

CHAPTER 4

PRESENTATION, ANALYSIS AND INTERPRETATION OF DATA

A. System Architecture

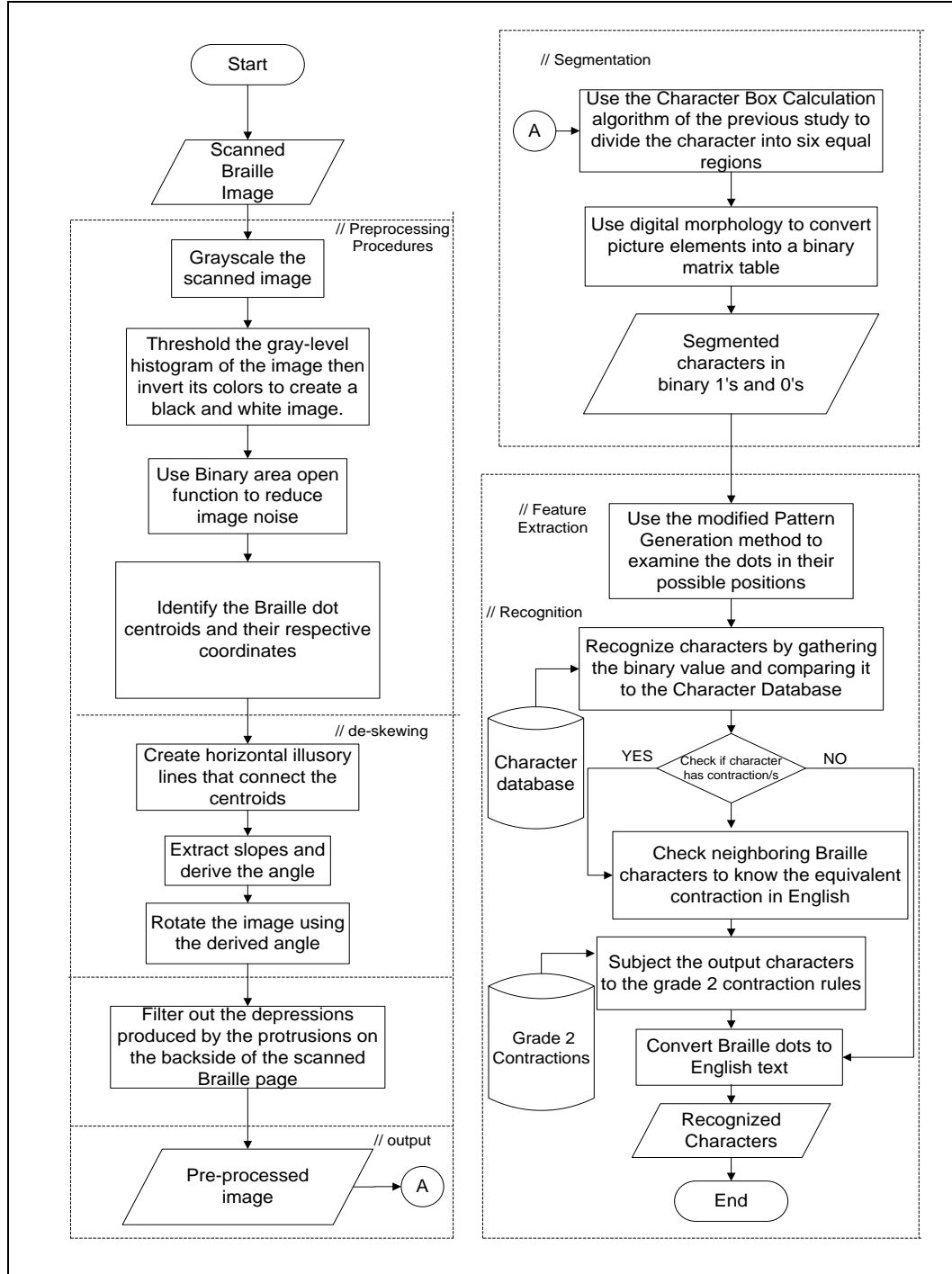


Figure 4.1
Proposed System Architecture for Optical Braille Recognition System

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 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.

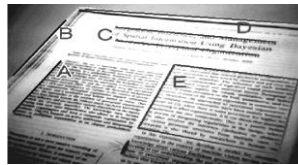


Figure 4.2

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

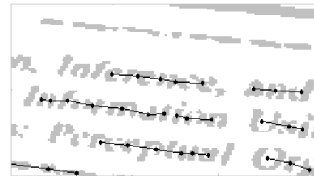


Figure 4.4
Groups of compact blobs

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

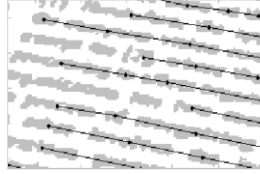


Figure 4.5
Groups of elongated blobs

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).

