
For each pixel i
    If (Magnitude of B (i) =1)
        If (gAng (i) < 180°)
            V (i) == 0.5;
        Else if (gAng > = 180°)
            V (i) == 1;
        End if
    Else
        V (i) == 0;
    End if
End for
Compute region properties (centroid, area and orientation of each dots) on
image V.
For each dot i on V
    If (area of the dot < 300)
        If (pixel value of the dot at its centroid = 1)
            Identify the dot as Verso.
        Else if (pixel value of the dot at its centroid = 0.5)
            Identify the dot as Recto.
        End if
    Else
        Identify the dot as overlapping
    End if
End for
Return image V.
```

Algorithm 4.1: *Dot Identification as Recto, Verso or Overlapping*

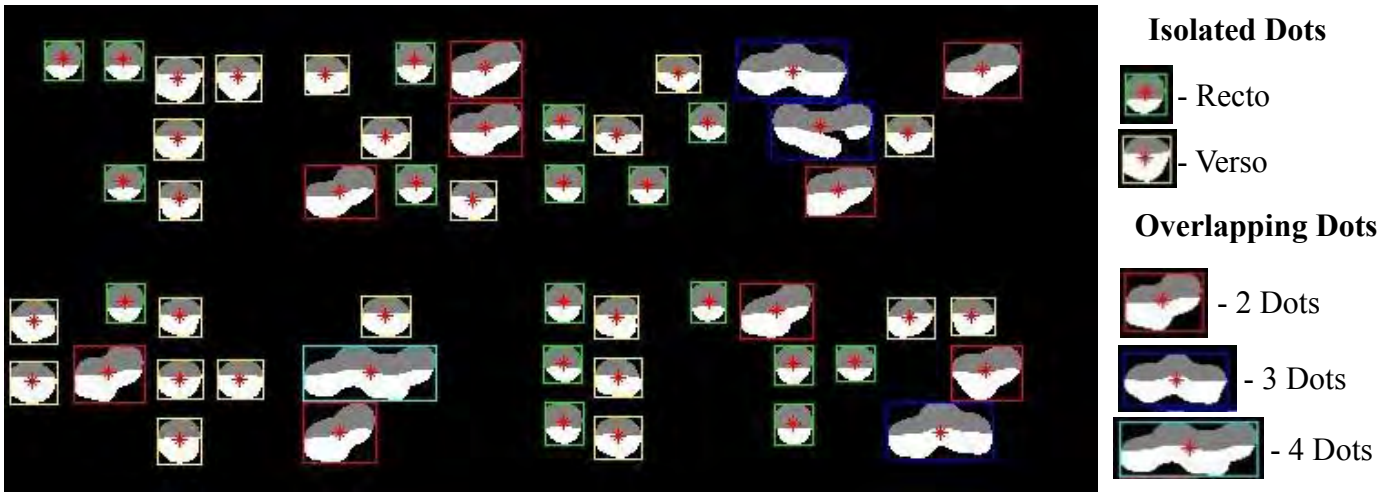


Figure 4.11: *Identified dots as Recto, Verso and Overlapping dots*

As we have discussed, overlap of dots in the vertical direction is already solved using morphological operators. In order to solve overlap of dots in the horizontal direction, we break down the problem into three overlapping cases as briefly discussed below. The problem is solved by using orientation of the overlap, centroid of the overlap and dimensions (width and height) of the box bounding the overlap.

Case 1 Overlap of Two Dots

Overlapping of two dots exists with orientation angle greater than zero or less than zero respectively as shown in Figure 4.9 (c) and Figure 4.9 (d). If the orientation angle is greater than zero, the dots are found at the first and third quadrants by assuming the centroid as an origin. On the other hand, the dots will be on the second and fourth quadrants for orientation angle less than zero. We use this pattern to find the center of each dot connected and identify them as recto and verso dots using Algorithm 4.2.

Input: - Two Overlapping Dots
Output: - Recto and Verso Dots

Compute the orientation of the overlap.
Compute the centroid of the overlap (X, Y).
Compute quarter of width and quarter of height of the box bounding the overlap as w and h respectively.
If Orientation > 0
Compute centroids $C_1 (X_1, Y_1)$ and $C_2 (X_2, Y_2)$ of the two dots as
$C_1 (X_1 = X + w \quad , \quad Y_1 = Y - h)$ and
$C_2 (X_2 = X - w \quad , \quad Y_2 = Y + h)$
Else if Orientation < 0
Compute centroids $C_1 (X_1, Y_1)$ and $C_2 (X_2, Y_2)$ of the two dots as
$C_1 (X_1 = X - w \quad , \quad Y_1 = Y - h)$ and
$C_2 (X_2 = X + w \quad , \quad Y_2 = Y + h)$
End if
Compute pixel value of each dots at their centroid
For each dot
If pixel value at the centroid is one
Identify the dot as Verso.
Else if pixel value at the centroid is half
Identify the dot as Recto
End if
End for

Algorithm 4.2: *Identification of Two Overlapping Dots*

Case 2 Overlap of Three Dots

Like that of the two overlapping dots, overlapping of three dots also exists in two different ways: concave and convex shapes as shown in Figure 4.9 (e) and Figure 4.9 (f). We can see the centroid of the overlapping located in two different regions, which measures intensity distribution of the overlapping. We recognized a centroid is located in gray region when pixels with gray intensity are larger in number than pixels with white intensity, and a centroid is located in white region when pixels with white intensity dominates gray intensity pixels. This is due to the cumulative effect of two similar dots. For instance, in Figure 4.9 (f) two recto dots bound a verso dot. For this case, the centroid is located over gray region, which tells domination of gray pixels (two bounding recto dots) over white pixels (middle verso dot). Thus, the bounding dots are identified as recto, and the middle one is identified as verso dot. On the other hand, if a centroid is located at white region, it tells domination of

white pixels (two bounding verso dots) over gray pixel (middle recto dot). Therefore, we used pixel value at the centroid of the overlapping dots (gray or white) to identify the type of the bounding and the middle dots as recto or verso. We used this feature to design and implement Algorithm 4.3 to identify three overlapping dots.

Input: - Three Overlapping Dots
Output: - Recto and Verso Dots
Compute the centroid of the overlap (X, Y).
Compute pixel value at the centroid.
If pixel value at the centroid is one
Identify the middle dot as Recto and the bounding dots as Verso**.**
Else if pixel value at the centroid is half
Identify the middle dot as Verso and the bounding dots as Recto.
End if

Algorithm 4.3: *Identification of Three overlapping dots*

Case 3 Overlap of Four Dots

Similar to overlap of two dots, four dots overlap in two ways: with orientation angle greater than zero and orientation angle less than zero as shown in Figure 4.9 (g) and Figure 4.9 (h). When orientation angle is less than zero, the first and third dots in the overlap are placed above the centroid, and the second and fourth dots are placed below the centroid. However, when the orientation angle is greater than zero, the dots will be reversed. The first and third dots are placed above the centroid, and the second and fourth dots are placed below the centroid. We used this features to design and implement Algorithm 4.4 in determining the centers of the overlapping dots. After determining the center of each dot, we look for pixel value at the center of each dot. A dot with pixel value of 1 at the center is verso and a dot with pixel value of 0.5 at the center is recto.

Input: - Four Overlapping Dots
Output: - Two Recto and two Verso Dots

Compute the orientation of the overlap.
Compute upper left corner coordinate(X, Y), one eighth of width and quarter of height of the box bounding the overlap as w and h respectively.
If Orientation > 0
Compute centroids of the four dots (C_1, C_2, C_3 and C_4) as
C_1 ($X_1 = X + w$, $Y_1 = Y + 3h$),
C_2 ($X_2 = X + 3w$, $Y_2 = Y + h$),
C_3($X_3 = X + 5w$, $Y_3 = Y + 3h$) and
C_4($X_4 = X+7w$, $Y_4 = Y + h$)
Else if Orientation < 0
Compute centroids of the four dots (C_1, C_2, C_3 and C_4) as
C_1($X_1 = X + w$, $Y_1 = Y + h$),
C_2($X_2 =X + 3w$, $Y_2 = Y + 3h$),
C_3($X_3 = X + 5w$, $Y_3 = Y + h$) and
C_4($X_4 = X + 7w$, $Y_4 = Y + 3h$)
End if
Compute pixel value of each dots at their centroid
For each dot
If pixel value at the centroid is one
Identify the dot as Verso.
