CHAPTER 4

NOVEL BRAILLE DOCUMENT IMAGE MOSAICING, INTER-POINT AND INTER-LINE BRAILLE DOCUMENT
Research has shown that the best way to be happy is to make each day happy.

- Deepak Chopra

CHAPTER – 4

NOVEL BRAILLE DOCUMENT IMAGE
MOSAICING, INTER-POINT AND INTER-LINE
BRAILLE DOCUMENT ANALYSIS

4.1 Preamble

Mosaic is an area of art, wherein small pieces of glass are assembled in a systematic way to get a decorative image. Glasses of different colors are used in mosaic art.

In Digital image processing, image mosaicing is the technique of assembling multiple pieces of images to get a combined image. This can be applied to a document image or a general image. It is the art of creating a big image with congregation of small images. There are many situations where it may not be possible to capture large documents with the given imaging media such as digital camera,
scanner or copying machines in a single stretch because of their inherent limitations. As a result the large document is captured as split components and they need to be mosaiced. Digital Image Mosaicing requires an overlapping region to generate a single and complete image. In the experiments conducted in this thesis, Braille documents have been considered for mosaicing and in the process novel approaches are evolved.

To reduce the cost of paper, Braille can be embossed on both sides of a sheet. In Inter-point Braille, Space between the dots of front embossed character is used to emboss the characters on the back side. Inter-point embossing leads to Smudging of dots of front and back embossed character and hence an alternate approach of back to back embossing is used called inter-line embossing. Space between the lines of front embossed character is used in embossing the back embossed character.

In this chapter, novel algorithms for Braille document mosaicing, Inter-point and Inter-line Braille document analysis has been presented. Comparison and superiority of the algorithms cannot be presented as there are no evidences for Braille mosaicing or inter-line mosaicing.

### 4.2 Image mosaicing techniques

Methods adopted for image mosaicing can be broadly classified into direct method and feature based methods. Direct methods are found to be useful for mosaicing large overlapping regions, small translations and rotations. Feature based methods can usually handle small overlapping regions and in general tend to be more accurate but computationally intensive.

Different techniques have been adopted by researchers for image mosaicing. Curvature domain image stitching approach, feature point detector, Local optimal flow registration method, column matching
approach are some of the techniques used for image mosaicing. Figure-4.1 shows pixel column based approach used in image mosaicing. Image blending and Image registration are the two key components for image mosaicing. Major task in image mosaicing is to find the geometrical relationship between the images to be mosaiced.

Figure – 4.1: Image Mosaicing

4.3 Document Image mosaicing

Image mosaicing, when applied to documents, results into document image mosaicing. Methods applied to normal images can also be applied to document images. Computation time can be reduced by applying suitable techniques depending on the nature of the document. Most of the document image processing is done on black and white images and hence mosaicing such images will further reduce the time complexity. Techniques like a Zernike moment use the relative grid transformation between the images for image mosaicing. Column wise, unidirectional and bidirectional scanning have also been used for document image mosaicing. Figure-4.2 shows an example of document image mosaicing. All the pixels of last column in ‘image1’ are mapped with first column of ‘image2’. If the pixels do not match, the process is repeated by moving from right to left in ‘image1’, keeping ‘image2 column constant. Once the mapping
is successful, then it indicates the beginning of overlapping region in ‘image1’. Image mosaicing is done by copying the entire image column from column1 to, the column before the mapping column, to the ‘RESULT’ image. All the columns of ‘image2’ are now copied to the ‘RESULT’ image appending to the existing image. In all, it is studied and noted that the concepts and methods to be adopted for Braille document mosaicing is different. The study has resulted into developing new Braille Document Mosaicing algorithm.

4.4 Braille document mosaicing

Digitization of Braille document is mainly done through a scanner. The sheet used for Braille document is usually bigger than A4 size sheet and hence it cannot be scanned in one shot using A4 size scanner. The alternate solution is to use A3 size scanner, which is expensive. Hence the document will be scanned in two stages and later they are mosaiced to get the full document for optical Braille recognition.

