ENHANCED OPTICAL BRAILLE RECOGNITION USING SLOPE DETERMINATION AND DOUBLE SIDED DOT DETECTION FOR GRADE 2 BRAILLE

A Thesis
Presented to the
Department of Information and Computer Studies
Faculty of Engineering
University of Santo Tomas

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Science in Computer Science

by

Martin Luis R. Faustino
Gian Gerard E. Libunao
Jessica Cassandra T. Magbitang

March 2010

APPROVAL SHEET

Thesis Title: ENHANCED OPTICAL BRAILLE RECOGNITION USING SLOPE DETERMINATION AND DOUBLE SIDED DOT DETECTION FOR GRADE 2 BRAILLE

Proponents:

1. **Martin Luis R. Faustino**
2. **Gian Gerard E. Libunao**
3. **Jessica Cassandra T. Magbitang**

In partial fulfillment of the requirements for the degree of ***Bachelor of Science in Computer Science***, the thesis mentioned above, has been adequately prepared and submitted by the proponents. This thesis was duly defended in an oral examination before a duly constituted tribunal with a grade of

_____
Ms. Charmaine Salvador-Ponay
Panel Member

_____
Assoc. Prof. Perla P. Cosme
Panel Member

_____
Mr. Chester Arvin R. Morales
Panel Member

_____
Asst. Prof. Vergil V. Reyes
Thesis Adviser

Accepted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

_____
Engr. Alex A. Santos
Chairperson
Information and Computer Studies Department
Faculty of Engineering

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENT

First of all, the proponents would like to thank God for giving them the knowledge, opportunity and strength to push this through.

The proponents would like to give their deepest appreciation to their very supportive parents and friends who never fail to give support in everything the proponents do.

The proponents would just like to acknowledge their resource person, Ms. Lorrie Barboza for being a part of the research process.

They would also like to acknowledge Mr. Vergil V. Reyes for the guidance that he had given to the proponents. He is the one, who painstakingly helped the proponents, starting from looking for the CS problems, while undergoing the research study, until from the preparation of the documentation of the thesis.

Moreover, they would like to acknowledge the author of the previous study, Alexander Lao for sharing his time and ideas.

Also, the proponents would like to thank the panel members namely Perla P. Cosme, Chester Arvin R. Morales and Charmaine S. Ponay for spending their time reading and understanding the proponents' research study and for challenging the students to think of topics that will be a good contribution in the field of Computer Science.

DEDICATION

First of all, this thesis is dedicated to God, who never fails to who is always there in good and bad times. Also, this thesis is dedicated to our ever supportive families who pushed us to go beyond our fears and thought us how to be strong and courageous. This thesis is dedicated to all our friends who never fail to support us in any way. Lastly, this thesis is dedicated to the students and teachers of the Resources for the Blind who never fail to give us all the information we need in the research process.

ABSTRACT

Braille is a communication system that's used by the blind to read and to write. The first system was created by Charles Babier and was intended to be use by the French military at war. The Babier's ruler as what they've called the system was complex to use and the problem of the system was spotted by Louis Braille who was a blind man at his teenage years. Louis Braille then modified Babier's ruler and made it simple and easy to use. Today, it has been in existence for almost two centuries.

Since technology is rapidly arising, researchers are finding ways to help the visually impaired in many ways. One of the most important purposes is to be able to convert Braille characters to regular text for preservation, duplication purposes and many more. As of now there are already several algorithms that have the capability to convert Braille to regular text but with limitations. As researchers, the proponents discover some problems on the algorithms which can be corrected. One of the problems on the existing algorithms on how the system to be developed will recognize slanted or skew Braille images. In addition, the problem on how it will recognize Grade 2 double-sided Braille documents.

To solve the problems mentioned, the proponents reinvented the algorithms and developed a system that clearly showed the whole operation. The system developed is composed of three different parts. The preprocessing part which has five levels: grayscale, thresholding, noise reduction, back side filtering and de-skewing. The segmentation process is the one responsible in partitioning the Braille characters into regions and collecting the raw data on the image. Lastly, the recognition part which is the one that compares the gathered data into the database and return the equivalent texts.

CHAPTER 1

THE PROBLEM AND ITS BACKGROUND

A. Introduction

A Braille character or cell which is depicted in Figure 1.1 is composed of six dots arranged in two columns and three rows. It has sixty-four possible combinations that contain letters, numbers and special characters [Lao,2009].

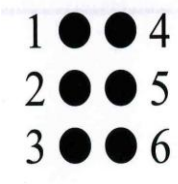


Figure 1.1
Numbered positions of dots on a Braille cell

Braille is a method of reading and writing text through touch, rather than sight. It is mainly used by those with impaired vision; however, sighted people can read Braille as well. There are many reasons for this, especially for those with a blind or visually impaired person in their household. There are many types of literary Braille. The most commonly taught and used is Grade 2 Literary Braille (see Figure 1.2), which is a space-saving alternative to Grade 1 Braille. In Grade 2 Braille, a cell can represent a shortened form of a word.

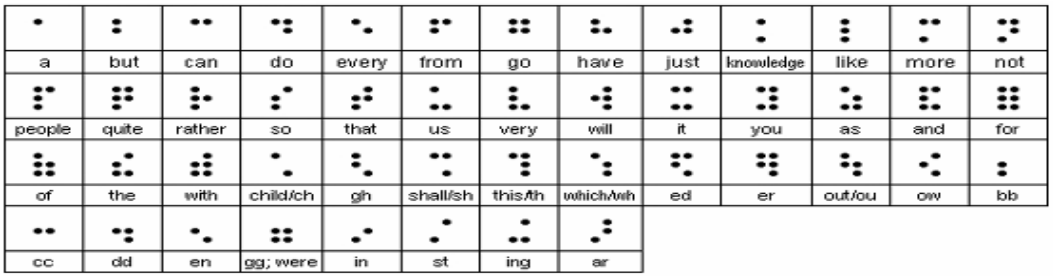


Figure 1.2
Example of Grade 2 Braille

B. Background of the Study

Nowadays, computers are very helpful and important. It help the people do a lot of things easier, from school work to office work. These computers are important due to the data it process and store because those significant bits are the computers' need in order to accomplish something. Data contain information that are stored inside the computer's hard disk which are very useful since it can easily be kept without the worry of deteriorating itself even as time goes by.

In the early years, important documents are being converted to data or being digitalized for an easier accessibility and storage but converting hard copies to their equivalent data form can be very hard and time consuming since one would need to retype everything in a given document. With the technology and OCR (Optical Character Recognition) these tasks are done easier. Through OCR, a scanned image of the document can be used to translate the document page to a word processor file, thus eliminating the process of retyping every letter in the document. OCR is the basis of other recognition systems that also deal with digitalizing documents, one example is OBR (Optical Braille Recognition). OBR is needed because of the need to digitalize Braille documents that are slowly deteriorating. Those Braille documents which are much more bulky than text only documents contain very important information that have been gathered and those Braille documents would often be needed by other people so digitalizing them would help a lot of people that would need it and also reading a Braille document would require a special course to be read thus those who do not know how to read would often just disregard Braille documents

in their research. Through OBR, Braille documents would be digitalized and also be translated to common English.

To describe a Braille document, it has a standard size of 11 by 11.5 inches. A Braille document may be single or double sided. This study will focus on double sided Braille documents. A double sided Braille document has 2 sides, the dots facing the reader is called the protrusions or called as the embossed side. The other side which is called as the depressed side has the dots called depressions which are the ones facing the other side

This study is based on a previous study on Optical Braille Recognition entitled Optical Braille Recognition Using Modified Character Isolation Box and Pattern Generation. The previous study deals with the recognition of Grade 1 Braille using Character Isolation Box and Pattern Generation algorithm. Pattern Generation was modified by the proponent to solve the problem in recognizing the Braille characters. In implementing his study, the proponent went through image preprocessing first. In the preprocessing phase, the image underwent noise reduction and thresholding. The image skews were also corrected to preserve the data in the scanned image. However, the cropping process of the de-skewing phase sometimes fails due to some poor edge detection problems.

C. Statement of the Problem

1. How will the system interpret the Braille document if upon scanning, the document/page is already skewed?

The absence of an input de-skewing algorithm leaves very little margin of error in terms of the angle of the scanned image input, the segmentation algorithm

immediately attempts to cut the image into equal and usable segments. If the image is skewed, the image will be cut into useless segments because the associated Braille dots that should go together in one segment will most likely end up in separate segments. The previous study lacks an input de-skewing process that comes before the segmentation algorithm.

2. How will the algorithm for character recognition identify it as a Grade 2 Braille?

Since Grade 2 Braille is used and the previous study is limited to Grade 1 Braille, there is a need to develop a system that is able to accommodate Grade 2 Braille. The previous study does not come with a grouping function that takes into consideration the preceding and succeeding characters, it only maps one cell to one character and does not recognize the arrangement of the letters. Because of this, the existing program can only identify one Braille character at a time thus limiting it to the system to recognizing only Grade 1 Braille while Grade 2 Braille is made up of at least 2 characters to form shortened words and is governed by very different contraction rules.

3. How will the system differentiate the embossed side from the depressed side since they both produce shadows that is recognized a character?

Since the current system applies thresholding and then commences the recognition of the dots based on a cluster of black pixels, a scanned image of a double sided Braille would not be properly interpreted due to the impressions caused by the other side of the document. These impressions would be identified as dots in the

faced side of the document in effect this would produce misleading dots that could be interpreted as a part of the Braille cell that is being read. The previous system is unable to differentiate the embossed dots from the impressed dots.

The Braille system has six possible places of dots (see Figure 1.3). For example a Braille cell that is equivalent to letter A (see Figure 1.4), the only dot that should be visible is the 1st dot, however the Braille cell could be misinterpreted as a Braille cell equivalent to letter B (see Figure 1.5). If there happens to be an impression on the 2nd dot on the Braille cell, that impression will be interpreted as a dot, the Braille cell would be interpreted as a Letter B instead of A.

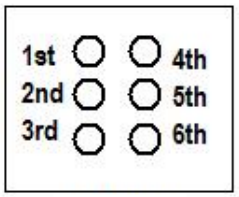


Figure 1.3
6 possible places of dots

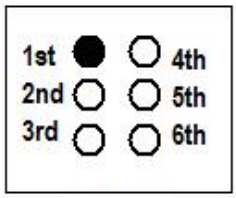


Figure 1.4
Braille cell equivalent
to letter A

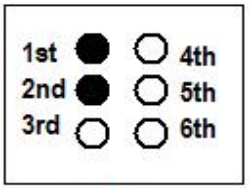


Figure 1.5
Braille cell equivalent
to letter B

D. Objectives of the Study

i. General Objective

The primary goal of the study is to modify the existing segmentation and recognition approaches to accurately and correctly translate scanned Grade 2 Braille documents into English text.

ii. Specific Objectives

1. To modify the existing image recognition algorithms to recognize Grade 2 Braille;

2. To create a stand alone application that interprets a double sided and Grade 2 Braille document; and

3. To improve de-skewing process of the scanned image.

E. Significance of the Study

Many people can benefit from this thesis. Everyone who works with blind people and does not read Braille can benefit from using the OBR. For example: teachers who do not read Braille, people who work in public organizations, those who communicate with the Blind individuals and people who work in computerized Braille libraries [Neovision]. In school, normal individuals like students and teachers may gain knowledge on Braille to help them communicate with the visually impaired ones and vice versa. Since Braille documents are stored in computers, there is a big chance that many people may access those Braille documents therefore, those people may not buy Braille documents because as everyone knows, buying Braille documents is a little bit costly and tiresome. Grade 2 Braille is actually an addition to Grade 1 Braille. The study can reduce the computer memory used by Grade 1 Braille documents because some Grade 2 Braille characters are abbreviated and combined to represent some words. The use of contractions of Grade 2 Braille reduces the bulk of the material and speed up the reading process [Wilson, 2005].

F. Scope and Delimitation of the Study

Scopes

- The system can only scan Grade 2 Braille documents.
- The system is focused on double sided Braille documents.
- The system can interpret a 6-dot Braille cell

- The system can accept lowercase Braille characters.
- The system can cover the Grade 2 Braille contractions.
- The flatbed scanner must be set into 200 dpi.

Limitations

- The system can only support JPEG & 24-bit BITMAP image formats.
- The system only accepts English alphabetic Braille and not other type of Braille documents like musical Braille or other languages like Arabic, Chinese and French.
- Numeric Braille, uppercase Braille letters and special Braille characters are not covered.
- The system can only translate Braille word or short phrase/s to its equivalent English word/phrase that is cut or cropped from the scanned page of the Braille document.
- The Braille document to be scanned should be in excellent condition.
- The Braille document to be scanned should not contain foreign objects.
- The Braille document to be scanned should not have dots that are very close/ joined with each other.
- The system can only correct slanted or skewed images at an angle less than or equal to 5.5 degrees.
- A single character input should have at least 1 dot per row and column.
- Only Braille documents that have a standard size of 11 x 11.5 inches can be used as source.

CHAPTER 2

RESUMÉ OF RELATED LITERATURE AND PROFESSIONAL STUDIES

A. Related Literature

De-skewing

Image de-skewing is the process of straightening an image that has been scanned or photographed crookedly- that is an image that is slanting too far in one direction or one that is misaligned. Skew is an artifact that can occur in scanned images because of the camera being misaligned, imperfections in the scanning or surface, or simply because the paper was not placed completely flat when scanned. An example of a skewed image is shown in Figure 2.1. [Lao, 2009]

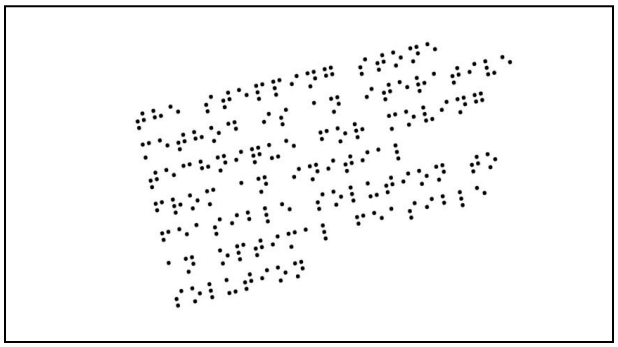


Figure 2.1
Skewed image tilted 13 degrees to the left

Double Sided Braille Documents

The scanned Braille page appears with a mid-gray background, and for each protrusion and depression, a highlight and shadow pair is present along the scanning direction (depressions are only present in double-sided documents). The order in which the shadow and highlight appear for each dot depends upon the model of the scanner involved. Some models represent protrusions as shadow areas over highlight

areas while other 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 [Dengel et al., 2004] An example of a typical scanned double-sided Braille document can be seen in Figure 2.2.

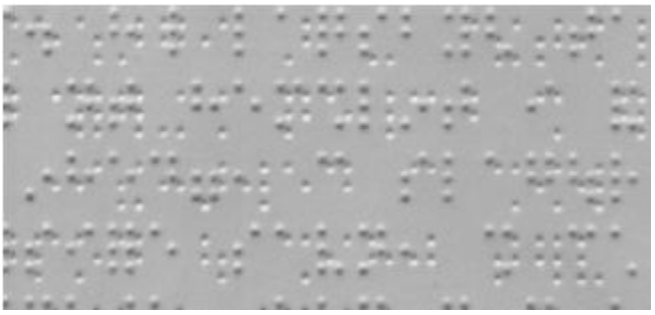


Figure 2.2
An example of scanned double-sided Braille document

Grade 2 Braille

Grade Two English Braille (America Edition) has 250 symbols for: letters, punctuation marks, composition signs, numerals, contractions, single-cell words, and short-form words [Charlie Web, 2005]. Grade 2 Braille was developed to reduce the size of books and make reading quicker. Other symbols are used to represent common letter combinations, for example in English 'OW', 'ER', and words such as 'AND' and 'FOR'. Combinations of two symbols are also used to represent some words, for example: 'THROUGH'. Some characters may change their meaning, depending on how they are spaced. [Sablé,2006]

Preprocessing

Since there are only three classes of useful information (shadows, light areas and background), a preprocessing step to reduce the gray levels in the image is necessary. To cope with significant (in many cases) variations in lightness across the

whole image, a local adaptive thresholding method was introduced. The method works by dividing the image into 32x32 pixel regions (the window size is experimentally derived) and assesses whether each region contains whole dots, highlight(s) only, shadow(s) only, or just background. This assessment is based on a comparison of sets of ranges of gray levels observed in the region against equivalent ranges that are expected when a particular feature (dot, highlight or shadow) is present or not. In each of those four different cases, a different threshold (or a fixed value in the case of background regions) is applied to the pixels of the region.

The resulting image will have only black regions (corresponding to the shadows), white regions (corresponding to the highlights) and mid-grey (the majority, corresponding to the background). An example of a region of the image after this stage is shown in Figure 2.3.

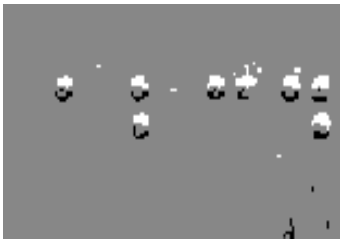


Figure 2.3
Example of a result of preprocessing

Thresholding

During the thresholding process, individual pixels in an image are marked as “object” pixels if their value is greater than some threshold value (assuming an object to be brighter than the background) and as “background” pixels otherwise. This convention is known as threshold above. Variants include threshold below, which is opposite of threshold above; threshold inside, where a pixel is labeled "object" if its value is between two thresholds; and threshold outside, which is the opposite of

threshold inside [Shapiro, et al., 2001]. Typically, an object pixel is given a value of "1" while a background pixel is given a value of "0." Finally, a binary image is created by coloring each pixel white or black, depending on a pixel's label. [www.wikipedia.com]

In an image (see Figure 2.4) that already underwent the process of thresholding, all image details are lost. The source image is scanned one pixel at a time and the pixels above the certain threshold are transformed into black pixels in their respective positions while the pixels below the threshold are transformed into white pixels, in effect the image is left with two pixel modes, black and white. A cluster of black pixels given a specific volume can represent a dot in a Braille cell.

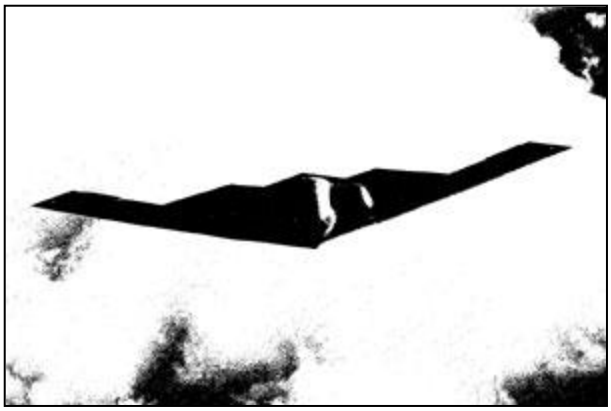


Figure 2.4
Example of thresholded image

B. Related Studies

Image pre-processing is an essential step during which errors that occurred while the images were taken are eliminated. Errors include 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. The algorithms used

differ from one system to another depending on the classification approach followed by researchers and developers. [Al-Salman et al., 2007]

In Optical Braille Recognition, the appropriate pre-processing algorithm must be chosen for the problems that occur in scanning a Braille document. The pre-processing algorithm in our study should focus on rotating skewed images.

There is a wealth of books and documents that only exist in Braille that, as with other rare/old documents, are deteriorating and must be preserved (digitized). Also, there is an everyday need for duplicating (the equivalent of photocopying) Braille documents and for translating Braille documents for use by non-Braille users. The latter application is quite important, as it forms the basis for written communication between visually impaired and sighted people (e.g., a blind student submitting an assignment in Braille). [Antonacopoulos and Bridson, 2004]

A percentage of these Braille documents are in grade 2, which cannot be processed by the previous study.

One of the first approaches to use a flatbed scanner to appear in the literature is that of Ritchings et al. It is applied to both single and double-sided Braille documents, scanned at 100dpi at 16 grey levels (for economic reasons at the time). It performs few image based operations and it is relatively flexible to skew as it identifies Braille characters based on character-region search. [Antonacopoulos and Bridson, 2004]

Attempts had been made to optically recognize embossed Braille using various methods. In 1998, Dubus and his team designed an algorithm called Lectobraille which translates relief Braille into an equivalent printed version on

paper. Since then, research has built on knowledge of image processing techniques towards the goal of Braille to text translation. In 1993, Mennens and his team designed an optical recognition system which recognized Braille writing that was scanned using a commercially available scanner. The result was satisfactory with reasonably well formed Braille embossing. However, the system cannot handle deformation in the dot grid alignment. In 1999, Ng and his team approached the problem using boundary detection techniques to translate Braille into English or Chinese. The recognition rates were good, however no mention was made of grid deformed input, nor its efficiency. In 2001, Murray and Dais designed a handheld device which handles the scanning as well as the translation. Since the user is in control of the scanning orientation, and only a small segment is scanned at each instance, grid deformation is not a major concern, and a simpler algorithm was used to yield efficient, real-time translation of Braille characters. In 2003, Morgavi and Morando published a paper where they described the use of a hybrid system using neural network to solve the recognition problem. The paper also provides a means of measuring accuracy in Braille recognition, and the results show the system can handle a larger degree of image degradation compared with the algorithms that use more conventional and rigid image processing techniques. [Wong et al., 2004]

This reveals the evolution of Optical Braille Recognition through the years. The different pros and cons of these previous studies showed which problems our study might also face. This gives us a wide array of algorithms that can be potentially useful.

Horizontal illusory clues originate from the arrangements of characters into words and lines. Henceforth we shall indicate with horizontal the clues belonging to this dominant direction. The algorithm proposed in this paper to extract the horizontal illusory lines is summarised as follows. A pre-processing stage binarizes the input image, turning it into blobs representing either single characters or (portion of) words or lines, depending upon the font size and the resolution considered. These blobs are divided into elongated (major axis longer than thrice the minor axis) or compact. A pairwise saliency measure is computed for pairs of neighbouring blobs that represent how likely they are to be part of a text line. A network is then built using the blobs and their associations. The network then transverses to extract salient linear groups of blobs which constitute the illusory horizontal clues. Isolate elongate blobs are also considered as individual clues. In the following sections we shall describe these stages is more detail. [Pilu, 2001]

This study deals with de-skewing English alphabet letters, the concept can also be used in de-skewing an already skewed Braille document.

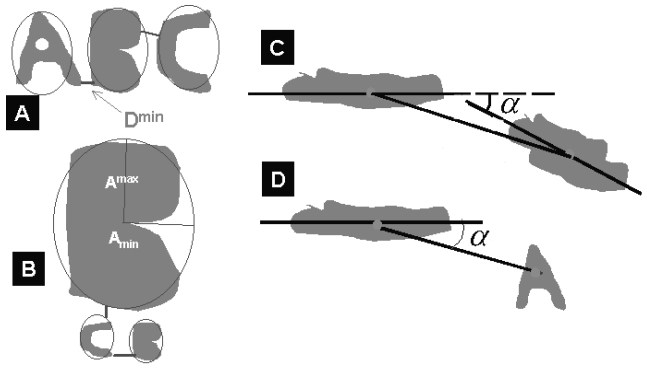


Figure 2.5
Illustration of the quantities used for the determination of the pairwise blob saliency

The reduction of noise (Figure 2.7) is valuable in correctly recognizing the Braille dots. Figure 2.6 shows a sample Braille image that underwent thresholding. From there, the image noises are more visible so it should undergo the process of noise reduction to be able to recognize the Braille dots correctly. Also, correcting the image skews is beneficial to preserve the data in the scanned image. However, the cropping process of the de-skewing phase sometimes fails due to some poor edge detection problems. The problem of recognizing the Braille characters was also solved by modifying the pattern generation algorithm. [Lao, 2009]

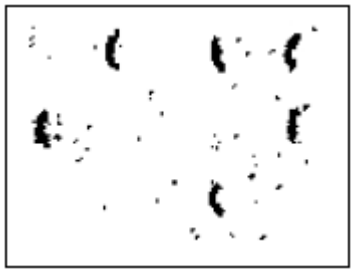


Figure 2.6
Scanned Braille image with noise

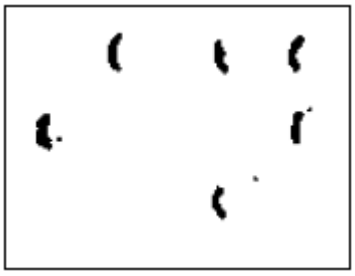


Figure 2.7
Noise reduction of Braille scanned image

1. Summary of Related Studies

Related Study	Description
An Arabic Optical Braille Recognition System [Al-Salman et al., 2007]	This project looks at developing a system to recognize an image of embossed Arabic Braille and then convert it to text. It particularly aims to build fully functional Optical Arabic Braille Recognition system. It has two main tasks, first is to recognize printed Braille cells, and second is to convert them to regular text. Converting Braille to text is not simply a one to one mapping, because one cell may represent one symbol (alphabet letter, digit, or special character), two or more symbols, or part of a symbol. Moreover, multiple cells may represent a single symbol. The project deals with a double sided Arabic Braille. One problem that the proponents tend to solve in the study is the de-skewing of the Braille document. The study yielded an accuracy rate of 98%.

Related Study	Description
A Robust Braille Recognition System [Antonacopoulos and Bridson, 2004]	This paper describes a new system that recognizes Braille characters in scanned Braille document pages. Unlike most other approaches, an inexpensive flatbed scanner is used and the system requires minimal interaction with the user. A unique feature of this system is the use of context at different levels (from the pre-processing of the image through to the post-processing of the recognition results) to enhance robustness and, consequently, recognition results.
A Software Algorithm Prototype for Optical Recognition of Embossed Braille [Wong et al., 2004]	This paper proposes a software solution prototype to optically recognize single sided embossed Braille documents using a simple image processing algorithm and probabilistic neural network. The output is a Braille text file formatted to preserve the layout of the original document which can be sent to an electronic embosser for reproduction.