Else if pixel value at the centroid is half
Identify the dot as Recto**.**
End if
End for

Algorithm 4.4: *Identification of Four Overlapping Dots*

After implementing all the techniques and algorithms discussed above, we identify dots (isolated or overlapping) as recto and verso dots. Finally, we represent pixel values of 1 and 0.5 to label recto and verso dots respectively as shown in Figure 4.11.

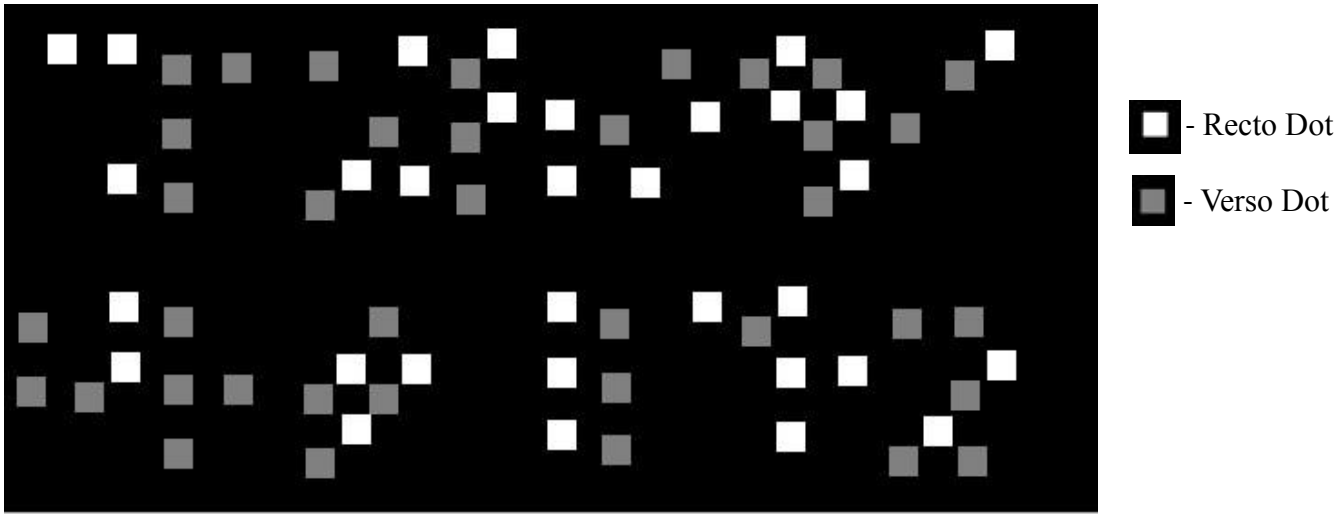


Figure 4.12: *Identified Recto and Verso Dots (Recto=1 and Verso =0.5)*

4.6 Page Formulation

For further processing, in transformation and recognition which are briefly discussed in Section 4.7 and Section 4.8 respectively, the recto and verso dots which are found on the same image or page (Figure 4.12) should be separated in two different pages: Recto and Verso Pages. Dots with pixel value of 1(white) are formulated into Recto Page (Figure 4.13), and dots with pixel value of gray (0.5) are formulated into Verso Page. After formulation, the Verso Page is binarized (Figure 4.14). Binarization of the page is required to use the same translation and Braille cell encoding algorithms for both of the pages, since the Braille cell encoding algorithm (Section 4.8.1) and translation algorithm (Section 4.8.2) designed depends on pixel value of dots.

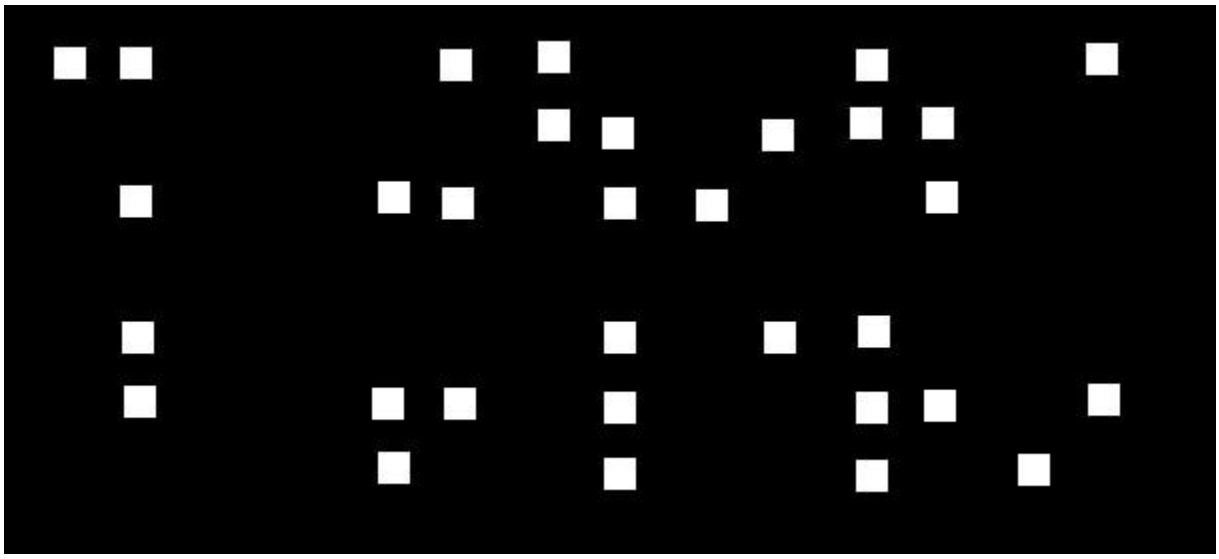


Figure 4.13: *Recto Dots /Page/*

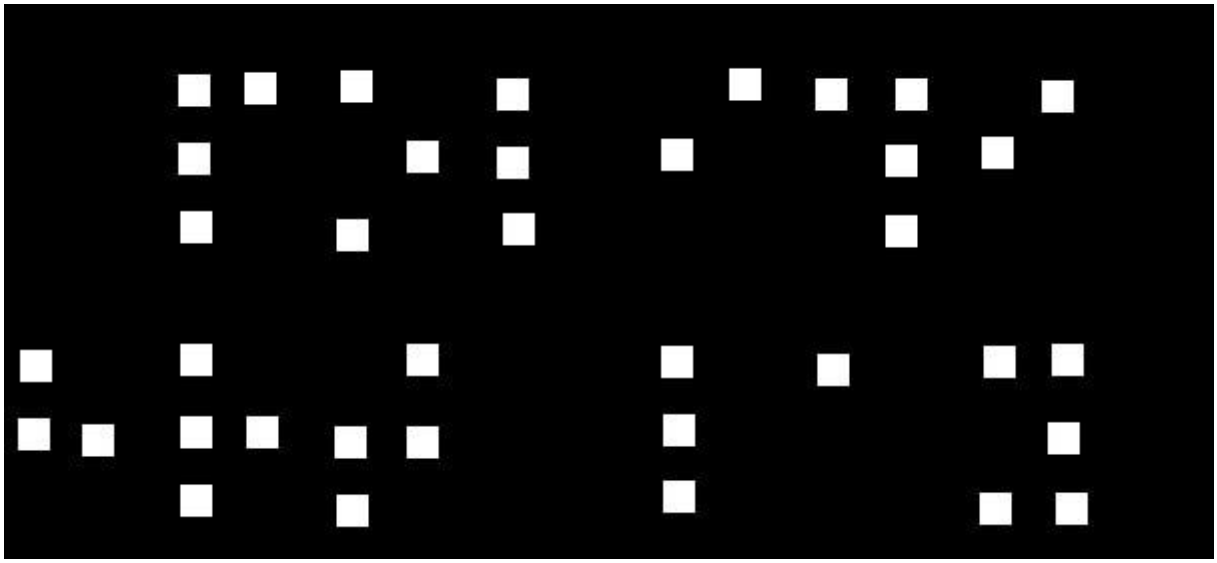


Figure 4.14: *Binarized Verso Dots /Page/*

4.7 Transformation

This is a process by which points in a plane or space are transformed into another point of interest using reflection, translation or rotation. We apply concept of reflection to turn a page, because the dot points (recto or verso) have similar shape which is invariant of reflection. In this component, a recto page is turned horizontally from right to left. Because the page cannot be read directly (without turning) unless a separate translation algorithm that can read from right to left is implemented. Thus, Algorithm 4.5 is

designed and implemented to reflect all points on the image about a vertical symmetric line. For instance, a Braille page containing recto dots in Figure 4.13 is turned into its verso page equivalent as shown in Figure 4.15. Designing this algorithm is required in order to use only one Braille cell encoder and translation algorithms (Algorithm 4.6 and Algorithm 4.7) for both pages. Otherwise, we need to design two algorithms for each of them, one reads from left to right downward for verso page and the other from right to left downward for recto page.