The techniques used for conventional document mosaicing cannot be used for Braille document mosaicing. The main reason being, the Braille characters are aligned in columns. The gap between characters
will be uniform throughout the document for all the lines and hence uniform pixel values in between the dots. This may mismatch the overlapping areas and the conventional mosaicing algorithms fail. Hence there is a need to design new algorithms for Braille document mosaicing. Figure-4.3 shows the process of Braille document mosaicing of two Braille document images that have a common portion in them.

![Figure – 4.3: Braille Document Mosaicing](image)

**4.4.1 Proposed method for Braille document mosaicing**

The experiment conducted is through scanning the Braille document via a Digital scanner. The document which is bigger than A4 size scanner is scanned in two shots. It is as shown in Figure-4.4. The process produces two digital Braille documents which have common portion. Consider them as ‘img1’ and ‘img2’.

![Figure– 4.4 : Scanning the Pieces of the Braille Document](image)

Column based approach followed in Document image mosaicing is used for mosaicing the Braille images, but the difference is, instead of pixel by pixel mapping, the embossed Braille dots are used for mapping.
The Braille documents are pre-processed to eliminate all the noise and correct the skew if any. Then they are further processed to find the overlapping regions.

Both the Braille document images are considered individually and processed row wise, from top left corner pixel to bottom right corner. The top and bottom edges of each Braille dot row are identified and the process is repeated till the end of the document. The process is repeated column wise and the left and right edge for every dot column is identified. The process is repeated till the end of the document.

The row and column numbers in the first Braille document image are recorded as \( r_{11}, r_{12}, r_{13}, \ldots, r_{1n} \) and \( c_{11}, c_{12}, c_{13}, \ldots, c_{1n} \) respectively and the pixel row and column values in the database are stored. Similarly the second Braille document’s row and column values are referenced as \( r_{21}, r_{22}, r_{23}, \ldots, r_{2n} \) and \( c_{21}, c_{22}, c_{23}, \ldots, c_{2n} \) respectively.

The sub-image between \( c_{1n} \) and \( c_{1(n-1)} \) between last two columns, gives the last Braille dot column in the first image ‘img1’. The resulting sub-image is formed out of co-ordinates:

\[
(r_{11}, c_{1(n-1)}) \text{ and } (r_{1n}, c_{1n})
\]

The objects within the sub-image and their centroid are identified. The objects are recorded as Obj11…Obj1n and their centroids as \( (x_{11}, y_{11}) \) to \( (x_{1n}, y_{1n}) \). The number of objects in this case is ‘N1’.

In the second Braille document image ‘img2’, first Braille dot column is extracted with the co-ordinates:

\[
(r_{21}, c_{21}) \text{ and } (r_{2n}, c_{22})
\]
The objects within the sub-image and their centroids are identified. The objects are Obj21…Obj2n and their centroids are (x21, y21) to (x2n, y2n). The number of objects in this case is ‘N2’.

If the numbers of objects in both the cases are same i.e, N1=N2, then further comparisons are done:
Get the absolute difference between the corresponding objects of each image. For ‘n’ objects:

\[ d_{11} = |(x_{11}-x_{21})| \] and \[ d_{12} = |(y_{11}-y_{21})| \]

\[ d_1 = d_{11} + d_{12} \]

\[ d_{12} = |(x_{12}-x_{22})| \] and \[ d_{22} = |(y_{12}-y_{22})| \]

\[ d_2 = d_{12} + d_{22} \]

\[ \ldots \]

\[ d_n = d_{1n} + d_{2n} \]

\[ \text{distance} = \sum d_i \text{ where } i=1 \text{ to } n \]

If the ‘distance’ is less than a pre-determined threshold value then it implies that the Braille dot columns overlap. In the above experiment the threshold fixed as10.

Else the process is repeated by considering each dot column lying to the left of the last column of ‘img1’ and mapping them with the first column in ‘img2’, till the mapping is successful.

