4.1. Introduction

The process of document conversion includes scanning, displaying, quality assurance, image processing, and text recognition. After document scanning, various data preprocessing operations are applied sequentially to the document images in order to put them in a suitable format ready for feature extraction. Section 4.2 discusses major preprocessing steps in handwritten character recognition systems. In section 4.3, reviews of different thinning algorithms are carried out. Section 4.4 proposes a novel modified thinning algorithm. In section 4.5, comparative analysis of performance of different thinning algorithms is presented. In section 4.6 reviews of contour tracing techniques and a novel contour tracing algorithms are explained. Finally, section 4.7 concludes the chapter.

4.2. Major Pre-processing Steps

The preliminary step for recognizing handwritten character is the preprocessing, which involves operation on the digitized image intended to reduce noise and to simplify extraction of structural and statistical features. The preprocessing algorithms employed on the scanned image depend on many factors such as age of the document, paper quality and resolution of the scanned image, amount of skew in the image, format and layout of the images and text, kind of the script used and also whether the characters are printed or handwritten. The preprocessing operations in character recognition include noise removal, slant correction, normalization, thinning, binarization, and edge detection. Preprocessing stage involves a set of operations to produce a
modified image which is less complex, so that it can be used directly and efficiently by the feature extraction stage. In this work focus is given to noise removal, size normalization, binarization, thinning and contour tracing.

4.2.1. Smoothing and Noise Removal

The process of document scanning introduces noises. Noise in image is a major obstruction in pattern recognition. Noise degrades the image quality. Smoothing is a widely used procedure for eliminating the noises introduced during image capture. Linear and non-linear filtering [Gonzalez & Woods, 2002] as well as morphological operations are widely employed in noise reduction.

4.2.2. Character Normalization

Character normalization is considered to be the most important preprocessing operation for character recognition. Character can have different sizes, positions and orientation. The goal for character normalization is to reduce the within-class variation of the shapes of the characters in order to facilitate feature extraction process and also improve their classification accuracy. Basically, there are two different approaches for character normalization: linear methods and nonlinear methods [Mohamed et al., 2007]. In this work the segmented character images obtained from document image are cropped to fit into minimum rectangle and subsequently normalized to a standard size 72x72 (Figure 4.1) using nearest neighbor interpolation method [Gonzalez & Woods, 2002]. The standard size is chosen considering maximum size reduction with minimum loss of information and shape distortion and is fixed by empirical analysis.
4.2.3. Binarization

A handwritten document is first scanned and is converted into a grey scale image. Binarization is a technique by which the grey scale images are converted to binary images, that is, it separates the foreground and background information. The most common method for binarization is to select a proper intensity threshold for the image and then to convert all the intensity values above the threshold to one intensity value, and to convert all intensity values below the threshold to the another chosen intensity. Thresholding (Binarization) methods can be classified into two categories: global and local (adaptive) thresholding. Global methods apply one threshold value to the entire image. Local or adaptive thresholding methods are performed in one pass using local information obtained from the image [Wu & Amin, 2003]. The threshold value is determined by the neighborhood of the pixel to which the thresholding is being applied. Among global techniques, Otsu’s thresholding technique [Otsu, 1979] has been cited as an efficient and frequently used technique [Leedham et al., 2003]. [Solihin & Leedham, 1999], [Berson, 1986], [Eikvil et al., 1991], and [Yanowitz and Bruckstein, 1989] used local methods to calculate the explicit thresholds. [Niblack, 1986] and [Zhang & Tan, 2001] used a gradient technique in local thresholding.

In this work, the Otsu’s thresholding selection technique [Otsu, 1979] is used. The implementation detail of this algorithm is described below.
1. Read an input image sample

2. Take the random intensity value in the range 0….L-1 as threshold value 
k and divide the total pixels S in the image in two groups.

\[ C_0 = [0, 1… k-1] \]

\[ C_1 = [k, k+1… L-1] \]

where L is the possible total intensity levels.

3. Compute

\[ W_0 = \sum_{q=0}^{k-1} P_q(r_q) \]

where

\[ P_q(r_q) = \frac{\text{Total pixels with intensity } q}{\text{Total pixels in the image}} \]

4. Compute \( W_1, u_0, u_1 \) and \( u_t \) with the following formulas

\[ W_1 = \sum_{q=k}^{L-1} P_q(r_q) \]

\[ u_0 = \frac{\sum_{q=0}^{k-1} qP_q(r_q)}{W_0} \]

\[ u_1 = \frac{\sum_{q=k}^{L-1} qP_q(r_q)}{W_1} \]

\[ u_t = \sum_{q=0}^{L-1} qP_q(r_q) \]

5. Find the class variability \( \lambda^2 \) between \( C_0 \) and \( C_1 \) where

\[ \lambda^2 = W_0(u_0 - u_t)^2 + W_1(u_1 - u_t)^2 \]
6. Repeat the above steps (for different k) until with maximum class variability is obtained

7. Choose Otsu’s threshold value (k) with maximum class variance.

The experimental results show that the Otsu’s thresholding algorithm can be effectively applied on the Tamil handwritten image database for threshold selection. The result of the Otsu’s thresholding algorithm when applied on Tamil handwritten character images is shown in Figure 4.2.

![Input: Gray Scale image and Output: Binary image](image)

Figure 4.2 Input gray-scale image and its binary output obtained using Otsu’s Algorithm
4.2.4. Thinning

Thinning has been used in different applications such as medical image analysis, bubble-chamber image analysis (a device for viewing microscopic particles), text and handwritten recognition analysis, materials analysis, fingerprint classification, printed circuit board design and robot vision. Thinning is the process of reducing an object in a digital image to the minimum size necessary for machine recognition of that object. After thinning, analysis on the size reduced image can be performed. Thinning is an important step for designing efficient handwritten character recognition (HCR) system. Offline handwritten characters have great variations in character thickness. Thinning eliminates the influence of character thickness on feature extraction.

4.2.5. Contour Tracing

The contour is essential for general shape representation. The boundary contour is a meaningful external representation of area of interest. It is widely used in many applications such as character recognition systems and content based image retrieval systems [Gonzalez & Woods, 2002].