Moreover, a page can be scanned wrongly in a reversed direction (header becomes footer), because most people are not aware of Braille characters. To solve this problem, the page (image) should be rotated by 180^{o} as shown in Figure 4.16. This figure depicts the reverse image of the page in Figure 4.15 by 180^{o}. The question is when to reverse or rotate the page automatically by the system. It needs performance evaluation of the translation. How much of the Braille cells are translated correctly into the corresponding characters before and after the image is reversed. In other words, it needs finding a threshold value (maximum translation performance value for Braille images scanned wrongly or in a reversed direction). According to the result shown in Table 5.5, the maximum threshold value is 68.5% for Amharic Braille documents. If a page is translated with a translation performance less than the threshold value, it tells the Braille image is scanned wrongly thus it needs rotation. Evaluating this threshold value optimizes performance of the system. Otherwise, we need to compare translation performance for both cases (wrongly and correctly scanned) and select a print text with better performance. Because, we don't know which one is correct or wrong before the recognition process unless we can read Braille.

Input: - Recto Page/ image
Output: - Turned/Reflected Page
Compute width (w) and height (h) of the input image.
Find the vertical symmetric line, VL=w/2;
For each pixel of the input **i**mage
If pixel value of the pixel is one
Compute horizontal distance D of the pixel from VL.
Take the pixel a distance equal to D in opposite direction from VL.
End if
End for

Algorithm 4.5: *Algorithm to Turn a Page*

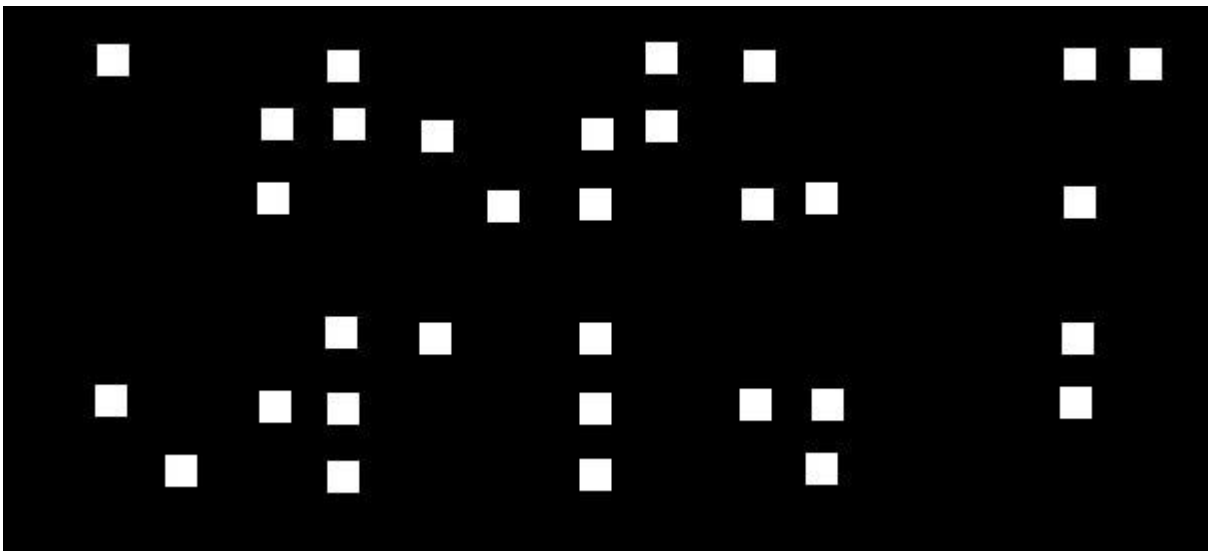


Figure 4.15: *Turned Image (Page) of Recto Page in Figure 4.13*

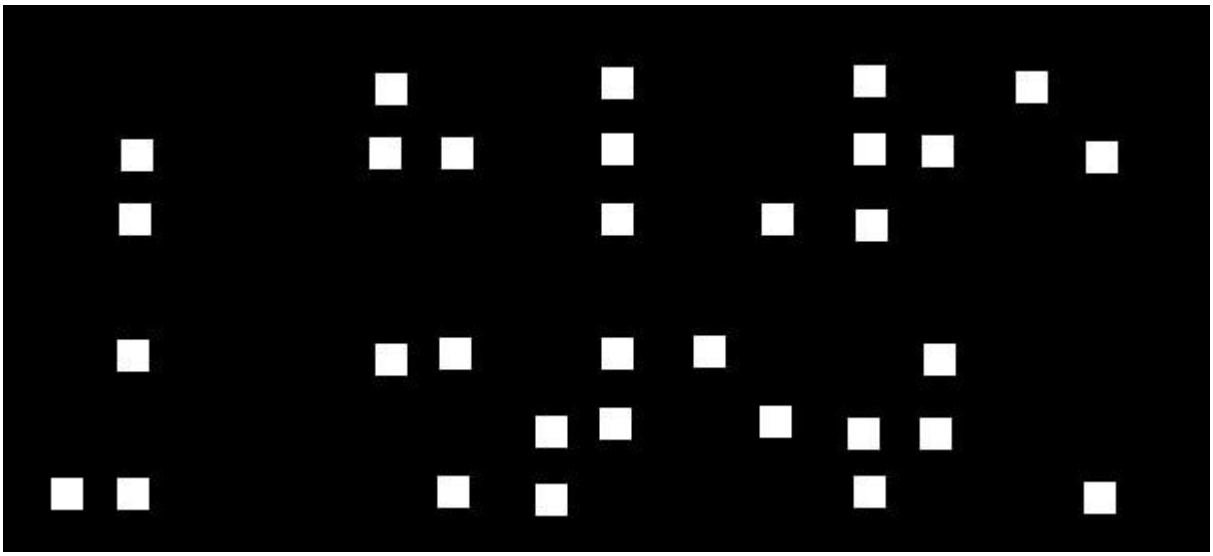


Figure 4.16: *Reversed Image of the Page in Figure 4.15 by 180º*

4.8 Recognition

In this component, the dots on each page are formulated into a Braille cell. The Braille cells are encoded into a Braille code. Finally, the Braille codes are translated into Amharic print text. The details of each process are explained briefly as follows.

4.8.1 Braille Cell Formulation

In this stage, after identifying the dots (recto and verso) and turning the page containing recto dots, a Braille cell is formulated. A Braille cell formulation refers to recognition of the cell that contains six dots, and converts them into the corresponding Braille code. This is done by using Braille cell attributes: Braille cell height, Braille cell width, space between Braille cells, space between Braille cell lines and average dot width and height for a Braille document scanned with 200 dpi as shown in Table 4.3.

