ENGLISH BRAILLE PATTERN RECOGNITION

HARKIRAT SINGH (071064)

RAGHAV MALOT (071076)

SAURABH SINGLA (071078)

NAME OF SUPERVISOR- MOHD. WAJID

May – 2011

Submitted in partial fulfilment of the Degree of

Bachelor of Technology

B. Tech

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY- WAKNAGHAT

TABLE OF CONTENTS

CERTIFICATE

This is to certify that the work titled **"ENGLISH BRAILLE PATTERN RECOGNITION"** submitted by **"SAURABH SINGLA (071078), HARKIRAT SINGH(071064) and RAGHAV MALOT(071076)"** in partial fulfilment for the award of degree of B.TECH of Jaypee University of Information Technology, Waknaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor........................... [illegible]

Name of Supervisor Mr. Mohd. Wajid

Designation Lecturer..........

Date 23-May-2011.

ACKNOWLEDGEMENT

We would like to express our sincere thanks to our esteemed project guide Mr. Mohammed Wajid for his immense contribution to the project. Without his assistance, progress thus made by us would not have been possible. His unwavering dedication marked with his profound knowledge of the subject matter has inspired us to work even harder and achieve above our capabilities. We would also like to express our deepest gratitude to the department of Electronics and Communication engineering, Aligarh Muslim University from where we obtained the prototype Urdu Braille reader.

Harkirat Singh
HARKIRAT SINGH(071064)

Raghav
RAGHAV MALOT(071076)

Saurabh Singla
SAURABH SINGLA(071078)

SUMMARY

The purpose of the aforementioned project is to develop and perfect a software which, when completed, will be able to recognize a pattern (words) written in the English Braille language and also provide an audio of the said pattern. The software will have profound benefits for blind people such as enabling them to learn Braille in an interactive manner and making possible written communication between blind people and normal people over long distances with less bandwidth.

Skill set and resources required: To create English Braille pattern recognition software requires basic to intermediate knowledge of MATLAB and also competent understanding of digital image processing.

Time needed for successful implementation of the software: Six to seven months. The first stage involves conceptualization of the project. This will be the longest stage in terms of time taken. Once the plan of action is finalized, we estimate that it will take only two months to design the software. Afterwards, considerable time will be devoted to testing of the software to ensure its workability.

Various stages of project: The project will consist of four different stages. In the first step the original Braille image will be scanned or a picture is taken of the Braille document. The original Braille image will be a Red-Green-Blue image also known as a RGB image. An RGB image has three channels: Red, Green and Blue. RGB channels are used in computer displays and image scanners. After the RGB Braille image is scanned it will be converted into a Gray image which in turn is converted into a binary image using MATLAB coding. The reason why the RGB image is converted into a binary image is to decrease the processing time of the image as the number of levels will get reduced to only 2 as compared to the Gray image which has 255 levels.

Cropping is done on the binary image to get rid of the unwanted area in the image. Noise such as salt and pepper noise which is prevalent in images will be removed by techniques such as median filtering. After cropping and noise reduction the next step in the implementation of the project is to assign binary vectors to each Braille block,

which is a 3x2 matrix. Therefore a vector will be of six bits. Now this vector will be mapped to a data base containing all twenty six alphabets of the English language and matched to a corresponding letter. Proceeding in a similar fashion, the software will keep on mapping the letters until it finds the space block. When the software finds the space block, it will know that the word is finished, and the audio component in the software will read aloud the finished word.

LIST OF FIGURES

ABSTRACT

The purpose of the aforementioned project is to develop a software which, when completed, will be able to recognize a pattern (words) written in the English Braille language and also provide an audio of the said pattern. The software will have profound benefits for blind people such as enabling them to learn Braille in an interactive manner and making possible written communication between blind people and normal people over long distances with less bandwidth.

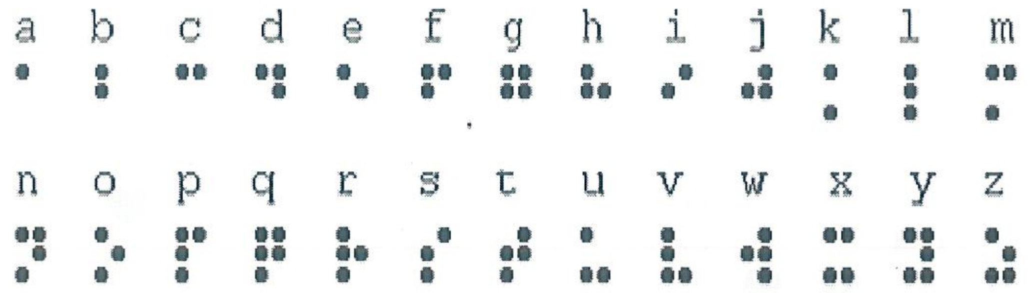
CHAPTER 1

1.1 AIM-

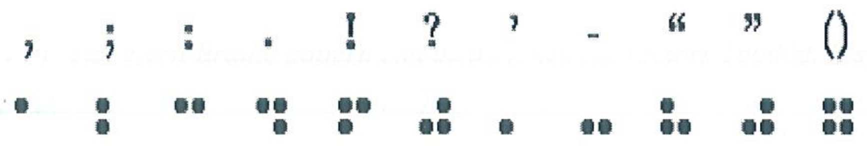
The purpose of the aforementioned project is to develop and perfect a software which, when completed, will be able to recognize the patterns (words) written in the English Braille language and convert it into corresponding English text. The software will have profound benefits for blind people such as enabling them to learn Braille in an interactive manner and making possible written communication between blind people and normal people over long distances with less bandwidth.

1.2 INTRODUCTION TO BRAILLE SYSTEM OF LANGUAGE-

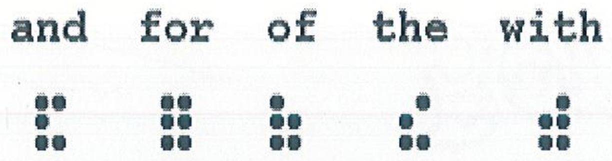
For the benefit of everyone concerned, it would be prudent to give a brief overview of the Braille language before moving onto the technical nitty-gritty of the project. The Braille system is a method that is widely used by blind people to read and write. Braille was devised by Louis Braille, a blind Frenchman. Each Braille character or cell is made up of six dot positions, arranged in a rectangular configuration of two columns of three dots each. A dot may be raised at any of the six positions to form sixty four possible subsets, including the arrangement in which no dots are raised. For reference purposes, a particular permutation may be described by naming the positions where dots are raised, the positions being universally numbered 1 to 3, from top to bottom, on the left, and 4 to 6, from top to bottom, on the right. For example, dots 1-3-4 would describe a cell with three dots raised, at the top and bottom in the left column and on top of the right column, i.e., the letter *m*. The lines of horizontal Braille text are separated by a space, much like visible printed text, so that the dots of one line can be differentiated from the Braille text above and below. Punctuation is represented by its own unique set of characters. The technique we are implementing is applicable in any language which uses 3 x 2 matrixes.[1]



- Punctuation marks:



- Full-word signs:



- Letter combinations:

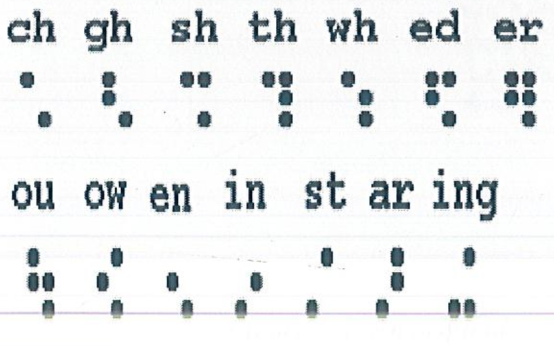


Figure1.1: The standard interpretation of Braille blocks in English

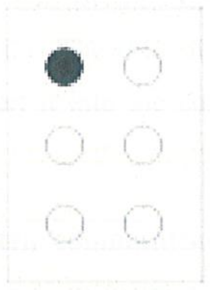


Figure (1.2)

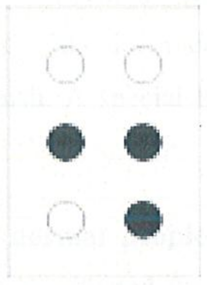


Figure (1.3)

Figure (1.2): The given Braille pattern can be assigned the vectors 100000. It symbolizes the letter 'a'.

Figure (1.3): The given Braille pattern can be assigned the vectors 010011. It symbolises the punctuation mark '-'.