Related Study	Description
Deskewing Perspectively Distorted Documents: An Approach Based on Perceptual Organization [Pilu, 2001]	This work deals with the recovery of illusory linear clues from perspectively skewed documents with the purpose of using them for rectification. The computational approach proposed implements the perceptual organization principles implicitly used in textual layouts. The numerous examples provided show that the method is robust and viewpoint and scale invariant.
Optical Braille Recognition Using Character Isolation Box and Pattern Generation [Lao, 2009]	This study deals with recognition of Grade 1 Braille by applying the Character Isolation Box algorithm and Pattern Generation algorithm. The main concept is to cut the whole Braille document into Braille cells that can be recognized by Pattern Generation algorithm. The study yielded an accuracy rate of 85 percent.

Table 2.1
Summary of Related Studies

CHAPTER 3

RESEARCH DESIGN AND METHODOLOGY

A. Definition of Terms

The proponents would like to define some terms used in the study to help the reader understand the study.

a. Braille System– a method that is widely used by blind people to read and write

b. Character Isolation Box – algorithm that cuts the image into segments that contain Braille character

c. Depression or Back Side Dot – flat Braille dot in a Braille document

d. Dpi (dots per inch) – a measure of printing resolution

e. Double sided Braille document – also known as contracted, inter-point or literary Braille. It has protrusions on both sides of the document page.

f. Grade 1 Braille – a direct, one to one substitution of normal print letters for letters from the Braille alphabet

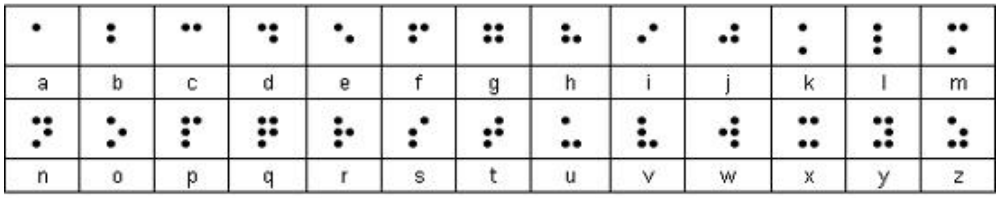


Figure 3.1
Grade 1 Braille Alphabet

g. Grade 2 Braille – a shorter form which makes reading and writing Braille much faster with the use of contraction rules

h. Image Skew– due to incorrectly scanned documents, the result of the image that has been scanned is sometimes slanted or skewed in an angle

i. Jpeg – commonly used method of compression for photographic images

j. Optical Braille Recognition – (OBR) is a Windows software package that allows to interpret single and double sided Braille documents with a standard scanner

k. Pattern Generation – a technique used in examining the presence of the dot in their possible positions/ predefined regions

l. Protrusion – raised or bulged Braille dot

m. Single sided Braille document – has protrusions on one side of the document page.

n. Slope –describes the steepness, incline, or grade of a line

o. Thresholding – is used to segment an image by setting all pixels whose intensity values are above a threshold to a foreground value and all the remaining pixels to a background value

B. Hypothesis

By improving the pattern generation algorithm to recognize Grade 2 Braille and successfully converting them to text documents, through adding an additional de-skew detection to make the system more flexible in the image inputs of the user and lastly to modify the segmentation of the current system to allow the input of double sided Braille because this would be the expected formats of published Braille documents.

C. Basic Assumption

The proponents assume the requirement for Braille recognition. The person assigned to scan the Braille document is assumed to have expertise in Braille and computer. The person is assumed to have knowledge in determining levels of Braille

and the front and back pages of the Braille document. The person is also assumed to have knowledge in scanning Braille documents and storing images. The system only accepts six Braille dots and assumed to interpret one row of Braille characters at a time. It is assumed that the scanned Braille documents are in Grades 1 and 2 alphabetic Braille and not in musical Braille. Other Braille languages aside from English are not accepted.

D. Research Design and Methodology

1. Specification of Research Data

The proponents' point of interest in this research was the limitations of the previous of the previous thesis because of those limitations; the system was far from being used in digitalizing Braille documents. After reading related documents, the proponents gain knowledge of certain solutions that the would take the previous on the next level such as improving then de-skewing, allowing Grade 2 Braille and double sided Braille documents as inputs.

2. Sources of Data

The proponents' sources come from books, online journals, internet and previous studies like an unpublished thesis on Optical Braille Recognition.

a. Books

The proponents are able to find books regarding image recognition and processing that can be helpful to know the methods and procedure.

b. Internet

The proponents are able to acquire much information in the internet. Most are whitepapers which contain discussions of the study and

explanations on the techniques used in the process of Braille recognition. The proponents also find some articles related to Optical Braille Recognition.

c. Previous Studies

The proponents are able to acquire many related studies that are done already. Those studies use many different approaches and procedures. With these researches, the proponents are able to determine the advantages and disadvantages of the algorithms used and plan to enhance those.

d. Resource Person

The resource person of the proponents is Miss Lorrie Barboza. She is a teacher, specifically for the blind. She teaches in the Resources for the Blind located in Cubao, Quezon City. Moreover, she is a chief Braillist in that organization.

3. Description of Research Methods

1. Information Gathering – The proponents find facts and other previous studies that can be used in the study.

2. Research Evaluation – The proponents study the facts gathered and try to know what to improve and add in the study, given those previous studies that are done before.

3. Identification of the Research Study – With the researches that the proponents are able to find, the proponents now identify the problems that occur in the previous studies which they want to focus on. Those problems that are image skewing, character recognition for Grade 2 Braille and

differentiation of the embossed side from the impressed side since the Braille document is double-sided.

4. Limitation of the Study – Optical Braille Recognition is a broad topic; the proponents limit the study in some areas like covering only until Grade 2 Braille, some contraction rules for Grade 2 Braille and recognition of a number of characters only at a time.

5. Outline of the System – This is also the creation and searching for algorithms. With the related studies that the proponents are able to find and study, they now combine, generate algorithms and sketch the system.

6. Code Formation for the System – The proponents convert the algorithms or methods applicable into program codes.

7. Testing and Debugging of the System – Testing and debugging is necessary to assure that the system will run, recognize Braille characters and converts those Braille characters in English text.

8. Maintenance of the System – The system should be maintained fully by updating the system since there is no system that is bug-free.

9. Implementation – At this point, the software created can be used not only by the visually impaired individuals but the normal ones too. The software created is also used to duplicate and preserve Braille documents.

CHAPTER 4

PRESENTATION, ANALYSIS AND INTERPRETATION OF DATA

A. System Architecture

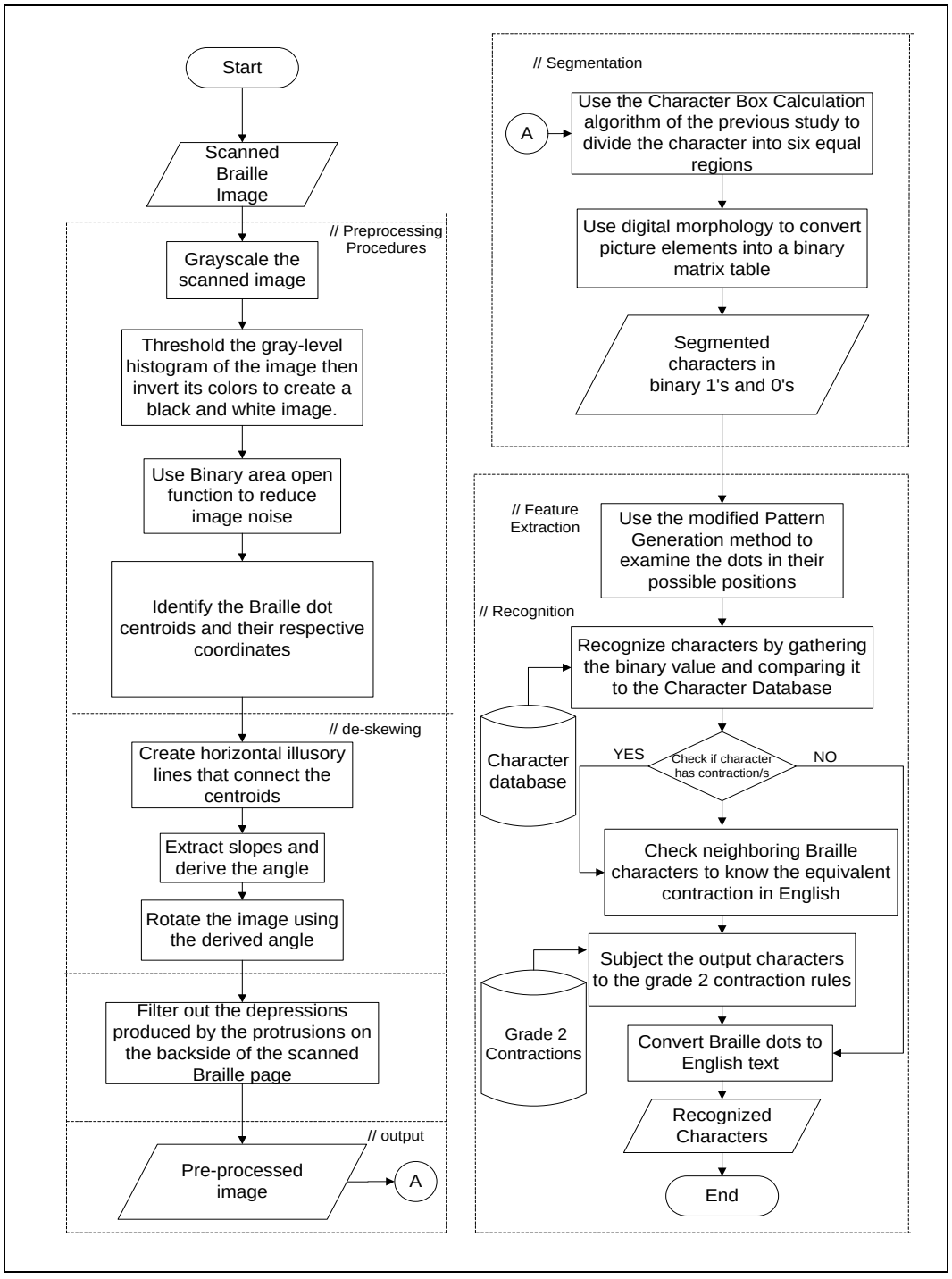


Figure 4.1
Proposed System Architecture for Optical Braille Recognition System

Based on the proponents' research and knowledge gained, the proponents come up with the proposed system architecture which consists of processes that would work together to implement the overall system.

Figure 4.1 shows the planned system architecture for the Optical Braille Recognition System. A Braille document should be scanned first, this would be the input image to the computer to digitalize it and also convert it to a text document. Next, the image would undergo the preprocessing procedures; this would include image grayscale, thresholding, noise reduction, de-skewing the image and filtering the depressions caused by the protrusions. The de-skewing process works by identifying the centroids of each Braille dot and its respective coordinates. It will use a series of horizontal lines as defining variables in finding the tilted angle that needs to be corrected. Preprocessing procedures are required to improve the quality of the image. To filter the depressions of the Braille document, the Braille centroids are used to determine the location of pixels that correspond to the depressions.

After the image undergoes preprocessing procedures, the image is much clearer and enhanced. This image now undergoes segmentation. Segmentation is needed to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. The Segmentation would have two sub phases one is the Character Isolation box. This will be the one responsible in determining the height and width of a single Braille character. After segmenting each character it would now be subject to Digital Morphology. This would produce a series of 1's and 0's or binary data that would be based on the Braille characters protrusions produced by the Character Isolation Box.

Shortly after, the modified Pattern Generation phase would start in the feature extraction. It will be the part wherein the system would examine each Braille character's dot possible positions. This phase can determine if the dots recognized are part of the correct side or the reverse one. The Pattern Generation is used to identify the parts in the Braille cell that contains dots then converts to binary data that will be used to match in the database of Braille characters.

Afterwards, the image now undergoes recognition. This phase translates the Braille character/s to English equivalent letter/s. It will ensure if that Braille character has a contraction equivalent. If there is, the neighboring characters should be verified to get the equivalent contraction in English. After checking, the characters should be validated by the Grade 2 Braille Contractions. On the other hand, if there is no contraction for that Braille character, that character is converted directly to English text. The characters would now produce English words in a text document.

B. System Development Tools

The proponents use Matlab for the entire process. Matlab stands for Matrix Laboratory which is a high level language created by Mathworks. Matlab is is a high-level language and interactive environment that enables you to perform computationally intensive tasks faster than with traditional programming languages such as C, C++, and Fortran. It provides a number of features for documenting and sharing people's work. It provides toolboxes which are used to solve specific problems like complex mathematical formulas. It can also import Java and C++ classes at the same time. The proponents also use the Image Processing and Image Acquisition toolboxes to accomplish some required image preprocessing phases.

C. Algorithms

De-skewing Algorithms

These algorithms are for the system to interpret the Braille document is skewed upon scanning it.

a. Computational Approach

When capturing a document with a hand-held camera, it is common to have an output image (Figure 4.2) that is perspectively distorted.

Figure 4.2
Illustration of perspective de-skew in documents and the linear clues

The algorithm done is a passive way of determining the illusory clues (such as text lines and paragraph margins) that can be used, along with clues such as document edges if available, to rectify the image in order produce an upright, undistorted document. The approach is based on a computational implementation of perceptual organization principles from text perception and done so using saliency measures and simple geometric reasoning.

Preprocessing stage binarizes (Figure 4.3) the input image, turning it into blobs (Figure 4.4) representing either single characters or (portion of) words or lines, depending upon the font size and the resolution considered.

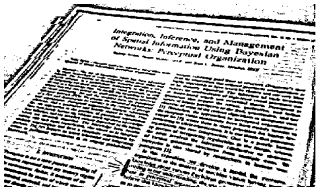


Figure 4.3
Binarized image

Figure 4.4
Groups of compact blobs

These blobs are divided into elongated (major axis longer than thrice the minor axis) or compact (Figure 4.5).

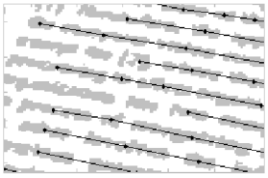


Figure 4.5
Groups of elongated blobs

b. Slope Determination Algorithm

All centroids are interconnected to form a network of lines. The slope of these lines is then used to filter out less useful lines. Lines that have slopes that do not fall in between -1.0 and +1.0 are considered insignificant and eliminated. To further filter out the less significant lines, the mode of the slopes is obtained. The mode of the slopes is considered to be the slope of the line that describes the angle of inclination. The angle of inclination is then derived by using the arctangent mathematical function to convert the slope into its equivalent angle.

$$\text{Slope} = \frac{Y2 - Y1}{X2 - X1}$$

Figure 4.6
Formula for Slope Determination

$$\text{Angle} = \arctan(\text{Slope})$$

Figure 4.7
Formula to convert slope to Angle

Double Sided Dot Detection Algorithm

This algorithm is used to detect all the Braille dots in the Braille document and then differentiates the embossed dots from the impressed dots. The first step in this

algorithm is to convert the scanned image in to grayscale mode so the pixel gray values only range from 0 to 255 where 0 is black and 255 is white and every value in between are different shades of gray(Figure 4.8).

Figure 4.8
Gray values

The light and shadow orientation is then identified by using the pixel coordinates and their respective gray values. High gray values result in a light pixel and low gray values result in a shadow pixel.

When the shadow pixel cluster is above the light pixel cluster (Figure 4.11) then the Braille dot is classified as impressed (Figure 4.9), however if the light pixel cluster is above the shadow pixel cluster, (Figure 4.12) then it is classified as an embossed Braille dot (Figure 4.10).

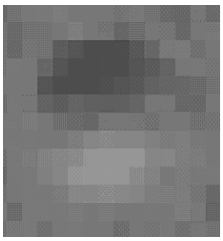


Figure 4.9
Impressed Braille dot in Grayscale

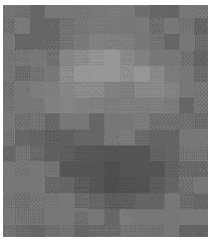


Figure 4.10
Embossed Braille dot in grayscale

111	111	107	114	114	113	115	119	115	111	107	105	114	113
115	116	104	107	112	113	116	115	118	115	111	114	112	104
110	114	115	91	99	78	81	74	77	80	90	104	111	114
114	113	114	95	87	82	82	75	73	87	82	96	108	108
104	112	80	85	80	85	84	90	92	95	96	98	112	112
114	113	90	94	97	88	82	75	94	87	82	96	108	108
104	112	109	89	88	85	91	90	92	101	96	100	112	112
117	111	112	108	115	114	104	114	114	119	111	111	111	109
113	115	115	126	128	126	127	126	124	123	125	119	107	113
108	110	122	128	126	137	140	145	145	145	133	128	114	113
108	110	122	125	135	140	143	145	147	140	136	128	114	113
114	109	112	117	129	127	130	140	144	143	128	119	118	109
116	112	112	114	114	111	110	107	116	114	114	110	115	115
108	109	107	111	112	104	109	109	112	112	112	110	106	104

Figure 4.11
Impressed Braille Dot pixels
categorized by gray values

111	111	107	114	114	113	115	119	115	111	107	105	114	113
115	116	104	107	112	113	116	115	118	115	111	114	112	104
115	107	106	108	112	116	126	127	126	118	112	103	110	109
113	113	110	111	130	140	145	147	138	126	110	110	111	101
114	109	112	117	140	150	152	151	144	143	137	119	118	109
108	110	122	125	135	140	140	126	123	133	141	128	114	113
113	115	115	126	128	126	127	126	124	123	125	119	107	113
117	111	112	108	115	114	104	114	114	119	111	111	111	109
110	104	99	92	94	99	101	110	116	109	104	102	110	110
104	112	109	94	93	85	84	90	92	101	96	100	112	112
114	113	114	112	87	82	82	75	94	87	82	96	108	108
110	114	115	108	99	90	81	74	77	80	90	104	111	114
112	111	113	113	114	109	101	91	95	98	105	110	109	107
116	112	112	114	114	111	110	107	116	114	114	110	115	115
108	109	107	111	112	104	109	109	112	112	112	110	106	104

Figure 4.12
Embossed Braille Dot pixels
categorized by gray level

Existing Segmentation Algorithm

Character Isolation Box Calculation

Character Isolation Box Calculation works by scanning each dot horizontally, making a line for every row in the Braille document. The scanning will stop if the last 3 lines are encountered. This is also the same when scanning vertically but making 2 vertical lines. The only downside is that if after preprocessing, the image has still noises. This will cause a misalignment because of the noises that might be recognized as dots. [Salim et al.]

The previous study comes up with the following steps for segmentation as in Figure 4.13.

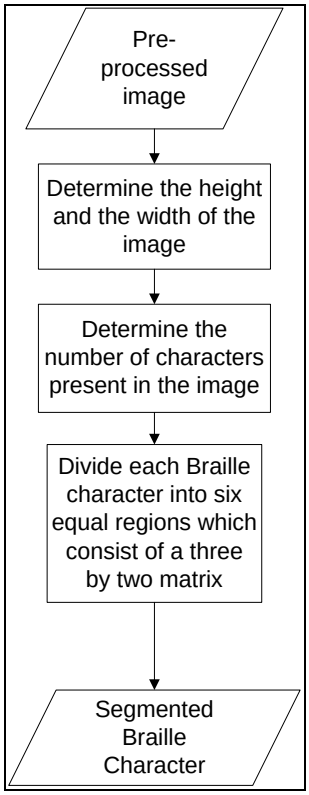


Figure 4.13
Segmentation algorithm of the previous study

Segmentation is the process of partitioning an image into multiple segments in which each Braille character will be isolated for easier translation and evaluation. The preprocessed image is used in the segmentation process. The character enclosed in a box is one that will be isolated (Figure 4.14). The isolated Braille character is then divided into six equal parts with one Braille dot for each subdivision. Figure 4.15 shows an example of a segmented Braille character.

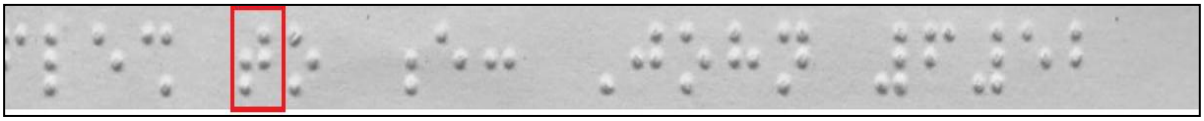


Figure 4.14
Preprocessed image to be used in the segmentation

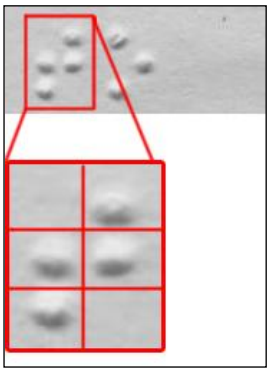


Figure 4.15
Segmented Braille Character

Proposed Segmentation Algorithm

This algorithm is used to get the dimensions of each Braille character within the Braille document. It works by scanning each dot horizontally, making a line for every row in the Braille document. This is also the same when scanning vertically but making 2 vertical lines. The only downside is that if after preprocessing, the image has still noises. This will cause a misalignment because of the noises that might be recognized as dots. In

a sentence, there may be many spaces in between depending on the number of words. Those spaces must be also part of the segmentation because a space also represents a character. In this algorithm, the proponents include space character as part of the segmentation. The proposed segmentation (Figure 4.16) is based on the distance of the x-coordinates of the Braille dot centroids. It takes into consideration the fixed horizontal distance between the dots.

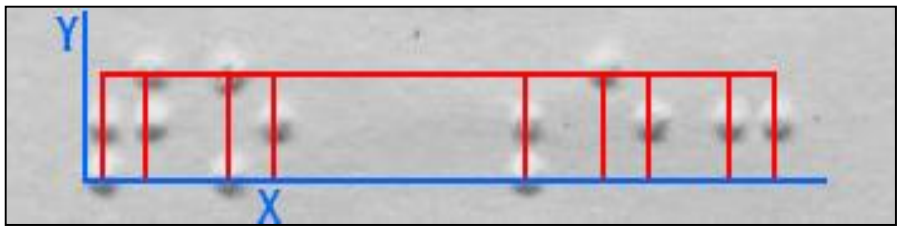


Figure 4.16
Visual representation of the proposed segmentation algorithm

Pattern Generation Algorithm

This algorithm is used after segmentation. This algorithm identifies which part of the 3x2 Braille cell contains a dot then converts to binary which can be used to match in the database of Braille characters. Pattern Generation is a technique used in examining the presence of the dot in their possible positions/ predefined regions. Originally, Pattern Generation algorithm works well if the Braille is not distorted but if it is, it will malfunction and will produce erroneous results so the previous proponent was able to solve that problem. As of now, Pattern Generation Algorithm recognizes Braille characters one by one. The proponents reinvented a way to be able to recognize many characters including the spaces.

Existing Recognition Algorithm

A character database (Table 4.1) in the system is made which in turn will be compared to the extracted results. It is very important that the character database is one hundred percent accurate to achieve the desired result. The Binary representations are based on the correct positions of the Braille character.

Table 4.1 shows the whole character database that is essential and the key component of the whole system.

Binary Representation	Letter
100000	A
110000	B
100100	C
100110	D
100010	E
110100	F
110110	G
110010	H
010100	I
010110	J
101000	K
111000	L
101100	M
101110	N
101010	O
111100	P
111110	Q
111010	R
011100	S
011110	T
101001	U
111001	V
010111	W
101101	X
101111	Y
101011	Z

Table 4.1
Character Database

The binary representations are not randomly generated but rather they are produced by plotting the dots on their designated regions (Figure 4.17). [Lao, 2009]

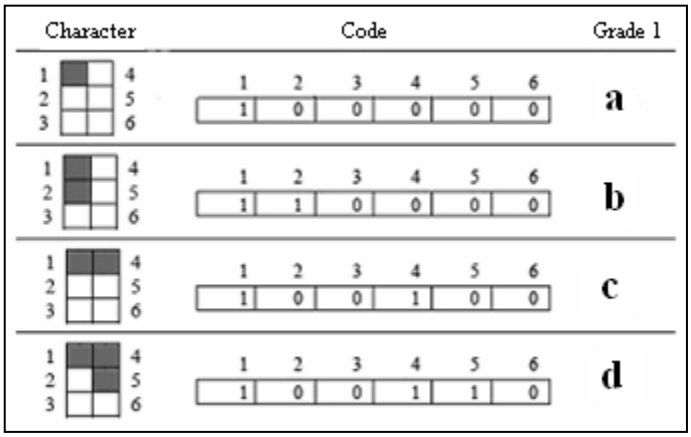


Figure 4.17
Illustration of Braille Codes and its equivalent letter

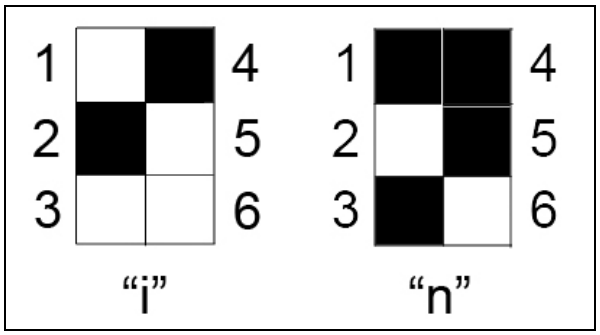


Figure 4.18
Braille Representation of the word "in"

To illustrate on how letters in the alphabet are represented by binary code, Figure 4.18 shows the Braille Coded Binary Representation of the word "in". By starting from 1 to 6, it is seen that the letter "i" is equivalent to the binary code "010100" and the letter "n" is equivalent to "101110".

Proposed Recognition Algorithm

This algorithm is for the system to accommodate Grade 2 Braille which is made up of other Braille characters to form shortened words and is governed by different

contractions. It checks if a certain Grade 2 Braille character has a contraction or not. If yes, that character should be checked for the equivalent contraction in the database and in the Grade 2 Contraction Rules set.

For example, the word “daddy” is written in Grade 1 Braille (Figure 4.19) which is just a conversion of Braille codes to its equivalent letter. On the other, in Figure 4.20, the word “daddy” is already contracted based on the set of contractions that starts with letter d (Figure 4.21).