Table 4.3: *Braille Cell Attributes in Braille Cell Formulation*

Attributes	Estimated Values (in pixels)
Dot width(dw)	12
Dot height(dh)	12
Cell width(w)	52
Cell height(h)	90
Space between Braille cells(sbbc)	28
Space between Braille cell lines(sbbcl)	44

Using this information, a Braille cell is formulated using the following procedures:

1. **Draw a Character Line:** A character line is drawn by first finding the top and bottom margin lines. To know the position of the top margin line, traverse from left top corner of the image to the right along each row of pixel until the top most Braille dot (pixel value of one) is found. For instance, if the top most dot is found at (row, column) = (5, 10), then the top margin line will be row 5. The same is true for the bottom margin line, the difference is the traverse starts from left bottom corner or the last row upward until the most bottom dot is found. This enhances system performance. After knowing the margin lines, using Braille cell attributes (cell height and space between Braille cell lines), we draw character lines starting from the top margin line to the bottom margin line as shown in Figure 4.17.
2. **Draw a Window of Size Equals to the Braille Cell:** Before drawing the windows, we determine the position of the left and right margins by applying the same logic as top and bottom margins. Then after, along each character line, we put a window of size equals to the width of a Braille cell starting from the left margin to the right margin line. A gap between windows is considered which is equal to the estimated space between Braille cells. After implementing the

above procedures we find a formulated Braille cell shown in Figure 4.17. Interpretation of each window into a Braille code is described in the following procedure.

3. **Find the Braille Code of Each Window:** This is the last stage in Braille cell formulation. In this stage, each window is encoded into a Braille code depending on the position of the dots bounded by the window. For instance, a Braille cell shown in Figure 4.18 is taken from the Figure 4.17 encoded in to a Braille code or cell code of '1235'. This process is done by searching a pixel value of one in the window that shows presence of a dot. The search process takes place at six different regions (small windows) with in the Braille cell window. These regions are the most probable positions of the dots. The size of the region considered is equal to average Braille dot size. The regions located at the four corners of the Braille cell window are encoded as 1, 3, 4 and 6. In addition, the regions located at half of the Braille cell window's height in both sides are encoded as 2 and 5. Checking presences of a dot (pixel value of one) at these regions (small window) instead of the whole Braille cell window optimizes the performance of the system. Taking this into account, we designed and implemented Algorithm 4.6 in order to encode each Braille cell window into the corresponding Braille code.

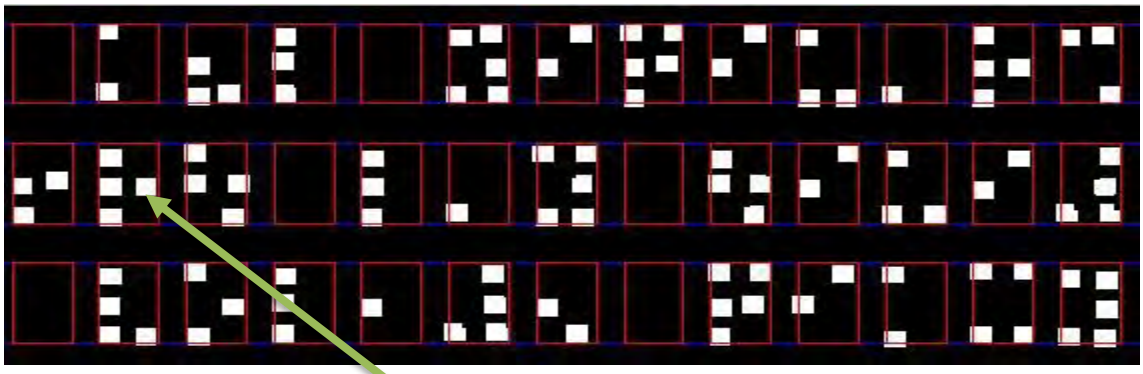


Figure 4.17: *Braille Cell Formulation*

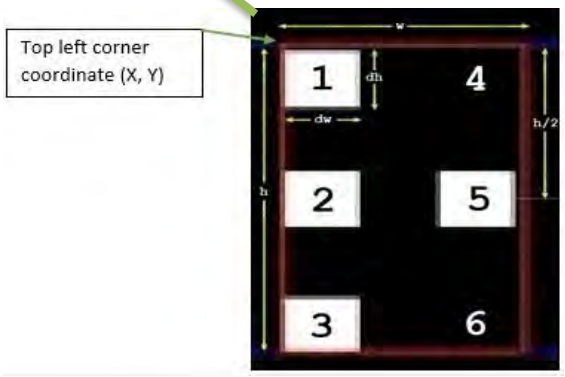


Figure 4.18: *A Single Braille Cell*

Input: -
A Braille Image whose Braille cells are formulated,
Braille dot dimensions. Width (dw) and height (dh).
Output: - Braille Code of each Braille cell

```
Take a Braille cell.
Compute upper left corner coordinate(x, y), width (w) and height (h) of the Braille cell window.
Initialize a string variable to hold the Braille code of the selected Braille cell, cellCode='';
Initialize an integer variable to hold position of a dot, dotPosition=1;
Create a small window of size equals to the Braille dot.
Put the upper left corner coordinate of the small window at X= x; and Y=y;
For each dot in the Braille cell window dot1::: dot6
    For each pixel in the Braille cell window bounded by the small window
        If (pixel value == 1)
            cellCode= strcat (cellCode, int2str (dotPosition));
            break;
        End if
    End for
    dotPosition = dotPosition + 1;
    If (dotPosition == 2)
        Put the upper left corner coordinate of the small window at
        X= x; and
        Y= y + h/2 – dh/2;
    Else iF (dotPosition == 3)
        Put the upper left corner coordinate of the small window at
        X= x; and
        Y= y + h – dh;
    Else iF (dotPosition == 4)
        Put the upper left corner coordinate of the small window at
        X= x + w - dw; and
        Y= y;
    Else iF (dotPosition == 5)
        Put the upper left corner coordinate of the small window at
        X= x + w - dw; and
        Y= y + h/2 – dh/2;
    Else iF (dotPosition == 6)
        Put the upper left corner coordinate of the small window at
        X= x + w + dw; and
        Y= y + h – dh;
    End if
End for
Return cellCode as Braille code of each cell;
```

Algorithm 4.6: *Braille Cell Encoding*

4.8.2 Braille Code Translation

In this stage, a Braille code is translated into the corresponding Amharic symbol (character). This is done using lookup tables and a translation algorithm shown in Algorithm 4.7. The translation algorithm takes a Braille code as an input from the formulated Braille cell in Section 4.8.1. The algorithm checks presence of the code in the database and returns the corresponding character. This algorithm doesn't go to the database for Braille codes representing mode indicators and vowels. These Braille codes are not directly seen or interpreted as a character on the text version of the Braille as discussed in Section 2.2. The vowels will have meaning if they come next to Braille code of consonants to represent syllable combinations (consonant + vowel). Similarly, the mode indicators have meaning when they come before numbers (Ge'ez or Arabic).

The lookup tables are designed in four tables as shown in Figure 4.19 based on the Amharic Braille character or symbol organization. Because symbols in different categories (Numbers both Arabic and Ge'ez, Punctuations, Consonants, Vowels and Syllable combination of Consonants and Vowels) are uniquely identified by a Braille code as presented in Section 2.2 (Table 2.1 to Table 2.2). Organizing the characters in four different lookup tables increases the performance of the system in searching characters for a given Braille code. In translating a Braille code, the worst case is to search the code in the four tables, and the best case is to search the code only in one of the tables. In addition, it enhances the translation process by removing ambiguity, because characters or symbols in different categories have the same Braille code. For instance, a Braille code 1:2:4 represents three characters (6, ፮ and ፱) as shown in Table 4.4 from two different categories tbl_numbers and tbl_consonants. The system recognizes as number, if a mode indicator of Arabic or Ge'ez comes before the Braille code. The system searches for '6' or '፮' respectively directly from tbl_numbers. If a mode indicator is not found, the system searches '፱' directly from tbl_consonants. Otherwise, the system will print at least two of them ('6' and '፱' or '፮' and '፱') if the lookups were not organized into four lookups. The values of the lookup tables are annexed (Annex E-H).