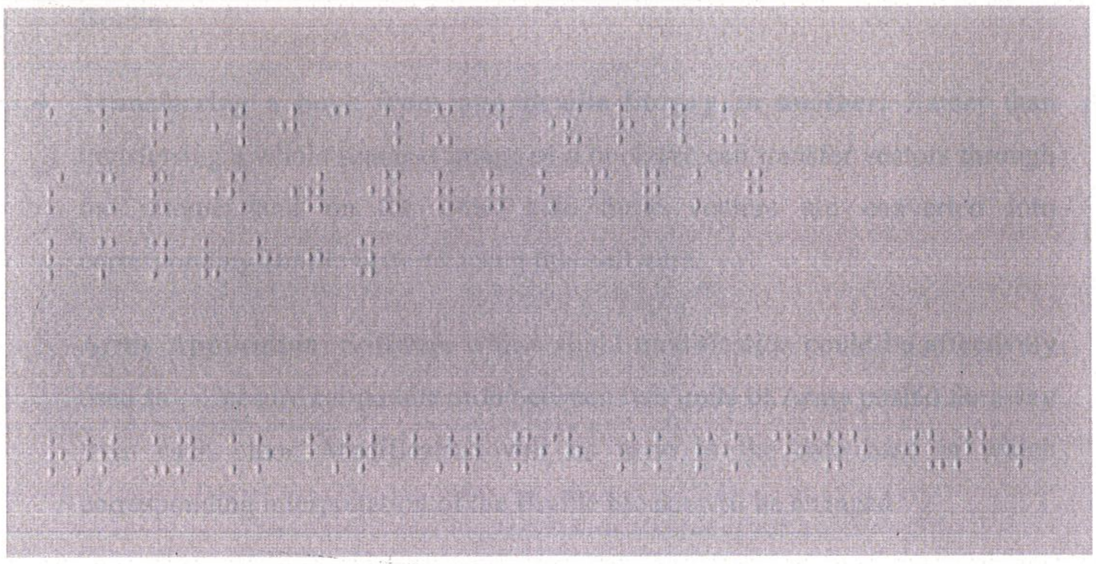


Figure 1.4: An original Braille image [2]

1.3 Applications:

The applications of such a project are many and varied-

1 **Photocopy of Braille text:** If ever the need arises to make photocopies of a particular Braille document, then we can use Braille pattern recognition software. The software will simply take a scanned image of the document and convert it into the corresponding interpretation in English. A special Braille printer is used to print the English text directly into a Braille document.

2 **Written communication between blind people and normal people:** If a blind person wants to communicate with a normal person situated far away from him, he/she can either send the Braille directly through post and the person at the receiving end can scan the Braille copy and use this software to interpret the writings of the blind person or the blind person can send the scanned image to the normal person which he/she can interpret by using this software to retrieve the information.

3 **Aiding blind people to learn Braille:** Audio implementation of Braille pattern recognition software will help visually impaired individuals to learn Braille.

4 **Transferring a book from one Braille library to another:** Rather than transferring a whole scanned image of a book we can transfer vectors through the internet and on the other side these vectors are converted into corresponding Braille patterns using this software.

5 **Army Application:** Software with a slight modification could be affectively used as secure communication between two units of Army posted far away from each other. Modification will be done in the data base in which corresponding interpretation of the Braille blocks will be changed.

CHAPTER 2
RGB, GRAY AND BINARY IMAGES

2.1 RGB: An RGB image has three channels: Red, Green and Blue. RGB channels are used in computer displays and image scanners. If the RGB image is 24-bit, each channel has 8 bits, for red, green, and blue—in other words, the image is composed of three images (one for each channel), where each image can store discrete pixels with conventional brightness intensities between 0 and 255.[3]

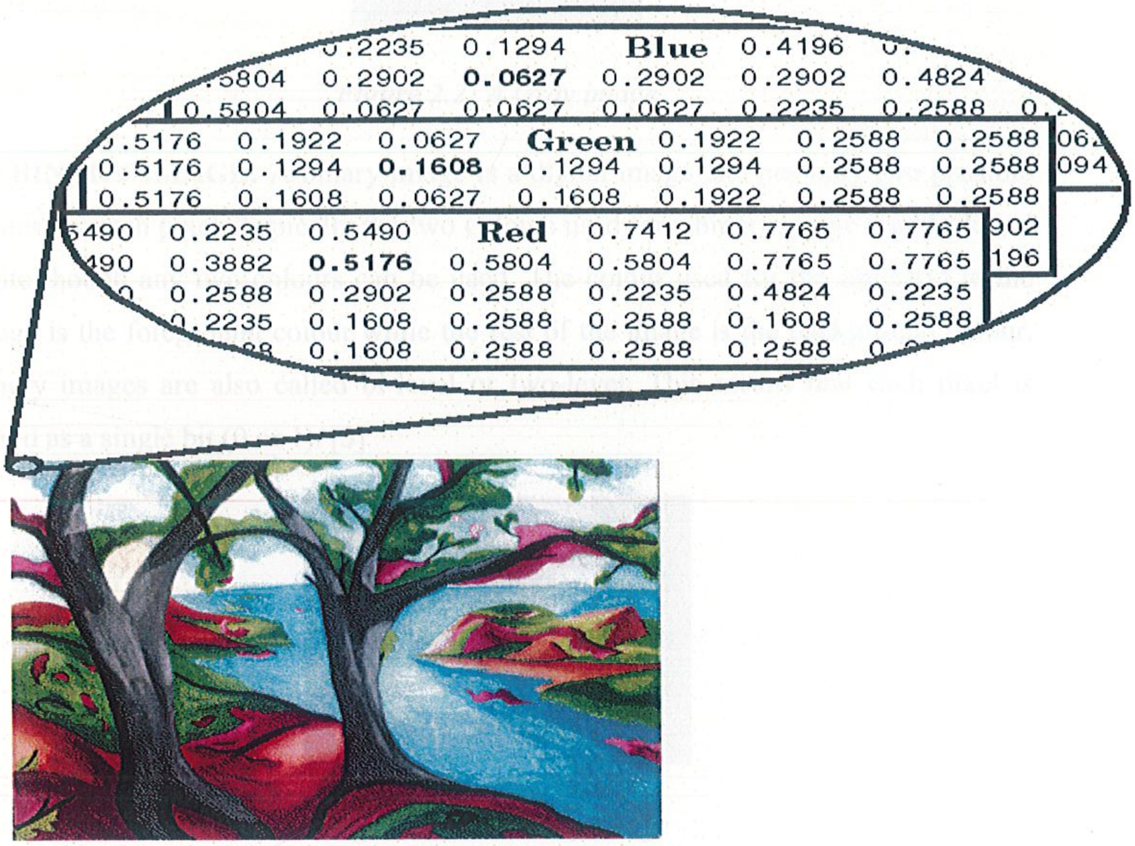


Figure 2.1: An RGB image with three channels.

2.2 GRAY IMAGE: In photography and computing, a gray digital image is an image in which the value of each pixel is a single sample, that is, it carries only intensity information. Images of this sort, also known as black-and-white, are composed exclusively of shades of gray, varying from black at the weakest intensity to white at the strongest. Gray images are distinct from one-bit black-and-white images, which in the context of computer imaging are images with only the two colours, black, and white (also called bi-level or binary images).[3]

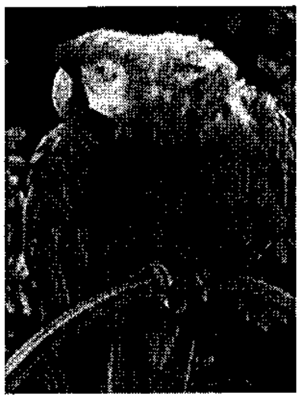


Figure 2.2: A Gray image

2.3 BINARY IMAGE: A binary image is a digital image that has only two possible values for each pixel. Typically the two colours used for a binary image are black and white though any two colours can be used. The colour used for the object(s) in the image is the foreground colour while the rest of the image is the background colour. Binary images are also called bi-level or two-level. This means that each pixel is stored as a single bit (0 or 1). [3]

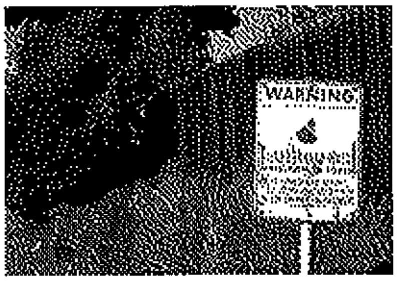


Figure 2.3: A Binary image

CHAPTER 3

STAGES OF THE PROJECT

The project will consist of four different stages.