Figure 4.19
“daddy” in Grade 1 Braille

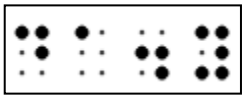


Figure 4.20
“daddy in Grade 2 Braille

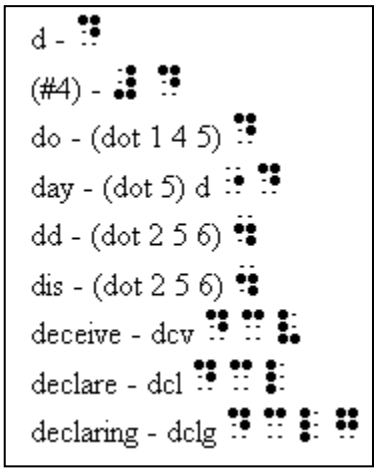


Figure 4.21
“d” Braille contractions

One example of a Grade 2 Braille contraction rule is seen in Figure 4.22 and Figure 4.23. The word “not” (Figure 4.22) is not the same as the contracted “cannot” (Figure 4.23). The reason behind it is that the word “not” is a stand alone of “n” in the contractions of Grade 2 Braille (Figure 4.24). Meaning, it cannot mix with other contractions. The word “not” in Grade 2 Braille is “not” as is.

Figure 4.22
“not” in Grade 2 Braille

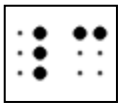


Figure 4.23
“cannot” in Grade 2 Braille

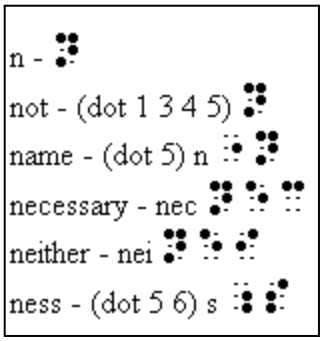


Figure 4.24
“n” Braille Contractions

In Figure 4.25, the phrase “still standing” is written in Grade 2 Braille. “St” and “still” have the same contractions but different translations due to neighboring Braille characters. For example, in Figure 4.25, the next character after “still” is a space and “st” has other contractions to mix with.

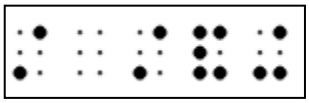


Figure 4.25
“still standing” in Grade 2 Braille

D. Processes

1. Preprocessing Procedures

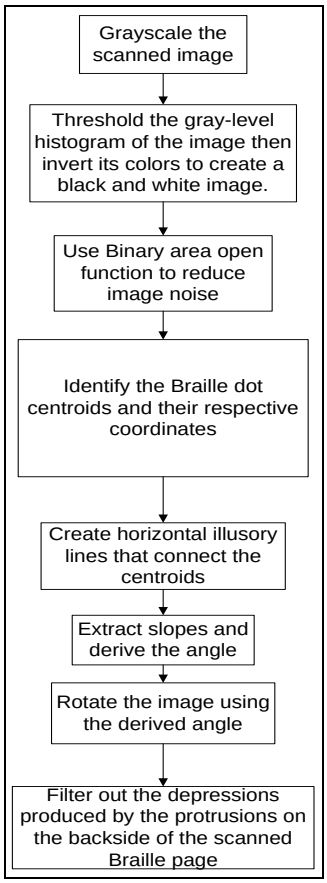


Figure 4.26
Preprocessing Phase

a. Grayscale/Thresholding/Inversion

Since modern scanners capture a document as it is, it may capture many different colors. It is better to make the color of the image monochrome thus it will become black and white so it will be easier to detect a certain pixel in an image. This is a very crucial part because the output will be used by the feature extraction part. In short, thresholding makes the image black and white to make it easier for the algorithm to detect particular pixels and differentiate them as to

what they are. For example: dots from open space. Inversion is useful in detecting the Braille dot centroids.

Figure 4.27
Example of a pre-cut scanned Braille document

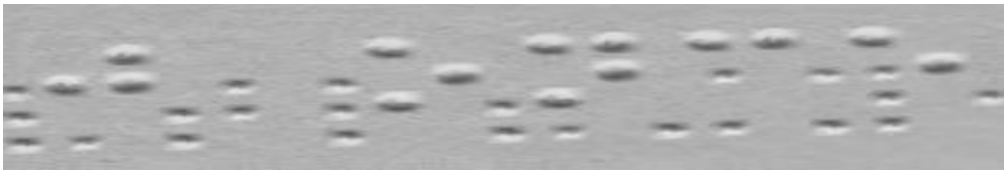


Figure 4.28
Gray level of Figure 4.27

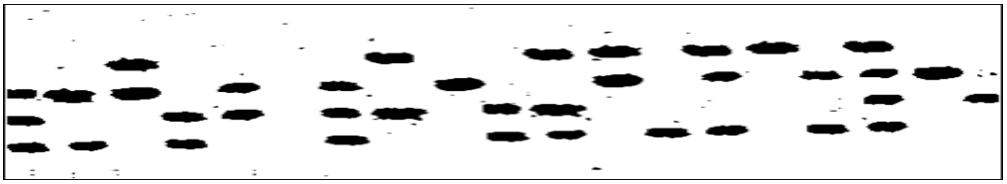


Figure 4.29
Threshold of Figure 4.28

Figure 4.30
Inverted Threshold Image of Figure 4.29

b. Noise Reduction

Using the Binary Area Open function, the amount of noise in the image is reduced. Groups of white adjacent pixels that are less than 30 will be removed because they are considered as noise because the estimated size of the shadow of

a Braille dot is 30. These groups of white pixels that are composed of less than 30 pixels are useless and can only cause inaccurateness so the logical thing to do is get rid of as much noise as possible.

As shown in Figure 4.31, some Braille dots are a bit flawed. Those faults may become a problem in analyzing the characters. On the other side, as shown in Figure 4.32, by using the Binary Area Open, the amount of noise that can be seen decreased.

Figure 4.31
Braille Image with noise

Figure 4.32
Braille image with no noise

c. De-skewing

The de-skewing process starts with the detection of the center of the Braille dots. These detected centers are then connected to each other to form the horizontal illusory lines (Figure 4.33). To eliminate the less significant lines, the line slopes are acquired. The lines with slopes outside the accepted slope range are eliminated (Figure 4.34). The range is -0.1 to 1.0. The mode of the slopes (Figure 4.35) is then acquired. This selected slope is used to determine the angle of inclination using the arctangent mathematical function. After the slope is converted into an angle, the system will rotate the image until the angle is equal to zero or a straight line is produced.

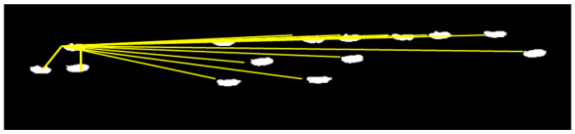


Figure 4.33
Horizontal Illusory Lines

Figure 4.34
Slopes within the range

Figure 4.35
Mode of the Slopes

This selected slope is used to determine the angle of inclination using the arctangent mathematical function. After the slope is converted into an angle, the

system will rotate the image until the angle is equal to zero or a straight line is produced (Figure 4.36).

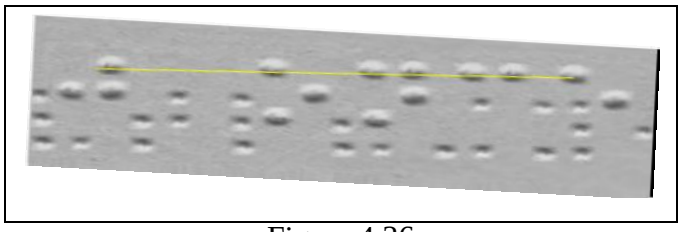


Figure 4.36
De-skewed Braille Image

d. Depression Filtering

The depression removal starts with the thresholded and inverted image to extract the coordinates of the centroids of each dot (red dot in Figure 4.37), using the grayscale version of the image the gray values of the pixels above and below the centroids (blue dots in Figure 4.38) are then used to determine whether the dot is classified as a protrusion or a depression. A high gray value pixel above the centroids that coincides with a low gray value pixel below the centroids indicate a protrusion while a low gray value pixel above the centroids that coincides with a high gray value pixel below the centroids is classified as a depression.

Figure 4.37
Inverted Thresholded Braille Dot Image

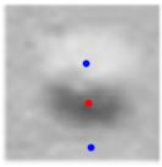


Figure 4.38
Grayscale Braille Dot Image

Once the depressions are differentiated from the protrusions the centroids of the depressions are then enclosed in a box. A blurring function is then implemented within the box to destroy the depressions as seen in figure 4.40.

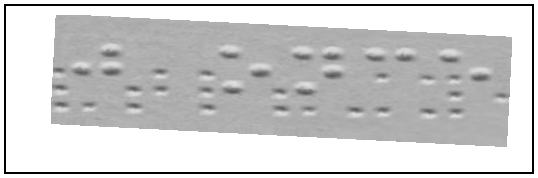


Figure 4.39
Braille Image with Depressions

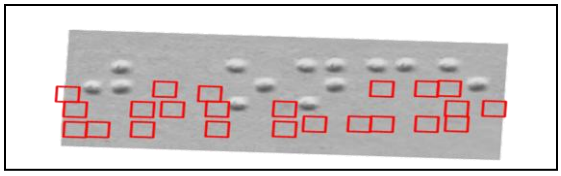


Figure 4.40
Braille Image without the Depressions

After performing the preprocessing procedures, the preprocessed image is then produced (Figure 4.41).

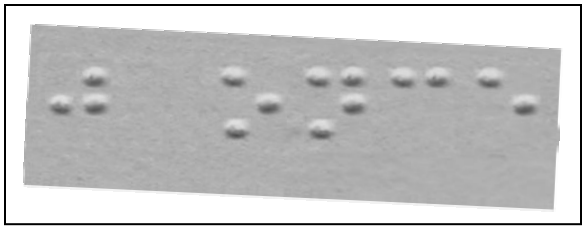


Figure 4.41
Example of Preprocessed Image

2. Segmentation

Segmentation is used to simplify and/or change the representation of an image into something that is more significant and easier to analyze. Image segmentation is usually used to detect objects and boundaries in images like lines and curves. Segmentation is done by getting the dimensions of the image like the height and the width. Then, determine how many Braille characters are in that image and then, divide each character into six equal areas that would signify a 3 by 2 matrix.

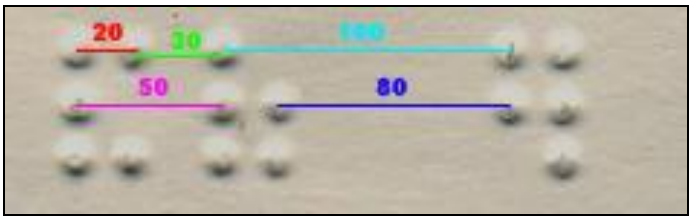


Figure 4.42
Braille Dot Pixel Distances

As seen in Figure 4.42, the Braille dots have standard distances from each other, when the scanner is set to 200 dpi, the estimates distances between dots are 18-22 pixels for dots within the same character, 28-52 pixels for dots from a character to the next character and 80-100 pixels for dots that have spaces between them.

3. Feature Extraction

Feature Extraction gets the relevant features of the Braille image to be recognized. This is where Pattern Generation which is the one in charge in examining the dots in its possible positions takes place. The scanning of dots is from left to right and top to bottom. If the system detects a dot, it will be represented by binary value 1 otherwise 0. Since there are 6 regions in a Braille cell, the system will just keep on scanning and converting until all regions are filled. Since the noise can be distinguished as a dot, it may lead to wrong representation.

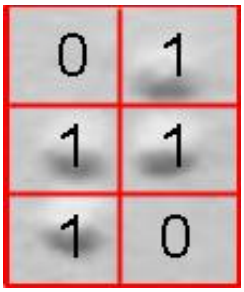


Figure 4.43
Example of Feature Extraction

4. Recognition

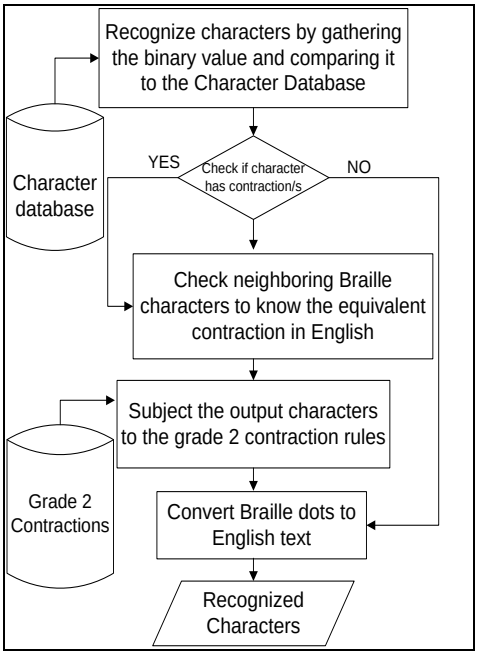


Figure 4.44
Recognition Phase

The approach for the recognition phase is shown in Figure 4.44. First, it will check if the scanned Braille character has a contraction equivalent. If there is, the characters on the left and on the right of that character should be verified to get the equivalent contraction in English. After checking, the characters should be validated by the Grade 2 Braille Contractions. On the other hand, if there is no contraction for that Braille character, that character is converted directly to English text. The characters would now produce English words in a text document.

E. Features of the System

One feature that the system has is that the user can view the Braille alphabet and the Grade 2 Braille contractions. These features are desired to validate if the scanned Braille image is actually equivalent to the converted output. The system can also allow the user to reset everything to try another scenario or to try another

translation/s. This feature clears out the previous image used and the output observed. The system also has exit button to close the program.

F. Comparative Analysis

This section tackles the differences of the previous study and the current study. Table 4.2 shows the differences in algorithms and processes of both studies.

Description	**Previous Study Optical Braille Recognition Using Modified Character Isolation Box and Pattern Generation**	**Current Study Enhanced Optical Braille Recognition Using Slope Determination and Double Sided Dot Detection for Grade 2 Braille**
Noise Filtering	• Uses Weiner Filter function	• Uses Binary Area Open function
Depression Filtering	• Limitation of the previous study	• Checks certain pixels and their corresponding color to identify if that dot is a depression or not
Segmentation	• Can only segment 5 characters at a time • Their position is pre defined due to fixed values of each character box	• Dynamically forms each Character box as it accepts more than 5 characters and also each Braille character is not bound to a specific position
Pattern Generation	• Can only generate a binary equivalent of a single Braille character	• Can identify if the group of characters is a contraction and can generate its own set of binary Braille character equivalents

Table 4.2
Comparative Analysis of the previous and current studies

The previous study included a de-skewing process that is reliant on the edge of the image to calculate the angle of inclination, if the actual angle of inclination of the image does not coincide with the angle of the edge, the image would not be rotated correctly for the image to be recognized accurately. The proponents introduced the Slope Determination algorithm to derive the actual angle of inclination without relying on the edge of the image, Slope Determination uses the rows of Braille dots to form a line and derive the actual angle of inclination. Weiner filter is an algorithm that mostly restores images that contain motion blurs and noise. Upon scanning a Braille document a motion blur is very unlikely to occur since the Braille document would be enclosed in the scanner. The proponents used Binary Area Open since it requires less CPU resources and yet able to remove noise from the scanned Braille document. The previous study was unable to translate double sided Braille documents. The proponents used Double Sided Dot Detection to solve this previous limitation. In terms of segmentation the previous study was also limited to Braille images that have constant distances from the image edges and can only segment up to five characters at a time, as opposed to the new segmentation algorithm that is reliant on image edges and is not limited to only a number of characters. The Pattern Generation of the previous study maps on Braille character to a single letter, this relationship is possible in Grade 1 Braille. However in Grade 2 Braille one to one mapping is not applicable because there are contractions that have identical binary codes that could cause ambiguity. The modified Pattern Generation is now able to read a contraction and translate it into its full word or partial word equivalent and at the same time differentiate the contractions that have identical binary codes by taking into consideration the position of the contraction within the word.

Overall, the previous study was constrained to images that contain Braille dots that have constant distances from the edge of the image. Image inputs that were not cut correctly or do not comply with the constant distance requirement will result in an erroneous output. The proponents introduced a set of algorithms that would make the de-skewing and segmentation dynamic, meaning the distances of the Braille dots from the edges does not need to be constant and the whole Braille document may be skewed upon scanning. The Double Sided Dot Detection algorithm was also added to enable the translation of double sided Braille documents.

G. Testing

1. Test Script

A test script is a short program written in a programming language used to test the functionality of a software system.

```
1       START
2               Load image
3               Undergo thresholding
4               detect Braille dot centroids
5               convert image to grayscale
6               identify depressed Braille dots
7               Loop
8                       erase the identified depressed Braille dots
9               end Loop
10              determine the angle of skew
11              based on the angle, rotate the image to de-skew
12              acquire the Braille dot coordinates
13              convert the data into binary code

14              if (binary code is present in Grade 2 table)
15                      get the corresponding contraction equivalent in Grade2 table
16              else
17                      get the corresponding Grade 1 Braille equivalent in Grade1 table
18              end if

19      END
```

Figure 4.45
Pseudo code for Optical Braille Recognition for Grade 2 Braille

2. Test Cases

Test cases are a set of scenarios to determine whether an application or a system is working correctly or not. In testing the system, the proponents use an American Bible and an instructional manual as test case subjects for scanning Braille documents. There are 100 different test cases, some cases are based on the study's limitations and some consist of samples which have different tilted angles and word/s. The test cases are presented in a table. The table consists of columns with different categories. The first category includes the test case number. The second category indicates the generated result of a certain test case. The result can be negative or positive. The result that produces a negative result means that the actual output did not match the expected output. The third and fourth categories consist of the expected and actual outputs respectively while the last column indicates the reason/s why the result is negative.

H. Findings

The previous algorithm was not successful in de-skewing the image if the image is skewed upon scanning the Braille document. The proponents' de-skewing algorithm is able to de-skew an image that is skewed upon scanning. Grade 2 Braille has some contractions which are physically identical but differ in the grammatical use, as seen in the test cases (see Appendix C). In eliminating the back side dots, some back side dots are not erased totally which can lead to protrusions later on. After the testing, it yielded that out of the 100 test cases, 89 are correct and has 89 percent accuracy. The remaining 11 gave negative results because some instances are

part of the study's limitations, some are invalid inputs and some of the negative results are caused by the problems with the Braille documents.

I. Problems Encountered in System Implementation

The proponents encountered many problems during the system implementation. One problem encountered is the attempt to solve the de-skewing process because the exact angle of inclination was difficult to identify since there is initially no angle to compare to. There is also a problem in eliminating the back side dots or depressions of the Braille document since some of those depressions are really difficult to identify because some of those are a bit identical to protrusions. Another problem that the proponents encountered was during the recognition process. Some dot combinations are not unique so the proponents made a way in order to translate the dots correctly. There are many rules in using the Grade2 Braille contractions so the proponents made a way in order to include all the rules in the system. Also, the implementation of the algorithms in Matlab is another problem that the proponents encountered because they do not have any experience in using the said application.

CHAPTER 5

SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

E. Summary

Braille is a writing system that has been commonly used by visually impaired people to read and write. As time goes by and as technology advances, researchers think of ways on how can they help bridge the gap between the normal and the visually impaired people. Because of that, the Optical Braille Recognition (OBR) is born. The OBR is a series of process that aims to convert Braille characters to normal texts. There are many algorithms that are available today that can convert Braille to normal texts but there are certain limitations that need to be considered. First, the image skews should be corrected to preserve the original data. Second, Grade 2 Braille characters should be recognized. Last but not the least, the ability to differentiate the depressions from the protrusions is required to recognize it correctly. As a response, the proponents developed a system that aims to solve those problems and produced superior results with only minimal errors. Through intensive research, researchers are now continually finding ways to improve those algorithms to help the visually impaired people, as well as normal people in exchanging information rapidly.

F. Conclusions

The proponents conclude that the deletion of back side dots is valuable in recognizing the Braille characters correctly. Also, correcting the image skews will be beneficial to preserve the data in the scanned image. The problem of recognizing Grade 2 Braille characters was also solved by modifying the pattern generation algorithm. The strength of the system is recognizing inputs that have Braille dots in

excellent condition and have protrusion Braille dots that are significantly apart from depression Braille dots. The proponents were able to modify the existing segmentation and recognition approaches to accurately and correctly translate scanned Grade 2 Braille documents into English text. They were also able to create a stand alone application that interprets a double sided and Grade 2 Braille document and they were able to improve the de-skewing process of the scanned image. Overall, the accuracy of the proposed algorithms is tested and thus achieved the desired results with minimal problems.

G. Recommendations

In this study, the proponents only developed a system that would recognize Grade 2 Braille characters on a double sided Braille scanned image and correct image skews.

The following recommendations are for further improvement of the study:

- One area of improvement is instead of using a scanner, try using a camera in capturing the Braille document.
- Another area of improvement is the ability to solve the problem with scanned Braille documents that contain foreign objects aside from the Braille document itself (example: dusts, hair strands, pen mark etc).
- Moreover, the recommended system should recognize complete sentences including punctuation marks and other symbols.
- Another improvement is instead of using English Braille documents, use Filipino Braille documents or other languages as source.

- Another improvement is by using a musical Braille instead of using alphabetic Braille.
- By changing (increase/decrease) the resolution (dpi) of the scanner is another way of improving the study.
- The ability to solve the problem with de-skewing the image for more than 5.5 degrees.
- The ability to solve the recognition of the dots which are very close with each other.
- The system should recognize other image formats (256 or 16 color BITMAP, GIF, PNG etc.) as image inputs.
- The system should be able to recognize and translate a whole Braille document. The system should also recognize and translate front and back sides of the whole document.
- The system should be integrated to a text to speech translator software to further help the visually impaired.
- The system should able be able to accept Braille documents that are written manually (using a stylus and a slate).
- Finally, the system should be able to improve the detection of the back side dots due to poor back side deletion that the proponents encountered in constructing the system.

BIBLIOGRAPHY

A. Books

Erwin M. Bakker, Thomas S. Huang. 2003. *Image and Video Retrieval*. Springer New York.

Craig A. Lindley. 2000. *Practical Image Processing in C*. John Wiley and Sons Inc Canada.

Mike Nachtegael, Dietrich Van der Weker. 2003. *Fuzzy Filters for Image Processing.* Springer Berlin.

J. Schalkoff. 2000. *Digital Image Processing and Computer Vision.* John Wiley and Sons Inc Canada.

B. Internet Sources

Optical Braille Recognition. Read Braille with a Standard Scanner. Retrieved March 9, 2009, from http://www.neovision.cz/prods/obr/

OBR (Optical Braille Recognition). Read Braille with a Standard A4 Scanner Retrieved March 9, 2009, from http://www.techno-vision.co.uk/obr.htm

Should we be Teaching Children Grade 1 or Grade 2 Braille? Retrieved March 9, 2009, from http://www.viewbraille.co.uk/grade1or.html

What is Braille? Tennessee Council of the Blind Retrieved March 9, 2009, from http://www.acb.org/tennessee/braille.html

Grade 1 Braille. Tennessee Council of the Blind Retrieved March 9, 2009, from http://www.acb.org/tennessee/braille.html

Grade 2 Braille. Tennessee Council of the Blind Retrieved March 9, 2009, from http://www.acb.org/tennessee/braille.html

Grade 3 Braille. Tennessee Council of the Blind Retrieved March 9, 2009, from http://www.acb.org/tennessee/braille.html

Grade two Braille Contractions Retrieved March 5, 2009 from www.99main.com/~charlief/brl/brl2.html

Adaptive Thresholding. Wikipedia. Retrieved March 5, 2009 from en.wikipedia.org/wiki/Adaptive_thresholding

Braille. Wikipedia. Retrieved March 5, 2009 from en.wikipedia.org/wiki/Braille

Deskewing Perspectively Distorted Documents: An Approach Based on Perceptual Organization. Retrieved March 22, 2009 from http://www.hpl.hp.com/techreports/2001/HPL-2001-100.pdf

C. Journal

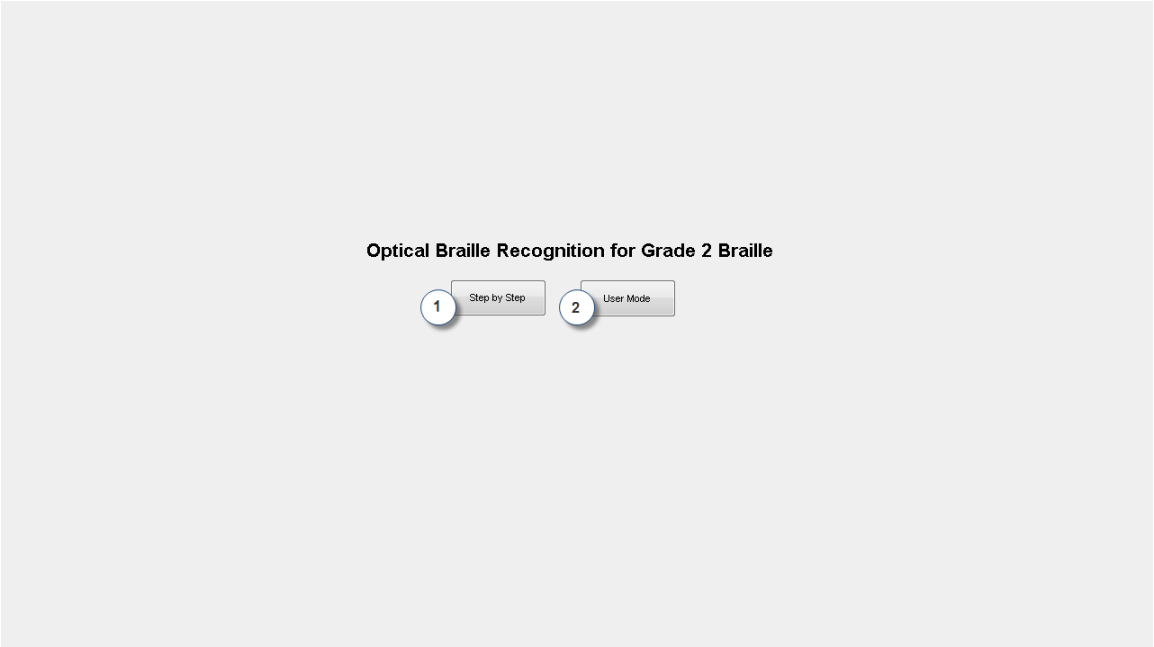
Gel Analysis Software. (2005, June 3). *Science*, Retrieved March 2, 2009, from MAS Ultra - School Edition database.