Consider ‘img1’ co-ordinates
(0, 0) – Top left co-ordinate
(m, n) – Bottom right co-ordinate

Similarly Consider ‘img2’ co-ordinates
(0, 0) – Top left co-ordinate
(p, q) – Bottom right co-ordinate
Once the mapping column for the first Braille dot column in ‘img2’ is found in ‘img1’, and with mapping Braille dot column co-ordinates in ‘img1’ are 

(0, m1)- top left co-ordinate and the other 
(m, n1)- is the Bottom right co-ordinate

Copy contents of ‘img1’ and ‘img2’ to ‘result’ image to be resulted as described below:

Copy the sub-image from ‘img1’ to ‘result’: (0, 0) to ((m1-1), n)
Append from ‘img2’ to the ‘result’ : (0, 0) to (x, y)

Now the resultant image ‘result’ is the mosaiced image of ‘img1’ and ‘img2’ eliminating the overlapping regions. It is shown in Figure-4.5.

Figure - 4.5: Row and Column Marking for the Braille Documents
4.4.2 Algorithm: Braille document mosaicing

Procedure begins

Step 1: Start

Step 2: Read the two Braille document image pieces ‘img1’ and ‘img2’ which are to be mosaiced.
  Read image (‘img1’) of size (a x b)
  Read image (‘img2’) of size (c x d)
  The number of rows of the two images must be equal a=c

Step 3: Preprocess the Braille documents to convert the images into Black and white images and eliminate the noise and extra dot which are not the part of Braille character.

Step 4: Process row wise from top left corner pixel to bottom right corner and identify the top and bottom edge of each Braille dot row. Repeat the process till the end of the document.

\[
\text{row= 1}
\]
\[
\text{For } i=1\ldots a
\]
\[
\text{For } j = 1\ldots b
\]
\[
\text{Read the pixel } p(i,j) \text{ and get the intensity of it.}
\]
\[
\text{If } (p(i,j) = 1) \text{ then white=}\text{white}+1
\]
\[
\text{Else black=}\text{black}+1
\]
\[
\text{End for}
\]
\[
\text{If } (\text{row mod 2 }<> 0)
\]
\[
\text{If } (\text{white}<>\text{b})
\]
\[
\text{It is the upper edge of the Braille dot}
\]
\[
\text{Record this as:}
\]
\[
\text{r1[row]}=i;
\]
\[
\text{row=}\text{row}+1
\]
\[
\text{End if}
\]
\[
\text{else (row mod 2=}0\text{)}
\]
\[
\text{If } (\text{white}=\text{b})
\]
\[
\text{It is the lower edge of the Braille dot}
\]
\[
\text{Record this as:}
\]
\[
\text{r1[row]}=i;
\]
\[
\text{row=}\text{row}+1
\]
\[
\text{End if}
\]
\[
\text{End if}
\]
\[
\text{End for}
\]
Step 6: Repeat the process column wise and identify the left and right edge for every dot column. Repeat the process till the end of the document.

Column=1
For i=1…b
  For j = 1…a
    Read the pixel p(i,j) and get the intensity of it.
    If (p(i,j) = 1) then white=white+1
    Else black=black+1
  End for
  If (column mod 2 <> 0)
    If (white<>a)
      It is the left edge of the Braille dot
      Record this as:
      c1[column]=i;
      column=column+1
    End if
  else (column mod 2=0)
    If (white=a)
      It is the right edge of the Braille dot
      Record this as:
      c1[column]=i;
      column=column+1
    End if
  End if
End for

Step 7: Recorded row and column numbers in the first Braille document image are

r11, r12, r13…r1n and c11, c12, c13 ...c1n

respectively and store the pixel row and column values in the database.

Step 8: Repeat step 5 and 6 and second image ‘img2’ with dimension (cxd)
Second Braille document’s row and column values are Identified as

r21, r22, r23…r2n   and c21, c22, c23,...c2n

respectively and store the pixel row and column values in the database.

Step 9: Between the c1n and c1 (n-1), the last two columns of first image we have the last Braille dot column. Extract this as sub-image and get the number of objects ‘ob1’ and the co-ordinate
values of centroids of each of this object. The co-ordinates of
the centroid are recorded as:

(x11, y11),(x12,y12)...(x1n,y1n).

Where each dot, is a true Braille dot of Braille character.