STEP 1: In the first step the original Braille image will be scanned or a picture is taken of the Braille document. The original Braille image will be a Red-Green-Blue image also known as a RGB image. After the RGB Braille image is scanned, it will be converted into a Gray image which in turn is converted into a binary image using MATLAB coding. A binary image is one whose pixels can take up only two possible values i.e. 0 and 1. Each pixel is stored as a single bit (0 or 1). Numerically, the two values are often 0 for black, and either 1 or 255 for white. The reason why the RGB image is converted into a binary image is to decrease the processing time of the image as the number of levels will get reduced to only 2 as compared to the Gray image which has 255 levels.

STEP 2: Cropping is done on the binary image to get rid of the unwanted area in the image. Noise such as salt and pepper noise which is prevalent in images will be removed by techniques such as median filtering.

STEP 3: The median filter is a nonlinear digital filtering technique, often used to remove noise. Median filtering is very widely used in digital image processing because under certain conditions, it preserves edges whilst removing noise.[3]

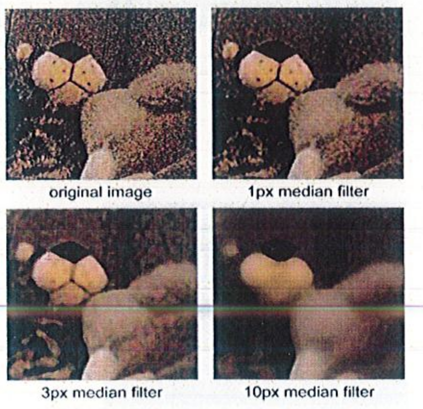


Figure 3.1: Image undergoing Median filtering

STEP 4: After cropping and noise reduction the next step in the implementation of the project is to assign binary vectors to each Braille block, which is a 3x2 matrix. Therefore a vector will be of six bits.

STEP 5: Now this vector will be mapped to a data base containing all twenty six alphabets of the English language and matched to a corresponding letter. Proceeding in a similar fashion, the software will keep on mapping the letters until it finds the space block. When the software finds the space block, it will know that the word is finished.

Flow chart showing all the stages of the project

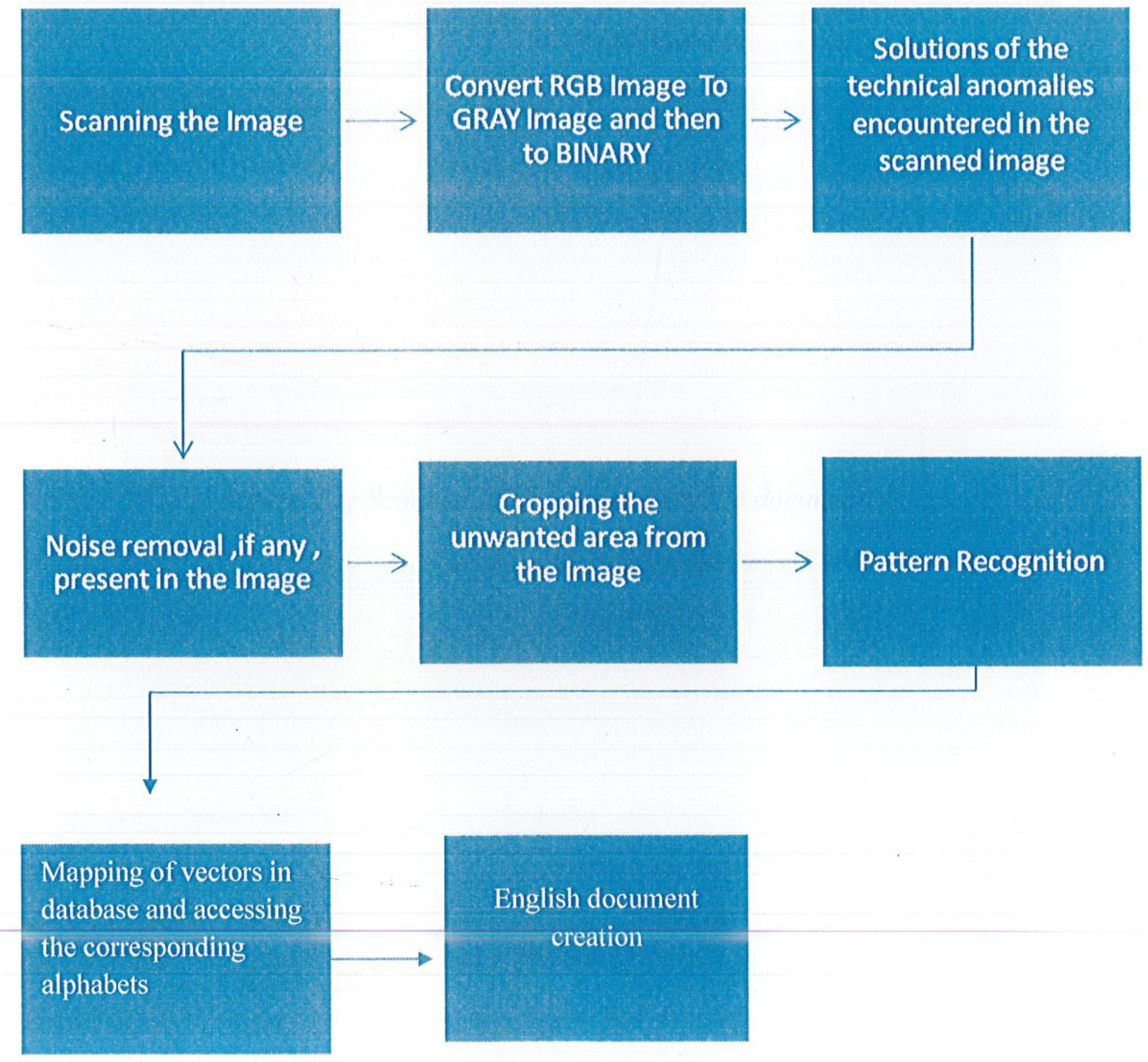

Scanning the Image
Convert RGB Image To GRAY Image and then to BINARY
Solutions of the technical anomalies encountered in the scanned image
Noise removal ,if any , present in the Image
Cropping the unwanted area from the Image
Pattern Recognition
Mapping of vectors in database and accessing the corresponding alphabets
English document creation

Figure 3.2: A block diagram representing various stages of Project

CHAPTER-4

IMAGE PROCESSING, NOISE REDUCTION AND CROPPING

STEP 1: We have converted our original RGB image of Braille document into a Gray image using MATLAB function "rgbtogray()"[4]. Further we have removed the unwanted area of the image i.e. we have cropped the image.

The progress is shown by the original images from MATLAB.