D. Unpublished Thesis

Lao, A. (2009). *Optical Braille Recognition Using Character Isolation Box and Pattern Generation.* Unpublished thesis, University of Santo Tomas, España, Manila

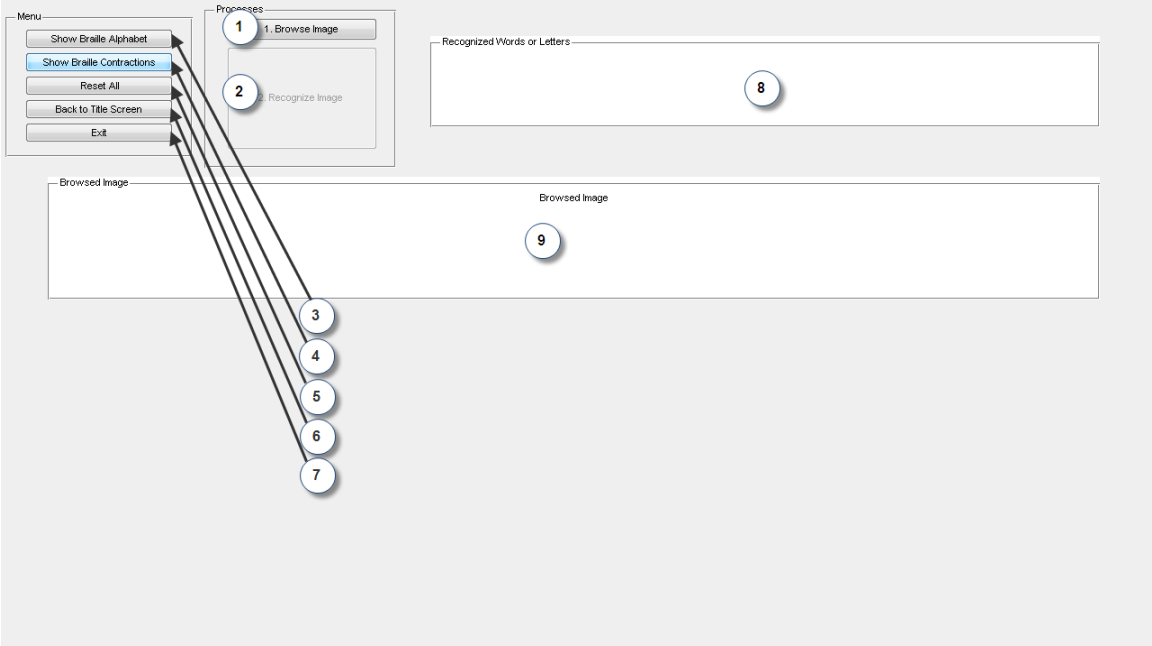
APPENDIX A

User Interface



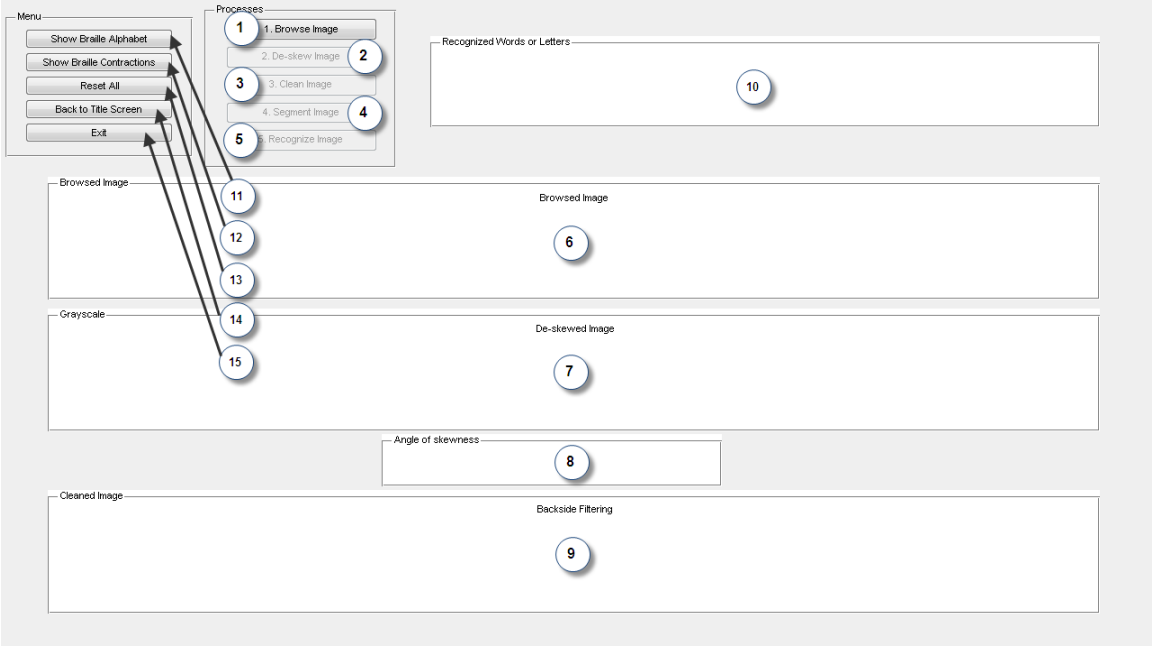
A.1
Title Screen Interface

1. Step by Step Button- enters step by step mode to enable an advanced user to view the Braille recognition processes one by one.
2. User mode – enters user mode which only displays strictly the input and the output of Braille recognition.



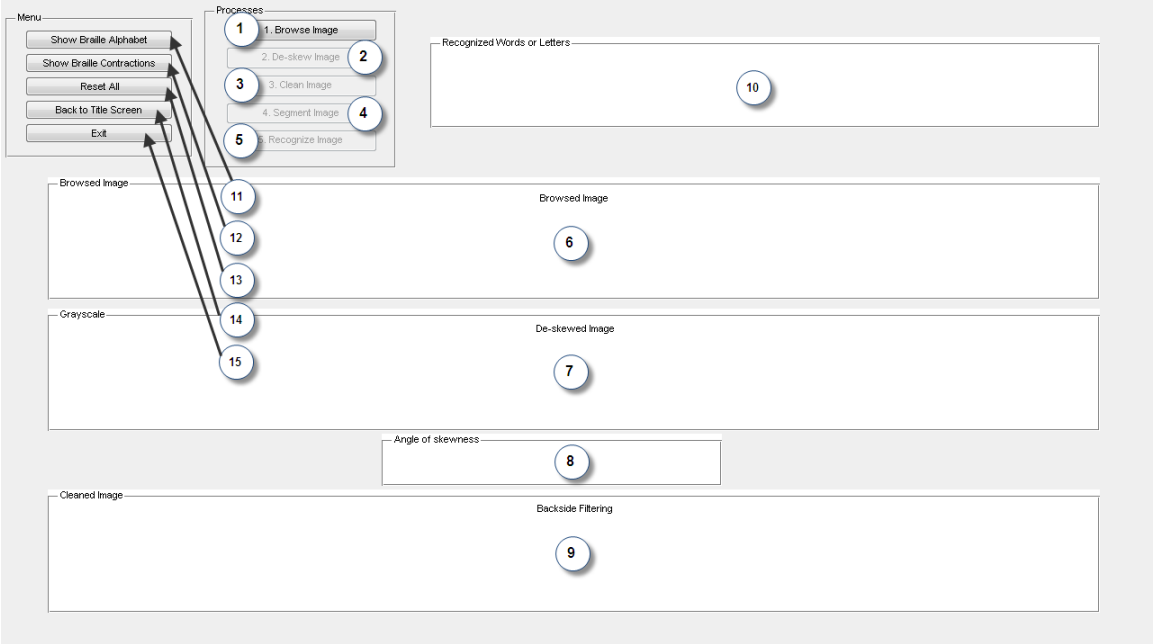
A.2
User Interface (User Mode)

1. Browse Image Button – allows the user to choose which Braille image to translate.
2. Recognize Image Button – used to translate the Braille image in English text.
3. Show Braille Alphabet Button – shows the Braille alphabet for comparison purposes.
4. Show Braille Contractions Button – shows the Braille contractions for comparison purposes.
5. Reset All Button – to clear all the data in the user interface.
6. Back to Title Screen Button – returns to the initial screen
7. Exit Button – to close the application or system.
8. Recognized Words or Letters Panel – displays the translated English text
9. Browsed Image Panel – displays the selected Braille image input



A.3
User Interface (Step by Step)

1. Browse Image Button – allows the user to choose which Braille image to translate.
2. De-skew Image Button – straightens the skewed input image
3. Clean Image Button – performs the deletion of back side Braille dots
4. Segment Image Button – divides the Braille character/s
5. Recognize Image Button - identifies the Braille character/s from the image and convert it to its respective value
6. Browsed Image Panel – shows the chosen image to be manipulated
7. De-skewed Image Panel – displays the result after manipulating the skewed image input.
8. Angle of Skewness Panel– displays the tilted angle of the input
9. Back side Filtering Panel – displays the result of the deleted back side Braille dots
10. Recognized Words or Letters Panel – where the converted value will be displayed



A.3
User Interface (Step by Step)

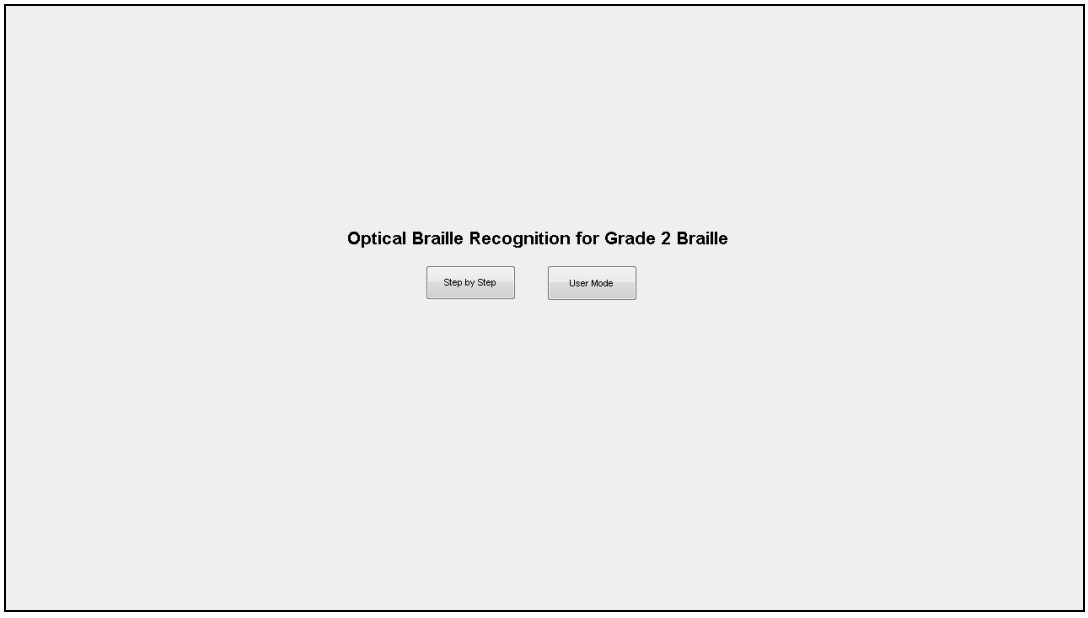
11. Show Braille Alphabet Button – shows the Braille alphabet for comparison purposes.
12. Show Braille Contractions Button – shows the Braille contractions for comparison purposes.
13. Reset All Button – to clear all the data in the user interface.
14. Back to Title Screen Button – returns to the initial screen
15. Exit Button – to close the application or system.

APPENDIX B

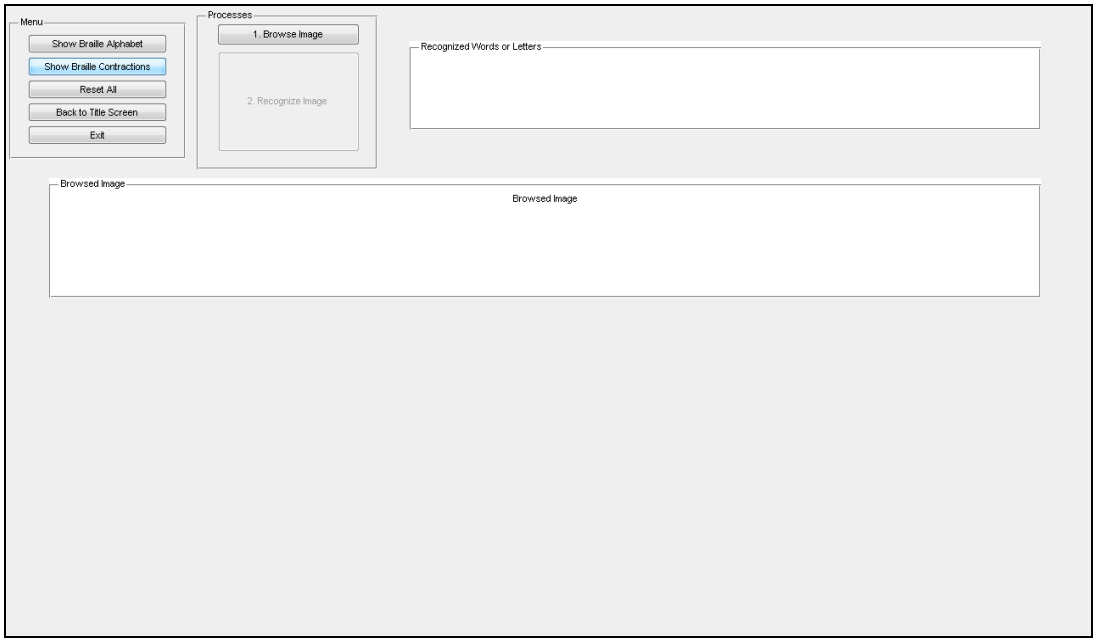
User's Manual

1. Double click the **OBR.exe** icon located on the desktop of the computer.

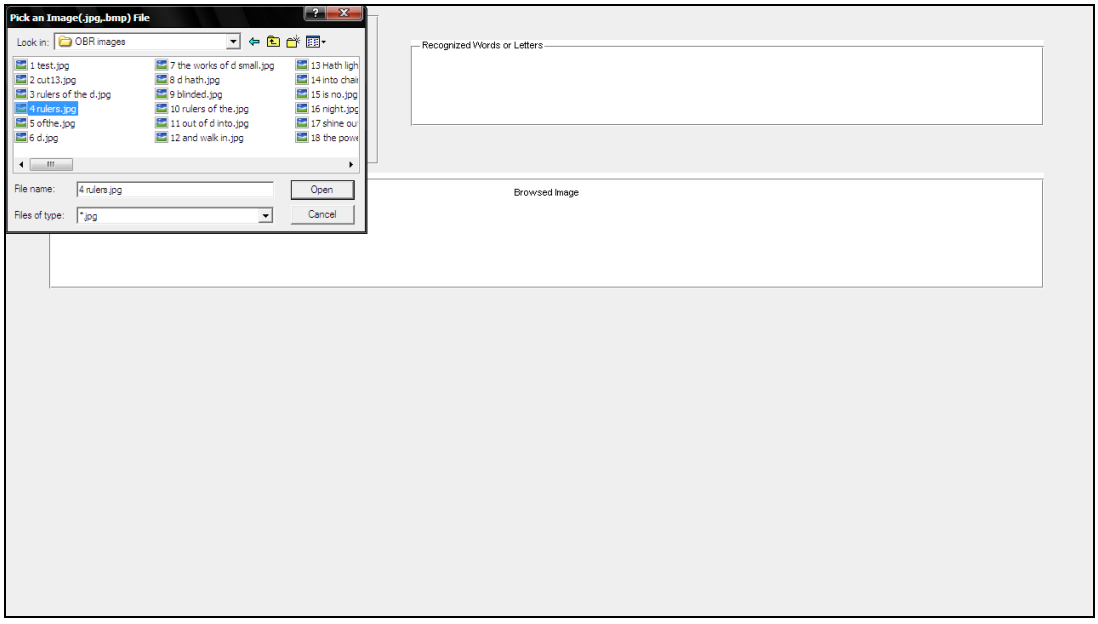
2. Wait for the application to load.
3. Choose either **Step by Step** or **User Mode** button then wait for the new window to load.



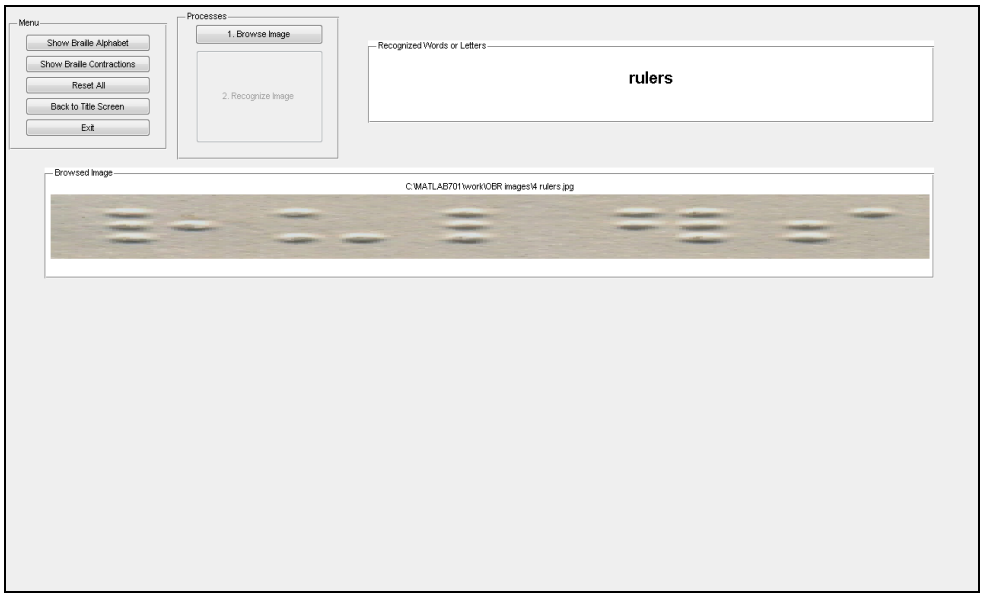
4. If you choose **Step by Step**, do steps 5 to 10. If you choose **User Mode**, click **Browse Image** to select an image to be manipulated.



Select the image and then click the **Open** button.



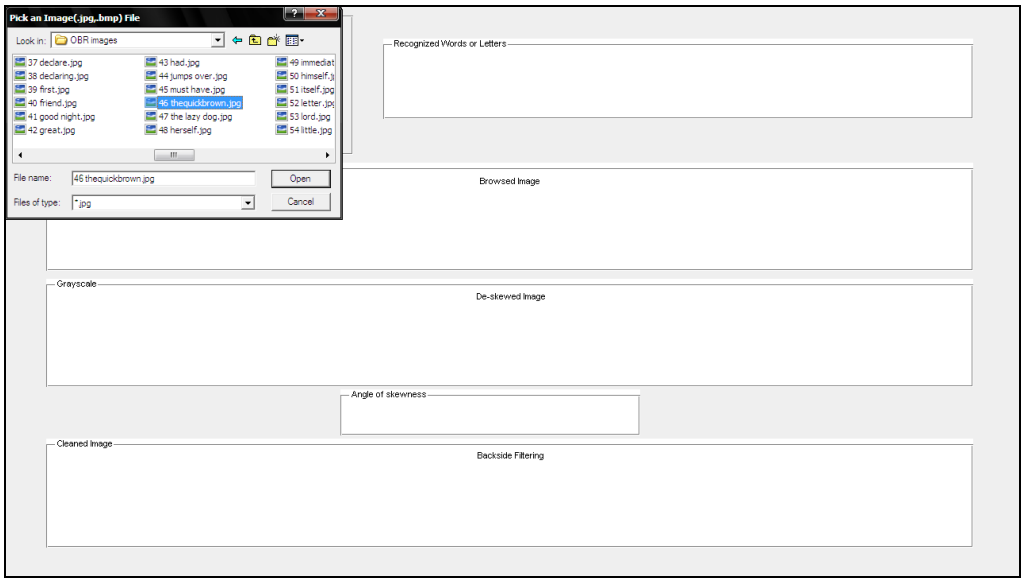
Then, click **Complete Process** button to view the translation. You may click the **Back to Title Screen** button anytime in case you want to go to the main window. Proceed to step 11 and/or 12 if necessary.



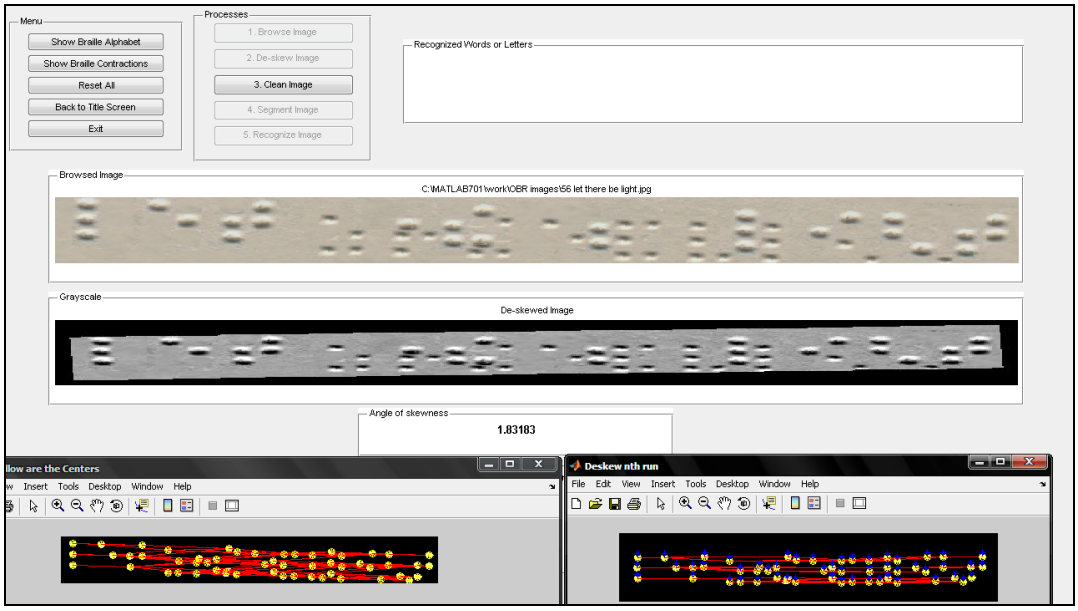
5. Click **Browse Image** button to select an image to be manipulated.

6. Select the image and then click the **Open** button.

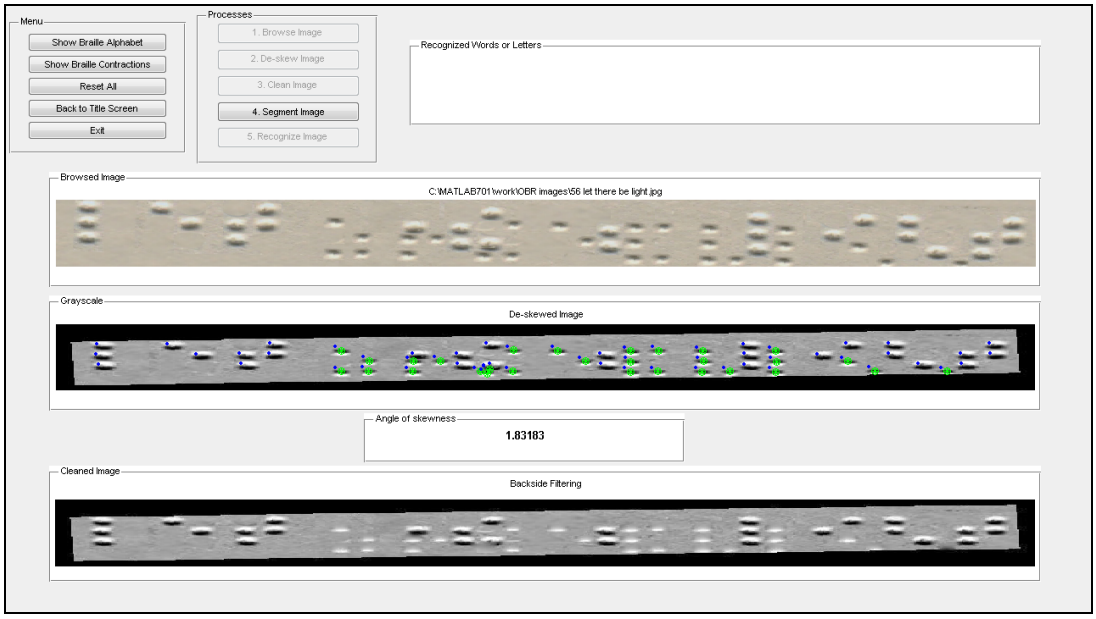
You may click the **Back to Title Screen** button anytime in case you want to go to the main window.