Step 10: In the second Braille document ‘img2’ extract the first Braille
dot column and get the count of objects ‘ob2’ in this sub-image
and the Co-ordinate values as

(X21, y21),(x22,y22)...(x2n,y2n)

Step 11:

If (ob1 = ob2)
   Get the Euclidean distance ‘d’ between set of co-ordinate
   values of
   ‘img1’ and ‘img2’
   If (d<10)
       then the two columns are the overlapping columns
       and record the overlapping pixel column as ‘m1’,
       Mosasicked image ‘result’ is constructed by first
       copying pixels column-wise from first column till
       (m1-1) pixel column in ‘img1’ to ‘result’. Now
       append all the pixel column of ‘img2’ to ‘result’.
       Go to step 13
   End if
else
   go to step 12.
End if

Step 12: Move from right to left in extracting next Braille dot column
in ‘img1’.

   If the end of Braille dot then
   Display (‘mosaicing is not possible’)  
   Go to step 13
   Else

   Get the count of objects ‘ob1’ and the object’s co-
   ordinates from ‘img1’. The count of objects ‘ob2’ and
   object’s co-ordinates of ‘img2’ are retained
   Go to step 11.

End if

Step 13: Stop.

Procedure ends
4.4.3 Experimental setup: Results and Discussion

The performance of the proposed algorithm is investigated on 20 sets of documents. The documents used are independent of the language. The documents are digitized using a standard A4 size scanner HPM1005 with 600 DPI. The proposed algorithm designed considers 2 pieces of Braille documents at a time for mosaicing.

Dimension of the documents used for the experimentation for both Document 1 ('img1') and Document 2 ('img2') have paper size of 8 inch by 10 inch and with other parameters:

Top Margin : 1 lines
Bottom Margin : 1 lines
Inner Margin : 5 Characters
Outer Margin : 3 Characters
Calculated Values : 25 Characters per line
Embossing : Single side embossing
Number of Braille dot columns : 50

Performance of this algorithm is given in Table-4.1. The performance is linear with the number of Braille dot columns overlapped. Table-4.2 gives the average performance in terms of number of Braille dot columns overlapped. Figure-4.6 shows the chart for the average performance by considering the number of Braille dot columns overlapped against the time taken for mosaicing.
Computation time is linear with number of Braille dot columns overlapped. As the number of Braille dot columns increases, computation time also increases. To reduce the time complexity, the overlapping regions are reduced.
4.5 Inter-point Braille

Using a Braille slate, Braille can be embossed on one side of the paper. Braille printer or Braille embosser can print simultaneously on both sides of a paper. This reduces the cost of paper by 50%. Double sided printout uses inter-point system of embossing. Only 22 lines can be embossed per sheet in normal embossing. In inter-point embossing with both side printing 44 lines can be embossed per sheet on a 8x10 inch sized sheet. To emboss on the back side, the space between the dots of the front side is used. Figure-4.7 shows the template of inter-point Braille.

Inter-point Braille document appears to be having little mountains (front side embossing-recto dots) and valleys (back side embossing-verso dots). The reading finger of visually impaired will read only the bulged mountains of the dots and hence only one side of the document can be read at a time by human reading. The valleys are to be read from other side of the document.

Figure – 4.6: Number of Overlapped Columns vs. Time Taken
Figure – 4.7 : Inter-Point Braille template

Figure – 4.8 : Digitized Back to Back Embossed Braille document (Scanner Light Focused From Top Edge)

Figure – 4.9 : Pre-Processed Back to Back Embossed Braille document (scanner Light Focused From Top Edge)
When the inter-point Braille is digitized, characters on both sides are recognized at one scan. Scanned Braille document will have semi circular patterns of dark regions for mountains and valleys depending on the position of the scanner light.

If the scanner lights fall from the top edge of the back to back embossed Braille document, then dark region will be semi-circular, for the mountains at the bottom half of the Braille dot and at the top half, for the valleys which is shown in Figure- 4.8 and Figure- 4.9. If the scanner light is focused from the left side of the Braille document, the dark regions will be semi-circular for the mountains at the right half of the Braille dot and at the left half of the Braille dot, for the valleys which is shown in Figure-4.10, and Figure-4.11.