Braille codes for mode indicators and vowels are statically hardcoded because we don't need to translate and print them. Because on Amharic print documents, the vowels and the mode indicators are not visible as print characters. The Braille codes for mode indicators tell the system the next Braille cell is an Arabic or Ge'ez number depending on the mode value. The Braille codes for vowels tell the system combination of the previous Braille code and the Braille code of the vowel together represent

syllable combination. Thus, hardcoding or not putting mode indicators and vowels in lookup tables has its own advantage in performance optimization; because the system doesn't go to the database.

```
Input: - Array of Braille code
Output: - print text
    Initialize a character and a string variable char='', str='';
    For each Braille code instance i, code[i]
        If code[i] is null
            Assign space character to char;
        Else if (isGeezNoIndicator (code[i]))
            Select geez-number equivalent of code [i+1] from tbl_number;
            Assign the character to char;
            Increase index i by 1;
        Else if (isArabicNoIndicator (code[i]))
            Select arabic-number equivalent of code [i+1] from tbl_number;
            Assign the character to char;
            Increase index i by 1;
        Else if (isConsonant (code[i]))
            If (isVowel (code [i+1]))
                brailleCode = strcat (code[i], code [i+1]);
                Select character equivalent of brailleCode from tbl_sylabel_combination;
                Assign the character to char;
                Increase index i by 1;
            Else
                Select character equivalent of code [i] from tbl_consonant;
                Assign the character to char;
            End if
        Else if (isPunctuation (code[i]))
            Select character equivalent of code [i] from tbl_punctuation;
            Assign the character to char;
        Else
            Has no equivalent character; //wrongly encoded Braille code
        End if
        str = strcat (str, char);
    End for
    Return str as print text;
```

Algorithm 4.7: *Braille Code Translation*

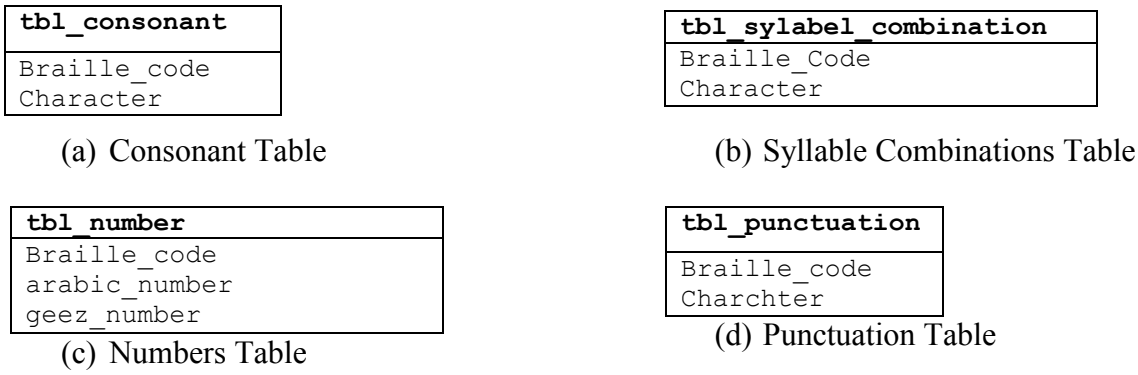


Figure 4.19: *Design of the Lookup Tables*

Table 4.4: *Braille Codes Representing More than One Symbol*

Braille Code	Symbols Represented
1	1, ⠁ and ኣ
2:4	9, ⠊ and ኢ
1:2	2, ⠃ and ብ
1:2:4	6, ⠆ and ፍ
1:2:4:5	7, ⠇ and ግ

4.9 Summary

The chapter systematically went through the designing of double sided Amharic Braille Document Recognition System. The designed system has components working together. In the first component, double sided Amharic Braille image is acquired and preprocessed to remove noises using Gaussian filters. In the second component, direction field tensor is used to segment dots from the background. In the third component, gradient field is implemented to identify the segmented dots as recto, verso or overlapping dots. Finally, we designed algorithms to segment and identify overlapping dots as recto and verso based on the features (centroid, area, and orientation) extracted. The recto and verso dots are separated into recto and verso pages. In the recognition component, we used Braille cell information (dot width and height, cell width and height, space between Braille cells and Braille cell lines) in order to formulate Braille cell using Braille cell encoder directly for Braille pages containing verso dots. Finally, the Braille code encoded by the encoder is translated into print text using the Braille code

translator. However, Braille pages containing recto dots or reversed pages need transformation. Hence, a third component is required. In this component, we used the concept of reflection and rotation before formulating the Braille cells. We used reflection to turn recto page into its verso page equivalent. This helps us to use the same Braille cell encoder and translation algorithm. In addition, we used rotation to reverse the page automatically in case there is error during image acquisition (wrong direction) depending on the performance of the translation.

Chapter Five : Experiment

5.1 Introduction

This chapter describes implementation detail of the proposed design for recognition of double sided Amharic Braille documents. Section 5.2 presents the datasets used in testing the system. Section 5.3 describes the tools used and the overview of the system. Section 5.4 presents the test results found. Finally, the proposed design is evaluated in the last section of the chapter.

5.2 Data Sets

In order to test the proposed design for recognition of double sided Amharic Braille documents, we used six Braille documents with the following properties. These documents are collected from Addis Ababa University Kennedy Library, Ethiopian Association for Blind Society and Misrach Center.

Table 5.1: *Experimental Data*

Braille Attribute	Value
Number of Braille Documents Tested	6
Resolution	200 dpi
Image Format	JPEG
Braille Type	Double Sided

5.3 Implementation

The prototype is implemented using MATLAB and MySQL. MATLAB is a high-level language and interactive environment used to design and test the tools and techniques used in our approach. The Braille codes and their corresponding symbols or characters are managed in MySQL database which is an open source database management system. The prototype takes a Braille image as an input using the graphical user interface shown in Figure 5.1. The designed system is implemented using Dell laptop core i5, processor speed of 1.6 GHz, 4.0GB RAM and 64 bits operating system. The system translates on average 6.8 Braille cells/ second into a print text. Table 5.2 shows output of the designed system.