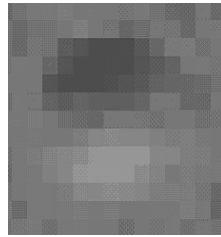


Figure 4.9
Impressed Braille dot in Grayscale

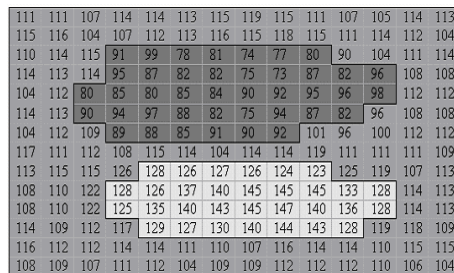


Figure 4.11
Impressed Braille Dot pixels
categorized by gray values

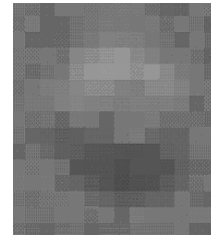


Figure 4.10
Embossed Braille dot in grayscale

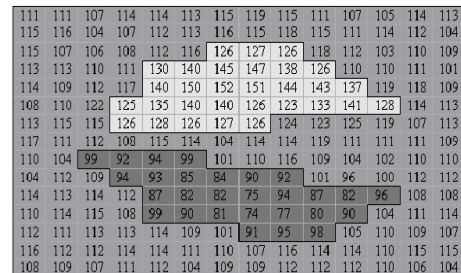


Figure 4.12
Embossed Braille Dot pixels
categorized by gray level

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.

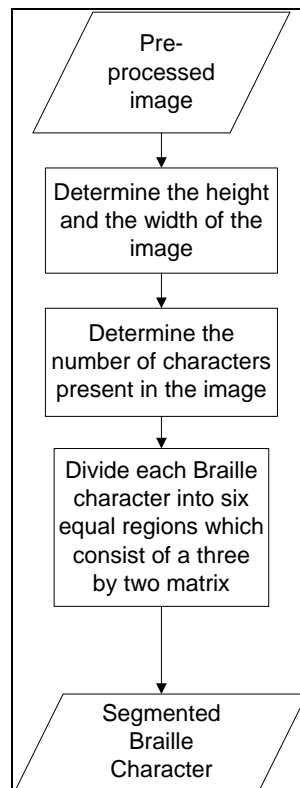


Figure 4.13
Segmentation algorithm of the previous study

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.

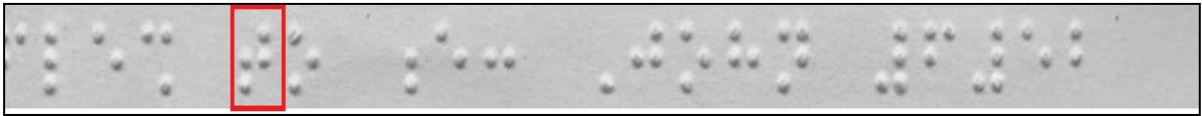


Figure 4.14
Preprocessed image to be used in the segmentation

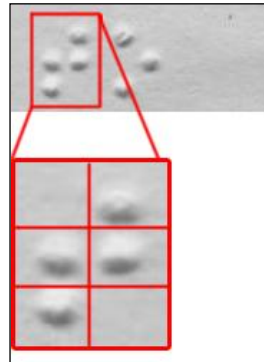


Figure 4.15
Segmented Braille Character

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.

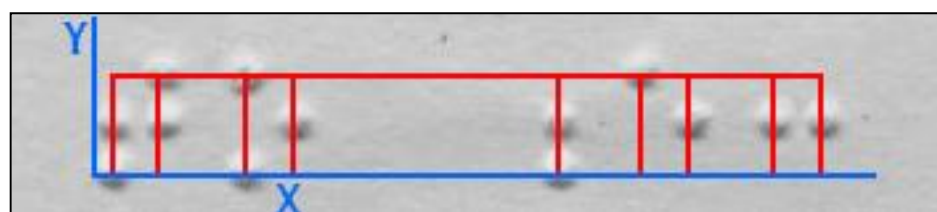


Figure 4.16
Visual representation of the proposed segmentation algorithm

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]


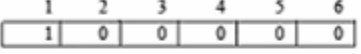

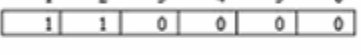

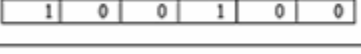

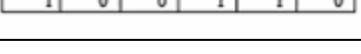
Character	Code	Grade 1
		a
		b
		c
		d

Figure 4.17
Illustration of Braille Codes and its equivalent letter













1			4	1			4
2			5	2			5
3			6	3			6
			“i”				“n”

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).