Figure – 4.10 : Digitized Interpoint Braille document
(Scanner Light Focused From Left Edge)
Before processing the Interpoint Braille document for character recognition, the front and back side of the embossing are to be separated. Very less research has been done in the area of separating interpoint braille document. The research carried out till dated is based on the dark pattern created. Efficiency of recognition depends on the dark pattern created and it also requires the knowledge of the light position. The work carried out in this thesis proposes a novel algorithm to overcome the problems faced by the contemporary researchers.

4.5.1 Proposed method

A interpoint braille document is pre-processed to eliminate all noise and correct the skew if any. The document is further processed to separate the recto and verso dots which are the front and back embossed dots respectively. The conventional methods which use the shape of the dots may fail because the shape may vary depending on the lighting condition. Smudging of verso and recto dots may also lead to creation of different shaped dots which may result in erroneous recognition. The proposed method overcomes all the drawbacks of the previous research work and uses dot position instead of the dot shape. This concept significantly enhances the accuracy of recognition, and increases the computation efficiency both in terms of time and space. The Average Braille dot size is computed using successive experiments. Dot distance within character, between the characters and between the lines is fixed in a Braille document and is as discussed in section 1.5.2.
The document is processed from top left edge, row wise till the upper edge of the first dot row is obtained. Similarly the left edge of the left Braille dot column is marked as shown in Figure-4.12.

Figure – 4.12: Marking Upper and Left Edge of The Braille Document

The Braille document is processed from bottom row to identify the bottom edge of the last Braille dot row. Similarly the right edge of the rightmost Braille dot column is identified. The four edges of the Braille dot column are identified as shown in Figure-4.13.

Figure – 4.13: Marking Upper, Bottom Left and Right Edge of the Braille Document
The edges of the Braille document which do not have any role in computation are removed, as shown in Figure-4.14.

The average size of the Braille dot is computed. The vertical space between the dots, within the character and space between the lines being known, moving from the top edge horizontal lines are marked till the end of the document and the coordinates are recorded. The process is repeated vertically from left edge moving towards right edge as shown in Figure-4.15.

Figure – 4.14: Pre-Processed Braille after Removing the Edges

Figure – 4.15: Braille Document with Horizontal and Vertical Dot Positions Marked

95
In the marked lines, the vertical odd lines (c1, c3, c5, c7, ...) represent the front faced Braille dots while the even vertical lines (c2, c4, c6, c8, ...) represent back embossed Braille dots.

The first character in the first line can have dots in positions, where c1,c3 meet r1,r3,r5. To find the presence of dots, in the six possible positions of the first front embossed character, a mask of size 15X15 ones (White) is multiplied element by element (dot multiplication), with the possible dot of size 15x15, having center at the row and column meeting positions. If in the resultant matrix all the contents are 1 then no dot is present else indicates the presence of dot. This is done with histogram computation. the process is repeated for all the six dot positions and the data about presence or absene of dots is obtained. The mask used in the process is shown in Figure-4.16.

![Figure - 4.16 : Mask Used for Detection of Dots](image)

The process is repeated till the end of the line, to obtain all the front faced Braille characters in the first line and the process is continued till the end of the document.
In the next iteration the entire process is repeated to find the back embossed character dots. Column numbers $c_n, c_{n-2}$ (alternate columns) and row number $r_2, r_4, r_6$ are positions at which first line first character dots will be found.

### 4.5.2 Algorithm: Inter-point Braille character recognition

**Procedure Begins**

Step 1: Start

Step 2: Read the Braille document image ‘d1’ of size ‘axb’

Step 3: Preprocess the document to convert the document into binary image and to eliminate the noise and extra dot which are not the part of Braille character.

Step 4: Eliminate the extra edge parts of the document which does not carry any data.

Step 5: Use the knowledge base about the size of the Braille dot, distance between the dots within the character and between the characters, between the Braille character lines and also between the inter-point Braille dots.