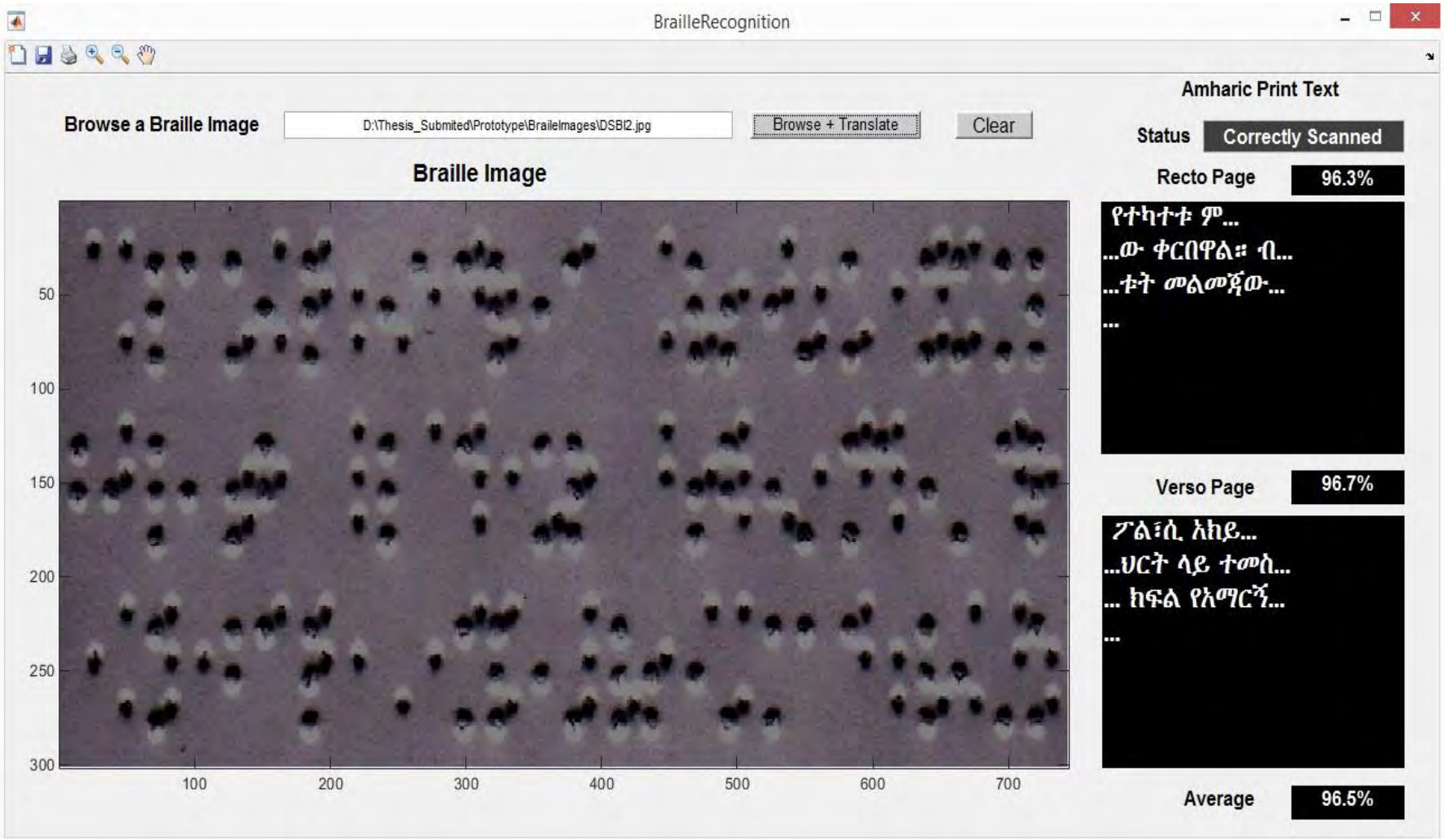


Figure 5.1: *Running Prototype*

Table 5.2: *Translation of Amharic Braille by Experts and System*

Braille Document	Experts Translation		System Translation	
	Recto Page	**Verso Page**	**Recto Page**	**Verso Page**
DSBI2	የተካተቱ ም..... ቀርበዋል። ብ.... መልመጃዎ	ፖሊሲ አክይ..... .....ሀርት ላይ ተመስ .... ክፍል የአማርኝ.....	የተካተቱ ም..... ቀርበዋል። ብ..... መልመጃው	ፖል፣ሲ አክይ..... .....ሀርት ላይ ተመስ ....ክፍል የአማርኝ.....
DSBI3	ሐይማኖት እና ልዩነቶች ፣ ሥርዓት	ማክበርና ማስከን ጠቀሜታ ተመራጩችና ላይ ገደብ ይ	ሐይማኖት እና ልዩነቶች ፣ ሥርዓጽ	ማክበርና ማስከን ጠቀሜታ ተመራጮችና ላይ ጨብ ይ
DSBI4	ይህንን ምዕራፍና ስታጠናቅ አስፈላጊነት	መሠረታዊ የሚከተሉትን ጥያቄዎች	ህንን ምዕራፍና ስታጠናቅ ስፈላጊነት	መሠረታዊ የሚከሠሉትን ጥያቄዎች
DSBI5	ሁለት በበላይነት ፣ ይህን ምዕራፍ	ደግሞ ራስን የክልሉ የልማት ፖሊሲ መሠረታዊ	ሁለት በላይነት ፣ ይህንን ምዕራፍ	ደግሞ ራስን የክልኩ የልማት ፖሊሲ መሠረታጂ
DSBI6	ጥቅሙስ ምንድን ነዉ ሥርዓት በተለይ እጅግ በጣም በምሳሌ	ሰነድ ነዉ እንዲጠቀሙና የፌዴራል በሀገር አቀፍ የኢኮኖሚን	ጥቅሙስ ምንድን ነው ሥቅዓት ብለሸ እጅግ በጣም በምሳሌ	ሰነድ ነው እንዲጠቀሙና ደፈዴራል በሀገር አቀፍ የኢኮኖሚን

5.4 Test Result

As we discussed in Section 4.4 and 4.5, apart from attribute of Braille dots such as centroid, orientation and area we used the two Eigen values, I_{11} and gradient angle. I_{11} is used to segment objects (dots) from the background. In addition, gradient angle is used to identify the segmented dots as recto and verso dots. Finally, after identification of dots as recto and verso, we formulate Braille cells in order to encode a Braille cell into a Braille code and then the Braille code is translated into a print text. In the experiment, we tried to test the performance of segmentation, identification and translation. Tables 5.3 to 5.5 show the performance of each phase.

Table 5.3: *Segmentation Accuracy of Dots*

No	Braille Documents	Ground Truth	Experiment		
		Number of Dots	Number of Non Overlapping Dots	Number of overlaps	Number of Dots in the Overlaps
1	DSBI1	70	39	13	31
2	DSBI2	203	100	32	103
3	DSBI3	273	187	36	87
4	DSBI4	210	139	27	72
5	DSBI5	236	169	27	62
6	DSBI6	360	235	52	125
	Total	1352	869	187	480
	Non Overlapping Dots %			64.3%	
	Overlapping Dots %			35.5%	

From the result shown in Table 5.3, we can see 35.5% of the dots (recto and verso) are overlapped. This infers, considering segmentation and identification of overlapping dots (Section 4.5) have significant effect on translation performance of Braille documents.

Table 5.4: *Identification Accuracy of Dots as Recto and Verso*

No	Braille Documents	Ground Truth		Experiment	
		Number of Recto Dots	Number of Verso Dots	Number of Recto Dots	Number of Verso Dots
1	DSBI1	32	38	32	38
2	DSBI2	99	104	100	103
3	DSBI3	142	131	144	129
4	DSBI4	120	90	121	89
5	DSBI5	88	148	89	147
6	DSBI6	158	202	160	197
	Total	639	713	646	703
Identification Accuracy		99.3%			

Table 5.4 shows identification accuracy of a dot as recto or verso, the designed system is tested with a total of 1352 dots (recto and verso). The system totally identifies 646 dots as recto from which 639 of them are correctly identified and 7 of them are wrongly identified, and the system correctly identifies 703 dots as verso. Using these information, we determine the identification accuracy of the proposed system using Equation 21.

$$Accuracy = \frac{correctly\ identified\ recto\ dots\ +\ correctly\ identified\ verso\ dots}{total\ number\ of\ dots}\ x\ 100\% \qquad (21)$$

This work which considers overlapping of recto and verso dots archives a remarkable identification accuracy of 99.3%.

Table 5.5: *Performance of Translation*

No	Braille Documents	Translation Performance Out of 100%			
		Wrongly Scanned		Correctly Scanned	
		Recto Page	Verso Page	Recto Page	Verso Page
1	DSBI1	63.6	58.3	100	100
2	DSBI2	60.6	51.5	96.3	96.7
3	DSBI3	57.1	51.1	94.7	94.7
4	DSBI4	67.5	68.5	88.6	97.0
5	DSBI5	57.5	56.3	97.4	95.0
6	DSBI6	64.2	50.8	93.5	93.8
Maximum		67.5	68.5	100	100
Minimum		57.1	50.8	88.6	93.8
Average Accuracy		61.8	56.1	95.1	96.2
Overall Accuracy		58.9		95.6	

Table 5.5 shows the translation performance of the proposed system. The table shows the translation accuracy of each document by page for two cases: correctly scanned and wrongly (reversed by 180^{o}) scanned documents. Both of them are calculated using Equation 22.