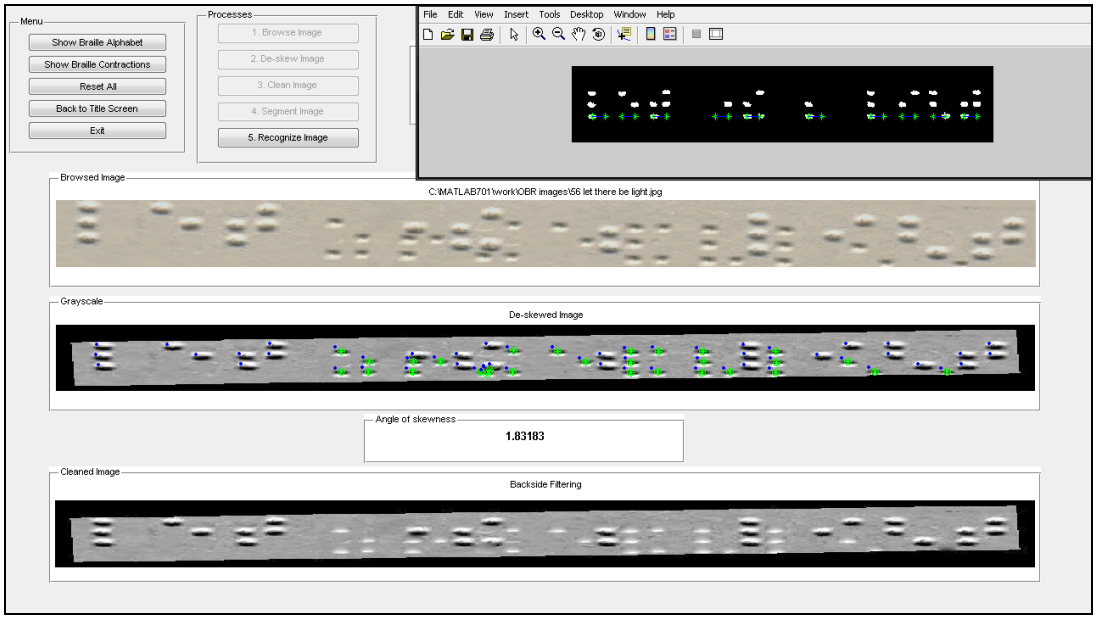
7. Click **De-skew Image** button to display the straightened image.



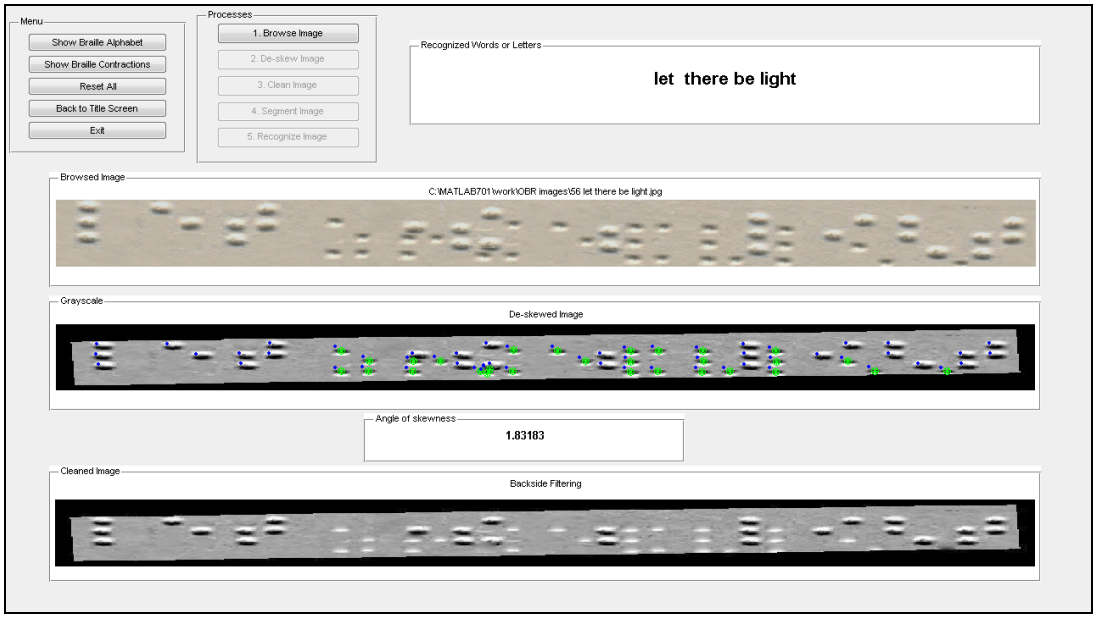
8. Click **Clean Image** to display the filtered back side Braille dots.



9. Click **Segment Image** to show the segmented Braille characters.



10. Click **Recognize Image** to know its converted value.

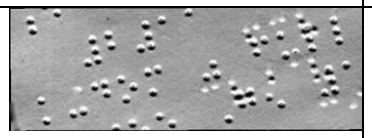
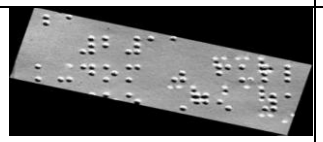
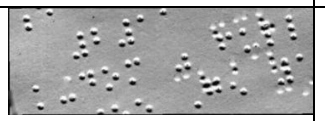
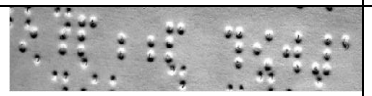
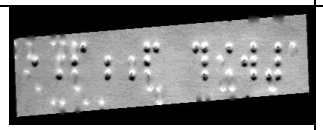
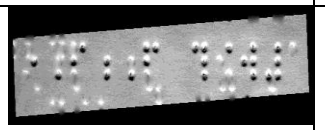
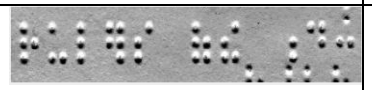
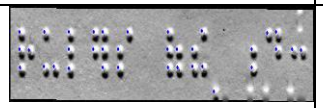
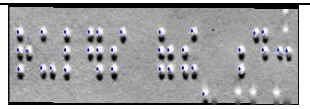


11. To try another test, just click **Reset All** and repeat the necessary steps.

12. Click the **X** button at the upper right corner or the **Exit** button to **close** the application.

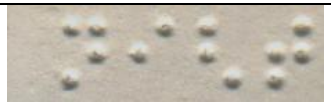
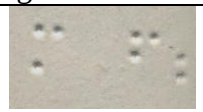
APPENDIX C

Test Cases

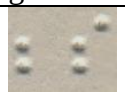
	Positive/ Negative	Word/Phrase	Input	Expected Output	Actual Output	Reason (if negative)
1.	Negative		 Skewed more than 5.5 degrees counterclockwise	 De-skewed Image	 Image is still skewed	An angle more than 5.5 degrees will not be extracted
2.	Positive		 Skewed 3.14 degrees clockwise	 De-skewed Image	 De-skewed Image	
3.	Positive		 Skewed 0.05 degrees counterclockwise	 Depressions are filtered	 Depressions are filtered	
4.	Positive	"rulers"	 "rulers" word	"rulers"	"rulers"	

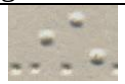
	Positive/ Negative	Word/Phrase	Input	Expected Output	Actual Output	Reason (if negative)
5.	Positive	“ofthe”	 “ofthe” phrase	“ofthe”	“ofthe”	
6.	Positive	“do”	 “d” character	“do”	“do”	
7.	Negative	“the works of d”	 low resolution input	“the works of d”	n/a	Resolution is under 200dpi
8.	Positive	“d hath”	 “d hath” phrase	“d hath”	“d hath”	
9.	Positive	“blinded”	 “blinded” word Skewed 0.06 degrees clockwise	“blinded”	“blinded”	
10.	Positive	“rulers ofthe”	 phrase skewed 0.40 degrees counterclockwise	“rulers ofthe”	“rulers ofthe”	

	Positive/ Negative	Word/Phrase	Input	Expected Output	Actual Output	Reason (if negative)
11.	Positive	“out of d into”	 “out of d into” phrase skewed 0.05 degrees counterclockwise	“out of d into”	“out of d into”	
12.	Positive	“and walk in”	 “and walk in” phrase skewed 0.71 degrees clockwise	“and walk in”	“and walk in”	
13.	Positive	“hath light with”	 “hath light with” phrase skewed 0.19 degrees counterclockwise	“hath light with” phrase	“hath light with” phrase	
14.	Positive	“into chains”	 “into chains” phrase skewed 0.77 degrees counterclockwise	“into chains”	“into chains”	
15.	Positive	“is no”	 “is no” phrase skewed 0.66 degrees counterclockwise	“is no”	“is no”	

	Positive/ Negative	Word/Phrase	Input	Expected Output	Actual Output	Reason (if negative)
16.	Positive	“night”	 “night” word skewed 0.83 degrees counterclockwise	“night”	“night”	
17.	Positive	“shine out”	 “shine out” phrase skewed 0.91 degrees counterclockwise	“shine out”	“shine out”	
18.	Positive	“the power”	 “the power” phrase skewed 0.73 degrees counterclockwise	“the power”	“the power”	
19.	Positive	“the d is”	 “the d is” skewed 4.45 degrees clockwise	“the d is”	“the d is”	
20.	Positive	No word	 Back Side Dots skewed 3.75 degrees clockwise	No Character Detected	No Character Detected	

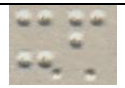
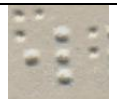
	Positive/ Negative	Word/Phrase	Input	Expected Output	Actual Output	Reason (if negative)
21.	Positive	“of”	 “of” word skewed 4.95 degrees clockwise	“of”	“of”	
22.	Positive	“out d”	 “out d” phrase skewed 5.27 degrees clockwise	“out d”	“out d”	
23.	Positive	“start here”	 “start here” phrase skewed 0.64 degrees counterclockwise	“start here”	“start here”	
24.	Positive	“no”	 “no” word skewed 0.64 degrees counterclockwise	“no”	“no”	
25.	Positive	“in him is”	 “in him is” phrase skewed 0.26 degrees clockwise	“in him is”	“in him is”	

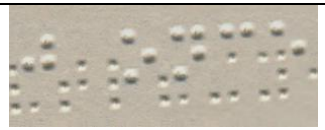
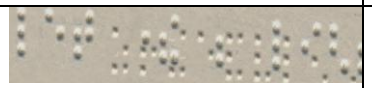
	Positive/ Negative	Word/Phrase	Input	Expected Output	Actual Output	Reason (if negative)
26.	Positive	“because”	 “because” word skewed 1.07 degrees clockwise	“because”	“because”	
27.	Positive	“about his”	 “about his” phrase skewed 0.26 degrees counterclockwise	“about his”	“about his”	
28.	Positive	“before”	 “before” word skewed 3.70 degrees counterclockwise	“before”	“before”	
29.	Positive	“below”	 “below” word skewed 1.08 degrees clockwise	“below”	“below”	
30.	Positive	“beneath”	 “beneath” word skewed 1.78 degrees clockwise	“beneath”	“beneath”	
31.	Positive	“beside”	 “beside” word skewed 0.9 degrees counterclockwise	“beside”	“beside”	

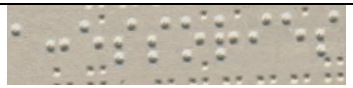
	Positive/ Negative	Word/Phrase	Input	Expected Output	Actual Output	Reason (if negative)
32.	Positive	“blind”	 “blind” word skewed 0.86 degrees counterclockwise	“blind”	“blind”	
33.	Positive	“cannot”	 “cannot” word skewed 0.4 degrees clockwise	“cannot”	“cannot”	
34.	Positive	“character”	 “character” word skewed 0.49 degrees clockwise	“character”	“character”	
35.	Positive	“children”	 “children” word skewed 2.17 degrees clockwise	“children”	“children”	
36.	Positive	“deceive”	 “deceive” word skewed 0.31 degrees clockwise	“deceive”	“deceive”	
37.	Positive	“declare”	 “declare” word skewed 2.33 degrees clockwise	“declare”	“declare”	

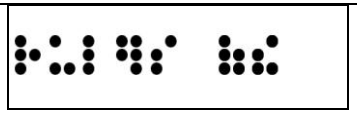
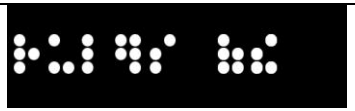
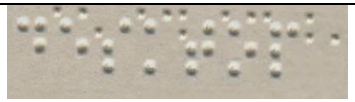
	Positive/ Negative	Word/Phrase	Input	Expected Output	Actual Output	Reason (if negative)
38.	Positive	“declaring”	 “declaring” word skewed 0.46 degrees clockwise	“declaring”	“declaring”	
39.	Positive	“first”	 “first” word skewed 1.37 degrees clockwise	“first”	“first”	
40.	Positive	“friend”	 “friend” word skewed 2.09 degrees clockwise	“friend”	“friend”	
41.	Positive	“good night”	 “good night” word skewed 0.49 degrees clockwise	“good night”	“good night”	
42.	Positive	“great”	 “great” word skewed 1.07 degrees clockwise	“great”	“great”	
43.	Positive	“had”	 “had” word skewed 1.76 degrees counterclockwise	“had”	“had”	

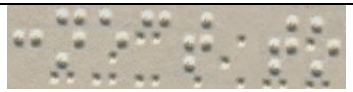
	Positive/ Negative	Word/Phrase	Input	Expected Output	Actual Output	Reason (if negative)
44.	Positive	“jumps over”	 “jumps over” phrase not skewed	“jumps over”	“jumps over”	
45.	Positive	“must have”	 “must have” phrase skewed 0.89 degrees counterclockwise	“must have”	“must have”	
46.	Positive	“the quick brown”	 “the quick brown” phrase not skewed	“the quick brown”	“the quick brown”	
47.	Positive	“the lazy dog”	 “the lazy dog” phrase not skewed	“lazy dog”	“lazy dog”	
48.	Positive	“herself”	 “herself” word skewed 0.55 degrees counterclockwise	“herself”	“herself”	
49.	Positive	“immediate”	 “immediate” word not skewed	“immediate”	“immediate”	

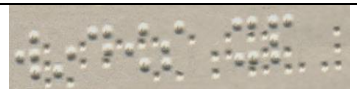
	Positive/ Negative	Word/Phrase	Input	Expected Output	Actual Output	Reason (if negative)
50.	Positive	“himself”	 “himself” word skewed 1.57 degrees counterclockwise	“himself”	“himself”	
51.	Positive	“itself”	 “itself” word skewed 2.42 degrees counterclockwise	“itself”	“itself”	
52.	Positive	“letter”	 “letter” word skewed 3.05 degrees clockwise	“letter”	“letter”	
53.	Positive	“lord”	 “lord” word skewed 4.3 degrees counterclockwise	“lord”	“lord”	
54.	Positive	“little”	 “little” word skewed 3.20 degrees counterclockwise	“little”	“little”	

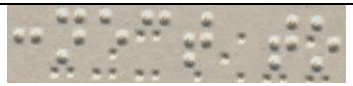
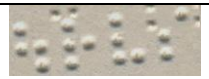
	Positive/ Negative	Word/Phrase	Input	Expected Output	Actual Output	Reason (if negative)
55.	Positive	“just once”	 “just once” phrase skewed 3.7 degrees counterclockwise	“just once”	“just once”	
56.	Positive	“let there be light”	 “let there be light” phrase skewed 1.83 degrees clockwise	“let there be light”	“let there be light”	
57.	Positive	No word	Blank input	Insufficient dots	Insufficient dots	
58.	Positive	“kingdom was full of”	 “kingdom was full of” phrase skewed 0.12 degrees counterclockwise	“kingdom was full of”	“kingdom was full of”	
59.	Negative	“how are you?”	 “how are you?” phrase skewed 3.7 degrees counterclockwise	“how are you?”	“how are yhis”	The system cannot process special character/s or signs. Question mark is a special character which is beyond the scope. Also the question mark has the same binary code as the “his” contraction.

	Positive/ Negative	Word/Phrase	Input	Expected Output	Actual Output	Reason (if negative)
60.	Negative	“hidden things of d”	 “hidden things of d” phrase skewed 4.2 degrees clockwise	“hidden things of d”	“hidden things of ever”	In the rightmost part, the depressions merged with the protrusion and combined as 1 dot.
61.	Positive	“just”	 “just” word cut vertically and skewed 0.26 degrees counterclockwise	“just”	“just”	
62.	Positive	“accessories”	 “accessories” word not skewed	“accessories”	“accessories”	
63.	Negative	“click icon”	 “click icon” phrase skewed 1.9 degrees counterclockwise	“click icon”	“cbick icon”	In the lowest left part, the depressions merged with the protrusion. In effect, the protrusion was also erased.

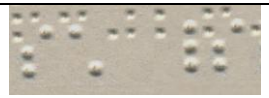
	Positive/ Negative	Word/Phrase	Input	Expected Output	Actual Output	Reason (if negative)
64.	Negative	“rulers of the”	 “rulers of the” phrase not skewed	“Insufficient Braille dots”	“No Dataowble ofa No Datak”	The input is not a scanned Braille document. It is an invalid input. There was a problem in extracting the centroids.
65.	Negative	“rulers of the”	 “rulers of the” phrase not skewed	“Insufficient Braille dots”	“Insufficient Braille dots”	The equivalent translation of the Braille characters in the image is “rulers of the”. However, this was not cut from a scanned Braille document.
66.	Positive	“desktop”	 “desktop” word skewed 1.30 degrees clockwise	“desktop”	“desktop”	
67.	Positive	“enter your name”	 “enter your name” phrase skewed 1.28 degrees clockwise	“enter your name”	“enter your name”	

	Positive/ Negative	Word/Phrase	Input	Expected Output	Actual Output	Reason (if negative)
68.	Positive	"installation successful"	 "installation successful" phrase not skewed	"installation successful"	"installation successful"	
69.	Positive	"internet"	 "internet" word skewed 0.17 degrees counterclockwise	"internet"	"internet"	
70.	Positive	"keyboard"	 "keyboard" word skewed 1.7 degrees clockwise	"keyboard"	"keyboard"	
71.	Positive	"program files"	 "program files" phrase skewed 1.5 degrees clockwise	"program files"	"program files"	

	Positive/ Negative	Word/Phrase	Input	Expected Output	Actual Output	Reason (if negative)
72.	Positive	“mouse over”	 “mouse over” phrase skewed 3.17 degrees counterclockwise	“mouse over”	“mouse over”	
73.	Positive	“right click”	 “right click” phrase skewed 0.8 degrees clockwise	“right click”	“right click”	
74.	Positive	“shut down”	 “shut down” phrase skewed 0.33 degrees clockwise	“shut down”	“shut down”	
75.	Positive	“windows xp”	 “windows xp” phrase skewed 2.9 degrees counterclockwise	“windows xp”	“windows xp”	

	Positive/ Negative	Word/Phrase	Input	Expected Output	Actual Output	Reason (if negative)
76.	Positive	“connect to”	 “connect to” phrase skewed 2.00 degrees clockwise	“connect to”	“connect to”	
77.	Positive	“great knowledge”	 “great knowledge” phrase skewed 2.05 degrees clockwise	“great knowledge”	“great knowledge”	
78.	Positive	“ourselves”	 “ourselves” word skewed 4.85 degrees counterclockwise	“ourselves”	“ourselves”	
79.	Positive	“paid people”	 “paid people” phrase skewed 0.45 degrees counterclockwise	“paid people”	“paid people”	

	Positive/ Negative	Word/Phrase	Input	Expected Output	Actual Output	Reason (if negative)
80.	Positive	“perceive”	 “perceive” word skewed 0.99 degrees clockwise	“perceive”	“perceive”	
81.	Positive	“world tonight”	 “world tonight” phrase skewed 4.2 degrees clockwise	“world tonight”	“world tonight”	
82.	Negative	“be”	 “be” word not skewed	“be”	“Insufficient Braille dots”	No dots exist in the second column and the first row
83.	Positive	“good news”	 “good news” word skewed 2.17 degrees counterclockwise	“good news”	“good news”	

	Positive/ Negative	Word/Phrase	Input	Expected Output	Actual Output	Reason (if negative)
84.	Positive	“music player”	 “music player” phrase skewed 0.51 degrees clockwise	“music player”	“music player”	
85.	Positive	“controller”	 “controller” word skewed 1.99 degrees counterclockwise	“controller”	“controller”	
86.	Positive	“such spirit”	 “such spirit” phrase skewed 1.77 degrees counterclockwise	“such spirit”	“such spirit”	
87.	Positive	“so young”	 “so young” phrase skewed 1.77 degrees counterclockwise	“so young”	“so young”	

	Positive/ Negative	Word/Phrase	Input	Expected Output	Actual Output	Reason (if negative)
88.	Positive	“counting”	 “counting” word skewed 1.38 degrees counterclockwise	“counting”	“counting”	
89.	Positive	“press enter”	 “press enter” phrase skewed 0.29 degrees counterclockwise	“press enter”	“press enter”	
90.	Positive	“network”	 “network” word skewed 0.84 degrees counterclockwise	“network”	“network”	
91.	Negative	“23”	 “23” number skewed 0.06 degrees counterclockwise	“23”	“blebc”	Numeric Braille is not part of the character database.

	Positive/ Negative	Word/Phrase	Input	Expected Output	Actual Output	Reason (if negative)
92.	Negative	“3:16”	 “3:16” phrase skewed 0.38 degrees counterclockwise	“3:16”	“blecccble after”	Numeric Braille and special characters are not part of the character database.
93.	Negative	“John”	 “John” word skewed 0.1 degrees clockwise	“John”	“No Datajohn”	Only lower case Braille characters are covered in the study. The letter sign before “j” is causing the mismatch.
94.	Positive	“free wind”	 “free wind” phrase skewed 0.31 degrees clockwise	“free wind”	“free wind”	
95.	Positive	“for children”	 “for children” phrase skewed 0.39 degrees clockwise	“for children”	“for children”	

	Positive/ Negative	Word/Phrase	Input	Expected Output	Actual Output	Reason (if negative)
96.	Positive	“angels”	 “angels” word skewed 0.53 degrees clockwise	“angels”	“angels”	
97.	Positive	“demons”	 “demons” word skewed 0.29 degrees counterclockwise	“demons”	“demons”	
98.	Positive	“escape”	 “escape” word skewed 1.58 degrees counterclockwise	“escape”	“escape”	
99.	Positive	“brother”	 “brother” word skewed 0.22 degrees clockwise	“brother”	“brother”	

	Positive/ Negative	Word/Phrase	Input	Expected Output	Actual Output	Reason (if negative)
100.	Positive	“water”	 “water” word skewed 0.84 degrees counterclockwise	“water”	“water”	

A.4
Test Cases

October 15, 2009

To whom it may concern:

This document is to certify that the translation made by the Optical Braille Recognition system created by Martin Luis Faustino, Gian Gerard Libunao and Jessica Cassandra Magbitang is accurate and complies with the general Grade 2 Braille contraction rules. Hence, this system produces accurate results. Moreover the system can be used by the organization for educational and informative purposes.

Lorrie Barboza
Chief Braillist
Resources for the Blind

RESOURCES FOR THE BLIND, INC.
4/F COTI Bldg., 623 EDSA Cubao, Quezon City

Lorrie G. Barboza
Braille Production Supervisor

Mobile: 0928-2414569
09274271775
E-mail: lorrie@blind.org.ph
lorrie21_b@yahoo.com

Office: 726-3021 to 24
Fax: 727-0077
Res: 2355686/2870532
Website: www.blind.org.ph

UNIVERSITY OF SANTO TOMAS
Information and Computer Studies Department

Proponents: Martin Luis R. Faustino Signature: _____
Gian Gerard E. Libunao _____
Jessica Cassandra T. Magbitang _____

Thesis Title: *Enhanced Optical Braille Recognition Using Character Isolation Box and Pattern Generation for Grade 2 Braille*

Thesis Adviser: Asst. Prof. Vergil V. Reyes Signature: _____
Oral Defense Date: Time:
Thesis Oral Defense Panel:

Panel 1 Ms. Charmaine S. Ponay Signature:
Panel 2 Assoc. Prof. Perla P. Cosme Signature:
Panel 3 Mr. Chester Arvin R. Moroles Signature:

Panel's Evaluation of Thesis Oral Defense

A. GRADE

1. ✓ Pass _____ Conditional Pass _____ Fail
2. ✓ No re-oral _____ re-Oral

B. Required revisions and Changes

1. _____ No Revisions ✓ Minor Revision _____ Major Revision
2. Revisions/Changes Submission Date (Specific) _____
3. Revisions/Changes to be reviewed by
   _____ Thesis Oral Defense Panel and Thesis Adviser
   ✓ Thesis Adviser Only

C. Specific Comments on Revisions, Changes and Recommendations

Good! Revise document & put cover page

UNIVERSITY OF SANTO TOMAS
Information and Computer Studies Department

Proponents: Martin Luis R. Faustino Signature: _____
Gian Gerard E. Libunao _____
Jessica Cassandra T. Magbitang _____

Thesis Title: *Enhanced Optical Braille Recognition Using Character Isolation Box and Pattern Generation for Grade 2 Braille*

Thesis Adviser: Asst. Prof. Vergil V. Reyes Signature: _____
Oral Defense Date: Time:
Thesis Oral Defense Panel:

Panel 1 Ms. Charmaine S. Ponay Signature:
Panel 2 Assoc. Prof. Perla P. Cosine Signature:
Panel 3 Mr. Chester Arvin R. Morales Signature: _____

Panel's Evaluation of Thesis Oral Defense

A. GRADE

1. ✓ Pass _____ Conditional Pass _____ Fail
2. ✓ No re-oral _____ re-Oral

B. Required revisions and Changes

1. _____ No Revisions ✓ Minor Revision _____ Major Revision
2. Revisions/Changes Submission Date (Specific) _____
3. Revisions/Changes to be reviewed by
   _____ Thesis Oral Defense Panel and Thesis Adviser
   ✓ Thesis Adviser Only

C. Specific Comments on Revisions, Changes and Recommendations

Background of the Study; Include previous study
More Recommendations
Interface of Software
Test Cases (include the different cases used & summary)
Scope & Limitation (organize)