In order to extract information about their general shape, the contour tracing is one of the many preprocessing techniques performed on digital images. Once the contour of a given pattern is extracted, its different characteristics can be examined and used as feature in pattern classification. The correct extraction of contours will produce more accurate features which in turn increase the chances of correctly classifying a given pattern [Mohamed et al., 2007]. The contour pixels are generally a small subset of the total number of pixels representing a pattern. Therefore, the amount of computation is greatly reduced when we run feature extraction algorithms on the contour instead of on the whole pattern.
4.3. Review of Thinning Algorithms

Pattern recognition and image processing applications frequently deal with raw inputs that contain lines of different thickness. In some cases, this variation in the thickness is an asset, enabling quicker recognition of the features in the input image. In other cases, the variation can be a liability and can cause degradation in the accuracy and the speed of recognition. For example, in the case of handwritten characters, the degree of uniformity of the thickness of individual strokes directly impacts the probability of successful recognition; especially if neural network based recognition techniques are employed. For the latter category of applications, uniform thickness can be attained, prior to recognition stage, by first thinning the input pattern to a thickness of a single pixel. Thinning or skeletonization is a process by which a one-pixel-width representation (or the skeleton) of an object is obtained, by preserving the connectedness of the object and its end points [Gonzalez & Woods, 2002]. Thinning algorithms should also preserve topological and geometric properties of the original object as much as possible. This includes connectedness of components, no spurious endpoints, and no excessive erosion of the original object.

Thinning algorithms can be classified into two broad categories (Figure 4.3):

1) Iterative thinning algorithms

2) Non-iterative thinning algorithms.

In general, iterative thinning algorithms perform pixel-by-pixel operations until a suitable skeleton is obtained. Non-iterative thinning methods use sequential pixel scan of an image. Iterative thinning algorithms delete successive layers on the edges of a pattern until a skeleton remains. Usually the pixels of an image are considered consecutively and a choice is made to either delete or keep the pixel. The criterion for this choice is a set of “rules” based on
the pixels in the neighborhood the current pixel. Usually the neighborhood is a 3x3 area around the pixel.

Iterative algorithms may be classified as either sequential or parallel. Sequential thinning algorithms [Davies & Plummer, 1981], [Hilditch, 1969] examine contour points of an object in a pre-determined order. Either raster scan or contour following algorithm accomplishes this. Raster scanning is essentially scanning an image until a contour point is found and then applying a set of rules. Contour following algorithms pre-compute the border pixels of connected objects and can provide a speed increase since every pixel of the image does not have to be examined. In parallel thinning algorithms [Zhang & Wang, 1988], [Guo & Hall, 1989], [Hall, 1989], [Ben & Roland, 1992], [Han & La, 1997] the decision for individual pixel deletion is based on the results of the previous iteration. Like sequential algorithms, parallel thinning usually considers a 3*3 neighborhood around the current pixel. A set of rules for deletion are applied based on pixels in the neighborhood. Fully parallel algorithms have trouble maintaining connectedness, so they are often broken into sub-iterations where only a subset of the pixels is considered for deletion.
Non-iterative thinning methods are not based on examining individual pixels. Some popular non-pixel based methods include medial axis transforms, distance transforms, and determination of centerlines by line following method. In line following methods, midpoints of black spaces in the image are determined and then joined to form a skeleton. This method is fast in computing but tends to produce noisy skeletons. It has been conjectured that human beings naturally perform thinning in a manner similar to this. Another method of centerline determination is by following contours of objects. By simultaneously following contours on either side of the object, a continual centerline can be computed. The skeleton of the image is formed from these connected centerlines. Medial axis transforms often use gray-level images where pixel intensity represents distance to the boundary of the object. The pixel intensities are calculated using distance transforms. Over the years, many thinning algorithms have been proposed and comprehensive survey of this method is contained in [Louisa et al., 1992]. Many thinning techniques [Ben & Roland, 1992], [Lei Huang et al., 2003], [Maher & Ward, 2002], [Stentiford & Mortimer, 1983], [Bunke & Wang, 1996] reported have obtained fairly good results. But there are still performance limitations in the context of different scripts as well as applications.

Based on the review of related works, we have chosen the following algorithms in our study: Parallel Thinning [Lei Huang et al., 2003], Rotation Invariant Thinning [Ahmed & Ward, 2002], Zhang Suen Thinning [Bunke & Wang, 1996], NWG Thinning [Nagendraprasad et al., 1993], Parker Thinning [parker, 1997] and Stentiford Thinning [Stentiford & Mortimer, 1983]. The implementation details of the above algorithms are given below.
4.3.1. Implementation Details of the Thinning Algorithms

4.3.1.1. An Improved Parallel Thinning Algorithm

The improved parallel thinning algorithm [Lei Huang et al., 2003] is fast and efficient for handwritten character recognition. Thinning is basically a search and deletes process that removes only those boundary pixels whose deletion do not change connectivity of their neighbors locally and do not reduce the length of an already thinned curve. The deleting process take place based on some elimination rules. Elimination rule is the kernel of a thinning algorithm which decides the performance of the thinning algorithm. All elimination rules are shown in Figure 4.4. These rules were classified according to the amount of black pixels in the eight neighbors. The first column denotes the number of black pixel in eight neighbors of image foreground pixel and the second column shows the elimination rules. All these rules are applied simultaneously to each pixel. In some peculiar cases these cannot be performed very well. The two pixel width or even pixel width in horizontal or vertical direction may be deleted, which would cause the loss of connectivity of object pattern. For example, a two pixel width rectangular pattern will disappear completely. In order to keep up the connectivity, these pixels should not be deleted. So some new rules are used to solve the above problem. These rules are shown in Figure 4.5.
### Table 4.4: Elimination Rules

<table>
<thead>
<tr>
<th>Amount</th>
<th>Elimination Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Never</td>
</tr>
<tr>
<td>1</td>
<td>Never</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Never</td>
</tr>
</tbody>
</table>

*Figure 4.4 Elimination rules [Lei Huang et al., 2003]*
The steps in parallel thinning algorithm [Lei Huang et al., 2003]:

1) Create the lookup table.
2) The index number is calculated for each object pixel.
3) Use the index number and lookup table to decide if the pixel is eliminable. If the pixel is not eliminable, go to step 5.
4) Get the width of object pixel. If it is not two-pixel width, delete it, otherwise, use the $3 \times 4$, $4 \times 3$ and $4 \times 4$ templates to decide if the pixel should be preserved. If the object pixel need not be preserved, delete it, or else preserve it.
5) Repeat 2-4 until no pixel can be eliminated.