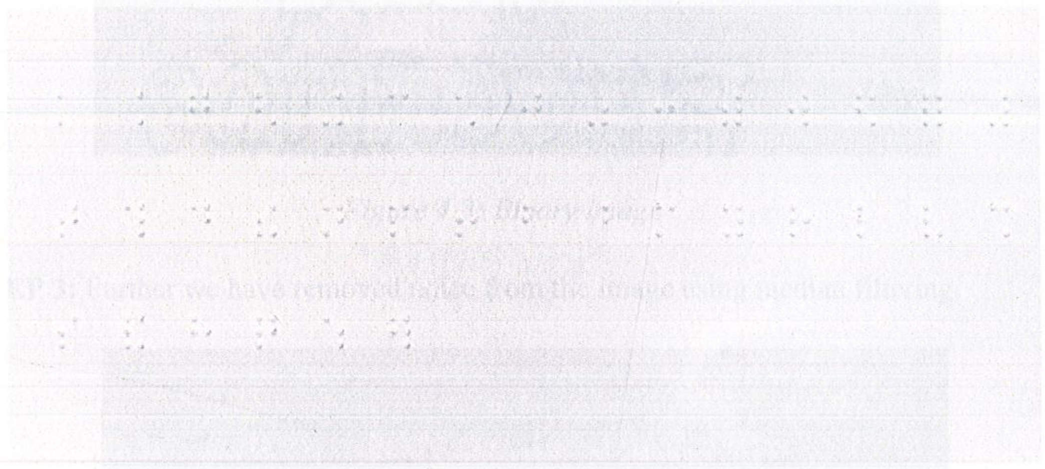


Figure 4.1: Scanned RGB image of Braille document

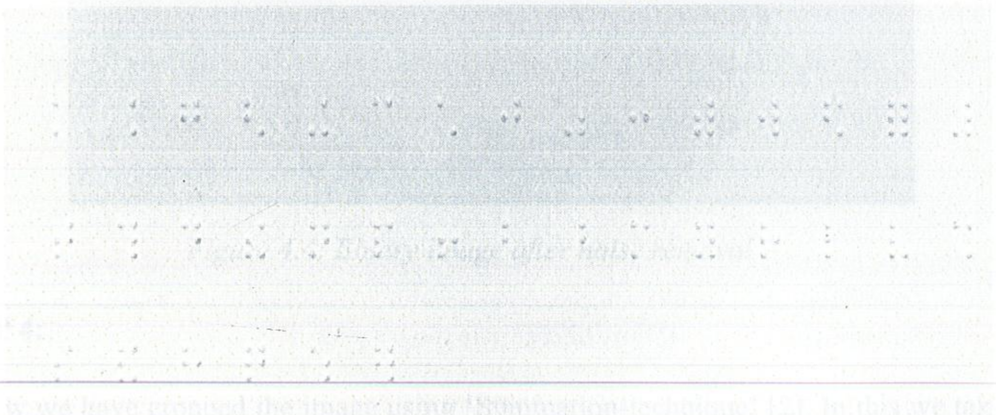


Figure 4.2: Above RGB image converted into Gray image

STEP 2: This gray image is converted to a Binary image using the logic that all the pixels in the gray image with values more than a particular threshold value (for eg.220) are converted into 0 values i.e. they will form the black portion and with pixel values less than that are converted into 255 values i.e. they will form the white portion of the binary image.

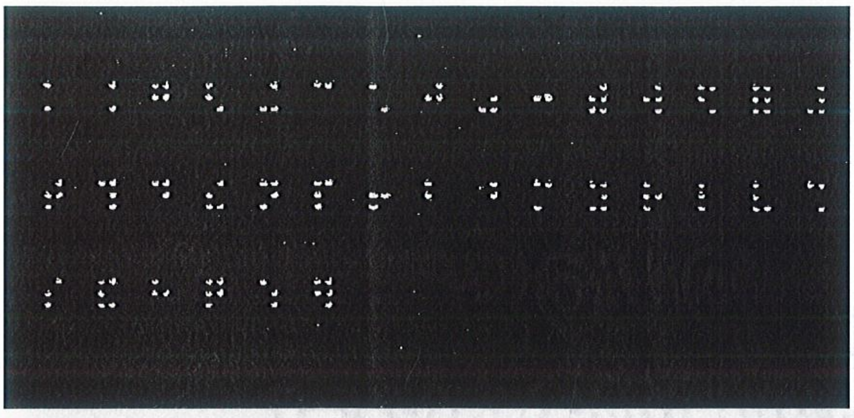


Figure 4.3: Binary image

STEP 3: Further we have removed noise from the image using median filtering.

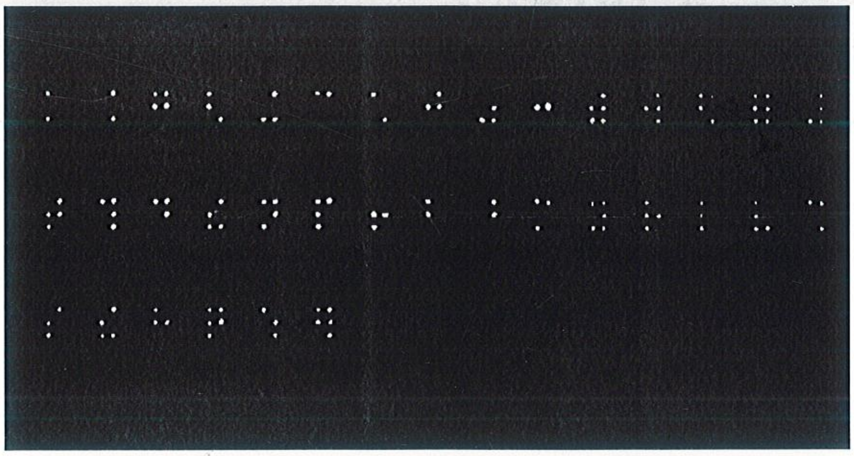


Figure 4.4: Binary image after noise removal

STEP 4:

a) Now we have cropped the image using 'Summation-technique' [2]. In this we take summation of each row of the above image or the 2-D array which will form a 1-D array with values corresponding to the summation results of each row in the image. We find that all the initial values in the 1-D array are 0s or less than a threshold value and as soon as we encounter value which is greater than the threshold value, from that point we cut the above portion of the image and process the rest. Hence we have removed unwanted portion from above the image.

b) Now we check if we could find a location from the 1-D array where from that point onwards all the values are less than threshold value. As soon as we encounter such point we remove the image from below that point. Hence we have now removed the unwanted part from bottom.

c) Proceeding in the similar manner we now take summation of columns and remove the un-wanted parts from the left and right portion of the image.

d) Hence we get a final cropped image as shown below.

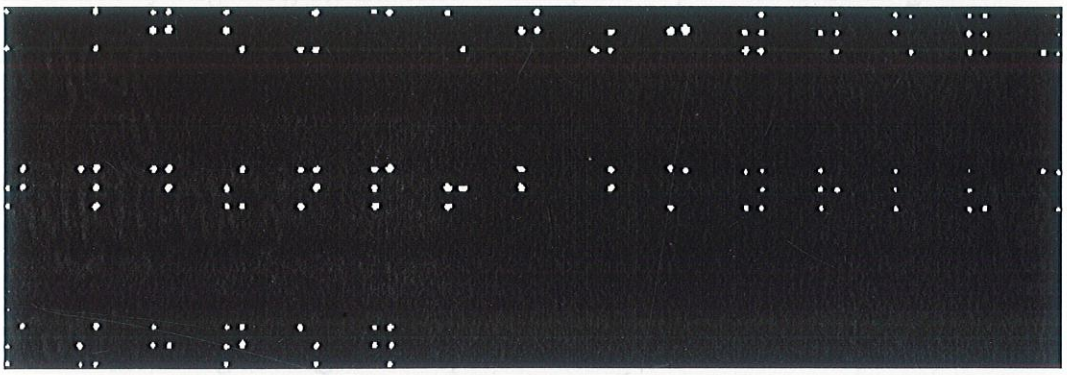


Figure 4.5: Cropped binary image

CHAPTER 5
VARIOUS FILTERING TECHNIQUES

5.1 Linear smoothing filters

One method to remove noise is by convolving the original image with a mask that represents a low-pass filter or smoothing operation. For example, the Gaussian mask comprises elements determined by a Gaussian function. This convolution brings the value of each pixel into closer harmony with the values of its neighbors. In general, a smoothing filter sets each pixel to the average value, or a weighted average, of itself and its nearby neighbors; the Gaussian filter is just one possible set of weights.

Smoothing filters tend to blur an image, because pixel intensity values that are significantly higher or lower than the surrounding neighborhood would "smear" across the area. Because of this blurring, linear filters are seldom used in practice for noise reduction; they are, however, often used as the basis for nonlinear noise reduction filters.

This method first gathering local displacement statistic around each points, filter out outliers and apply smoothing filter. Here are original matches; matches after applying consistency checks and matches after applying smoothing filter.[3]

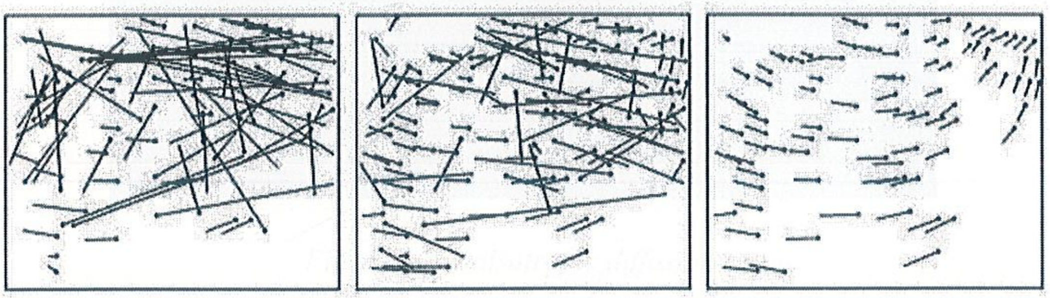


Figure 5.1: Here are original matches; matches after applying consistency checks and matches after applying smoothing filter