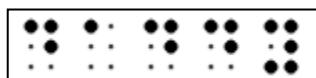


Figure 4.19
“daddy” in Grade 1 Braille

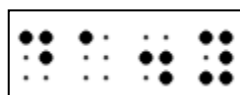


Figure 4.20
“daddy in Grade 2 Braille

d -	⠠
(#4) -	⠠ ⠠
do - (dot 1 4 5)	⠠
day - (dot 5) d	⠠ ⠠
dd - (dot 2 5 6)	⠠
dis - (dot 2 5 6)	⠠
deceive - dcv	⠠ ⠠ ⠠
declare - dcl	⠠ ⠠ ⠠
declaring - dclg	⠠ ⠠ ⠠ ⠠

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

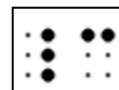


Figure 4.23
“cannot” in Grade 2 Braille

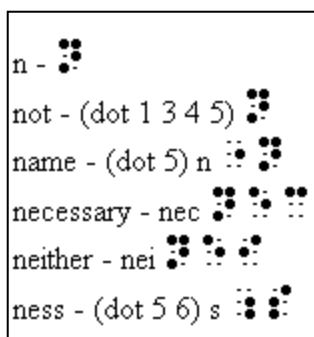


Figure 4.24
“n” Braille Contractions

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.

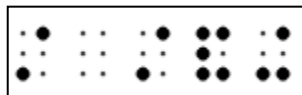


Figure 4.25
“still standing” in Grade 2 Braille

D. Processes

1. Preprocessing Procedures

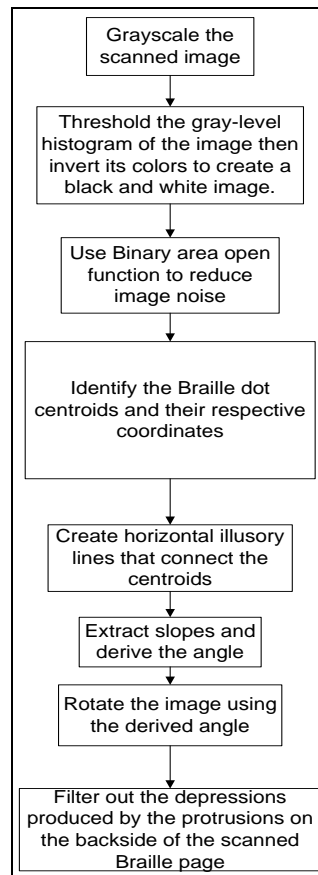


Figure 4.26
Preprocessing Phase

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

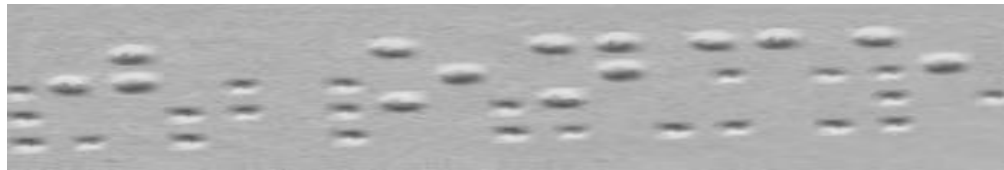


Figure 4.28
Gray level of Figure 4.27

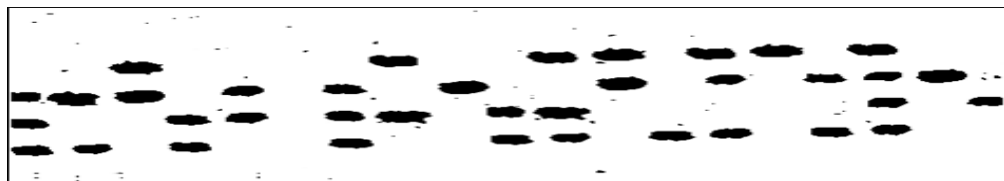


Figure 4.29
Threshold of Figure 4.28

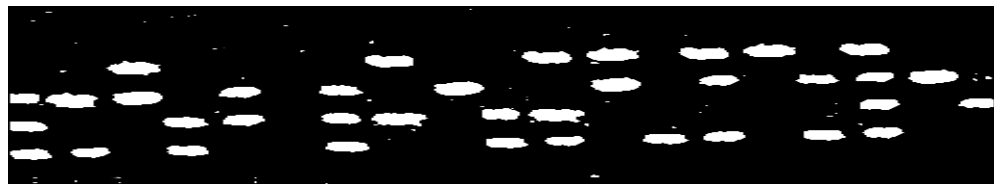


Figure 4.30
Inverted Threshold Image of Figure 4.29

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 inaccuracy 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.