$$Accuracy = \frac{number\ of\ correctly\ translated\ charachters}{total\ no\ of\ chrachters\ considered} x100\%$$

Finally, from accuracy of each document by page, we determine the average accuracy for the documents by page and the overall translation accuracy of the proposed system. The result shows, the proposed system translates double sided Braille documents with accuracy of 95.6% when a Braille document is scanned in its correct direction.

5.5 Discussion

The proposed system has six components. However, we tried to evaluate three of them: Braille dot segmentation, Braille dot identification and recognition components independently. This helps us to compare our work (Braille dot identification component) with other attempts for other languages. Recent works Attempted by [8] and [31] , both have an average accuracy of 99% in identifying dots as recto and verso. However, both works don't consider overlapping dots. Our attempt that considers overlap of recto and verso dots has an average accuracy of 99.3% in identifying a dot as recto or verso, which is remarkably appreciable performance. Thus, recognition of double sided Braille documents using direction field and gradient field has better performance in segmentation and identification of recto and verso dots from the background than thresholding algorithms used in [8, 31].

As this work is the first in recognition of double sided Braille documents for Amharic language, we can't compare the translation (recognition) accuracy with other attempts. Previous attempts focused only on recognition of single sided Amharic Braille documents. The average translation accuracy of this attempt is 95.6% for correctly scanned documents, which is a remarkable performance as compared with the English character recognition system attempted in [25] with average translation accuracy of 98.7%. For wrongly scanned documents, the average translation performance is 58.9%; this tell us that we can define a threshold value to reverse wrongly scanned documents automatically, if the translation performance is less than the threshold. In our design we select the maximum value 68.5% as discussed in Section 4.7 as a threshold value. However, the threshold value can be changed depending on the Braille document recognized, because this value is found for clean double sided Braille documents. Therefore, it is advisable to take a document with better translation performance.

In this work, we tried to show the segmentation accuracy, which is 64.3 %. As compared to the Braille dot identification and translation accuracies, the segmentation accuracy is significantly small. This shows us, in double sided braille documents; even if there is a diagonal offset angle to prevent overlap, there are overlap of dots which are created during Braille embossing or writing process. This work shows 35.5% of the dots are overlapped. Therefore, in designing a system to recognize double sided Braille document, segmentation and identification of overlapping dots is vital. Hence, it improves the recognition process of double sided Braille documents.

Chapter Six : Conclusion and Future Work

6.1 Conclusion

In this study, an attempt has been made to design and implement double sided Braille recognition system for Amharic language. We adopted and designed different techniques for the different steps of the recognition process. Gaussian filters and their derivatives, direction field tensor are used as tools in removing noises and segmenting Braille dots from the background. Gradient field has differentiated recto and verso dots by counting gradient angles over the dots. Among the three features (bright, dark and mid gray) observed on Braille images, recto dots have large number of pixels with bright intensity or gradient angle value less than 180^o than verso dots, and verso dots have large number of pixels with dark intensity than recto dots or gradient angle value greater than 180^o. Braille dot features like centroid, orientation and area can also be used to identify overlapping dots as recto and verso.

We have attempted to estimate Braille cell attributes such as Braille dot height, dot width, vertical and horizontal space between Braille cells, Braille cell height and width. These values tend to be different depending on the resolution level used during image capturing. 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 in formulating Braille cells. Braille cells are formulated by using a sliding window designed using the Braille cell attribute. The Braille cells are encoded into a Braille code and then translated into their corresponding print characters using lookup tables, which contains list of Braille code values and the associated print characters.

Before the translation process we use concept of reflection to turn a page and rotation to reverse a page in case a Braille is scanned in the wrong direction. Turning the page helps the same translation and Braille cell encoding algorithms. Moreover, reversing wrongly scanned Braille documents automatically optimizes the translation performance.

The prototype we have designed for our system has been experimented against Braille documents from different places: Addis Ababa University Kennedy Library, Misrach center and Ethiopian Association for Blind Society. We have achieved an average accuracy of 99.3% and 95.6% in identification of dots and translation of the Braille into print text respectively. The small errors observed in identifying the dots are attributed to overlapping dots. For instance, two consecutive similar dots (recto) may overlap with other type of dots (verso); due to some stain effect a single dot is taken as two dots, overlapping of two dots treated overlap of three dots. Moreover, estimation of Braille cell attributes which depends

on resolution of the scanner also affects the encoding process that encodes the Braille cells into a Braille code. Hence, these affect the translation performance. To summarize, this work in general depicts a design and implementation of a double sided Braille recognition system for Amharic Braille documents.

6.2 Future Work

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.

- In order to enhance the performance of the translation, a post-processing activity can be incorporated at word level using Amharic dictionary or performing a sentence level analysis in correcting words contextually.
- Keeping the format of Braille documents is sometimes very important. For instance, the content in the document can be a poem. Therefore, considering the structure of the Braille document can be a further work.

References

[1] "http://www.iefusa.org/download/2012 IEF Annual Report.pdf," IEF Foundation, [Online]. Available: http://www.iefusa.org. [Accessed 12 10 2014].

[2] Yemane Berhan, Alemayehu Worku, and Abebe Bejiga, "National Survey on Blindness, Low Vision and Trachoma in Ethiopia," *National Blindness & Low Vision Survey,* pp. 1-6, 2006.

[3] Miftah Hassen, and Yaregal Assabie, "Recognition of Ethiopic Braille Characters," *In Proceedings of the 4th International ACM Conference on Management of Emergent Digital Eco Systems,* 2012.

[4] Ebrahim Chekol, "Recognition of Amharic Braille Documents," Unpublished Master Thesis, Faculty of Informatics, Addis Ababa University, Addis Abeba, Ethiopia, 2010.

[5] Teshome Alemu, "Recognition of Amharic Braille.," Unpublished Master Thesis, Faculty of Informatics, Addis Ababa University, Addis Ababa, Ethiopia, 2009.

[6] Shumet Tadesse, "Feature Extraction and Classification Schemes for Enhancing Amharic Braille Recognition System," Unpublished Master Thesis, Addis Ababa University, Addis Ababa, 2011.

[7] Lisa W., Waleed A., and Stephan H., "A Software Algorithm Prototype for Optical Recognition of Embossed Braille," in *the 17th conference of the International Conference in Pattern Recognition*, Cambridge, UK, August 2004.

[8] Abdul Malik S., Yosef O., Mohammed K., and Abdullah R., "An Arabic Optical Braille Recognition System," in *Proceedings of the First International Conference in Information and Communication Technology and Accessibility (ICITA 2007)*, Hammamet, Tunisia, 2007.

[9] Ng C., NG V., and Lau Y., "Regular Feature Extraction for Recognition of Braille," *In Proceedings of Third International Conference on Computational Intelligence and Multimedia Applications, ICCIMA'99,* 1999.

[10] Hermida F., Corbacho A., and Martín F., "A Braille OCR for Blind People," *In Proceedings of ICSPAT-96*, Boston (U.S.A.), 1999.

[11] Shreekanth T., and Udayashankara V., "A Review on Software Algorithms for Optical Recognition of Embossed Braille Characters," *International Journal of Computer Applications, ISSN: 0975 – 8887,* Vol. 81, November 2013.

[12] Ritchings R., Antonacopoulos A., and Drakopoulos D., "Analysis of Scanned Braille Documents," *Document Analysis Systems: World Scientific Publishing Company,* pp. 413-421, 1995.

[13] ኑበየ ልዑል ዮሃንስ, *ዘመነ ብርሃን,* አዲስ አበባ: ብርሃን ና ሰላም, 1954.

[14] Apostolos A., and David B., "A Robust Braille Recognition System," *Pattern Recognition and Image Analysis group, S. Marinai and A. Dengel (Eds.): DAS 2004, LNCS 3163,* pp. 533-545, 2004.

[15] Néstor F., Carlos M., Jesús B., and Miguel A., "Image Processing Techniques for Braille Writing Recognition," *EUROCAST, LNCS3643,* pp. 379-385, 2005.