A drawback of this algorithm is that a large black block whose length and width are both greater than 1, is usually changed to a line or a dot [Lei Huang et al., 2003]. In the field of handwritten character recognition this is not acceptable in some cases. In handwritten Tamil character recognition system, there are a lot of samples such as ஓ/uni0B9F_uni0BC0, எ/uni0BD1 Uni0BD3/Uni0BD5 which contains the black block. Figure 4.6 shows result of improved parallel thinning algorithm applied on a character எ. As seen, the thinning results in extra lines or loss of information.

4.3.1.2. Rotation Invariant Rule Based Algorithm

This algorithm proposed by [Ahmed & Ward, 2002] is used to thin the character patterns to their central line. This algorithm proceeds by deriving a set of 20 rules over the eight neighbors of the pixel, which is a candidate for
deletion. The 20 rules are shown in Figure 4.7. It is iterative in nature. In each iteration, it deletes every point that lies on the outer boundaries of the character pattern, as long as the width of the pattern is more than one pixel wide. Which are applied simultaneously at every iteration to each pixel. The iterations are repeated until no further changes occur. If the resultant pattern, at some point, has width (measured in any direction) equal to one pixel then this pixel belongs to the central line of the symbol and will not be deleted. If the width at any point is two pixels, then the central line passes between these two pixels. This case is separately treated in the algorithm. The implementation steps of this algorithm are discussed below:

Consider a binary image of an isolated handwritten character. It can be considered as a graph of black and white pixels. The white pixels represent the background and are denoted by 0’s. The black pixels represent the character curve and are denoted by 1’s. Every pixel x has eight neighbors with a 3 ×3 pixel neighborhoods as shown in Figure 4.8. The pixel x is deleted if any one of the twenty conditions listed below is satisfied. After each iteration, the parallel application of the 20 rules (listed in Figure 4.7) to each pixel in the image can be easily shown to result in peeling off the outer and the inner boundaries of character curve.

Figure 4.6 Result of improved parallel thinning algorithm
1. \( x_3 = x_4 = x_5 = x_6 = 1 \) and \( x_1 = x_8 = 0 \)
2. \( x_4 = x_5 = x_6 = x_7 = 1 \) and \( x_1 = x_2 = 0 \)
3. \( x_1 = x_6 = x_7 = x_6 = 1 \) and \( x_3 = x_4 = 0 \)
4. \( x_5 = x_6 = x_7 = x_8 = 1 \) and \( x_2 = x_3 = 0 \)
5. \( x_1 = x_2 = x_3 = x_8 = 1 \) and \( x_5 = x_6 = 0 \)
6. \( x_1 = x_2 = x_7 = x_8 = 1 \) and \( x_4 = x_5 = 0 \)
7. \( x_1 = x_2 = x_3 = x_4 = 1 \) and \( x_6 = x_7 = 0 \)
8. \( x_2 = x_3 = x_4 = x_5 = 1 \) and \( x_7 = x_8 = 0 \)
9. \( x_5 = x_6 = 1 \) and \( x_1 = x_2 = x_3 = x_8 = 0 \)
10. \( x_6 = x_7 = 1 \) and \( x_1 = x_2 = x_3 = x_4 = 0 \)
11. \( x_4 = x_5 = 1 \) and \( x_1 = x_2 = x_7 = x_8 = 0 \)
12. \( x_3 = x_4 = 1 \) and \( x_1 = x_6 = x_7 = x_8 = 0 \)
13. \( x_7 = x_8 = 1 \) and \( x_2 = x_3 = x_4 = x_5 = 0 \)
14. \( x_1 = x_8 = 1 \) and \( x_3 = x_4 = x_5 = x_6 = 0 \)
15. \( x_2 = x_3 = 1 \) and \( x_5 = x_6 = x_7 = x_8 = 0 \)
16. \( x_1 = x_2 = 1 \) and \( x_4 = x_5 = x_6 = x_7 = 0 \)
17. \( x_2 = x_3 = x_4 = x_5 = x_6 = x_7 = x_8 = 1 \) and \( x_1 = 0 \)
18. \( x_1 = x_2 = x_4 = x_5 = x_6 = x_7 = x_8 = 1 \) and \( x_3 = 0 \)
19. \( x_1 = x_2 = x_3 = x_4 = x_6 = x_7 = x_8 = 1 \) and \( x_5 = 0 \)
20. \( x_1 = x_2 = x_3 = x_4 = x_5 = x_6 = x_8 = 1 \) and \( x_7 = 0 \)

**Figure 4.7** 20 rules [Ahmed & Ward, 2002]

<table>
<thead>
<tr>
<th>( X_6 )</th>
<th>( X_7 )</th>
<th>( X_8 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_5 )</td>
<td>( X )</td>
<td>( X_1 )</td>
</tr>
<tr>
<td>( X_4 )</td>
<td>( X_3 )</td>
<td>( X_2 )</td>
</tr>
</tbody>
</table>

**Figure 4.8** Pixel \( x \) and its eight neighbors
Pre-processing

Drawback of the above procedure is that, when a part of a pattern is two pixels wide in the horizontal or vertical direction, these two pixels may be deleted and results in disconnected central lines. In order to solve this problem, in each iteration we must first check whether the width of the pattern is two pixels or not. This can be achieved by introducing certain modification of the algorithm [Ahmed & Ward, 2002]. The following section describes the implementation details of the modified rule based thinning algorithm.

Repeat the following steps until no changes occur from one iteration to the next.

For iteration i, for every pixel w in the document do the following:

**Step 1:** If w belongs to two pixels wide in the vertical direction, go to step 2.

If w belongs to two pixels wide in the horizontal direction, go to step 3.

Otherwise go to step 6.

**Step 2:** If w belongs to

```
  x 0 x
  1 w 1
  1 1 1
  x 0 x
```

Stop calculations for this pixel w,

Else check if w belongs to

```
  x 0 x
  1 1 1
  1 1 1
  x 0 x
```

If yes, delete w and stop calculations for this pixel.

Otherwise, go to step 4.
Step 3: If \( w \) belongs to

\[
\begin{array}{ccc}
\times & 1 & 1 \\
0 & w & 1 \\
\times & 1 & 1
\end{array}
\]

Stop calculations for this pixel \( w \),

Else check if \( w \) belongs to

\[
\begin{array}{ccc}
\times & 1 & 1 \\
0 & 1 & w \\
\times & 1 & 1
\end{array}
\]

If yes, delete \( w \) and stop calculations for this pixel,

Otherwise, go to step 5.