5.2 Anisotropic diffusion

In image processing and computer vision, **anisotropic diffusion**, also called **Perona–Malik diffusion**, is a technique aiming at reducing image noise without removing significant parts of the image content, typically edges, lines or other details that are important for the interpretation of the image. Anisotropic diffusion resembles the process that creates a scale-space, where an image generates a parameterized family of successively more and more blurred images based on a diffusion process. Each of the resulting images in this family are given as a convolution between the image and a 2D isotropic Gaussian filter, where the width of the filter increases with the parameter. This diffusion process is a linear and space-invariant transformation of the original image. Anisotropic diffusion is a generalization of this diffusion process: it produces a family of parameterized images, but each resulting image is a combination between the original image and a filter that depends on the local content of the original image. As a consequence, anisotropic diffusion is a non-linear and space-variant transformation of the original image.[1]

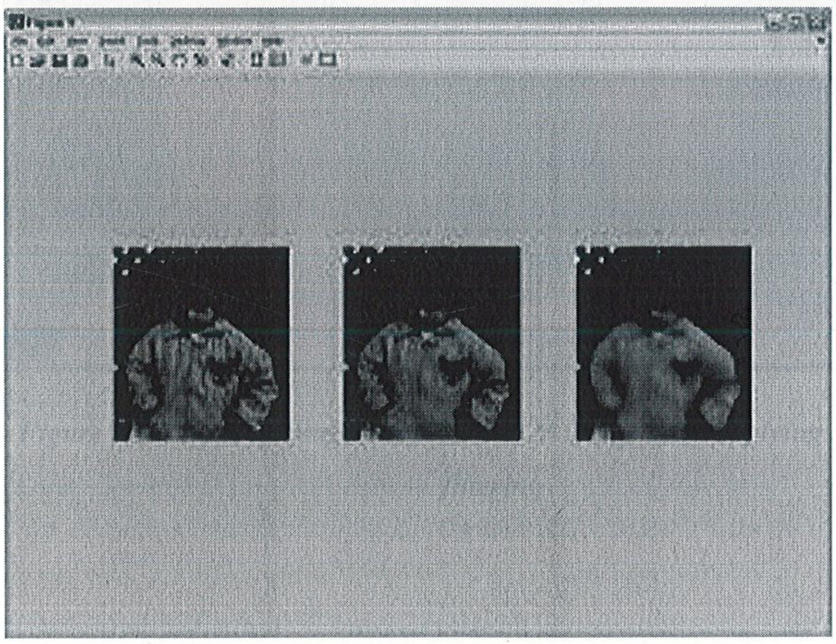


Figure 5.2: Anisotropic diffusion results

5.3 Nonlinear filters

A median filter is an example of a non-linear filter and, if properly designed, is very good at preserving image detail. To run a median filter: Consider each pixel in the image. Sort the neighboring pixels into order based upon their intensities. Replace the original value of the pixel with the median value from the list.

A median filter is a rank-selection (RS) filter, a particularly harsh member of the family of rank-conditioned rank-selection (RCRS) filters; a much milder member of that family, for example one that selects the closest of the neighboring values when a pixel's value is external in its neighborhood, and leaves it unchanged otherwise, is sometimes preferred, especially in photographic applications.

Median and other RCRS filters are good at removing salt and pepper noise from an image, and also cause relatively little blurring of edges, and hence are often used in computer vision applications.[1]

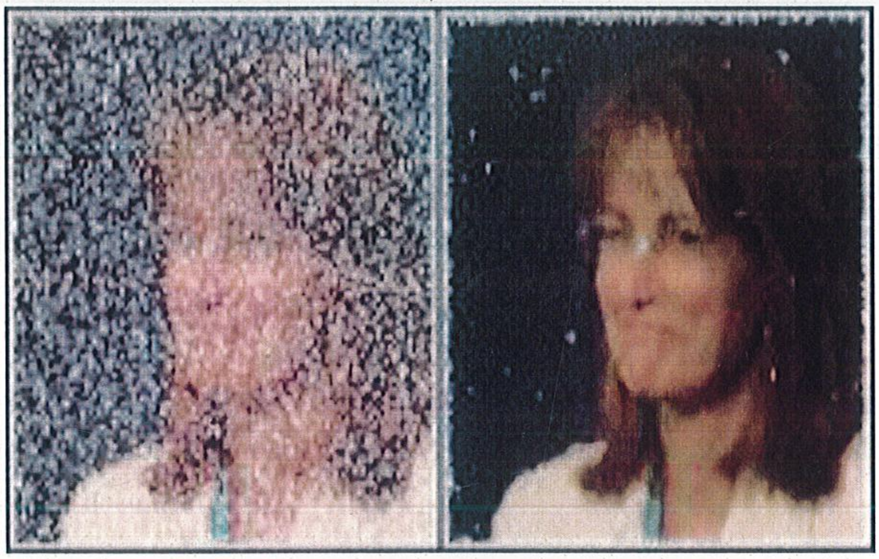


Figure 5.3: Showing image improvement in image after applying Non-linear filtering

CHAPTER 6
ALGORITHM TO EXTRACT BRAILLE BLOCKS FROM CROPPED IMAGE

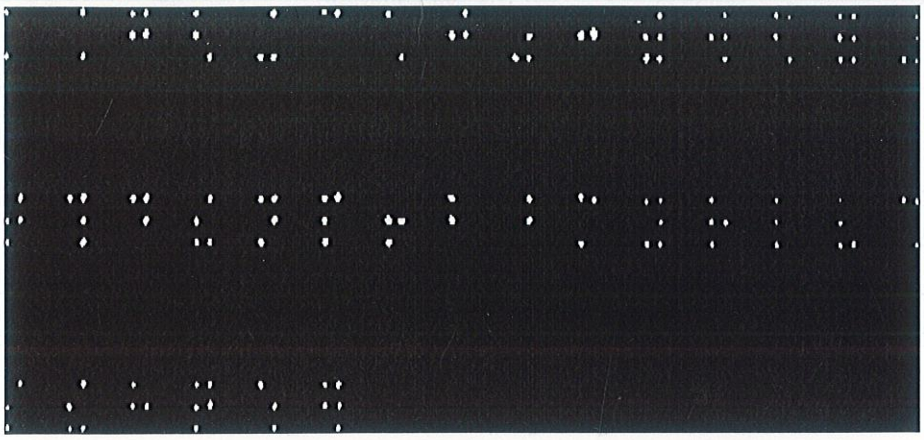


Figure 6.1: Cropped image

This image is a binary cropped image or a 2-D array consisting of 0s at the black portion and cluster of 255 values at white portions (which are the raised dots in the original Braille document)

In order to extract individual Braille blocks from the cropped image we first:

STEP 1: Take summation of each row in the 2-D array which forms a 1-D array consisting of elements corresponding to summation values of each row in the image.[2]

STEP 2: We find that the values in the 1-D array are increasing which increase to a peak value and then it decreases to a low point; this low marks the end of set of dots in the first row. Again the values in the 1-D array will increase to a peak and again decrease to a low and so on. As soon as we encounter the 3rd low, we see that it marks the end of first line of the Braille document.[2]

Hence we get a strip as the first line. We store this strip as the first element in an array and the rest of image is being processed further.

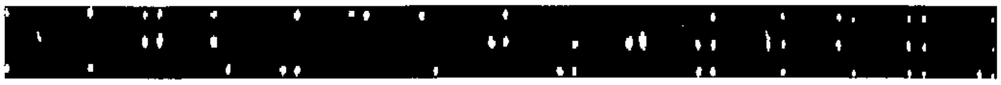


Figure 6.2: First horizontal strip

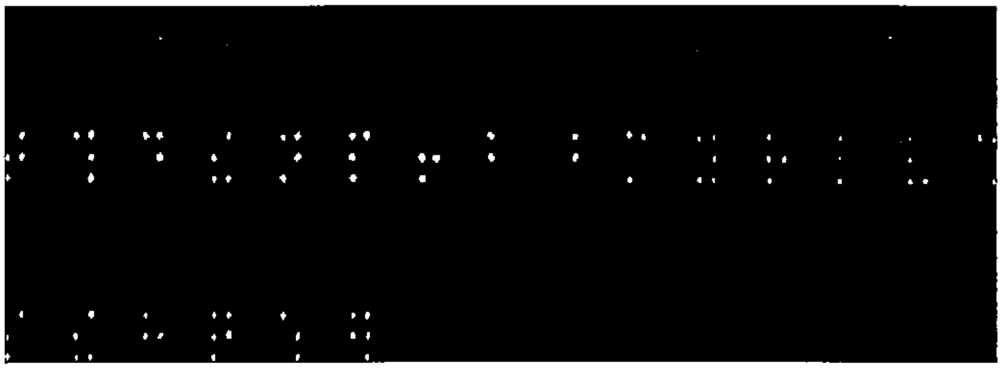


Figure 6.3: Remaining image

STEP 3: Again we will crop the rest of image shown above to obtain the next line of the Braille text.