Step 6: Top edge of the document represents the upper edge of the first Braille dot row. Based on the Knowledge base of step 5, draw the marking line horizontally in the middle of the dot. Repeat the process from top to bottom of the document. Record the ‘x’ axis value in the database as

$$r[1], r[2], r[3], \ldots r[n]$$

Step 7: Left edge of the document will contain the left most Braille dot column. Based on the Knowledge base of step 5, draw the marking line vertically in the middle of the dot. Repeat the process from left to right edge of the document. Record the ‘y’ axis value in the database as

$$c[1], c[2], c[3], \ldots c[n]$$

Step 9: Front embossed dots are associated with rows and columns (odd numbers)

$$(r_1, r_3, r_5, r_7, r_9\ldots)$$ and

$$(c_1, c_3, c_5, c_7, c_9\ldots)$$
Back embossed dots are associated with rows and columns

(Even numbers)

(r2, r4, r6, r8, r10 ...) and

(c2, c4, c6, c8, c10 ...)

Step 10: First two odd number columns and three odd number rows intersect points can have Braille dots corresponds to first character of first line front faced Braille character. Read all the front embossed characters in the document having ‘x’ number of characters and ‘y’ lines in the document.

(front embossed)

while(1) (for number of characters)

while (1) (for number of lines)

Keep (r[j], c[i] ) as center and extract 15x15 pixels from the document image and this sub image is ‘dot’

\[
dot[15][15] = d1[r[j]-7][c[i]-7]: d[r[j]+7][c[i]+7]\]

Dot multiply ‘dot’ image with ‘mask’ image of 15x15 ones.

res = ‘dot’ (dot multiplied by) ‘mask’

If ‘res’ is all ones then

Dot absent

pattern = pattern (append) 0

Else

Dot present

pattern = pattern (append) 1

End if

j = j + 2

count = count + 1

If (count=3)

i = i + 2

j = 1;

count = 0

End if

If(c=6)

then

Based on the ‘pattern’ analyze the Braille character and
convert into normal language character
set pattern='nil'
char=char+1
(prepare to read next character)
End if

End while
If (char=x)
  line=line+1
  j=j+1
  i=i
  char=0
End If
If (line=y)
  End of document reached
End if

Step 11: Repeat step 10 for all characters of the back embossed by setting
         i=2 and j=2
Step 12: End.

Procedure Ends.

4.5.3 Experimental setup : Results and Discussion

The performance of the proposed system is evaluated on 10 different documents of Inter-point Braille document. It is independent of the language. The documents used are digitized using a standard A4 size scanner HPM1005 with 600 DPI. Arrangements were done to get the scanner light from the top edge of the Braille document.

Paper dimension Used for the experiment
Custom paper of size of 8 inch by 10 inch.
Top Margin : 1 line
Bottom Margin : 1 line
Inner Margin : 5 Characters
Outer Margin : 3 Characters
Calculated Values : 25 Characters per line with
                   : 22 Front embossed lines
                   : 22 Back embossed lines
                   : Maximum of 1100 Braille cells per page
                   (including blank characters)
95.39% of success rate was achieved for the given document. The success largely depends on the quality of the document and the inter point Braille embossing. Table-4.3 shows the performance analysis of algorithm on Inter-point braille document. Performance is compared with number of true dots against number of dots recognized. Performance analysis of front and back embossed dots with actual and recognized dots on different documents under experimentation are shown in Figure-4.17 and Figure-4.18 respectively.