Step 4: If \( w \) belongs to

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & w & 0 \\
0 & 1 & 1 \\
0 & 0 & 1
\end{array}
\quad \text{or}\quad
\begin{array}{ccc}
0 & 0 & 0 \\
0 & w & 0 \\
1 & 1 & 0 \\
1 & 0 & 0
\end{array}
\]

Stop calculations for this pixel

Else check if \( w \) belongs to

\[
\begin{array}{ccc}
1 & 0 & 0 \\
1 & 1 & 0 \\
0 & w & 0 \\
0 & 0 & 0
\end{array}
\quad \text{or}\quad
\begin{array}{ccc}
0 & 0 & 1 \\
0 & 1 & 1 \\
0 & w & 0 \\
0 & 0 & 0
\end{array}
\]

If yes, stop calculations for this pixel \( w \)

Otherwise go to step 6.

Step 5: If \( w \) belongs to

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & w & 1 \\
0 & 0 & 1
\end{array}
\quad \text{or}\quad
\begin{array}{ccc}
0 & 0 & 1 \\
0 & w & 1 \\
0 & 0 & 0
\end{array}
\]
Stop calculations for this pixel

Otherwise, check if w belongs to

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & w \\
1 & 1 & 0
\end{array}
\quad \text{or} \quad
\begin{array}{ccc}
1 & 1 & 0 \\
0 & 1 & w \\
0 & 0 & 0
\end{array}
\]

If yes, stop calculations for this pixel

Otherwise go to (6).

If yes, stop calculations for this pixel w

Otherwise go to step 6.

Step 6: Apply the 20 rules and stop calculations for this pixel w.

By applying the above modified procedure in handwritten character, the characters are thinned to their central lines without extraneous branches. The resultant central lines are connected and one pixel width. In some cases, the algorithm may result in extra branches. A white pixel inside a black boundary will turn to an extra hole after thinning as shown in Figure 4.9.

![Figure 4.9 Results of rotation invariant algorithm applied on character ஐ](image)

4.3.1.3. Nagendra Prasad-Wang-Gupta (NWG) Thinning Algorithm

Nagendra prasad et al proposed an algorithm which is simpler and produced more elegant skeletons of handwritten characters at a reduced
computational cost [Nagendraprasad et al., 1993]. The algorithm uses masks in order to select pixels to be turned off. The eight closest neighbors are numbered following a clockwise walk around the pixel p, which starts at the upper edge as shown in Figure 4.10.

![Figure 4.10 Numbering of neighboring pixels](image)

The NWG algorithm is given below [Nagendraprasad et al., 1993]

Start
\[ g = 1; \ h = 1; \ Q_0 = Q; \text{ where } Q \text{ is an input image} \]

Do While (\( h = 1 \))
\[ h = 0; \ Q = Q_0; \]
If (\( g = 1 \)) then \( g = 0; \)
\[ \text{Else } g = 1; \]
End if

For (every pixel \( p \in Q \))
If (\( 1 < b(p) < 7 \) and \( a(p) = 1 \) or \( c(p) = 1 \)) then
\[ \text{If } (g = 0 \text{ and } e(p) = 0) \text{ then} \]
\[ \text{Erase } p \text{ in } Q_0; \ h = 1; \]
End if
End if
End for
End do
EndNWG

Here \( b(p) \) is the number of neighbors of \( p \) which are on (pixels with value 1), \( a(p) \) is the number of off-to-on transitions when the neighbors are
visited following a clockwise walk around p in the order of p(0), p(1),.....p(7). p(0) and the functions c(p), e(p) and f(p) are given by:

\[
c(p) = \begin{cases} 
1 & \text{if } p(0) = p(1) = p(2) = p(5) = 0 \text{ and } p(4) = p(6) = 1 \\
1 & \text{if } p(2) = p(3) = p(4) = p(7) = 0 \text{ and } p(6) = p(0) = 1 \\
0 & \text{otherwise}
\end{cases}
\]

\[
e(p) = (p(2) + p(4)) * p(0) * p(6)
\]

\[
f(p) = (p(6) + p(0)) * p(4) * p(2)
\]

The skeleton obtained is connected and has redundant pixels eliminated in order to recognize the handwritten character. However, due to an asymmetry in the algorithm, some superfluous pixels are not removed. This may be corrected by simply changing the function c(p) in those iterations where g = 1 (this means odd iterations). That is, the condition

\[
a(p) = 1 \text{ or } c(p) = 1
\]

is replaced by

\[
a(p) = 1 \text{ or } (1 - g) * c(p) + g * d(p) = 1
\]

with

\[
d(p) = \begin{cases} 
1 & \text{if } p(1) = p(4) = p(5) = p(6) = 0 \text{ and } p(2) = 1 \\
1 & \text{if } p(0) = p(3) = p(6) = p(7) = 0 \text{ and } p(2) = p(4) = 1 \\
0 & \text{otherwise}
\end{cases}
\]

The result of NWG thinning algorithm usually produce more elegant skeleton in the sense that some redundant or confusing pixels are removed. It may result in loss of information in some cases as shown in Figure 4.11.
4.3.1.4. Zhang and Suen Thinning Algorithm

Zhang-Suen thinning algorithm [Zhang & Suen, 1988] has been widely used as it is fast and simple to implement. It is a parallel method, meaning that the new value for any pixel can be computed only using the values known from the previous iteration. This algorithm repeatedly deletes contour pixels satisfying a number of conditions until a one-pixel wide 8-connected skeleton is obtained. In this algorithm, a 3 x 3 window is used and each element is connected to its eight neighboring elements. The pixels in the window are labelled as in Figure 4.12.

<table>
<thead>
<tr>
<th>P9</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>P8</td>
<td>P1</td>
<td>P4</td>
</tr>
<tr>
<td>P7</td>
<td>P6</td>
<td>P5</td>
</tr>
</tbody>
</table>

*Figure 4.12 Designations of the nine pixels in a 3x3 window*

Let NT (P1) be the number of zero (white) to non-zero (black) transitions in the ordered sequence P2, P3…P9, P2, and NZ (P1) be the number of nonzero neighbours of P1. Pixel P1 is deleted (set to zero) if it stratifies the following conditions:
i. \( 2 \leq NZ (P1) \leq 6 \)

ii. \( NT (P1) = 1 \) and \( P2.P4.P8 = 0 \) or

iii. \( NT (P2) \neq 1 \) and \( P2.P4.P6 = 0 \) or

iv. \( NT (P4) \neq 1 \)

The process is repeated until there are no more changes in the image.