APPENDIX D

Program Listing

```
function varargout = OBR(varargin)
% OBR M-file for OBR.fig
%      OBR, by itself, creates a new
OBR or raises the existing
%      singleton*.
%
%      H = OBR returns the handle to
a new OBR or the handle to
%      the existing singleton*.
%
%
OBR('CALLBACK',hObject,eventData,han
dles,...) calls the local
%        function named CALLBACK in
OBR.M    with    the    given    input
arguments.
%
%        OBR('Property','Value',...)
creates a new OBR or raises the
%              existing  singleton*.
Starting  from  the  left,  property
value pairs
%      are
%         applied to the GUI before
OBR_OpeningFunction gets called.  An
%      unrecognized property name or
invalid    value    makes    property
application
%       stop.  All inputs are passed
to OBR_OpeningFcn via varargin.
%
%       *See GUI Options on GUIDE's
Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
%    See    also:    GUIDE,    GUIDATA,
GUIHANDLES

% Copyright 2002-2003 The MathWorks,
Inc.

% Edit the above text to modify the
response to help OBR

% Last Modified by GUIDE v2.5 23-
Oct-2009 14:20:06

% Begin initialization code - DO NOT
EDIT
gui_Singleton = 1;
gui_State    =    struct('gui_Name',
mfilename, ...
                   'gui_Singleton',
gui_Singleton, ...
                   'gui_OpeningFcn',
@OBR_OpeningFcn, ...
                   'gui_OutputFcn',
@OBR_OutputFcn, ...
                   'gui_LayoutFcn',
[] , ...
                   'gui_Callback',
[]);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback         =
str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}]         =
gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State,
varargin{:});
end
% End initialization code - DO NOT
EDIT


% --- Executes just before OBR is
made visible.
function     OBR_OpeningFcn(hObject,
eventdata, handles, varargin)
% This function has no output args,
see OutputFcn.
% hObject    handle to figure
%  eventdata    reserved  -  to  be
defined  in  a  future  version  of
MATLAB
% handles     structure with handles
and user data (see GUIDATA)
% varargin    command line arguments
to OBR (see VARARGIN)
set(handles.BrowseImage,'Enable','on
');
set(handles.CleanImage,'Enable','off
');
set(handles.DeskewImage,'Enable','of
f');
set(handles.SegmentImage,  'Enable',
'off');
set(handles.RecognizeImage,'Enable',
'off');
set(handles.ImageDisplay,'xcolor','w
','ycolor','w','xtick',[],'ytick',[]
);
set(handles.Deskew,'xcolor','w','yco
lor','w','xtick',[],'ytick',[]);
set(handles.Backside,'xcolor','w','y
color','w','xtick',[],'ytick',[]);
set(handles.Oneclick,'Enable','off')
;
```

```
% Choose default command line output
for OBR
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes OBR wait for user
response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are
returned to the command line.
function varargout =
OBR_OutputFcn(hObject, eventdata,
handles)
% varargout cell array for
returning output args (see
VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be
defined in a future version of
MATLAB
% handles structure with handles
and user data (see GUIDATA)

% Get default command line output
from handles structure
varargout{1} = handles.output;


% --- Executes on button press in
BrailleContractions.
function
BrailleContractions_Callback(hObject
, eventdata, handles)
% hObject handle to
BrailleContractions (see GCBO)
% eventdata reserved - to be
defined in a future version of
MATLAB
% handles structure with handles
and user data (see GUIDATA)
%set(handles.contractionsmenu,'visib
le','on');
backgroundImage =
importdata('C:\MATLAB701\work\image\
Braille Contractions (new).jpg');
figure('Name','Braille
Alphabet','NumberTitle','off'),imsho
w(backgroundImage);

% --- Executes on button press in
ResetAll.
function ResetAll_Callback(hObject,
eventdata, handles)
% hObject handle to ResetAll (see
GCBO)
% eventdata reserved - to be
defined in a future version of
MATLAB
% handles structure with handles
and user data (see GUIDATA)
cla(handles.ImageDisplay,'reset');
cla(handles.Deskew,'reset');
cla(handles.Backside,'reset');
cla;
set(handles.text1,'String','Browsed
Image');
set(handles.BrowseImage,'Enable','on
');
set(handles.DeskewImage,'Enable','of
f');
set(handles.CleanImage,'Enable','off
');
set(handles.SegmentImage,'Enable','o
ff');
set(handles.RecognizeImage,'Enable',
'off');
set(handles.RecognizedImage,'String'
,' ');
set(handles.ImageDisplay,'xcolor','w
','ycolor','w','xtick',[],'ytick',[]
);
set(handles.Deskew,'xcolor','w','yco
lor','w','xtick',[],'ytick',[]);
set(handles.Backside,'xcolor','w','y
color','w','xtick',[],'ytick',[]);
set(handles.text1,'visible','on');
set(handles.Oneclick,'Enable','off')
;
set(handles.Angle,'String',' ');
clear

% --- Executes on button press in
BrailleAlphabet.
function
BrailleAlphabet_Callback(hObject,
eventdata, handles)
% hObject handle to
BrailleAlphabet (see GCBO)
% eventdata reserved - to be
defined in a future version of
MATLAB
% handles structure with handles
and user data (see GUIDATA)
backgroundImage =
importdata('C:\MATLAB701\work\image\
braille.jpg');
figure('Name','Braille
Alphabet','NumberTitle','off'),imsho
w(backgroundImage);

% --- Executes on button press in
BrowseImage.
function
BrowseImage_Callback(hObject,
eventdata, handles)
% hObject handle to BrowseImage
(see GCBO)
% eventdata reserved - to be
defined in a future version of
MATLAB
```

```
% handles     structure with handles
and user data (see GUIDATA)
set(handles.RecognizedImage,'String'
,' ');
[filename,        pathname]        =
uigetfile({'*.jpg';'*.bmp'},'Pick an
Image(.jpg,.bmp) File');
%[filename,        pathname]        =
uigetfile({'*.jpg';'*.jpeg';'*.bmp'}
,'File
%Selector');
if  ~ischar([pathname  filename])  ||
isempty([pathname filename])
else
info= imfinfo([pathname filename]);
I   =   imread([pathname,filename]);
%Place   the   image   file   to   the
variable I;
save I;
set(handles.text1,'String',info.File
name);
set(handles.Oneclick,'Enable','on');
set(handles.BrowseImage,'Enable','of
f');
set(handles.CleanImage,'Enable','off
');
set(handles.DeskewImage,'Enable','on
');
set(handles.SegmentImage,   'Enable',
'off');
set(handles.RecognizeImage,'Enable',
'off');
axes(handles.ImageDisplay)
image(I);
axis off
save info
end


% --- Executes on button press in
DeSkewImage.
function
DeSkewImage_Callback(hObject,
eventdata, handles)
% hObject     handle to DeSkewImage
(see GCBO)
%  eventdata     reserved  -  to  be
defined  in  a  future  version  of
MATLAB
% handles     structure with handles
and user data (see GUIDATA)

load I;
cropedimage   =   I;   %copies   I   to
cropedimage
save cropedimage;
gray=imadjust(rgb2gray(cropedimage))
; %Grayscales cropedimage and then
increase its contrast
imwrite(gray,'C:\MATLAB701\work\dump
\gray.jpg'); %saves the image bw
imwrite(gray,'C:\MATLAB701\work\dump
\gray.jpg'); %saves the image bw
level=graythresh(gray);
bw=im2bw(gray,level);
imwrite(bw,'C:\MATLAB701\work\dump\b
w.jpg');
img=imread('C:\MATLAB701\work\dump\b
w.jpg');
img=double(img);
inv(:,:,:)=255-img(:,:,:);
inv=uint8(inv);
inv=im2bw(inv);
inv=bwareaopen(inv,30);
imwrite(inv,'C:\MATLAB701\work\dump\
inverse.jpg');

%rotated initalization
rotated = inv;
mediangray=gray;
save mediangray;
totalangle=0;

%Showing the skewed image with lines
L= bwlabel(rotated);
s = regionprops(L,'centroid');
figure('Name','Slopes:   Yellow   are
the
Centers','NumberTitle','off'),imshow
(rotated);
hold on
numObj = numel(s);

if numObj<3

ee='Insufficient Braille Dots'
cla(handles.ImageDisplay,'reset');
cla(handles.Deskew,'reset');
cla(handles.Backside,'reset');
cla;
set(handles.text1,'String','Browsed
Image');
set(handles.BrowseImage,'Enable','on
');
set(handles.DeskewImage,'Enable','of
f');
set(handles.CleanImage,'Enable','off
');
set(handles.SegmentImage,'Enable','o
ff');
set(handles.RecognizeImage,'Enable',
'off');
set(handles.RecognizedImage,'String'
,ee);
set(handles.ImageDisplay,'xcolor','w
','ycolor','w','xtick',[],'ytick',[]
);
set(handles.Deskew,'xcolor','w','yco
lor','w','xtick',[],'ytick',[]);
set(handles.Backside,'xcolor','w','y
color','w','xtick',[],'ytick',[]);
set(handles.text1,'visible','on');
set(handles.Oneclick,'Enable','off')
;
set(LOADING,'visible','off');
clear
```

```
else

for k = 1 :  numObj
    plot(s(k).Centroid(1),
s(k).Centroid(2), 'y*');
    plot(s(k).Centroid(1),
s(k).Centroid(2), 'yo');
    for i=1 : numObj
        if         (s(i).Centroid(1)-
s(k).Centroid(1)) >0
    slope=double((s(i).Centroid(2)-
s(k).Centroid(2))/(s(i).Centroid(1)-
s(k).Centroid(1)));
    distance                          =
sqrt((s(i).Centroid(1)-
s(k).Centroid(1))^2+(s(i).Centroid(2
)-s(k).Centroid(2))^2);
    format long
    slope;
             if  ((slope  <  0.1)  &&
(slope > -0.1)) && distance <150

plot([s(k).Centroid(1)
s(i).Centroid(1)],  [s(k).Centroid(2)
s(i).Centroid(2)],'r');
             end
        end
    end
end

hold off;


%Loop start and angle extraction
loop=1;
angle=1;
z=0;
while z<=5

L= bwlabel(rotated);
s = regionprops(L,'centroid');

%get total number of accepted slopes
ctrr=0;
numObj = numel(s);
for ctr = 1 :  numObj
    for ctr2=1 : numObj
        if      (s(ctr2).Centroid(1)-
s(ctr).Centroid(1))>0

slope=double((s(ctr2).Centroid(2)-
s(ctr).Centroid(2))/(s(ctr2).Centroi
d(1)-s(ctr).Centroid(1)));
    distance                          =
sqrt((s(ctr2).Centroid(1)-
s(ctr).Centroid(1))^2+(s(ctr2).Centr
oid(2)-s(ctr).Centroid(2))^2);
    format long
    slope;
           if loop==1
             if  ((slope  <  0.1)  &&
(slope > -0.1)) && distance <150
                ctrr=ctrr+1;
             end
           else
             if  ((slope  <  0.05)  &&
(slope > -0.05)) && distance <150
                ctrr=ctrr+1;
             end
           end
        end
    end
end

%Declaration of array and pointer
array =1:ctrr;
point=0;

%Extraction of the useful slopes
hold on
numObj = numel(s);
for k = 1 :  numObj
    for i=1 : numObj
        if         (s(i).Centroid(1)-
s(k).Centroid(1)) >0
    slope=double((s(i).Centroid(2)-
s(k).Centroid(2))/(s(i).Centroid(1)-
s(k).Centroid(1)));
    distance                          =
sqrt((s(i).Centroid(1)-
s(k).Centroid(1))^2+(s(i).Centroid(2
)-s(k).Centroid(2))^2);
    format long
    slope;
          if loop==1
             if  ((slope  <  0.1)  &&
(slope > -0.1)) && distance <150
                point=point+1;

array(:,point)=slope;

%plot([s(k).Centroid(1)
s(i).Centroid(1)],  [s(k).Centroid(2)
s(i).Centroid(2)],'r');
             end
           else
             if  ((slope  <  0.05)  &&
(slope > -0.05)) && distance <150
                point=point+1;

array(:,point)=slope;
             end
           end
        end
    end
end
hold off;

array;
medianave=median(array);

%Deskewing
angle=atan(medianave);
angle=(angle*180)/pi;
rotated= imrotate(rotated, angle);
```

```
mediangray=imrotate(mediangray,
angle);
save mediangray;
imwrite(mediangray,'C:\MATLAB701\wor
k\dump\mediangray.jpg');
imwrite(rotated,'C:\MATLAB701\work\d
ump\Rotated.jpg');

clear array;
loop=loop+1;
totalangle=totalangle+angle;
z=z+1;
end

%final display label
figure('Name','Deskew             nth
run','NumberTitle','off'),imshow(rot
ated);
L = bwlabel(rotated);
s = regionprops(L,'centroid');
hold on
numObj = numel(s);
for k = 1 :  numObj
    plot(s(k).Centroid(1),
s(k).Centroid(2), 'y*');
    plot(s(k).Centroid(1),
s(k).Centroid(2), 'yo');

    %backside filtering
    x=s(k).Centroid(1);
    y=s(k).Centroid(2)-7.5;
    plot(x,y, 'b*');
    p=impixel(mediangray,x,y);

    for i=1 : numObj
        if         (s(i).Centroid(1)-
s(k).Centroid(1))>0
    slope=double((s(i).Centroid(2)-
s(k).Centroid(2))/(s(i).Centroid(1)-
s(k).Centroid(1)));
    distance                        =
sqrt((s(i).Centroid(1)-
s(k).Centroid(1))^2+(s(i).Centroid(2
)-s(k).Centroid(2))^2);
    format long

            if loop==1
                if ((slope < 0.1) &&
(slope > -0.1)) && distance <150

plot([s(k).Centroid(1)
s(i).Centroid(1)],  [s(k).Centroid(2)
s(i).Centroid(2)],'r');
                end
            else
                if  ((slope  <  0.05)
&& (slope > -0.05)) && distance <150

plot([s(k).Centroid(1)
s(i).Centroid(1)],  [s(k).Centroid(2)
s(i).Centroid(2)],'r');
                end
            end
        end
    end
end
hold off;


%dis1=mediangray;
%dis1=im2bw(dis1);

dis1=imread('C:\MATLAB701\work\dump\
mediangray.jpg');
dis1 =cat (3, dis1 ,dis1 ,dis1);
axes(handles.Deskew);
image(dis1);
axis off;

totalangle
set(handles.Angle,'String',totalangl
e);

set(handles.BrowseImage,'Enable','of
f');
set(handles.CleanImage,'Enable','on'
);
set(handles.DeskewImage,'Enable','of
f');
set(handles.SegmentImage,  'Enable',
'off');
set(handles.RecognizeImage,'Enable',
'off');
end

% --- Executes on button press in
RecognizeImage.
function
RecognizeImage_Callback(hObject,
eventdata, handles)
%   hObject             handle   to
RecognizeImage (see GCBO)
%  eventdata    reserved  -  to  be
defined  in  a  future  version  of
MATLAB
% handles     structure with handles
and user data (see GUIDATA)

set(LOADING,'Visible','on')


load charcount;
charcount

if charcount==0
ee='No Character Detected'
set(LOADING,'Visible','off')
set(handles.RecognizedImage,'String'
,ee);
set(handles.BrowseImage,'Enable','on
');
set(handles.CleanImage,'Enable','off
');
set(handles.DeskewImage,'Enable','of
f');
```

```
set(handles.SegmentImage,   'Enable',
'off');
set(handles.RecognizeImage,'Enable',
'off');


elseif charcount>=1
load binary;
binary

%for u = 1 : charcount;
u=1
while(u<=charcount)
    z=num2str(binary(1,u));
    b=[z];
    for o = 2 : 6
    a=binary(o,u);
    str1=num2str(a);
    b=[b str1];
    end
    db(u,:)=strvcat(b);
    u=u+1;
end

db

    db1='123456';
    conn  =  database('Braille',  '',
'');

setdbprefs('DataReturnFormat','cella
rray');
    curs   =   exec(conn,   ['select
Letter from Grade1 where ID= ' ''''
db1 '''']);
    curs = fetch(curs);
    bb = curs.data;

    db1='000000';
    conn  =  database('Braille',  '',
'');

setdbprefs('DataReturnFormat','cella
rray');
    curs   =   exec(conn,   ['select
Letter from Grade1 where ID= ' ''''
db1 '''']);
    curs = fetch(curs);
    space = curs.data;

end

if charcount>=1
set(handles.RecognizedImage,'String'
,'Loading...');
db1=[db(1,:)]
met=1;
epox=1;
while(epox<=charcount)
    epox
    lft=0;
    rgt=0;
    if epox<charcount && charcount>1
    db2=[db1 db(epox+1,:) '%']
    elseif charcount==1
    db2='12345'
    end
    conn  =  database('Braille',  '',
'');

setdbprefs('DataReturnFormat','cella
rray')
    curs = exec(conn, ['select Words
from Grade2 where ID like ' '''' db2
''''])
    curs = fetch(curs)
    aa = curs.data

    %Empty query checker
    n = numel(curs.data)
    b=0;
    if n==1
    b = strcmp(bb,aa)
    end
    %left & right checker
    if epox+1<charcount

rgt=strcmp(db(epox+1,:),'000000')
    end
    if  epox-met>=1%error  catcher  if
ctr-met will equal to 0
    lft=strcmp(db(epox-
met,:),'000000')
    end

    %query  for  standalone  grade  2
braille  characters(group)  for  the
first
    %including the first char
    if  (b  ==  1  &&  n  ==  1)  &&  rgt==1
&& met>1 && epox-met<1
        curs  =  exec(conn,  ['select
Words  from  Grade2  where  ID=  '  ''''
db1 '''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa

        spaceb=met-1
        while spaceb>=1
        arrayy(epox-spaceb)=space;
        spaceb=spaceb-1;
        end

        db1=db(epox+1,:)
        met=1
        epox=epox+1;

     %Sure standalone
     elseif  (b  ==  1  &&  n  ==  1)  &&
lft==1 &&  rgt==1 && met>1
        curs  =  exec(conn,  ['select
Words  from  Grade2  where  ID=  '  ''''
db1 '''''])
```

```
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa

        spaceb=met-1
        while spaceb>=1
        arrayy(epox-spaceb)=space;
        spaceb=spaceb-1;
        end

        db1=db(epox+1,:)
        met=1
        epox=epox+1;


      %Prefix
      elseif (b == 1 && n == 1) &&
lft==1 &&  rgt==0  && met>1
         curs = exec(conn, ['select
Words from Pre_fix where ID= ' ''''
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)

        if  (b == 1 && n == 1)
        ppp=met-1
        while ppp>=1
        epox=epox-ppp
        db1=db(epox,:)
        curs = exec(conn, ['select
Letter from Grade1 where ID= ' ''''
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa
        db1=db(epox+1,:)
        met=1
        epox=epox+ppp;
        ppp=ppp-1;
        end

        else
        curs = exec(conn, ['select
Words from Pre_fix where ID= ' ''''
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa

        spaceb=met-1
        while spaceb>=1
        arrayy(epox-spaceb)=space;
        spaceb=spaceb-1;
        end

        db1=db(epox+1,:)
        met=1
        epox=epox+1;
        end

      %Suffix
      elseif (b == 1 && n == 1) &&
lft==0 &&  rgt==1  && met>1

        curs = exec(conn, ['select
Words from Suffix where ID= ' ''''
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)

        if  (b == 1 && n == 1)
        ppp=met-1
        while ppp>=1
        epox=epox-ppp
        db1=db(epox,:)
        curs = exec(conn, ['select
Letter from Grade1 where ID= ' ''''
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa
        db1=db(epox+1,:)
        met=1
        epox=epox+ppp;
        ppp=ppp-1;
        end

        else
        curs = exec(conn, ['select
Words from Suffix where ID= ' ''''
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa

        spaceb=met-1
        while spaceb>=1
        arrayy(epox-spaceb)=space;
        spaceb=spaceb-1;
        end

        db1=db(epox+1,:)
        met=1
        epox=epox+1;
        end


       %Midfix
       elseif (b == 1 && n == 1) &&
lft==0 &&  rgt==0  && met>1
```

```
        curs  =  exec(conn,  ['select
Words from Mid_fix where ID= ' ''''
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)

        if  (b == 1 && n == 1)
        ppp=met-1
        while ppp>=1
        epox=epox-ppp
        db1=db(epox,:)
        curs  =  exec(conn,  ['select
Letter from Grade1 where ID= ' ''''
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa
        db1=db(epox+1,:)
        met=1
        epox=epox+ppp;
        ppp=ppp-1;
        end

        else
        curs  =  exec(conn,  ['select
Words from Mid_fix where ID= ' ''''
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa

        spaceb=met-1
        while spaceb>=1
        arrayy(epox-spaceb)=space;
        spaceb=spaceb-1;
        end

        db1=db(epox+1,:)
        met=1
        epox=epox+1;
        end


        %try
    elseif  (b==1  &&  n==1)&&  met==1
&& lft==1 && rgt==1
        curs  =  exec(conn,  ['select
Words  from  Stand_Alone  where  ID=  '
'''' db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa
        if epox<charcount
        db1=db(epox+1,:)
        end
        met=1
        epox=epox+1;

    elseif  (b==1  &&  n==1)&&  met==1
&& lft==1 && epox+1>charcount
        curs  =  exec(conn,  ['select
Words  from  Stand_Alone  where  ID=  '
'''' db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa
        if epox<charcount
        db1=db(epox+1,:)
        end
        met=1
        epox=epox+1;

     elseif  (b==1  &&  n==1)&&  met==1
&& rgt==1 && epox-met<1
        curs  =  exec(conn,  ['select
Words  from  Stand_Alone  where  ID=  '
'''' db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa
        if epox<charcount
        db1=db(epox+1,:)
        end
        met=1
        epox=epox+1;

        elseif   (b==1   &&   n==1)&&
met==1  &&  epox+1>charcount  &&  epox-
met<1
        curs  =  exec(conn,  ['select
Words  from  Stand_Alone  where  ID=  '
'''' db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa
        if epox<charcount
        db1=db(epox+1,:)
        end
        met=1
        epox=epox+1;
%not sure



    %query for grade 1 braille(since
grade  2  1  braille  char  standalone
are unique)
    elseif    (b==1  &&  n  ==  1)  &&
met==1
        curs  =  exec(conn,  ['select
Letter from Grade1 where ID= ' ''''
db1 ''''])
```

```
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)

        %query for grade 2 single
braille char standalone(condition is
if
        %there's no match in the
grade 1 DB)
        if (b == 1 && n == 1)
        curs = exec(conn, ['select
Words from Grade2 where ID= ' '''' 
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)

            if n>1 && epox-met<1 &&
rgt==0
                curs = exec(conn,
['select Words from Pre_Fix where
ID= ' '''' db1 ''''])
                curs = fetch(curs)
                aa = curs.data
                n = numel(curs.data)
                b=strcmp(bb,aa)

            elseif n>1 && lft==0 &&
rgt==1 && epox==1
                curs = exec(conn,
['select Words from Stand_Alone
where ID= ' '''' db1 ''''])
                curs = fetch(curs)
                aa = curs.data
                n = numel(curs.data)
                b=strcmp(bb,aa)

            elseif n>1 && lft==0 &&
rgt==0
                curs = exec(conn,
['select Words from Mid_Fix where
ID= ' '''' db1 ''''])
                curs = fetch(curs)
                aa = curs.data
                n = numel(curs.data)
                b=strcmp(bb,aa)

            elseif n>1 && lft==0 &&
rgt==1
                curs = exec(conn,
['select Words from Suffix where ID=
' '''' db1 ''''])
                curs = fetch(curs)
                aa = curs.data
                n = numel(curs.data)
                b=strcmp(bb,aa)

            elseif n>1 && lft==1 &&
rgt==0
                curs = exec(conn,
['select Words from Pre_Fix where
ID= ' '''' db1 ''''])
                curs = fetch(curs)
                aa = curs.data
                n = numel(curs.data)
                b=strcmp(bb,aa)

            elseif n>1 && lft==1 &&
rgt==1
                curs = exec(conn,
['select Words from Stand_Alone
where ID= ' '''' db1 ''''])
                curs = fetch(curs)
                aa = curs.data
                n = numel(curs.data)
                b=strcmp(bb,aa)

            end
        end

        %proceed here if match is
already found
        arrayy(epox) = aa
        if epox<charcount
        db1=db(epox+1,:)
        end
        met=1
        epox=epox+1;

    else
    %db1 incrementation if there's
no match in any of the database
    if epox+1<=charcount
    db1=[db1 db(epox+1,:)]
    met=met+1
    epox=epox+1;

    %trial
    else
        curs = exec(conn, ['select
Words from Grade2 where ID= ' '''' 
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa

        spaceb=met-1
        while spaceb>=1
        arrayy(epox-spaceb)=space;
        spaceb=spaceb-1;
        end

        epox=epox+1
    end
    %trial
end
    close(conn)
    close(curs)
```

```
end

result = [arrayy(:)]

ee=strvcat(arrayy(1));
for cc = 2 : charcount
ff=strvcat(arrayy(cc));
ee=[ee ff];
end
ee

set(LOADING,'Visible','off')
set(handles.RecognizedImage,'String'
,ee);


set(handles.BrowseImage,'Enable','on
');
set(handles.CleanImage,'Enable','off
');
set(handles.DeskewImage,'Enable','of
f');
set(handles.SegmentImage,  'Enable',
'off');
set(handles.RecognizeImage,'Enable',
'off');
end

% --- Executes on button press in
AnalyzeImage.
function
AnalyzeImage_Callback(hObject,
eventdata, handles)
% hObject    handle to AnalyzeImage
(see GCBO)
% eventdata    reserved - to be
defined in a future version of
MATLAB
% handles    structure with handles
and user data (see GUIDATA)



% --- Executes on button press in
Exit.
function      Exit_Callback(hObject,
eventdata, handles)
% hObject      handle to Exit (see
GCBO)
% eventdata    reserved - to be
defined in a future version of
MATLAB
% handles    structure with handles
and user data (see GUIDATA)
close

%handles.output = hObject;
%if             ispc             &&
isequal(get(hObject,'BackgroundColor
'),
get(0,'defaultUicontrolBackgroundCol
or'))
%
set(hObject,'BackgroundColor','white
');
%end

% --- Executes when figure1 is
resized.
function  figure1_ResizeFcn(hObject,
eventdata, handles)
% hObject    handle to figure1 (see
GCBO)
% eventdata    reserved - to be
defined in a future version of
MATLAB
% handles    structure with handles
and user data (see GUIDATA)


% --- Executes on button press in
pushbutton16.
function
pushbutton16_Callback(hObject,
eventdata, handles)
% hObject    handle to pushbutton16
(see GCBO)
% eventdata    reserved - to be
defined in a future version of
MATLAB
% handles    structure with handles
and user data (see GUIDATA)

%erasing
load mediangray;
for po =1 : 3
rotated=imread('C:\MATLAB701\work\du
mp\rotated.jpg');
rotated=uint8(rotated);
rotated=im2bw(rotated);

%figure('Name','Erasing','NumberTitl
e','off'),imshow(mediangray);
hold on
L = bwlabel(rotated);
s = regionprops(L,'centroid');
numObj = numel(s);
for k = 1 :  numObj

    %backside filtering
    x=s(k).Centroid(1);
    y=s(k).Centroid(2);
    x2=s(k).Centroid(1)-4;
    y2=s(k).Centroid(2)-7;
    plot(x2,y2, 'b.');
    p=impixel(mediangray,x2,y2);
    if p(1,1)<=210
       plot(s(k).Centroid(1),
s(k).Centroid(2),'g*');
       plot(s(k).Centroid(1),
s(k).Centroid(2),'go');
        I2                          =
imread('C:\MATLAB701\work\dump\media
ngray.jpg');
```

```
        K                           =
imread('C:\MATLAB701\work\dump\rotat
ed.jpg');
        col  =  [(s(k).Centroid(1)-7)
(s(k).Centroid(1)+7)
(s(k).Centroid(1)+7)
(s(k).Centroid(1)-7)];
        row  =  [(s(k).Centroid(2)-7)
(s(k).Centroid(2)-7)
(s(k).Centroid(2)+7)
(s(k).Centroid(2)+7)];
        J = roifill(I2,col,row);
        M = roifill(K,col,row);

imwrite(J,'C:\MATLAB701\work\dump\me
diangray.jpg');

imwrite(M,'C:\MATLAB701\work\dump\ro
tated.jpg');
        rotated=M;
        mediangray=J;
        save mediangray;
    end

end
hold off;
end

dis2=imread('C:\MATLAB701\work\dump\
mediangray.jpg');
dis2 =cat (3, dis2 ,dis2 ,dis2);
axes(handles.Backside);
image(dis2);
axis off;



set(handles.BrowseImage,'Enable','of
f');
set(handles.CleanImage,'Enable','off
');
set(handles.DeskewImage,'Enable','of
f');
set(handles.SegmentImage,   'Enable',
'on');
set(handles.RecognizeImage,'Enable',
'off');



% --- Executes on button press in
SegmentImage.
function
SegmentImage_Callback(hObject,
eventdata, handles)
% hObject     handle to SegmentImage
(see GCBO)
%  eventdata     reserved  -  to  be
defined  in  a  future  version  of
MATLAB
% handles     structure with handles
and user data (see GUIDATA)
%segmentation

rotated=imread('C:\MATLAB701\work\du
mp\rotated.jpg');
rotated=uint8(rotated);
rotated=im2bw(rotated);
figure('Name','Segments','NumberTitl
e','off'),imshow(rotated);

L = bwlabel(rotated);
s = regionprops(L,'centroid');
hold on
numObj = numel(s);

if numObj>0
%xholder
xholder=1:numObj;
for iz = 1 : numObj
xholder(:,iz)=s(iz).Centroid(1);
format short;
end
xholder

%yholder
yholder=1:numObj;
for iz = 1 : numObj
yholder(:,iz)=s(iz).Centroid(2);
format short;
end
yholder

%xyholder
xyholder=[1:numObj;1:numObj];
for iz = 1 : numObj
xyholder(1,iz)=s(iz).Centroid(1);
xyholder(2,iz)=s(iz).Centroid(2);
format short;
end
xyholder



y1=-999;
y2=-999;
y3=-999;

for pt=1 : numObj
    y2=(yholder(:,pt));

    for pt2=1 : numObj
        if  yholder(:,pt2)>=y2-22  &&
yholder(:,pt2)<y2-9
        y1=yholder(:,pt2);
        break;
        end
    end

    for pt2=1 : numObj
        if  yholder(:,pt2)>=y2+9  &&
yholder(:,pt2)<y2+22
        y3=yholder(:,pt2);
        break;
        end
    end
```

```
if y3>=y2+9 && y3<y2+22 && y1>=y2-22
&& y1<y2-9
break;
end

end


%charcount
charcount=1;
pt=0;
firstx=xholder(:,1);
secondx=0;
for pt=1 : numObj
    save charcount;
        if  (xholder(:,pt)>=firstx+16
&&    xholder(:,pt)<firstx+21)     &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4)
            secondx=xholder(:,pt);
            break;
        elseif
(xholder(:,pt)>=firstx+21          &&
xholder(:,pt)<=firstx+32)          &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+14)
            secondx=firstx;
            firstx=secondx-20;
            break;
         elseif
(xholder(:,pt)>=firstx+47          &&
xholder(:,pt)<=firstx+51)          &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4)
            secondx=firstx+20;
            break;
         %subject to change
         elseif
(xholder(:,pt)>=firstx+74          &&
xholder(:,pt)<=firstx+83)          &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4)
            secondx=firstx;
            firstx=secondx-20;
            break;
         elseif
(xholder(:,pt)>=firstx+96          &&
xholder(:,pt)<=firstx+102)         &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4)
            secondx=firstx+20;
            break;
          elseif
(xholder(:,pt)>=firstx+113         &&
xholder(:,pt)<=firstx+124)         &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4)
            secondx=firstx+20;
            break;
         %end subject to change
        else
            firstx=xholder(:,pt);
        end
end

plot(firstx,80,'g*');
plot(secondx,80,'g*');
plot([firstx secondx], [80 80],'b');

 for i = 1 : 50

    for pt=1:numObj
        if
(xholder(:,pt)>=secondx+21         &&
xholder(:,pt)<secondx+34)          &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4)
            firstx=xholder(:,pt);
            for pt2=1:numObj
                if
(xholder(:,pt2)>=firstx+16         &&
xholder(:,pt2)<firstx+21)          &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4)

secondx=xholder(:,pt2);
                    break;
                else

secondx=firstx+20;
                    break;
                end
            end
            charcount=charcount+1;
            break;
        elseif
(xholder(:,pt)>=secondx+44         &&
xholder(:,pt)<secondx+53)          &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4)
            secondx=xholder(:,pt);
            firstx=secondx-20;
            charcount=charcount+1;
            break;
        elseif
(xholder(:,pt)>=secondx+74         &&
xholder(:,pt)<secondx+83)          &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4)
            firstx=xholder(:,pt);
            for pt2=1:numObj
                if
(xholder(:,pt2)>=firstx+16         &&
xholder(:,pt2)<firstx+21)          &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4)

secondx=xholder(:,pt2);
                    break;
                else

secondx=firstx+20;
                    break;
                end
            end
            charcount=charcount+2;
```

```
            break;
        elseif
(xholder(:,pt)>=secondx+96          &&
xholder(:,pt)<secondx+102)          &&
(yholder(:,pt)>=y1-4                &&
yholder(:,pt)<=y3+4)

            secondx=xholder(:,pt);
            firstx=secondx-20;
            charcount=charcount+2;
            break;

        end
    end
plot(firstx,80,'g*');
plot(secondx,80,'g*');
plot([firstx secondx], [80 80],'b');
end

charcount
save charcount;



y1
y2
y3

%test

binary=[1:charcount  ;  1:charcount  ;
1:charcount    ;    1:charcount    ;
1:charcount ; 1:charcount];
for cl=1:6
    for cl2=1:charcount
        binary(cl,cl2)=0;
    end
end

marker=1;
firstx=xholder(:,1);
secondx=0;
for pt=1 : numObj
        if  (xholder(:,pt)>=firstx+16
&&    xholder(:,pt)<firstx+21)     &&
(yholder(:,pt)>=y1-4                &&
yholder(:,pt)<=y3+4)
           secondx=xholder(:,pt);
           break;

        elseif
(xholder(:,pt)>=firstx+21           &&
xholder(:,pt)<=firstx+32)           &&
(yholder(:,pt)>=y1-4                &&
yholder(:,pt)<=y3+4)
            secondx=firstx;
            firstx=secondx-20;
            break;

        elseif
(xholder(:,pt)>=firstx+47           &&
xholder(:,pt)<=firstx+51)           &&
(yholder(:,pt)>=y1-4                &&
yholder(:,pt)<=y3+4)
            secondx=firstx+20;
            break;

        else
            firstx=xholder(:,pt);
        end
end

                    for  pt2  =  1  :
numObj
                    if
xyholder(1,pt2)>=firstx-10          &&
xyholder(1,pt2)<=firstx+10
                    if
xyholder(2,pt2)>=y1-8               &&
xyholder(2,pt2)<=y1+8

binary(1,marker)=1;
                    elseif
xyholder(2,pt2)>=y2-8               &&
xyholder(2,pt2)<=y2+8

binary(2,marker)=1;
                    elseif
xyholder(2,pt2)>=y3-8               &&
xyholder(2,pt2)<=y3+8

binary(3,marker)=1;
                    end
                    end
                    end

                    for  pt2  =  1  :
numObj
                    if
xyholder(1,pt2)>=secondx-10         &&
xyholder(1,pt2)<=secondx+10
                    if
xyholder(2,pt2)>=y1-8               &&
xyholder(2,pt2)<=y1+4

binary(4,marker)=1;
                    elseif
xyholder(2,pt2)>=y2-8               &&
xyholder(2,pt2)<=y2+8

binary(5,marker)=1;
                    elseif
xyholder(2,pt2)>=y3-8               &&
xyholder(2,pt2)<=y3+8

binary(6,marker)=1;
                    end
                    end
                    end



for i = 1 : 50
    for pt=1:numObj
```

```
        if
(xholder(:,pt)>=secondx+21         &&
xholder(:,pt)<secondx+32)          &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4)
            firstx=xholder(:,pt);

            for pt2=1:numObj
                if
(xholder(:,pt2)>=firstx+16         &&
xholder(:,pt2)<firstx+21)          &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4)

secondx=xholder(:,pt2);
                    break;
                else

secondx=firstx+20;
                    break;
                end
            end
            marker=marker+1;
            break;

        elseif
(xholder(:,pt)>=secondx+47         &&
xholder(:,pt)<secondx+51)          &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4)