Table – 4.3 : Performance Analysis of Inter-Point Braille

<table>
<thead>
<tr>
<th>No.</th>
<th>Total number of true dots in the document</th>
<th>Number of true front embossed dots recognized</th>
<th>Number of true back embossed dots recognized</th>
<th>Total number of dots recognized (both front and back)</th>
<th>% of success</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3567</td>
<td>1835</td>
<td>1769</td>
<td>1732</td>
<td>1634</td>
</tr>
<tr>
<td>2</td>
<td>3528</td>
<td>1930</td>
<td>1802</td>
<td>1646</td>
<td>1678</td>
</tr>
<tr>
<td>3</td>
<td>2457</td>
<td>1423</td>
<td>1390</td>
<td>1059</td>
<td>992</td>
</tr>
<tr>
<td>4</td>
<td>2693</td>
<td>1676</td>
<td>1598</td>
<td>1017</td>
<td>956</td>
</tr>
<tr>
<td>5</td>
<td>2950</td>
<td>1614</td>
<td>1514</td>
<td>1336</td>
<td>1279</td>
</tr>
<tr>
<td>6</td>
<td>3310</td>
<td>1581</td>
<td>1499</td>
<td>1729</td>
<td>1656</td>
</tr>
<tr>
<td>7</td>
<td>3274</td>
<td>1871</td>
<td>1707</td>
<td>1403</td>
<td>1289</td>
</tr>
<tr>
<td>8</td>
<td>3900</td>
<td>2259</td>
<td>2167</td>
<td>1641</td>
<td>1456</td>
</tr>
<tr>
<td>9</td>
<td>3081</td>
<td>1626</td>
<td>1578</td>
<td>1455</td>
<td>1398</td>
</tr>
<tr>
<td>10</td>
<td>4002</td>
<td>2029</td>
<td>1987</td>
<td>1973</td>
<td>1899</td>
</tr>
</tbody>
</table>

Average success: 0.953923
4.6 Inter-line Braille

Though interpoint braille is optimal in utilizing the space on the sheet, there is a possibility of smudging of the front embossed dots by the back embossed dots or vice-versa. As an alternative and little more complex Inter-Line Braille can be used where in alternate lines are embossed front and back. It can have 18 front embossed and 17 back embossed lines in a sheet of 8x10 inch size.

Figure – 4.18 : Inter-point Braille - Back Embossed dot recognition

Figure – 4.19 : Inter-line Braille document
Figure-4.19 is an example document for inter-line Braille document. Pre-processed version of this document is as shown in Figure-4.20.

![Figure - 4.20 : Pre-processed Inter-line Braille document](image)

In an inter-line document, the dots might overlap column wise. With this knowledge the inter-line braille document is separated line wise. The Braille dots are processed and upper, lower, left and right edges are recorded as shown in Figure-4.21. Document is trimmed to the extreme edge dot markings as shown in Figure-4.22.

![Figure - 4.21 : Identification of Upper, Lower, Right and Left Edges for the Document](image)
In the next step the document is segmented linewise. The first line of braille character line will be having Braille dots in three rows which are extracted as shown in Figure-4.23 and processed to identify the upper, lower, left and right edges of the dots in that line as shown in Figure-4.24.
Dark regions which are in semi-circular shape are analysed to find the front or back embossed dots. A mask of size 15x24 is designed by looking into the pattern of the front embossed dot as shown in Figure-4.25. The dot under consideration is extracted and its histogram is computed and recorded in the database ‘d1’. The dot is multiplied element wise (dot multiplication) with the mask. The resultant matrix histogram is computed and recorded in the database ‘d2’. If the euclidean distance is less than 30 then it is front embossed else back embossed dot.

Depending on whether they are front or back embossed, they are processed separately for recognizing the Braille characters separately. If front embossed, the processing is done from left to right else form right to left for the back embossed line. Algorithm presented in the next section gives the steps for Inter-Line Braille character recognition.

![Figure – 4.25: Mask to Identify Back and Front Embossed Dot](image)

**Figure – 4.25 : Mask to Identify Back and Front Embossed Dot**
4.6.1 Algorithm: Inter-line Braille Character Recognition

Procedure Begins

Step 1: Start

Step 2: Read the inter-line Braille document ‘d1’.

Step 3: Preprocess the Braille documents to eliminate the noise and extra dot which are not the part of Braille character.

Step 4: Compute the upper, lower, right and left edges of the Braille dots.

Step 5: Edges which do not carry any Braille dot are trimmed.

Step 6: First Braille ‘bl1’ line having three Braille dot rows are extracted.

Step 7: Extracted dots are processed to get mark the upper, lower, left and right edges of the dots.

Step 8: Detect whether the dots in this line are representing front or back embossed dot by passing a suitable mask.

Step 9: Extract one Braille character at a time:

If it is front embossed,
   then use the dimension of the Braille cell and extract two Braille columns at a time from left moving towards right.
Else
   It is back embossed. Extract two Braille column at a time from right moving towards left.