Although Zhang-Suen thinning algorithm has advantages of implementation speed and topological preservation, it still have other weaknesses. For example this algorithm deletes all 2X2 pixel image and some of the skeletons contain a defects such as necking and line fuzz as shown in figure 4.13. Therefore, parker [Parker, 1997] has introduced an improved thinning algorithm to address the problems posed by Zhang Suen thinning algorithm.

![Original image and Thinned images](image)

**Figure 4.13** Results of Zhang Suen thinning algorithm applied on character image அ

### 4.3.1.5. Parker Thinning Algorithm

Parker [Parker, 1997] has introduced a hybrid thinning algorithm in order to produce a good quality skeleton. This algorithm merges three methods in a sequence order; Stentiford’s pre-processing [Stentiford & Mortimer, 1983] scheme feeding images into Zhang-Suen’s basic algorithm [Zhang & Wang, 1988], followed with Holt’s staircase removal [Holt et al., 1987] as a post-
The implementation details of the parker thinning algorithm are described below:

i) **Pre-processing Stage:** The stentiford’s pre-processing [Stentiford & Mortimer, 1983] is used to reduce the failure rate due to the defects in the data. The classic thinning artefacts are necking, tailing and spurious projection, hairs or line fuzz. Necking happens when a narrow point at the intersection of two lines is stretched into a small line segment. Spurious projection refers to the creation of extra line segments joining a real skeletal segment.

   Basically, smoothing is done by making a pass over all pixels, deleting those having two or fewer black neighbours and having connectivity less than two. A procedure called acute angle emphasis is used to deal with necking [Adeline, 2005]. This procedure involves the detection of upward and downward acute angles between limbs by scanning the character with pattern of bits. This is done using the templates ($D_i$ and $U_i$) as shown in Figure 4.14. If there is any match with templates, the central pixel is marked for deletion, and then causes another iteration of less severe acute angle emphasis using only the first three templates of $D_i$ and $U_i$. If any pixels are deleted, one last pass using only the first templates of $D_i$ and $U_i$ is performed.

ii) **Staircase Removal:** Sometimes, when thinning is complete, there are still pixels that could be deleted. Principal among these are pixels that form a staircase; clearly half of the pixels in a staircase could be removed without affecting the shape or connectedness of the overall pattern [Holt et al., 1987]. Figure 4.15 shows the examples of pixels that form a staircase in each windows and the central pixel in one of windows can be deleted. The 1 values refers to the object pixels, 0 values refers to the background pixels and the x values refers to the ‘don’t care’ values, meaning, it can be either 0 or 1 values.
Figure 4.14 Templates used for the acute angle emphasis in pre-processing step [Adeline, 2005].

In order to avoid creating new hole after deleting, a condition is added, that is one of the x values is changed to 0. For windows having a northward bias (Figure 4.15 (a) and (b)) the logical expression for survival of a pixel in the staircase-removal iteration is

\[ v(C) \land \neg(v(N) \land (v(E) \land \neg v(NE) \land \neg v(NW) \land \neg v(SE) \land (v(W) \land \neg v(S)))) \]

where \( \neg \) denotes Not, and the \( v \) function gives the value of the pixel at a location (1 = true for an object pixel, 0 = false for background). The letters E, S, N, W, NE, SW, NW and SE represent pixels in a particular direction from the centre pixel C; E means east, S means south and so on as shown in Figure 4.16.
The pass having a southward bias is the same, but with north and south exchanged. The logical expression represents the condition under which the centre pixel C survives the iteration.

<table>
<thead>
<tr>
<th>NW</th>
<th>N</th>
<th>NE</th>
</tr>
</thead>
<tbody>
<tr>
<td>(north west)</td>
<td>(north)</td>
<td>(north east)</td>
</tr>
<tr>
<td>W</td>
<td>C</td>
<td>E</td>
</tr>
<tr>
<td>(west)</td>
<td>(centre)</td>
<td>(east)</td>
</tr>
<tr>
<td>SW</td>
<td>S</td>
<td>SE</td>
</tr>
<tr>
<td>(south west)</td>
<td>(south)</td>
<td>(south east)</td>
</tr>
</tbody>
</table>

Figure 4.16 Pixel locations in a 3x3 window [Adeline, 2005].

The result of Parker thinning algorithm usually produce more elegant skeleton. But some of the skeletons contain defects as shown in figure 4.17.

![Original Image](image1.png) ![Thinned Images](image2.png)

Figure 4.17 Results of parker thinning algorithm

4.3.1.6. Stentiford Thinning Algorithm

Stentiford thinning algorithm [Stentiford & Mortimer, 1983] uses some pre-processing stages before a Zhang Suen thinning algorithm is applied. These pre-processing heuristics are specifically aimed at reducing the failure rate due
to the defects in the data. The pre-processing steps such as hole removal, smoothing and acute angle emphasis are described below.

**i) Hole removal:** Hole removal is done using six patterns of bits or templates as shown in figure 4.18. These holes are removed by either merging the hole with adjacent white areas by the removal of black elements or filling it with additional black elements.

![Patterns of hole removal](image)

*Figure 4.18 Shows patterns of hole removal [Stentiford & Mortimer, 1983]*

**ii) Smoothing:** Smoothing is the removal of all black elements having less than three black neighbours and having connectivity 1. Stentiford thinning algorithm uses Yokoi’s 8 connectivity number [Yokoi et al., 1973]. This has the effect of removing single elements projections and all isolated spots having one or two elements.

**iii) Acute angle emphasis:** The final pre-processing stage involves the detection of upward and downward acute angles (Di and Ui) between limbs by again scanning the character with the patterns of bits [Adeline, 2005]. Five of these (Di) will fit the sharpest forms of downward pointing acute angles and
(U_i) will fit the sharpest forms of upward pointing acute angles. After a fit is found, the central black element is deleted and the process is repeated twice more. Stentiford and Mortimer described the summary of all preprocessing steps in [Stentiford & Mortimer, 1983].

After pre-processing Zhang Suen thinning algorithm is applied. Four matrices \(M_i\) (where \(i = 1, 2, 3\) and 4) in Figure 4.20 are scanned over the character and wherever a matrix fits the central black element is marked for deletion. Elements are not so marked if they are limb endpoints or if the 8-connectivity measure for that point is greater than one. An endpoint is defined as a black element which is eight connected to only one other black element. Elements already marked are considered to be white for the purpose of subsequent end point or connectivity calculations. When all four matrices have been scanned in this way, all marked elements are deleted and the process repeated until no more erosion can take place.