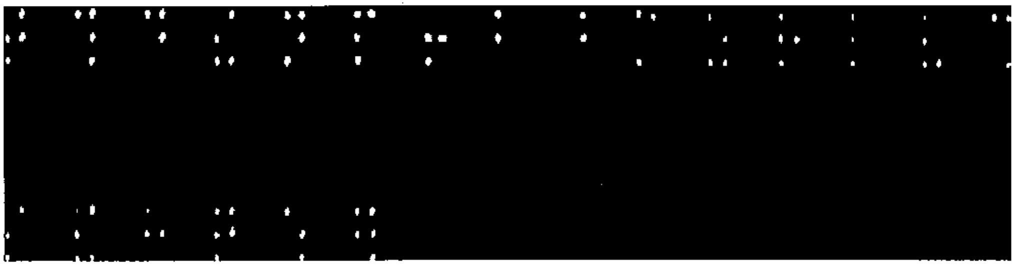


Figure 6.4: Cropped image till the second strip

And the same procedure of taking summation of each row and making 1- D array to extract the lines (strips) in the document is repeated which are being stored in an array simultaneously, till we get the last line of the document.[2]

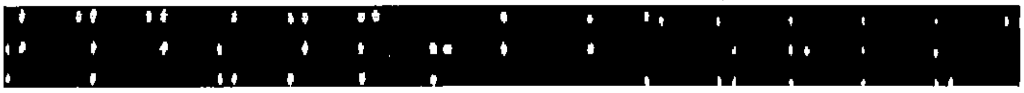


Figure 6.5: Second horizontal strip

Figure 6.6: Remaining image after extracting the second strip

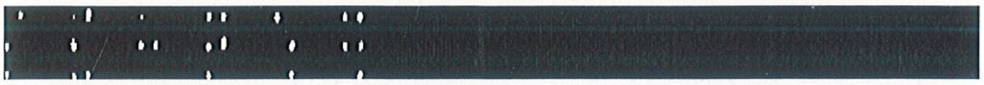


Figure 6.7: Last horizontal strip

Hence we obtain the last line (strip) of the document.

STEP 4: We again analyse the original cropped image shown below. But this time we analyse it vertically.[2]

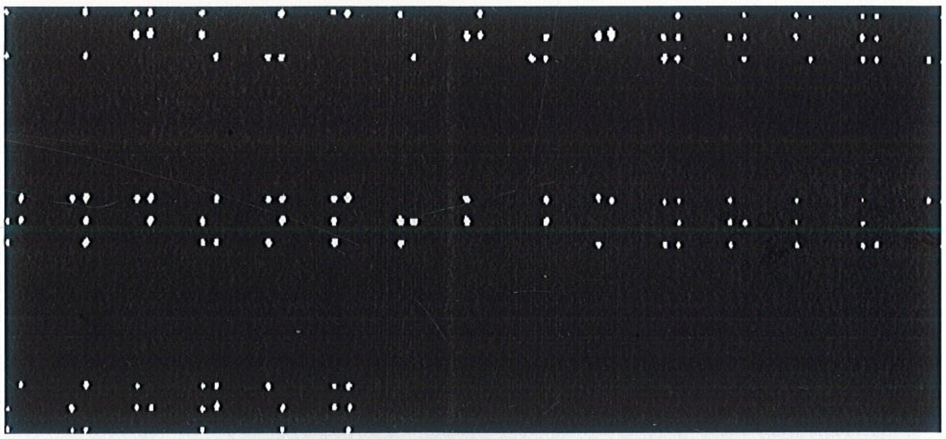


Figure 6.8: Original cropped image

STEP 5: Now we take summation of each column of the image or 2-D array to again form a 1-D array consisting of elements corresponding to summation values of each row in the image.[2]

STEP 6: We again find that the values in the 1-D array are increasing which increases to a peak value and then it decreases to a low point. This low marks the end of set of dots in the first column. Again the values in the 1-D array will increase to a peak and again decreases to a low and so on.[2]

STEP 7: As soon as we encounter the 2nd low, we see that it marks the end of first set of characters in first column of the Braille document. Hence, after finding that location in the 2-D array we extract the 3x2 Braille blocks from the strips obtained earlier.

The purpose of vertically analysing the original cropped image again rather than analysing individual strips and obtaining the Braille blocks is that there might be the chance that there are dots in only one column of the blocks which will not allow us to extract the individual blocks properly. And also it will be unable to find the space that we will encounter in lines.[2]

STEP 8: After extracting blocks from the first column of the document we again crop the image to further extract the Braille blocks from the 2nd column of the document. This process continues till we reach the last set of blocks in the last column.[2]

These blocks are also stored in an array simultaneously for further analysis.

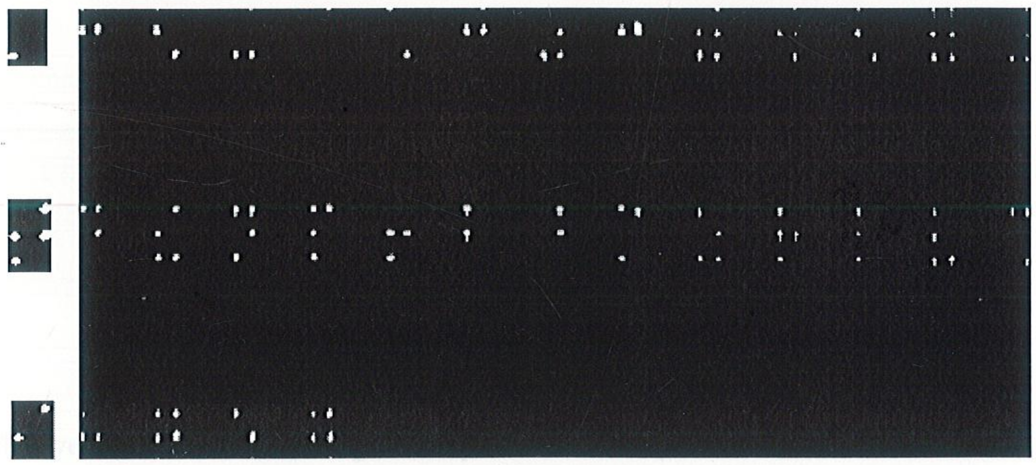


Figure 6.9: Extraction of the blocks from the first column of the document

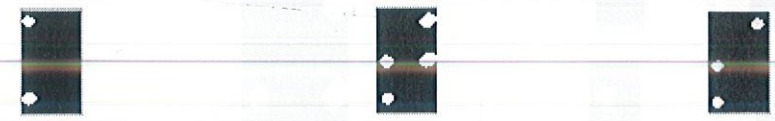


Figure 6.10: Extracted blocks

CHAPTER-7

CREATING ENGLISH DOCUMENT FROM THE BRAILLE

7.1 Vector Extraction from the Braille blocks: A vector is a binary representation of a particular Braille block.

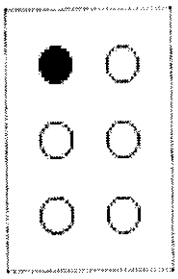


Figure 7.1: Its vector will be 100000[1]

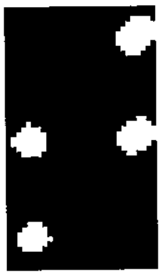


Figure 7.2: Its vector will be 011110

This vector shall be a six digit binary number as represented above. Now we will divide the Braille block into two parts vertically and those parts into three parts horizontally. As shown below:

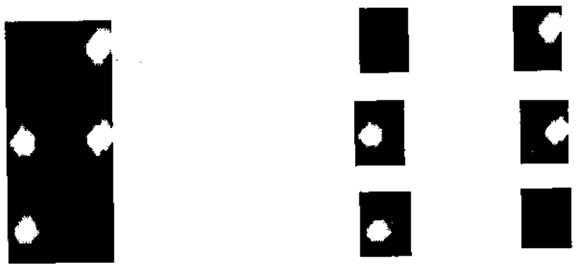


Figure 7.3 Subdivisions of Braille block

Now, the blocks with the white dots will represent the binary number "1"and the blank blocks will represent "0".