End if

Step 10: Dots are analyzed and processed to get the equivalent normal language character.

Step 11: Repeat steps 9 and 10 till the end of the line ‘bl1’.

Step 12: Steps 6 to 11 for the Braille lines till the end of the document ‘d1’.

Step 13: Stop.

Procedure Ends
4.6.2 Experimental Setup : Results and Discussion

The performance of the proposed system is evaluated on 10 different documents of Inter-line Braille document. It is independent of the language. The documents used are digitized using a standard A4 size scanner HPM1005 with 600 DPI. Arrangements were done to get the scanner light from the top edge of the Braille document.

Paper dimension Used for the experiment
Custom paper of size of 8 inch by 10 inch.

- Top Margin : 1 lines
- Bottom Margin : 1 lines
- Inner Margin : 5 Characters
- Outer Margin : 3 Characters
- Calculated Values : 25 Characters per line with
  - 18 Front embossed lines
  - 17 Back embossed lines
  - Maximum of 875 Braille cells per page (including blank characters)

Table – 4.4 : Inter-Line Braille Document Recognition Rate

<table>
<thead>
<tr>
<th>Sl.No.</th>
<th>Total number of true dots in the document</th>
<th>Number of true front embossed dots recognized</th>
<th>Number of true back embossed dots recognized</th>
<th>Total number of dots recognized (both front and back)</th>
<th>% of success</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2400</td>
<td>1440</td>
<td>1236</td>
<td>960</td>
<td>0.887083</td>
</tr>
<tr>
<td>2</td>
<td>2900</td>
<td>1656</td>
<td>1598</td>
<td>1244</td>
<td>0.976000</td>
</tr>
<tr>
<td>3</td>
<td>1986</td>
<td>1403</td>
<td>1398</td>
<td>583</td>
<td>0.978852</td>
</tr>
<tr>
<td>4</td>
<td>769</td>
<td>769</td>
<td>723</td>
<td>0</td>
<td>0.940182</td>
</tr>
<tr>
<td>5</td>
<td>2900</td>
<td>1614</td>
<td>1575</td>
<td>1286</td>
<td>0.968621</td>
</tr>
<tr>
<td>6</td>
<td>3100</td>
<td>1745</td>
<td>1742</td>
<td>1355</td>
<td>0.987419</td>
</tr>
<tr>
<td>7</td>
<td>2740</td>
<td>1403</td>
<td>1326</td>
<td>1337</td>
<td>0.958759</td>
</tr>
<tr>
<td>8</td>
<td>1490</td>
<td>1341</td>
<td>1289</td>
<td>149</td>
<td>0.960403</td>
</tr>
<tr>
<td>9</td>
<td>2001</td>
<td>1423</td>
<td>1403</td>
<td>598</td>
<td>0.984508</td>
</tr>
<tr>
<td>10</td>
<td>3002</td>
<td>1678</td>
<td>1624</td>
<td>1324</td>
<td>0.967022</td>
</tr>
</tbody>
</table>

Document is machine embossed and an overall accuracy of 96.02 % is achieved with the given experimental setup. The result analysis is given in Table -4.4. Number of true dots are compared against the number of dots recognized. Table also gives information
about the front and back embossed true dots vs. number of dots recognized. Performance analysis of dot recognition for front and back embossed dots are shown in Figure-4.26 and Figure-4.27 respectively.

**Figure – 4.26 : Inter-line Braille - Front Embossed dot recognition**

**Figure – 4.27 : Inter-Line Braille - Back Embossed Dot Recognition**
4.7 Conclusion

In this chapter three new algorithms are presented for pre-processing the Braille document. The proposed algorithm for Braille document image mosaicing is the first appearance in literature. The chapter also covers algorithms for Interpoint Braille document processing and Inter-line Braille document analysis. All the presented algorithms are novel and highly contributing to the Braille world. With the set of algorithms presented in chapter 3 and 4, Braille document is pre-processed and it is ready for recognizing the characters which has been presented in chapters 5 and 6.