![Figure 4.19 Templates used for the acute angle emphasis.](image-url)
Pre-processing

![Figure 4.20 Thinning matrices [Stentiford & Mortimer, 1983]](image)

The results of above process are sensitive both to the order of application of the matrix $M_i$ and also the direction of scan. An ordering which minimize spurious tail production is given in Table 4.1. For example $M_3$, the south edge eroding matrix is scanned from right to left moving upwards across the character.

**Table 4.1 Matrix Ordering and Scan Direction**

<table>
<thead>
<tr>
<th>Order of Application</th>
<th>Direction of Single scan line</th>
<th>Direction of successive scan line</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>Left to Right</td>
<td>Downwards</td>
</tr>
<tr>
<td>$M_2$</td>
<td>Upwards</td>
<td>Left to Right</td>
</tr>
<tr>
<td>$M_3$</td>
<td>Right to Left</td>
<td>Upwards</td>
</tr>
<tr>
<td>$M_4$</td>
<td>Downwards</td>
<td>Right to Left</td>
</tr>
</tbody>
</table>

4.4. Modified Stentiford Thinning (MST) Algorithm

In this section, we propose a modification to the stentiford thinning algorithm. It also uses three steps such as pre-processing, thinning and post processing. The MST pre-processing steps are described given below:

4.4.1. Pre-processing

Hole removal is done using six patterns of bits or templates as shown in Figure 4.18. Smoothing is the removal of all black elements, having less than three black neighbours and connectivity 1. The Stentiford thinning algorithm
uses Yokoi’s eight connectivity number. But it does not preserve all the connectivity in a Tamil character image. So, the proposed MST thinning algorithm uses Zhang Suen eight connectivity number [Zhang & Suen, 1984]. A 3x3 template window is used to define the pre-processing step as shown in Figure 4.12, where P1 refers to the current pixel of interest and P2 to P9 are considered as neighbouring pixels. This pre-processing step is applied only to pixels with value 1. A decision value is computed as the number of times the neighbourhood pixel value change from 0-1, when the neighbours are visited in the order P2, P3, P4, P5, P6, P7, P8, P9 and P2. If this decision value is 1 then the pixel under consideration is set to zero. For example, if the neighbourhood is visited in order P2, P3, P4, P5, P6, P7, P8, P9 and P2 results in a vector 0010000010, then the decision value is 2. This has the effect of removing single elements projections and all isolated spots having one or two elements. This step is followed by the detection of upward and downward acute angles as described in the case of stentiford thinning algorithm (Section 4.3.1.6).

4.4.2. Thinning

After pre-processing, Zhang Suen thinning algorithm is applied as described in Section 4.3.1.4.

4.4.3. Post processing

In post processing, extra lines in thinned images are removed using endpoint removal matrices (Figure 4.21). The endpoint removal matrices (Mi, Bi, Ci) are scanned over the character and wherever a matrix fits, the central black pixel is marked for deletion. In stentiford thinning algorithm, only four matrices Mi are used to delete a central black pixel. But the proposed MST algorithm, we used two new matrices Bi and Ci in addition to Mi where i= 1 2 3 and 4. Because Mi scans matrices remove only left, right upward and downward extra lines. In order to remove slanting extra lines, the MST algorithm is used
Bi and Ci scan matrices as shown in Figure 4.21. All matrices are scanned over the character and wherever a matrix fits, the central black pixel is marked for deletion. When all four matrices of Mi, Bi and Ci have been scanned in this way, all marked pixels are deleted and the process is repeated until no more erosion can take place. The pixel deletion processes are sensitive to both the order of application of the scan matrices (Mi, Bi, Ci) and the direction of scan (given in Table 4.2).

The post processing in the proposed MST algorithm can be summarized as follows:

Step1. Scan matrix M1 across character according to table 4.2 and identify next fit position.

Step2. If the central element at a fit is not an endpoint and has connectivity value one, and then mark it for deletion.

Step3. Repeat Steps 1 and 2 for all fit positions.

Step4. Repeat Step 1-3 for each scan of matrix M2, M3, M4.

Step5. Delete all marked elements.

Step6. If one or more elements are deleted in Step 4 then return to Step1.

Step7. Repeat Step 1-6 for scan matrices Bi and Ci.

Step8. Exit.

Table 4.2 Matrix ordering and scanning directions of modified Thinning algorithm

<table>
<thead>
<tr>
<th>Order of Application</th>
<th>Direction of Single scan line</th>
<th>Direction of successive scan line</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1, B1, C1</td>
<td>Left to Right</td>
<td>Downwards</td>
</tr>
<tr>
<td>M2, B2, C2</td>
<td>Upwards</td>
<td>Left to Right</td>
</tr>
<tr>
<td>M3, B3, C3</td>
<td>Right to Left</td>
<td>Upwards</td>
</tr>
<tr>
<td>M4, B4, C4</td>
<td>Downwards</td>
<td>Right to Left</td>
</tr>
</tbody>
</table>
The Figure 4.22 illustrates the comparison of stentiford thinning and modified stentiford thinning algorithms. It is clear that the modified stentiford algorithm thinning algorithm removes all spurious tails but stentiford thinning algorithm removes only vertical and horizontal lines.
4.5. Analysis and Comparison of MST Algorithms

For evaluating the quality of thinning algorithms we consider connectivity preservation, the width of skeleton, shape of the character, classification performance and execution time. The seven thinning algorithms described above are tagged as Parallel Thinning (PT), Rotation Invariant Thinning (RIT), Zhang Suen Thinning (ZST), Nagendra prasad-Wang-Gupta Thinning (NWGT), Parker Thinning (PKT), Stentiford thinning (ST) and Modified Stentiford Thinning (MST). Table 4.3 shows the result of different thinning algorithms applied on five handwritten Tamil character images. Table 4.4 shows the execution time of different thinning algorithms applied on five handwritten Tamil character images. Figure 4.23 shows the graphical representation of the execution time when applied to five handwritten Tamil characters.