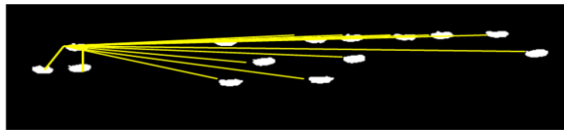


Figure 4.33
Horizontal Illusory Lines

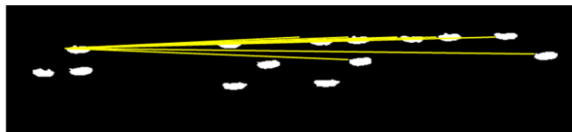


Figure 4.34
Slopes within the range



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.

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

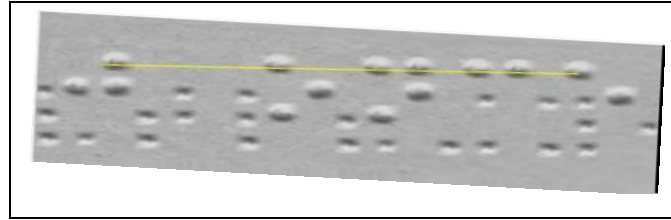


Figure 4.36
De-skewed Braille Image

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

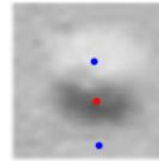


Figure 4.38
Grayscale Braille Dot Image

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.

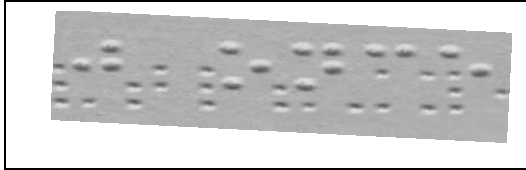


Figure 4.39
Braille Image with Depressions

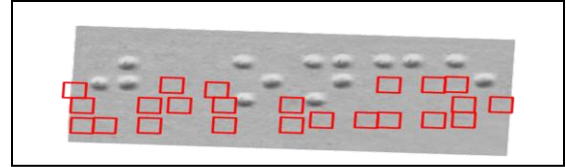


Figure 4.40
Braille Image without the Depressions

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

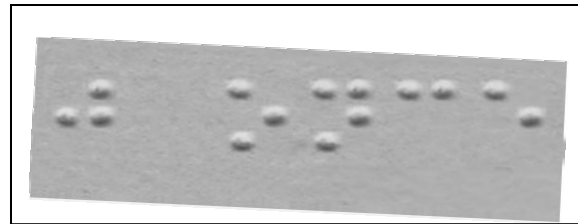


Figure 4.41
Example of Preprocessed Image

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.

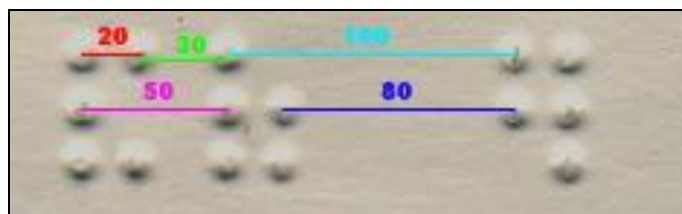


Figure 4.42
Braille Dot Pixel Distances

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.

0	1
1	1
1	0

Figure 4.43
Example of Feature Extraction

4. Recognition

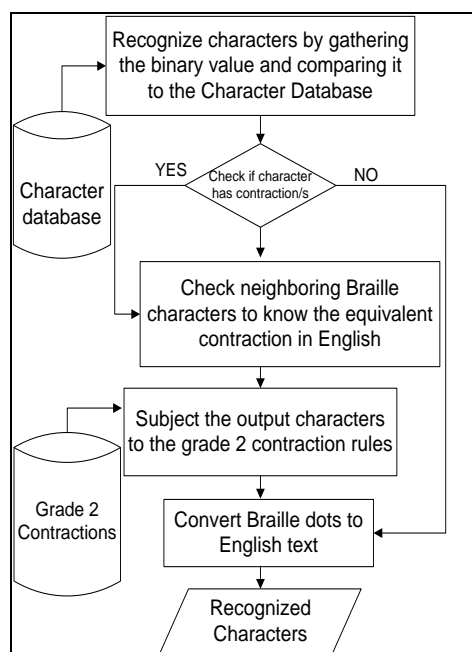


Figure 4.44
Recognition Phase

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	<ul style="list-style-type: none"> • Uses Weiner Filter function 	<ul style="list-style-type: none"> • Uses Binary Area Open function
Depression Filtering	<ul style="list-style-type: none"> • Limitation of the previous study 	<ul style="list-style-type: none"> • Checks certain pixels and their corresponding color to identify if that dot is a depression or not
Segmentation	<ul style="list-style-type: none"> • Can only segment 5 characters at a time • Their position is pre defined due to fixed values of each character box 	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • Can only generate a binary equivalent of a single Braille character 	<ul style="list-style-type: none"> • 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.