            secondx=xholder(:,pt);
            firstx=secondx-20;
            marker=marker+1;
            break;

        elseif
(xholder(:,pt)>=secondx+74         &&
xholder(:,pt)<secondx+83)          &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4)
            firstx=xholder(:,pt);

            for pt2=1:numObj
                if
(xholder(:,pt2)>=firstx+16         &&
xholder(:,pt2)<firstx+21)          &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4)

secondx=xholder(:,pt2);
                break;

                else
                secondx=firstx+20;
                break;
                end
            end
            marker=marker+2;
            break;

        elseif
(xholder(:,pt)>=secondx+96         &&
xholder(:,pt)<secondx+102)         &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4)

secondx=xholder(:,pt);
                firstx=secondx-20;
                marker=marker+2;
            break;

                    end

    end


                    for  pt2  =  1  :
numObj
                    if
xyholder(1,pt2)>=firstx-10         &&
xyholder(1,pt2)<=firstx+10
                    if
xyholder(2,pt2)>=y1-8              &&
xyholder(2,pt2)<=y1+8

binary(1,marker)=1;
                    elseif
xyholder(2,pt2)>=y2-8              &&
xyholder(2,pt2)<=y2+8

binary(2,marker)=1;
                    elseif
xyholder(2,pt2)>=y3-8              &&
xyholder(2,pt2)<=y3+8

binary(3,marker)=1;
                    end
                    end
                    end

                    for  pt2  =  1  :
numObj
                    if
xyholder(1,pt2)>=secondx-10        &&
xyholder(1,pt2)<=secondx+10
                    if
xyholder(2,pt2)>=y1-8              &&
xyholder(2,pt2)<=y1+8

binary(4,marker)=1;
                    elseif
xyholder(2,pt2)>=y2-8              &&
xyholder(2,pt2)<=y2+8

binary(5,marker)=1;
                    elseif
xyholder(2,pt2)>=y3-8              &&
xyholder(2,pt2)<=y3+8

binary(6,marker)=1;
                    end
                    end
                    end

save binary;
end
```

```
elseif numObj==0
charcount=0
save charcount;
end

set(handles.BrowseImage,'Enable','of
f');
set(handles.CleanImage,'Enable','off
');
set(handles.DeskewImage,'Enable','of
f');
set(handles.SegmentImage,   'Enable',
'off');
set(handles.RecognizeImage,'Enable',
'on');



% --- Executes on button press in
Oneclick.
function   Oneclick_Callback(hObject,
eventdata, handles)
% hObject    handle to Oneclick (see
GCBO)
%  eventdata     reserved  -  to  be
defined  in  a  future  version  of
MATLAB
% handles     structure with handles
and user data (see GUIDATA)


set(LOADING,'Visible','on')


load I;
cropedimage   =   I;   %copies   I   to
cropedimage
save cropedimage;
gray=imadjust(rgb2gray(cropedimage))
; %Grayscales cropedimage and then
increase its contrast
imwrite(gray,'C:\MATLAB701\work\dump
\gray.jpg'); %saves the image bw
imwrite(gray,'C:\MATLAB701\work\dump
\gray.jpg'); %saves the image bw
level=graythresh(gray);
bw=im2bw(gray,level);
imwrite(bw,'C:\MATLAB701\work\dump\b
w.jpg');
img=imread('C:\MATLAB701\work\dump\b
w.jpg');
img=double(img);
inv(:,:,:)=255-img(:,:,:);
inv=uint8(inv);
inv=im2bw(inv);
inv=bwareaopen(inv,30);
imwrite(inv,'C:\MATLAB701\work\dump\
inverse.jpg');

%rotated initalization
rotated = inv;
mediangray=gray;
save mediangray;
totalangle=0;

%Showing the skewed image with lines
L= bwlabel(rotated);
s = regionprops(L,'centroid');
%figure('Name','Slopes','NumberTitle
','off'),imshow(rotated);
%title('Yellow are the centers');
hold on
numObj = numel(s);

if numObj<3

ee='Insufficient Braille Dots'
cla(handles.ImageDisplay,'reset');
cla(handles.Deskew,'reset');
cla(handles.Backside,'reset');
cla;
set(handles.text1,'String','Browsed
Image');
set(handles.BrowseImage,'Enable','on
');
set(handles.DeskewImage,'Enable','of
f');
set(handles.CleanImage,'Enable','off
');
set(handles.SegmentImage,'Enable','o
ff');
set(handles.RecognizeImage,'Enable',
'off');
set(handles.RecognizedImage,'String'
,ee);
set(handles.ImageDisplay,'xcolor','w
','ycolor','w','xtick',[],'ytick',[]
);
set(handles.Deskew,'xcolor','w','yco
lor','w','xtick',[],'ytick',[]);
set(handles.Backside,'xcolor','w','y
color','w','xtick',[],'ytick',[]);
set(handles.text1,'visible','on');
set(handles.Oneclick,'Enable','off')
;
set(LOADING,'visible','off');
clear



else
for k = 1 :  numObj
    for i=1 : numObj
        if         (s(i).Centroid(1)-
s(k).Centroid(1)) >0
    slope=double((s(i).Centroid(2)-
s(k).Centroid(2))/(s(i).Centroid(1)-
s(k).Centroid(1)));
    distance                        =
sqrt((s(i).Centroid(1)-
s(k).Centroid(1))^2+(s(i).Centroid(2
)-s(k).Centroid(2))^2);
    format long
    slope;
             if  ((slope  <  0.1)  &&
(slope > -0.1)) && distance <150
             end
```

```
        end
    end
end

hold off;


%Loop start and angle extraction
loop=1;
angle=1;
z=0;
while z<=5

L= bwlabel(rotated);
s = regionprops(L,'centroid');

%get total number of accepted slopes
ctrr=0;
numObj = numel(s);
for ctr = 1 :  numObj
    for ctr2=1 : numObj
        if      (s(ctr2).Centroid(1)-
s(ctr).Centroid(1))>0

slope=double((s(ctr2).Centroid(2)-
s(ctr).Centroid(2))/(s(ctr2).Centroi
d(1)-s(ctr).Centroid(1)));
    distance                        =
sqrt((s(ctr2).Centroid(1)-
s(ctr).Centroid(1))^2+(s(ctr2).Centr
oid(2)-s(ctr).Centroid(2))^2);
    format long
    slope;
           if loop==1
             if  ((slope  <  0.1)  &&
(slope > -0.1)) && distance <150
                 ctrr=ctrr+1;
             end
           else
             if  ((slope  <  0.05)  &&
(slope > -0.05)) && distance <150
                 ctrr=ctrr+1;
             end
           end
        end
    end
end

%Declaration of array and pointer
array =1:ctrr;
point=0;

%Extraction of the useful slopes
hold on
numObj = numel(s);
for k = 1 :  numObj
    for i=1 : numObj
        if         (s(i).Centroid(1)-
s(k).Centroid(1)) >0
    slope=double((s(i).Centroid(2)-
s(k).Centroid(2))/(s(i).Centroid(1)-
s(k).Centroid(1)));
    distance                        =
sqrt((s(i).Centroid(1)-
s(k).Centroid(1))^2+(s(i).Centroid(2
)-s(k).Centroid(2))^2);
    format long
    slope;
          if loop==1
             if  ((slope  <  0.1)  &&
(slope > -0.1)) && distance <150
                point=point+1;

array(:,point)=slope;

%plot([s(k).Centroid(1)
s(i).Centroid(1)],  [s(k).Centroid(2)
s(i).Centroid(2)],'r');
            end
           else
             if  ((slope  <  0.05)  &&
(slope > -0.05)) && distance <150
                point=point+1;

array(:,point)=slope;
            end
           end
        end
    end
end
hold off;

array;
medianave=median(array);

%Deskewing
angle=atan(medianave);
angle=(angle*180)/pi;
rotated= imrotate(rotated, angle);
mediangray=imrotate(mediangray,
angle);
save mediangray;
imwrite(mediangray,'C:\MATLAB701\wor
k\dump\mediangray.jpg');
imwrite(rotated,'C:\MATLAB701\work\d
ump\Rotated.jpg');

clear array;
loop=loop+1;
totalangle=totalangle+angle;
z=z+1;
end

%final display label
L = bwlabel(rotated);
s = regionprops(L,'centroid');
hold on
numObj = numel(s);
for k = 1 :  numObj
    %backside filtering
    x=s(k).Centroid(1);
    y=s(k).Centroid(2)-7.5;
    p=impixel(mediangray,x,y);

    for i=1 : numObj
```

```
        if         (s(i).Centroid(1)-
s(k).Centroid(1))>0
    slope=double((s(i).Centroid(2)-
s(k).Centroid(2))/(s(i).Centroid(1)-
s(k).Centroid(1)));
    distance                        =
sqrt((s(i).Centroid(1)-
s(k).Centroid(1))^2+(s(i).Centroid(2
)-s(k).Centroid(2))^2);
    format long

            if loop==1
                if ((slope < 0.1) &&
(slope > -0.1)) && distance <150
                end
            else
                if  ((slope  <  0.05)
&& (slope > -0.05)) && distance <150
                end
            end
        end
    end
end
hold off;


dis1=imread('C:\MATLAB701\work\dump\
mediangray.jpg');
dis1 =cat (3, dis1 ,dis1 ,dis1);
axes(handles.Deskew);
image(dis1);
axis off;

totalangle

%erasing
load mediangray;
for po =1 : 3
rotated=imread('C:\MATLAB701\work\du
mp\rotated.jpg');
rotated=uint8(rotated);
rotated=im2bw(rotated);

%figure('Name','Erasing','NumberTitl
e','off'),imshow(mediangray);
hold on
L = bwlabel(rotated);
s = regionprops(L,'centroid');
numObj = numel(s);
for k = 1 :  numObj

    %backside filtering
    x=s(k).Centroid(1);
    y=s(k).Centroid(2);
    x2=s(k).Centroid(1)-4;
    y2=s(k).Centroid(2)-7;
    p=impixel(mediangray,x2,y2);
    if p(1,1)<=210
        I2                          =
imread('C:\MATLAB701\work\dump\media
ngray.jpg');
        K                           =
imread('C:\MATLAB701\work\dump\rotat
ed.jpg');
        col  =  [(s(k).Centroid(1)-7)
(s(k).Centroid(1)+7)
(s(k).Centroid(1)+7)
(s(k).Centroid(1)-7)];
        row  =  [(s(k).Centroid(2)-7)
(s(k).Centroid(2)-7)
(s(k).Centroid(2)+7)
(s(k).Centroid(2)+7)];
        J = roifill(I2,col,row);
        M = roifill(K,col,row);

imwrite(J,'C:\MATLAB701\work\dump\me
diangray.jpg');

imwrite(M,'C:\MATLAB701\work\dump\ro
tated.jpg');
        rotated=M;
        mediangray=J;
        save mediangray;
    end

end
hold off;
end



dis2=imread('C:\MATLAB701\work\dump\
mediangray.jpg');
dis2 =cat (3, dis2 ,dis2 ,dis2);
axes(handles.Backside);
image(dis2);
axis off;


%segmentation
rotated=imread('C:\MATLAB701\work\du
mp\rotated.jpg');
rotated=uint8(rotated);
rotated=im2bw(rotated);

L = bwlabel(rotated);
s = regionprops(L,'centroid');
hold on
numObj = numel(s);

if numObj>0
%xholder
xholder=1:numObj;
for iz = 1 : numObj
xholder(:,iz)=s(iz).Centroid(1);
format short;
end
xholder

%yholder
yholder=1:numObj;
for iz = 1 : numObj
yholder(:,iz)=s(iz).Centroid(2);
format short;
```

```
end
yholder

%xyholder
xyholder=[1:numObj;1:numObj];
for iz = 1 : numObj
xyholder(1,iz)=s(iz).Centroid(1);
xyholder(2,iz)=s(iz).Centroid(2);
format short;
end
xyholder


y1=-999;
y2=-999;
y3=-999;

for pt=1 : numObj
    y2=(yholder(:,pt));

    for pt2=1 : numObj
        if  yholder(:,pt2)>=y2-22  &&
yholder(:,pt2)<y2-9
        y1=yholder(:,pt2);
        break;
        end
    end

    for pt2=1 : numObj
        if  yholder(:,pt2)>=y2+9  &&
yholder(:,pt2)<y2+22
        y3=yholder(:,pt2);
        break;
        end
    end

if y3>=y2+9 && y3<y2+22 && y1>=y2-22
&& y1<y2-9
break;
end

end

%charcount
charcount=1;
pt=0;
firstx=xholder(:,1);
secondx=0;
for pt=1 : numObj
    save charcount;
        if  (xholder(:,pt)>=firstx+16
&&    xholder(:,pt)<firstx+21)    &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4);
            secondx=xholder(:,pt);
            break;
        elseif
(xholder(:,pt)>=firstx+21          &&
xholder(:,pt)<=firstx+32)          &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+14);
            secondx=firstx;
            firstx=secondx-20;
            break;
        elseif
(xholder(:,pt)>=firstx+47          &&
xholder(:,pt)<=firstx+51)          &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4);
            secondx=firstx+20;
            break;
        %subject to change
        elseif
(xholder(:,pt)>=firstx+74          &&
xholder(:,pt)<=firstx+83)          &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4);
            secondx=firstx;
            firstx=secondx-20;
            break;
        elseif
(xholder(:,pt)>=firstx+96          &&
xholder(:,pt)<=firstx+102)         &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4);
            secondx=firstx+20;
            break;
         elseif
(xholder(:,pt)>=firstx+113         &&
xholder(:,pt)<=firstx+124)         &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4);
            secondx=firstx+20;
            break;
         %end subject to change
        else
            firstx=xholder(:,pt);
        end
end

plot(firstx,80,'g*');
plot(secondx,80,'g*');
plot([firstx secondx], [80 80],'b');

 for i = 1 : 50

    for pt=1:numObj
        if
(xholder(:,pt)>=secondx+21         &&
xholder(:,pt)<secondx+34)          &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4);
            firstx=xholder(:,pt);
            for pt2=1:numObj
                if
(xholder(:,pt2)>=firstx+16         &&
xholder(:,pt2)<firstx+21)          &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4);

secondx=xholder(:,pt2);
                    break;
                else

secondx=firstx+20;
                    break;
```

```
                end
            end
            charcount=charcount+1;
            break;
        elseif
(xholder(:,pt)>=secondx+44          &&
xholder(:,pt)<secondx+53)           &&
(yholder(:,pt)>=y1-4                &&
yholder(:,pt)<=y3+4);
            secondx=xholder(:,pt);
            firstx=secondx-20;
            charcount=charcount+1;
            break;
        elseif
(xholder(:,pt)>=secondx+74          &&
xholder(:,pt)<secondx+83)           &&
(yholder(:,pt)>=y1-4                &&
yholder(:,pt)<=y3+4);;
            firstx=xholder(:,pt);
            for pt2=1:numObj
                if
(xholder(:,pt2)>=firstx+16          &&
xholder(:,pt2)<firstx+21)           &&
(yholder(:,pt)>=y1-4                &&
yholder(:,pt)<=y3+4);

secondx=xholder(:,pt2);
                    break;
                else

secondx=firstx+20;
                    break;
                end
            end
            charcount=charcount+2;
            break;
        elseif
(xholder(:,pt)>=secondx+96          &&
xholder(:,pt)<secondx+102)          &&
(yholder(:,pt)>=y1-4                &&
yholder(:,pt)<=y3+4);

            secondx=xholder(:,pt);
            firstx=secondx-20;
            charcount=charcount+2;
            break;

        end
    end
plot(firstx,80,'g*');
plot(secondx,80,'g*');
plot([firstx secondx], [80 80],'b');
end

charcount
save charcount;




y1
y2
y3
```

```
%test

binary=[1:charcount  ;  1:charcount  ;
1:charcount    ;    1:charcount    ;
1:charcount ; 1:charcount];
for cl=1:6
    for cl2=1:charcount
        binary(cl,cl2)=0;
    end
end

marker=1;
firstx=xholder(:,1);
secondx=0;
for pt=1 : numObj
        if  (xholder(:,pt)>=firstx+16
&&    xholder(:,pt)<firstx+21)     &&
(yholder(:,pt)>=y1-4                &&
yholder(:,pt)<=y3+4);
           secondx=xholder(:,pt);
           break;

        elseif
(xholder(:,pt)>=firstx+21           &&
xholder(:,pt)<=firstx+32)           &&
(yholder(:,pt)>=y1-4                &&
yholder(:,pt)<=y3+4);
            secondx=firstx;
            firstx=secondx-20;
            break;

        elseif
(xholder(:,pt)>=firstx+47           &&
xholder(:,pt)<=firstx+51)           &&
(yholder(:,pt)>=y1-4                &&
yholder(:,pt)<=y3+4);
            secondx=firstx+20;
            break;

        else
            firstx=xholder(:,pt);
        end
end

                    for  pt2  =  1  :
numObj
                    if
xyholder(1,pt2)>=firstx-10          &&
xyholder(1,pt2)<=firstx+10;
                    if
xyholder(2,pt2)>=y1-8               &&
xyholder(2,pt2)<=y1+8;

binary(1,marker)=1;
                    elseif
xyholder(2,pt2)>=y2-8               &&
xyholder(2,pt2)<=y2+8;

binary(2,marker)=1;
                    elseif
xyholder(2,pt2)>=y3-8               &&
xyholder(2,pt2)<=y3+8;
```

```
binary(3,marker)=1;
                    end
                    end
                    end

                    for  pt2  =  1  :
numObj
                    if
xyholder(1,pt2)>=secondx-10        &&
xyholder(1,pt2)<=secondx+10;
                    if
xyholder(2,pt2)>=y1-8              &&
xyholder(2,pt2)<=y1+4

binary(4,marker)=1;
                    elseif
xyholder(2,pt2)>=y2-8              &&
xyholder(2,pt2)<=y2+8;

binary(5,marker)=1;
                    elseif
xyholder(2,pt2)>=y3-8              &&
xyholder(2,pt2)<=y3+8;

binary(6,marker)=1;
                    end
                    end
                    end



for i = 1 : 50;
    for pt=1:numObj;
        if
(xholder(:,pt)>=secondx+21         &&
xholder(:,pt)<secondx+32)          &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4);
            firstx=xholder(:,pt);

            for pt2=1:numObj;
                if
(xholder(:,pt2)>=firstx+16         &&
xholder(:,pt2)<firstx+21)          &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4);

secondx=xholder(:,pt2);
                    break;
                else

secondx=firstx+20;
                    break;
                end
            end
            marker=marker+1;
            break;

        elseif
(xholder(:,pt)>=secondx+47         &&
xholder(:,pt)<secondx+51)          &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4);
            secondx=xholder(:,pt);
            firstx=secondx-20;
            marker=marker+1;
            break;

        elseif
(xholder(:,pt)>=secondx+74         &&
xholder(:,pt)<secondx+83)          &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4);
            firstx=xholder(:,pt);

            for pt2=1:numObj;
                if
(xholder(:,pt2)>=firstx+16         &&
xholder(:,pt2)<firstx+21)          &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4);

secondx=xholder(:,pt2);
                break;

                else
                secondx=firstx+20;
                break;
                end
            end
            marker=marker+2;
            break;

        elseif
(xholder(:,pt)>=secondx+96         &&
xholder(:,pt)<secondx+102)         &&
(yholder(:,pt)>=y1-4               &&
yholder(:,pt)<=y3+4);

secondx=xholder(:,pt);
                firstx=secondx-20;
                marker=marker+2;
            break;

                    end

    end


                    for  pt2  =  1  :
numObj;
                    if
xyholder(1,pt2)>=firstx-10         &&
xyholder(1,pt2)<=firstx+10;
                    if
xyholder(2,pt2)>=y1-8              &&
xyholder(2,pt2)<=y1+8;

binary(1,marker)=1;
                    elseif
xyholder(2,pt2)>=y2-8              &&
xyholder(2,pt2)<=y2+8;

binary(2,marker)=1;
```

```
                    elseif
xyholder(2,pt2)>=y3-8               &&
xyholder(2,pt2)<=y3+8;

binary(3,marker)=1;
                    end
                    end
                    end

                    for  pt2  =  1  :
numObj;
                    if
xyholder(1,pt2)>=secondx-10         &&
xyholder(1,pt2)<=secondx+10;
                    if
xyholder(2,pt2)>=y1-8               &&
xyholder(2,pt2)<=y1+8;

binary(4,marker)=1;
                    elseif
xyholder(2,pt2)>=y2-8               &&
xyholder(2,pt2)<=y2+8;

binary(5,marker)=1;
                    elseif
xyholder(2,pt2)>=y3-8               &&
xyholder(2,pt2)<=y3+8;

binary(6,marker)=1;
                    end
                    end
                    end
save binary;
end
elseif numObj==0
charcount=0
save charcount;
end


load charcount;
charcount

if charcount==0
ee='No Character Detected'
set(LOADING,'Visible','off')
set(handles.RecognizedImage,'String'
,ee);

set(handles.BrowseImage,'Enable','on
');
set(handles.CleanImage,'Enable','off
');
set(handles.DeskewImage,'Enable','of
f');
set(handles.SegmentImage,   'Enable',
'off');
set(handles.RecognizeImage,'Enable',
'off');
set(handles.Oneclick,'Enable','off')
;

elseif charcount>=1

load binary;
binary

u=1
while(u<=charcount)
    z=num2str(binary(1,u));
    b=[z];
    for o = 2 : 6
    a=binary(o,u);
    str1=num2str(a);
    b=[b str1];
    end
    db(u,:)=strvcat(b);
    u=u+1;
end

db

    db1='123456';
    conn  =  database('Braille',  '',
'');

setdbprefs('DataReturnFormat','cella
rray');
    curs   =   exec(conn,   ['select
Letter from Grade1 where ID= ' ''''
db1 '''']);
    curs = fetch(curs);
    bb = curs.data;

    db1='000000';
    conn  =  database('Braille',  '',
'');

setdbprefs('DataReturnFormat','cella
rray');
    curs   =   exec(conn,   ['select
Letter from Grade1 where ID= ' ''''
db1 '''']);
    curs = fetch(curs);
    space = curs.data;


end

if charcount>=1

db1=[db(1,:)]
met=1;
epox=1;
while(epox<=charcount)
    epox
    lft=0;
    rgt=0;
    if epox<charcount && charcount>1
    db2=[db1 db(epox+1,:) '%']
    elseif charcount==1
    db2='12345'
    end
    conn  =  database('Braille',  '',
'');
```

```
setdbprefs('DataReturnFormat','cella
rray')
    curs = exec(conn, ['select Words
from Grade2 where ID like ' '''' db2
''''])
    curs = fetch(curs)
    aa = curs.data

    %Empty query checker
    n = numel(curs.data)
    b=0;
    if n==1
    b = strcmp(bb,aa)
    end
    %left & right checker
    if epox+1<charcount

rgt=strcmp(db(epox+1,:),'000000')
    end
    if epox-met>=1%error catcher if
ctr-met will equal to 0
    lft=strcmp(db(epox-
met,:),'000000')
    end

    %query for standalone grade 2
braille characters(group) for the
first
    %including the first char
    if (b == 1 && n == 1) && rgt==1
&& met>1 && epox-met<1
        curs = exec(conn, ['select
Words from Grade2 where ID= ' ''''
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa

        spaceb=met-1
        while spaceb>=1
        arrayy(epox-spaceb)=space;
        spaceb=spaceb-1;
        end

        db1=db(epox+1,:)
        met=1
        epox=epox+1;


     %Sure standalone
     elseif (b == 1 && n == 1) &&
lft==1 &&  rgt==1 && met>1
        curs = exec(conn, ['select
Words from Grade2 where ID= ' ''''
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa

        spaceb=met-1
        while spaceb>=1
        arrayy(epox-spaceb)=space;
        spaceb=spaceb-1;
        end

        db1=db(epox+1,:)
        met=1
        epox=epox+1;

      %Prefix
      elseif (b == 1 && n == 1) &&
lft==1 &&  rgt==0  && met>1
         curs = exec(conn, ['select
Words from Pre_fix where ID= ' ''''
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)

        if  (b == 1 && n == 1)
        ppp=met-1
        while ppp>=1
        epox=epox-ppp
        db1=db(epox,:)
        curs = exec(conn, ['select
Letter from Grade1 where ID= ' ''''
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa
        db1=db(epox+1,:)
        met=1
        epox=epox+ppp;
        ppp=ppp-1;
        end

        else
        curs = exec(conn, ['select
Words from Pre_fix where ID= ' ''''
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa

        spaceb=met-1
        while spaceb>=1
        arrayy(epox-spaceb)=space;
        spaceb=spaceb-1;
        end

        db1=db(epox+1,:)
        met=1
        epox=epox+1;
        end

      %Suffix
```

```
      elseif (b == 1 && n == 1) &&
lft==0 &&  rgt==1  && met>1

        curs  =  exec(conn,  ['select
Words from Suffix where ID= ' ''''
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)

        if  (b == 1 && n == 1)
        ppp=met-1
        while ppp>=1
        epox=epox-ppp
        db1=db(epox,:)
        curs  =  exec(conn,  ['select
Letter from Grade1 where ID= ' ''''
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa
        db1=db(epox+1,:)
        met=1
        epox=epox+ppp;
        ppp=ppp-1;
        end

        else
        curs  =  exec(conn,  ['select
Words from Suffix where ID= ' ''''
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa

        spaceb=met-1
        while spaceb>=1
        arrayy(epox-spaceb)=space;
        spaceb=spaceb-1;
        end

        db1=db(epox+1,:)
        met=1
        epox=epox+1;
        end


       %Midfix
       elseif (b == 1 && n == 1) &&
lft==0 &&  rgt==0  && met>1

        curs  =  exec(conn,  ['select
Words from Mid_fix where ID= ' ''''
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)

        if  (b == 1 && n == 1)
        ppp=met-1
        while ppp>=1
        epox=epox-ppp
        db1=db(epox,:)
        curs  =  exec(conn,  ['select
Letter from Grade1 where ID= ' ''''
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa
        db1=db(epox+1,:)
        met=1
        epox=epox+ppp;
        ppp=ppp-1;
        end

        else
        curs  =  exec(conn,  ['select
Words from Mid_fix where ID= ' ''''
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa

        spaceb=met-1
        while spaceb>=1
        arrayy(epox-spaceb)=space;
        spaceb=spaceb-1;
        end

        db1=db(epox+1,:)
        met=1
        epox=epox+1;
        end


%try
    elseif  (b==1  &&  n==1)&&  met==1
&& lft==1 && rgt==1
        curs  =  exec(conn,  ['select
Words from Stand_Alone where ID= '
'''' db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa
        if epox<charcount
        db1=db(epox+1,:)
        end
        met=1
        epox=epox+1;

    elseif  (b==1  &&  n==1)&&  met==1
&& lft==1 && epox+1>charcount
```

```
        curs = exec(conn, ['select
Words from Stand_Alone where ID= '
'''' db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa
        if epox<charcount
        db1=db(epox+1,:)
        end
        met=1
        epox=epox+1;

     elseif (b==1 && n==1)&& met==1
&& rgt==1 && epox-met<1
        curs = exec(conn, ['select
Words from Stand_Alone where ID= '
'''' db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa
        if epox<charcount
        db1=db(epox+1,:)
        end
        met=1
        epox=epox+1;

     elseif (b==1 && n==1)&& met==1
&& epox+1>charcount && epox-met<1
        curs = exec(conn, ['select
Words from Stand_Alone where ID= '
'''' db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa
        if epox<charcount
        db1=db(epox+1,:)
        end
        met=1
        epox=epox+1;
%not sure


    %query for grade 1 braille(since
grade 2 1 braille char standalone
are unique)
    elseif    (b==1  &&  n  ==  1)  &&
met==1
        curs = exec(conn, ['select
Letter from Grade1 where ID= ' ''''
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)


        %query for grade 2 single
braille char standalone(condition is
if
        %there's no match in the
grade 1 DB)
        if (b == 1 && n == 1)
        curs = exec(conn, ['select
Words from Grade2 where ID= ' ''''
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)

            if n>1 && epox-met<1 &&
rgt==0
                curs   =   exec(conn,
['select Words from Pre_Fix where
ID= ' ''' db1 ''''])
                curs = fetch(curs)
                aa = curs.data
                n = numel(curs.data)
                b=strcmp(bb,aa)

            elseif n>1 && lft==0 &&
rgt==1 && epox==1
                curs   =   exec(conn,
['select   Words   from   Stand_Alone
where ID= ' ''' db1 ''''])
                curs = fetch(curs)
                aa = curs.data
                n = numel(curs.data)
                b=strcmp(bb,aa)

            elseif n>1 && lft==0 &&
rgt==0
                curs   =   exec(conn,
['select Words from Mid_Fix where
ID= ' ''' db1 ''''])
                curs = fetch(curs)
                aa = curs.data
                n = numel(curs.data)
                b=strcmp(bb,aa)

            elseif n>1 && lft==0 &&
rgt==1
                curs   =   exec(conn,
['select Words from Suffix where ID=
' ''' db1 ''''])
                curs = fetch(curs)
                aa = curs.data
                n = numel(curs.data)
                b=strcmp(bb,aa)

            elseif n>1 && lft==1 &&
rgt==0
                curs   =   exec(conn,
['select Words from Pre_Fix where
ID= ' ''' db1 ''''])
                curs = fetch(curs)
                aa = curs.data
                n = numel(curs.data)
                b=strcmp(bb,aa)
```

```
            elseif n>1 && lft==1 &&
rgt==1
                curs    =    exec(conn,
['select   Words   from   Stand_Alone
where ID= ' '''' db1 ''''])
                curs = fetch(curs)
                aa = curs.data
                n = numel(curs.data)
                b=strcmp(bb,aa)

            end
        end

        %proceed here if match is
already found
        arrayy(epox) = aa
        if epox<charcount
        db1=db(epox+1,:)
        end
        met=1
        epox=epox+1;

    else
    %db1 incrementation if there's
no match in any of the database
    if epox+1<=charcount
    db1=[db1 db(epox+1,:)]
    met=met+1
    epox=epox+1;

    %trial
    else
        curs = exec(conn, ['select
Words from Grade2 where ID= ' ''''
db1 ''''])
        curs = fetch(curs)
        aa = curs.data
        n = numel(curs.data)
        b=strcmp(bb,aa)
        arrayy(epox) = aa

        spaceb=met-1
        while spaceb>=1
        arrayy(epox-spaceb)=space;
        spaceb=spaceb-1;
        end

        epox=epox+1
    end
    %trial


    end

    close(conn)
    close(curs)


end

result = [arrayy(:)]
```

```
ee=strvcat(arrayy(1));
for cc = 2 : charcount
ff=strvcat(arrayy(cc));
ee=[ee ff];
end
ee


set(LOADING,'Visible','off')
set(handles.RecognizedImage,'String'
,ee);
set(handles.BrowseImage,'Enable','on
');
set(handles.CleanImage,'Enable','off
');
set(handles.DeskewImage,'Enable','of
f');
set(handles.SegmentImage,  'Enable',
'off');
set(handles.RecognizeImage,'Enable',
'off');
set(handles.Oneclick,'Enable','off')
;
end

end

% --- Executes on button press in
Showall.
function   Showall_Callback(hObject,
eventdata, handles)
% hObject    handle to Showall (see
GCBO)
% eventdata    reserved - to be
defined in a future version of
MATLAB
% handles    structure with handles
and user data (see GUIDATA)

set(handles.BrowseImage,'Visible','o
n');
set(handles.CleanImage,'Visible','on
');
set(handles.DeskewImage,'Visible','o
n');
set(handles.SegmentImage, 'Visible',
'on');
set(handles.RecognizeImage,'Visible'
,'on');
set(handles.Oneclick,'Visible','Off'
);
set(handles.BrailleAlphabet,'Visible
','on');
set(handles.BrailleContractions,'Vis
ible','on');
set(handles.ResetAll,'Visible','on')
;
set(handles.Exit,'Visible','on');
set(handles.uipanel2,'Visible','on')
;
set(handles.uipanel4,'Visible','on')
;
```

```
set(handles.RecognizedImage,'Visible
','on');
set(handles.uipanel7,'Visible','on')
;
set(handles.uipanel8,'Visible','on')
;
set(handles.text1,'Visible','on');
set(handles.ImageDisplay,'Visible','
on');
set(handles.uipanel6,'Visible','on')
;
set(handles.text3,'Visible','on');
set(handles.Deskew,'Visible','on');
set(handles.uipanel10,'Visible','on'
);
set(handles.text12,'Visible','on');
set(handles.Backside,'Visible','on')
;
set(handles.Showall,'Visible','off')
;
set(handles.Back,'Visible','on');
set(handles.Oneprocess,'Visible','of
f');
set(handles.Banner,'Visible','off');
set(handles.uipanel12,'Visible','on'
);

% --- Executes on button press in
Back.
function       Back_Callback(hObject,
eventdata, handles)
% hObject      handle to Back (see
GCBO)
% eventdata    reserved - to be
defined in a future version of
MATLAB
% handles     structure with handles
and user data (see GUIDATA)
set(handles.BrowseImage,'Visible','o
ff');
set(handles.CleanImage,'Visible','of
f');
set(handles.DeskewImage,'Visible','o
ff');
set(handles.SegmentImage, 'Visible',
'off');
set(handles.RecognizeImage,'Visible'
,'off');
set(handles.Oneclick,'Visible','Off'
);
set(handles.BrailleAlphabet,'Visible
','off');
set(handles.BrailleContractions,'Vis
ible','off');
set(handles.ResetAll,'Visible','off'
);
set(handles.Exit,'Visible','off');
set(handles.uipanel2,'Visible','off'
);
set(handles.uipanel4,'Visible','off'
);
set(handles.RecognizedImage,'Visible
','off');
set(handles.uipanel7,'Visible','off'
);
set(handles.uipanel8,'Visible','off'
);
set(handles.text1,'Visible','off');
set(handles.ImageDisplay,'Visible','
off');
set(handles.uipanel6,'Visible','off'
);
set(handles.text3,'Visible','off');
set(handles.Deskew,'Visible','off');
set(handles.uipanel10,'Visible','off
');
set(handles.text12,'Visible','off');
set(handles.Backside,'Visible','off'
);
set(handles.Showall,'Visible','on');
set(handles.Back,'Visible','off');
set(handles.Oneprocess,'Visible','on
');
set(handles.Banner,'Visible','on');
set(handles.uipanel12,'Visible','off
');

cla(handles.ImageDisplay,'reset');
cla(handles.Deskew,'reset');
cla(handles.Backside,'reset');
cla;
set(handles.text1,'String','Browsed
Image');
set(handles.BrowseImage,'Enable','on
');
set(handles.DeskewImage,'Enable','of
f');
set(handles.CleanImage,'Enable','off
');
set(handles.SegmentImage,'Enable','o
ff');
set(handles.RecognizeImage,'Enable',
'off');
set(handles.RecognizedImage,'String'
,' ');
set(handles.ImageDisplay,'xcolor','w
','ycolor','w','xtick',[],'ytick',[]
);
set(handles.Deskew,'xcolor','w','yco
lor','w','xtick',[],'ytick',[]);
set(handles.Backside,'xcolor','w','y
color','w','xtick',[],'ytick',[]);
set(handles.text1,'visible','on');
set(handles.Oneclick,'Enable','off')
;
set(handles.Angle,'String',' ');
clear

% --- Executes on button press in
Oneprocess.
function
Oneprocess_Callback(hObject,
eventdata, handles)
% hObject      handle to Oneprocess
(see GCBO)
```

```
% eventdata    reserved - to be
defined in a future version of
MATLAB
% handles     structure with handles
and user data (see GUIDATA)

set(handles.BrowseImage,'Visible','o
n');
set(handles.CleanImage,'Visible','of
f');
set(handles.DeskewImage,'Visible','o
ff');
set(handles.SegmentImage, 'Visible',
'off');
set(handles.RecognizeImage,'Visible'
,'off');
set(handles.Oneclick,'Visible','on')
;
set(handles.BrailleAlphabet,'Visible
','on');
set(handles.BrailleContractions,'Vis
ible','on');
set(handles.ResetAll,'Visible','on')
;
set(handles.Exit,'Visible','on');
set(handles.uipanel2,'Visible','on')
;
set(handles.uipanel4,'Visible','on')
;
set(handles.RecognizedImage,'Visible
','on');
set(handles.uipanel7,'Visible','on')
;
set(handles.uipanel8,'Visible','on')
;
set(handles.text1,'Visible','on');
set(handles.ImageDisplay,'Visible','
on');
set(handles.uipanel6,'Visible','off'
);
set(handles.text3,'Visible','off');
set(handles.Deskew,'Visible','off');
set(handles.uipanel10,'Visible','off
');
set(handles.text12,'Visible','off');
set(handles.Backside,'Visible','off'
);
set(handles.Showall,'Visible','off')
;
set(handles.Back,'Visible','on');
set(handles.Oneprocess,'Visible','of
f');
set(handles.Banner,'Visible','off');
```