Table 4.3 Different thinning algorithms applied on five handwritten Tamil characters

<table>
<thead>
<tr>
<th>Methods \ Images</th>
<th>Sample 1</th>
<th>Sample 2</th>
<th>Sample 3</th>
<th>Sample 4</th>
<th>Sample 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Images</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
</tr>
<tr>
<td>PT</td>
<td><img src="image6.png" alt="Image" /></td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
</tr>
<tr>
<td>RIT</td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
<td><img src="image15.png" alt="Image" /></td>
</tr>
<tr>
<td>ZST</td>
<td><img src="image16.png" alt="Image" /></td>
<td><img src="image17.png" alt="Image" /></td>
<td><img src="image18.png" alt="Image" /></td>
<td><img src="image19.png" alt="Image" /></td>
<td><img src="image20.png" alt="Image" /></td>
</tr>
<tr>
<td>Methods</td>
<td>Sample 1</td>
<td>Sample 2</td>
<td>Sample 3</td>
<td>Sample 4</td>
<td>Sample 5</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>NWGT</td>
<td><img src="image1" alt="" /></td>
<td><img src="image2" alt="" /></td>
<td><img src="image3" alt="" /></td>
<td><img src="image4" alt="" /></td>
<td><img src="image5" alt="" /></td>
</tr>
<tr>
<td>PKT</td>
<td><img src="image6" alt="" /></td>
<td><img src="image7" alt="" /></td>
<td><img src="image8" alt="" /></td>
<td><img src="image9" alt="" /></td>
<td><img src="image10" alt="" /></td>
</tr>
<tr>
<td>ST</td>
<td><img src="image11" alt="" /></td>
<td><img src="image12" alt="" /></td>
<td><img src="image13" alt="" /></td>
<td><img src="image14" alt="" /></td>
<td><img src="image15" alt="" /></td>
</tr>
<tr>
<td>MST</td>
<td><img src="image16" alt="" /></td>
<td><img src="image17" alt="" /></td>
<td><img src="image18" alt="" /></td>
<td><img src="image19" alt="" /></td>
<td><img src="image20" alt="" /></td>
</tr>
</tbody>
</table>

**Table 4.4** Execution time of different thinning algorithms applied on five handwritten Tamil characters

<table>
<thead>
<tr>
<th>Thinning methods</th>
<th>Sample1</th>
<th>Sample2</th>
<th>Sample3</th>
<th>Sample4</th>
<th>Sample5</th>
</tr>
</thead>
<tbody>
<tr>
<td>PT</td>
<td>9.109</td>
<td>5.969</td>
<td>11.766</td>
<td>5.875</td>
<td>7.641</td>
</tr>
<tr>
<td>ZST</td>
<td>9.422</td>
<td>7.188</td>
<td>13.328</td>
<td>6.875</td>
<td>9.218</td>
</tr>
<tr>
<td>NWGT</td>
<td>2.5</td>
<td>2.156</td>
<td>1.641</td>
<td>1.454</td>
<td>2.719</td>
</tr>
<tr>
<td>ST</td>
<td>10.24</td>
<td>7.852</td>
<td>14.119</td>
<td>7.453</td>
<td>8.918</td>
</tr>
<tr>
<td>MST</td>
<td>10.28</td>
<td>7.902</td>
<td>14.12</td>
<td>7.543</td>
<td>8.924</td>
</tr>
</tbody>
</table>
The performance comparison of different thinning algorithms is described in Section 7.4. The proposed modified stentiford thinning algorithm gives better output and high recognition accuracy (90.9%) than the other six thinning algorithms. The execution time of the proposed algorithm is slightly greater than the other methods except NWGT. But the output of NWGT is relatively inferior to MST.

![Figure 4.23 Execution time of five handwritten Tamil characters.](image)

### 4.6. Contour Tracing

Contour extraction is important in many scientific fields such as digital image processing, computer vision pattern recognition etc. Boundary contour, a meaningful external representation of area of interest, is widely used in many applications such as parametric curve fitting and Hough’s transformations [Pratt, 1991]. We can represent the region of an image in terms of its external characteristics (its boundary) or an internal characteristic (the pixels comprising the region) [Gonzalez & Woods, 2002]. An external representation is selected when the interest is on shape characteristics. An internal representation is selected when the principal focus is on regional properties such as color and texture.
4.6.1. Literature Review

The contour, skeleton and the Medial Axis Transform (MAT) [Choi et al., 2003] were introduced to describe the global properties of objects and to reduce the original image to a more compact highly structured representation. In computer vision, this kind of global representation [Grkgoire & Sara, 1998] has become increasingly more useful in object identification and recognition. Canny [Canny, 1986] have described a procedure for the design of edge detectors for arbitrary edge profiles. The design was based on the specification of detection and localization criteria in a mathematical form. It was necessary to augment the original two criteria with a multiple response measure in order to fully capture the intuition of good detection. A detector was proposed which used adaptive threshold with hysteresis to eliminate streaking of edge contours. The thresholds were set according to the amount of noise in the image, as determined by a noise estimation scheme. Canny edge detection algorithm is one of the best edge detection algorithms of the present era, and it has also become a standard in edge detection.

The most common contour tracing algorithms includes the Moore–neighbor tracing algorithm [Ghuneim, 2009], the radial sweep algorithm [Mirante & Weingarten, 1982], square tracing algorithm [Mohamed et al., 2007] and Theo Pavlidi’s algorithm [Pavlidis, 1982] and all of them uses three steps for contour tracing. The steps are

a) Searching for a starting pixel on a contour,

b) Examining its neighborhoods (how to find the next boundary point)

c) Selecting one of its neighborhoods as a next pixel on contour.

The last two steps are repeated until encountering the starting pixel. Bug following is another basic algorithm, but rather than considering neighborhood area, it makes decision only based on a sense of direction whether to make a left
or right turn. These algorithms have number of limitations which cause them to fail in tracing the contour of a large classes of patterns due to their special kind of connectivity.

Moore’s contour tracing algorithm [Grkgoire & Sara, 1998] uses the binary image, the pixel value of the binary image represented as 0 and 1 where 0 represents the background pixels or black pixels and 1 represents the foreground pixels or white pixels. The four neighbor’s adjacent method is adopted in this work. The boundary extracting algorithm [Canny, 1986] checks whether the foreground pixels is a contour pixel or not. If it is a contour pixel then it is retained otherwise it is replaced by the background pixel. [Schlei, 2009] introduced a new framework for one-dimensional contour extraction from discrete two dimensional data sets. A closed chain representing a contour can be obtained by using active contours that have the capability of morphological adaptation. In the component labeling concept [Chang et al., 2004], all contours are assigned with unique labels. The contour search starts with raster scan from the top left corner of an image, line by line. Once the first point on a contour is found, a label is assigned and contour tracing is activated to search on one closed contour. The following section describes a novel contour tracing algorithm using maximal disk and eight sequential Euclidean distance map and connectivity criteria based on maximal disk.