7.2 Technique used for detecting a "0"or a "1":

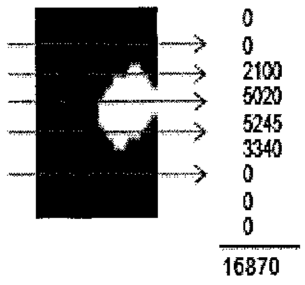


Figure 7.4: To take summation of each row in subdivided block

As shown above, in this small block obtained by dividing the original Braille block, we take summation of the elements of the rows of the 2-D array (image). This will form a 1-D array whose elements will correspond to summation of each row of the image. Again we will take summation of each element of the 1-D array to get a single decimal number. Since we know that black portion represent 0s and white portion represents 255 values. So if the final number obtain by above process is greater than 0 or 255 or less than threshold value (say 1000), then it represents "1" in the binary vector and if it is 0 or less than the threshold value, then it represent "0" In the binary vector.

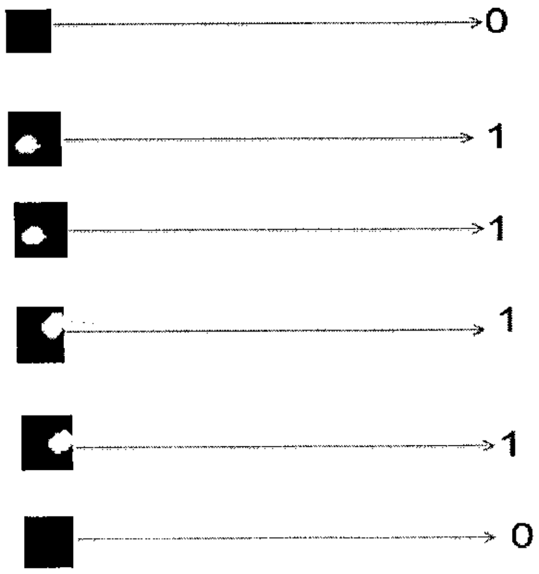


Figure 7.5: Binary interpretation of the individual block

7.3 Vector matching: Earlier we were thinking on the lines of making a database to match the vectors with corresponding English alphabets but then using database in this application will make it more complex and time consuming and also since our data is not dynamic i.e. we will not need to change it in future.

So we decided to include the vector matching process in the main code only by using 'Switch-case' method. Here we are comparing vectors obtained with the corresponding vectors of standard Braille alphabets.

Following is the standard representation of the Braille blocks in English language.

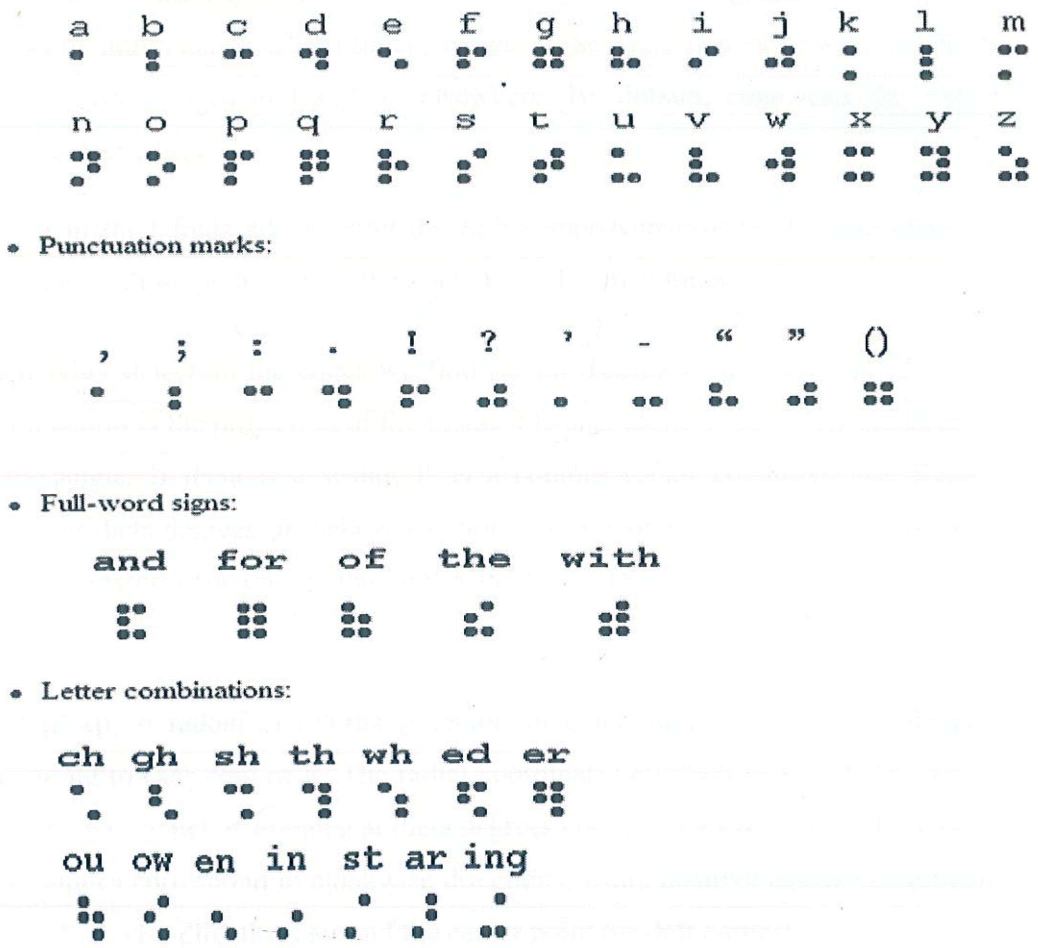


Figure 7.6: Standard English Braille[1]

7.4 English document creation: In this we will be using file-handling technique to create a new text document and every time as soon as we get our vectors matched the corresponding alphabets or special characters are appended in the same text document. Finally, after the whole process we will get an English text document corresponding to the initial Braille document.

7.5 SOLUTIONS OF THE TECHNICAL ANAMOLIES

7.5.1 If the image is scanned slightly tilted- To resolve this issue first

STEP 1: need to detect edges we detect edges by "edge" function in MATLAB [4]. Before using this function the image is first checked whether it is grey image or if it is not a grey image then first it is converted into a gray scale image.

STEP 2: Then we use edge function as follows. edge (I) takes a grayscale or a binary image I as its input, and returns a binary image of the same size as I, with 1's where the function finds edges in I and 0's elsewhere. By default, edge uses the Sobel method to detect edges.

The Sobel method finds edges using the Sobel approximation to the derivative. It returns edges at those points where the gradient of I is maximum.

STEP 3: After detecting the edges we find Radon transform of the image [5]. The Radon transform is the projection of the image intensity along a radial line oriented at a specific angle. If theta is a scalar, R is a column vector containing the Radon transform for theta degrees. If theta is a vector, R is a matrix in which each column is the Radon transform for one of the angles in theta. Here we use theta range from-90:90.

STEP 4:[R,xp] = radon(...) returns a vector xp containing the radial coordinates corresponding to each row of R. The radial coordinates returned in xp are the values along the x'-axis, which is oriented at theta degrees counter-clockwise from the x-axis. Negative angles correspond to clockwise directions, while positive angles correspond to counterclockwise directions around the center point (up-left corner).

STEP 5: R1 is a 1x180 vector in which, each element is equal the maximum value of Radon transform along each angle. This value reflects the maximum number of pixels along each direction.

[R1,r_max] = max(R);

r_max is a A 1x180 vector, which includes corresponding radii of 'R1'

7.5.1.1 Line detection

This section performs a search operation. It finds maximum value of Radon Transform over all radii and angles in angles greater than 50 or less than -50. First detected angle indicates the slope of the upper bond of the image.

7.5.1.2 Rotation correction: Finally after obtaining the value of the angle through which the image is tilted. The image is rotated in the opposite direction by the same angle to get the required image aligned perfectly to being processed further.