Martin Luis R. Faustino

12 Wisdom St., Capitol Estates 2,
Commonwealth Avenue, Quezon City
430-06-08(residence), 09272582580(mobile)
mart.faustino@gmail.com

EDUCATION

College	4th Year BS Computer Science University of Santo Tomas, España, Manila	2006-Present
Secondary	University of the Philippines Integrated School Katipunan Road, Diliman, Quezon City	2002-2006
Elementary	University of the Philippines Integrated School Katipunan Road, Diliman, Quezon City	1996-2002

WORK EXPERIENCE

April-May 2009: On-the-job Trainee, Smart Telecommunications, Ayala Ave, Makati

- 240 hours
- Position: Programmer

EXTRA AND CO-CURRICULAR INVOLVEMENT

Member	Information and Computer Organization	2006-2008
Member	Computer Science Society	2008-Present

SEMINARS ATTENDED

IT Infrastructure	TARC Auditorium University of Santo Tomas	Sept. 13, 2008
MySQL	TARC Auditorium University of Santo Tomas	Sept. 15, 2008
Introduction to SAP	Rizal Auditorium St. Raymund's Building, University of Santo Tomas	Sept. 22, 2008
Social Web and Media	Rizal Auditorium St. Raymund's Building, University of Santo Tomas	Sept. 29, 2008

SKILLS/ASSETS

Skills:

- Fluent in English and Filipino
- Programming Languages:

  Turbo C, C++, Java, Visual Basic 6, Assembly

- Web Graphics and Multimedia Development Tools:

  HTML, Drupal, Adobe Photoshop, Macromedia Flash

- Operating Systems:

  Windows 95/98/XP/Vista

- Database Application and Programming:

  Microsoft SQL, Microsoft Access

- Other Applications and Programming:

  Microsoft Office, Cisco Networking

Interests:

- Computer games
- Graphic design
- Sports Entertainment
- Various fields of Science

PERSONAL INFORMATION

Birth date: December 6, 1988 Religion: Roman Catholic

Weight: 130 lbs Height: 5 ft. 4in.

Place of Birth: Manila Languages Spoken: English, Filipino

Gian Gerard E. Libunao

11 Tiwi St., NPC Village,
Tnadang Sora, Quezon City
454-59-62(residence), 09276719174(mobile)
arcamcross21@yahoo.com

EDUCATION

College	4th Year BS Computer Science University of Santo Tomas, España, Manila	2006-Present
Secondary	Colegio de San Lorenzo Congressional Ave, Quezon City	2002-2006
Elementary	Saint Claire School Villa Corrina, Tandang Sora, Quezon City	1996-2002

WORK EXPERIENCE

April-May 2009: On-the-job Trainee, UST Edtech, España, Manila

- 240 hours
- Position: Research Assistant

EXTRA AND CO-CURRICULAR INVOLVEMENT

Member	Information and Computer Organization	2006-2008
Member	Computer Science Society	2008-Present

SEMINARS ATTENDED

IT Infrastructure	TARC Auditorium University of Santo Tomas	Sept. 13, 2008
MySQL	TARC Auditorium University of Santo Tomas	Sept. 15, 2008
Introduction to SAP	Rizal Auditorium St. Raymund's Building, University of Santo Tomas	Sept. 22, 2008
Social Web and Media	Rizal Auditorium St. Raymund's Building, University of Santo Tomas	Sept. 29, 2008

SKILLS/ASSETS

Skills:

- Management – influence and optimize people to meet objectives.

- Experimentation - relentless probing for solutions and approaches.
- Computer – Windows, Internet, Microsoft Office.
- Photo editing – Photoshop
- Web designing – Dreamweaver
- Programming – C, C++, Java, Visual Basic, Assembly Language

Assets:

- Creative
- Energetic
- Resourceful
- Hardworking
- Open minded
- Team player

PERSONAL INFORMATION

Birth date: March 19, 1989 Religion: Roman Catholic

Weight: 186 lbs Height: 5 ft. 11 in.

Place of Birth: Manila Languages Spoken: English, Filipino

Jessica Cassandra T. Magbitang

38 Yellowstone St., Greenview Executive Village,
West Fairview Quezon City
930-56-58(residence), 09179064352(mobile)
blacksandwhites28@yahoo.com

EDUCATION

College	4th Year BS Computer Science University of Santo Tomas, España, Manila	2006-Present
Secondary	Miriam College High School Katipunan Road, Loyola Heights, Quezon City	2002-2006
Elementary	Miriam College Grade School Katipunan Road, Loyola Heights, Quezon City	1995-2002

WORK EXPERIENCE

April-May 2009: On-the-job Trainee, UST Edtech, España, Manila

- 240 hours
- Position: Research Assistant

EXTRA AND CO-CURRICULAR INVOLVEMENT

Member	Vespers Choir	2003-2004
Member	Immaculate Heart of Mary Choir	2004-2006
Member	Information and Computer Organization	2006-2008
Member	Computer Science Society	2008-Present

SEMINARS ATTENDED

IT Infrastructure	TARC Auditorium University of Santo Tomas	Sept. 13, 2008
MySQL	TARC Auditorium University of Santo Tomas	Sept. 15, 2008
Introduction to SAP	Rizal Auditorium St. Raymund's Building, University of Santo Tomas	Sept. 22, 2008
Social Web and Media	Rizal Auditorium St. Raymund's Building, University of Santo Tomas	Sept. 29, 2008

SKILLS/ASSETS

Skills:

- Fluent in English and Filipino
- Speed Typing

- Computer: MS Office applications, Internet, Windows 98, NT, XP and Vista
- Programming Languages: Assembly, C, C++, Java, Visual Basic 6.0
- Web Development: HTML, Macromedia Dream Weaver, Adobe Photoshop

Assets:

- Creative
- determined
- energetic
- goal-oriented
- hardworking
- resourceful
- willing to learn

PERSONAL INFORMATION

Birth date: April 28, 1989 Religion: Roman Catholic

Weight: 94 lbs Height: 5 ft. 4in.

Place of Birth: Manila Languages Spoken: English, Filipino