[16] Million Meshesha, "A Generalized Approach to Character Recognition of Amharic Texts," Unpublished Masters Thesis, Addis Ababa University, Addis Ababa, 2000.

[17] በልዩ ትምህርት ቡድን ሥ/ት/ዝ/ጥናትና ምርምር ኢንስቲትዩት, "የብሬል ማስተማሪያ የመምህሩ መመሪያ," በልዩ ትምህርት ቡድን ሥ/ት/ዝ/ጥናትና ምርምር ኢንስቲትዩት, አዲስ አበባ, 1991.

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

[19] በኢትዮጵያ አይነ ስ ውራን ብሔራዊ ማህበር 12ኛ ዙር ስራ አስፈፃ ሚ ኮሚቴ የተቋቋመው የአማረኛ ብሬል አሻሻይ ኮሚቴ, "የ አማርኛ ብሬል አ ሻሻይ ኮሚቴ ዘ ገ ባ::," in *ታህሳስ 19 እና 20 ቀን 1995 ዓ.ም. ለተጠራው አገ ር አቀፍ ጉባኤ የ ቀረበ ::*, አዲስ አበባ, 1995.

[20] "http://www.dotlessbraille.org," dotlessbraille.org, 9 June 2002. [Online]. Available: http://www.dotlessbraille.org/braillewritingmethods.htm. [Accessed 12 March 2015].

[21] "brailleliteracy.weebly.com," [Online]. Available: brailleliteracy.com/methods-for-writing-braille.html. [Accessed 12 March 2015].

[22] Mennens J., "Optical recognition of Braille writing," *In Proceedings of the Second International Conference of the IEEE*, Tsukuba Science City, 1993.

[23] Dubus J., Benjelloun M., Devlaminck V., Wauquier F., and Altmayer P., "Image Processing Techniques to Perform an Autonomous System to Translate Relief Braille into Black-Ink, Called: Lectobraille," in *Engineering in Medicine and Biology Society, Proceedings of the Annual International Conference of the IEEE*, New Orleans, USA, 1988.

[24] François G. and Calders P., " The Reproduction of Braille Originals by Means of Optical Pattern Recognition," *In Proccedings 5th International Workshop on Computer Braille*, Heverlee, 1985.

[25] Antonacopoulos A. and Bridson D., "A Robust Braille Recognition System," *Document Analysis Systems VI, A. Dengel and S. Marinai (Eds.), Springer Lecture Notes in Computer Science, LNCS 3163,* pp. 533-545, 2004.

[26] Mennens J., Tichelen L., Francois G., and Engelen J., "Optical Recognition of Braille Writing Using Standard Equipment," *IEEE Trans. on Rehabilitation Engineering,* Vol. 2, pp. 207–212, 1994.

[27] Bigun J., Bigun T., and Nilsson K., "Recognition by Symmetry Derivatives and the Generalized Structure Tensor," *IEEE TPAMI 26 (2),* pp. 1590-1605, 2004.

[28] Yaregal Assabie. and Bigun J., "Offline Handwritten Amharic Word Recognition," *Pattern Recognition Letters, 32 (2011),* pp. 1089-1099, 2011.

[29] Thorstein S., "CIE Div 1, R1-47 Hue angles of Elementary Colours," International Commission on Illumination-CIE, Norway, 2004.

[30] Li J., Yan X., and Zhang D., "Optical Braille Recognition with Haar Wavelet Features and Support-Vector Machine," in *International Conference on ComputerMechatronics, Control and Electric Engineering (CMCE)*, 2010.

[31] Shreekanth T. and Udayashankara U., "An Algorithmic Approach for Double Sided Braille Dot Recognition Using Image Processing Techniques," *International Journal of Image Processing and Visual Communication ISSN (Online),* Vol. 2, No. 4, pp. 2319-1724, 2014.

[32] Santanu H., Abul H., Amina K., Debotosh B., and Mita N., "Development of a Bangla Character Recognition (BCR) System for Generation of Bengali Text from Braille Notation," *International Journal of Innovative Technology and Exploring Engineering (IJITEE),* Vol. 3, No. 1, pp. 5-10, 2013.

Annex A: Amharic Characters

Ge'ez	Ka'eb	Salis	Rab'e	Hamis	Sadis	Sab'e
ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ
ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ
ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ
ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ
ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ
ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ
አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ
ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ
ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ
ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ
ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ

ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ
የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ
ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ
ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ
ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ
ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ
ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ
ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ
ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ

Annex B: Amharic Lablization Characters

ኰ	ኩ	ኲ	ካ	ኴ	ጐ	ጒ
ጕ	ጓ	ጔ	ፏ	ቈ	ቍ	ቌ
ቋ	ቌ	ኈ	ኍ	ኌ	ኋ	ኄ
ኳ	ሟ	ሯ	ሳ	ሿ	ቧ	ቷ
ቿ	ኗ	ኟ	ኧ	ፉ	ዟ	ዧ
ጧ	ዷ	ጷ	ዃ	ቈ		

Annex C: Amharic Numerals

፩	፪	፫	፬	፭	፮	፯	፰	፱	፲	፳	፴	፵	፶	፷	፸	፹	፺	፻
1	2	3	4	5	6	7	8	9	10	20	30	40	50	60	70	80	90	100

Annex D: Amharic Punctuation Marks

Punctuation Mark/Symbol	Name
?	Question Mark
!	Exclamation Mark
( )	Parenthesis
“ “	Quotes
፣	Coma
።	Full Stop
፤	Colon
፥	Semi Colon
፦	Preface Colon

Annex E: Sample Lookup Tables for Consonants

Braille Code	Unicode	Symbol/Character
125	4613	ህ
123	4621	ል
126	4629	ሕ
134	4637	ም
234	4645	ሥ
1235	4653	ር
1456	4661	ስ
146	4669	ሽ
12345	4677	ቅ
12	4709	ብ
1236	4717	ቭ
2345	4725	ት
16	4733	ች
156	4741	ኀ
1345	4757	ን
346	4765	ኝ
12356	4773	እ
13	4781	ከ
236	4797	ኽ
2456	4813	ው
1256	4821	ዕ
1356	4829	ዝ
356	4837	ዥ
13456	4845	ይ
145	4853	ድ
245	4869	ጅ
1245	4877	ግ
23456	4901	ጥ

Annex F: Sample lookup Tables for Syllable Combinations

Braille Code	Unicode	Symbol/Character
12526	4608	ሀ
125136	4609	ሁ
12524	4610	ሂ
1251	4611	ሃ
12515	4612	ሄ
125135	4614	ሆ
12326	4616	ለ
123136	4617	ሉ
12324	4618	ሊ
1231	4619	ላ
12315	4620	ሌ
123135	4622	ሎ
12626	4624	ሐ
126136	4625	ሑ
12624	4626	ሒ
1261	4627	ሓ
12615	4628	ሔ
126135	4630	ሖ
13426	4632	መ
134136	4633	ሙ
13424	4634	ሚ
1341	4635	ማ
13415	4636	ሜ
134135	4638	ሞ
23426	4640	ሠ
234136	4641	ሡ
23424	4642	ሢ

Annex G: Sample Lookup Tables for Numbers

Braille Code	Arabic Numbers /Symbol	Ge'ez Numbers /Symbol
245	0	፩
1	1	፪
12	2	፫
14	3	፬
145	4	፭
15	5	፮
124	6	፯
1245	7	፰
125	8	፱
24	9	፲
123456	Ge'ez Number indicator	
3456	Arabic Number indicator	

Annex H: Sample Lookup Tables for Punctuations

Braille Code	Symbol
2	⁝
23	⁞
235	!
2356	()
236	?
25	:-
256	⁘
3	.
52	/
63	:

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 source of materials used for the thesis have been duly acknowledged.

Declared by:

Name: **Hassen Seid Ali**

Signature: _____

Date: _____

Confirmed by advisor:

Name: **Dr. Yaregal Assabie**

Signature: _____

Date: _____