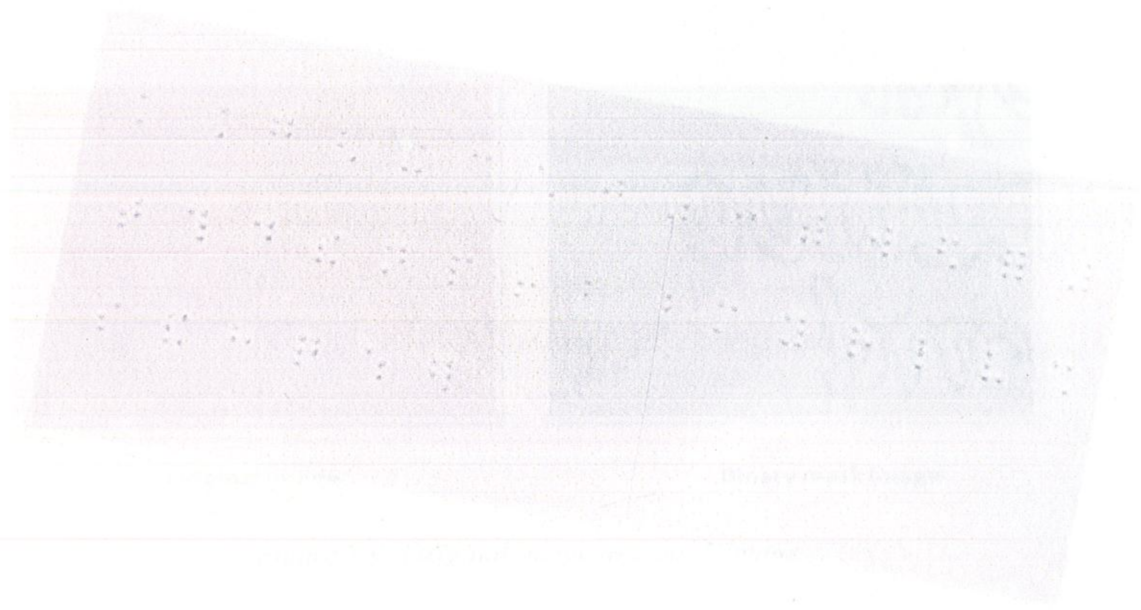


Figure: 7.7: Tilted image

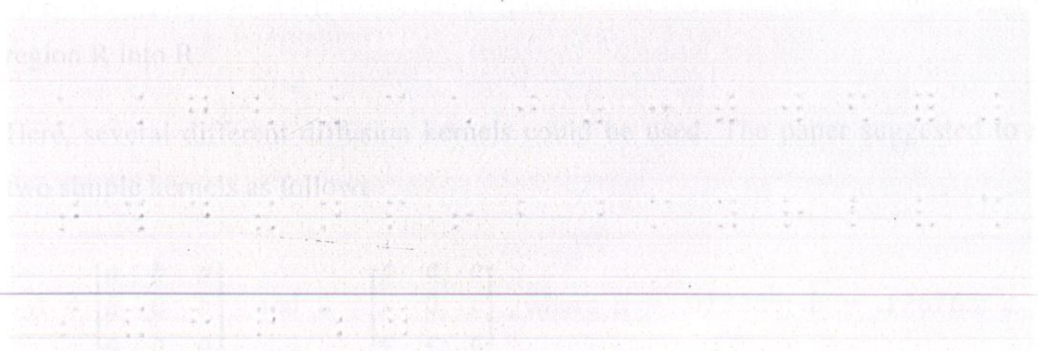
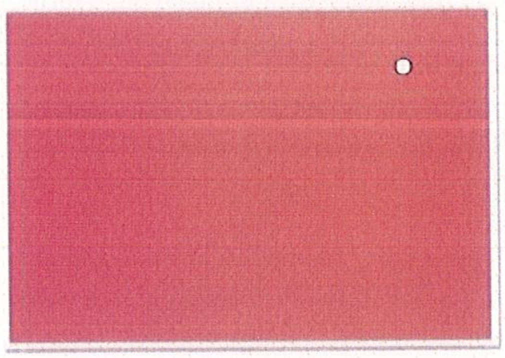


Figure 7.8: Image after rotation correction.

7.5.2 When the Braille document has a damaged portion: Scanned image obtained in such case will have a distortion or a spot. So it becomes compulsory for us to remove this distortion in order to further process the image.

7.5.2.1 Solution for such problem: Naturally, we should first define the area of missing or damaged in the original image. We call this image mask. Notice here mask image is a binary image, where only the missing and damaged area has been defined to one. As we see below, the left image is the original input and the right image is its mask.


Original image

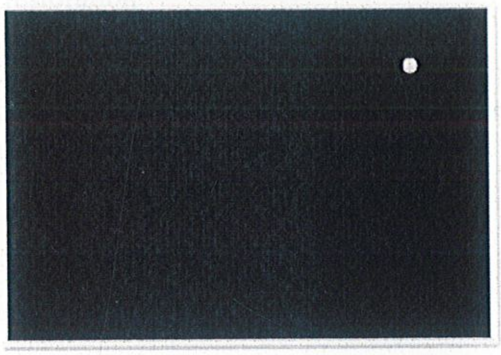

Binary mask image

Figure 7.9: Original image and mask image

After defining the repair region (missing or damaged area), then we run diffusion. Since the repair region R is very small, the in painting procedure can be approximated by an isotropic diffusion process that propagates information from the boundary of the region R into R.

Here, several different diffusion kernels could be used. The paper suggested to use two simple kernels as follows

$$A = \begin{bmatrix} a & b & a \\ b & 0 & b \\ a & b & a \end{bmatrix} \text{ and } B = \begin{bmatrix} c & c & c \\ c & 0 & c \\ c & c & c \end{bmatrix}, \text{ where } a = .073235;\ b = .176765;\ c = .125$$

Then after several diffusion iterations, we will achieve the desired restoration.[5]

7.5.2.2 INPAINTING ALGORITHM: The whole process of Image Inpainting with Fast Digital Image Inpainting approach could be stated as follows:

1. Read damaged image and its mask image

2. Clear damaged area in the original damaged image

3. Based neighborhoods, do isotropic diffusions inside of damaged area

4. Keep step 3 until the result satisfies some desirable condition

5. Retrieve the result of step 4 and let it be the restoration

UNSOLVED ISSUES AND FUTURE CORRECTIONS OF THE GIVEN PROJECT

1) **Curved Strips:** There might be the chances that there is a problem in which the image scanned has strips (text) which are curved in nature. Hence we have the task to extract Braille blocks in that condition by either properly aligning the curved strip horizontally or extracting Braille blocks from the curved strip only.

2) **Hardware implementation:** After interpretation of the image and converting into English text we can now implement our software to hardware part. Hence whenever a person touches the alphabets of original Braille document he or she can hear the corresponding interpretation in English through audio channels.

APPENDIX

Binary image: A binary image is a digital image that has only two possible values for each pixel. Typically the two colours used for a binary image are black and white though any two colours can be used.

Braille: The Braille system is a method that is widely used by blind people to read and write. Braille was devised in 1825 by Louis Braille, a blind Frenchman.

Edge detection: Find edges of objects in images using Sobel, Prewitt, Roberts and Canny methods.

Filter: It's is a device or process that removes from a signal some unwanted component or feature. Filtering is a class of signal processing, the defining feature of filters being the complete or partial suppression of some aspect of the signal.

Gray Image: Image in which the value of each pixel is a single sample, that is, it carries only intensity information. Images of this sort, also known as black-and-white, are composed exclusively of shades of gray, varying from black at the weakest intensity to white at the strongest.

Image processing: In electrical engineering and computer science, image processing is any form of signal processing for which the input is an image, such as a photograph or video frame; the output of image processing may be either an image or, a set of characteristics or parameters related to the image.

MATLAB: Numerical computing environment and fourth-generation programming language. Developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran.

Median filtering: The median filter is a nonlinear digital filtering technique, often used to remove noise. Such noise reduction is a typical pre-processing step to improve the results of later processing (for example, edge detection on an image). Median filtering is very widely used in digital image processing because, under certain conditions, it preserves edges while removing noise.

RGB Image: An RGB image has three channels: red, green, and blue. RGB channels roughly follow the color receptors in the human eye, and are used in computer displays and image scanners. If the RGB image is 24-bit (the industry standard as of 2005), each channel has 8 bits, for red, green, and blue—in other words, the image is composed of three images (one for each channel), where each image can store discrete pixels with conventional brightness intensities between 0 and 255. If the RGB image is 48-bit (very high resolution), each channel is made of 16-bit images.

Radon transform: The Radon transform is the projection of the image intensity along a radial line oriented at a specific angle.

Sobel method: The Sobel method finds edges using the Sobel approximation to the derivative. It returns edges at those points where the gradient of image is maximum.

Vector: Here vector is a 6-bit binary number which represent individual Braille blocks in the given image.

REFERRENCES

1. *www.wikipedia.com*
2. *M. Waris Abdullah and M. Wajid, "Urdu Braille reader", Department of Electronics, AMU Aligarh, 2006*
3. *Digital Image Processing Using MATLAB, 2nd ed. by Rafael C. Gonzalez*
4. *MATLAB help*
5. *Mathwork.com*