4.6.2. A Novel Contour Tracing Algorithm

In this section a new fast, efficient and accurate contour extraction method using eight sequential Euclidean distance map and connectivity criteria based on maximal disk [Choi et al., 2003] is presented. The connectivity criterion is based on a set of point pairs along the image boundary pixels, which are the nearest point under consideration and its neighbours. Most of the contour algorithm is time consuming and computationally intensive. The
proposed algorithm generates a contour of an image with less number of iterations compared to existing methods. The performance of the proposed algorithm in terms of execution time, image shape and classification performance is compared with two existing contour tracing algorithms - the Moore method and the Canny edge detection method. The results established the efficacy of the proposed algorithm as it has less computing time, results in high recognition accuracy and never loses the connectivity. The connectivity criteria based on maximal disk and sequential Euclidean distance mapping are described below.

**Definition of Medial axis transforms**

The medial axis transform is the set of ordered pairs of the centers and radii of maximal disks in the planar shape $F$ [Choi et al., 2003]. That is:

$$\text{SK}(F) = \{ (p, r) \in F \mid B(p, r) \in \text{MaxDisk}(F) \}$$  \hspace{1cm} (4.1)

where $\text{MaxDisk}(F)$ is the set of all maximal disks in the planar shape $F$ and $B(p, r)$ is a disk with radius $r$ centered at the point $p$. It can be observed that almost all skeleton points are associated with at least two boundary points whose respective distances to the skeleton point are the shortest except the end points of the skeleton. These boundary points divide the contour into different separate segments. In Figure 4.24, with a maximal disk centered at $p$, the object’s contour and the maximal disk touch each other at $q_1$ and $q_2$. These two points divide the contour $C$ into two segments $A$ and $B$.

**Euclidean distance mapping**

In order to find the maximal disks enclosed in an object efficiently, a distance map should be generated before locating the centers of the maximal disks. Different algorithms based on different distance maps have been proposed [Butt & Maragos, 1998], [Saadia et al., 2010], [Cuisenaire & Macq,
1999(a) and (b)], [Hinnik, 1996], [Ching, 1992]. Approximate distance maps such as the city-block, chessboard, and Chamfer Distance Transform (CDT) [Butt & Maragos, 1998] can be used to extract the maximal disks.

![Medial Axis Transform (MAT)](image)

**Figure 4.24 Medial Axis Transform (MAT)**

Distance mapping is frequently used in image processing. Usually, it is based on one of the metrics

$$d_4((i, j)(h,k)) = |i-h| + |j-k|$$

which is called the ‘city block distance’ or

$$d_4((i, j)(h,k)) = |i-h| + |j-k|$$

which is called the ‘chessboard distance’ or a combination of these called octagonal distance. We assume that \(i, j, h, k\) are integers in the two dimensional rectangular space. Given a binary image with two sets of pixels

- \(S = \) the set of 1’s representing the objects
- \(\overline{S} = \) set of 0’s, the background
A distance map \( L(S) \) is an integer such that for each pixel \((i, j) \in S\) there is a corresponding pixel in \( L(S) \) where

\[
L(i, j) = \min d[(i, j), \bar{S}]
\]

i.e., each pixel in \( S \) has been assigned a label in \( L(S) \) that amounts to the distance to the nearest background \( \bar{S} \). Obviously we can define a similar map \( L(\bar{S}) \) for the background. The computational procedure

\[ L : S \rightarrow L(S) \]

is called distance mapping. Distance mapping can be used for several purposes [Dattorro, 2005]. Two Sequential Euclidean Distance (SED) mapping algorithms viz four-point Sequential Euclidean Distance (4SED) and eight-point Sequential Euclidean Distance (8SED) have been proposed for the efficient computation of the Euclidean distance map. 4SED and 8SED algorithms are described in [Danielsson, 1980].

4.6.2.1. Proposed Framework for Contour Tracing Algorithm

The key component of this algorithm is to apply 8SED mapping and connectivity criteria to achieve the correct contour point of an image. The proposed algorithm is illustrated in Figure 4.25. This algorithm assumes the value of background points as 0 and is represented by black color, the value of object point as 1 and is represented by white color. To find a contour of an image one needs to traverse the entire image. First we select a background pixel at random and set its value to 1 (set it as foreground pixel, say P). This pixel can be far from the foreground pixel. Next we compute the 8SED mapping and set \( Q = DM(P) \). In Figure 4.24 the eight-point sequential Euclidean distance (8SED) mapping of an image is shown.
Start

INPUT: Original image

Convert original image to binary image (1 for foreground and 0 for background)

Set any background pixel of an image to a foreground pixel, \( P \), this is far from the foreground pixel.

Compute 8SED map and set \( Q = DM(P) \)

Set \( R \) to be the first foreground pixel

Find the 8-neighborhood of \( R \), \( N_R(i) \), i=1 to 8, and compute \( Q_i = DM(N_R(i)) \) + \( P(N_R(i)) \)

Apply the connectivity criteria for \( (Q, Q_i) \):
\[
D^2 = |Q_i - Q|^2 \geq \delta \quad \text{and} \quad |Q_i|^2 - |Q|^2 \leq \max(X(Q_i - Q), Y(Q_i - Q))
\]

Check any one of the point pairs \( (Q, Q_i) \), satisfies the connectivity criterion

The pixel \( R \) is a contour pixel

Set \( R \) as the next foreground pixel

Until the last foreground pixel of the image is checked

OUTPUT: Contour of an image

End

Figure 4.25 Flowchart of the new contour tracing algorithm
To determine whether a pixel point is contour point, the corresponding point for each of the eight neighboring points are determined, and the connectivity criterion is then applied to this set of eight point pairs. The relative positions of the nearest points for each of the pixel can be obtained by using 8SED map. The eight neighboring points of the point under consideration can be obtained by adding relative position of its nearest point to the relative position of its neighborhoods.

**Figure 4.26** Eight-point Sequential Euclidean Distance (8SED) mapping of an image.
The eight neighboring points of the pixel under consideration form an eight point pair. If any one of the point pairs satisfies the connectivity criterion [Choi et al., 2003], the pixel can be declared as a contour point. Otherwise, if all the eight point pair fails to fulfill the connectivity criterion, the pixel is not a contour point.

The procedure of our proposed algorithm is described as follows:

Set any background pixel of an image to a foreground pixel P, this is far from the original image pixel or foreground pixel.

Compute eight point sequential Euclidean distance map and set \( Q = DM(P) \) where DM(.) represents the 8SED map.

Set R to be the first foreground pixel.

Do

Find the 8-neighborhood of R, \( N_R(i), i = 1 \) to 8, and

Compute \( Q_1 = DM(N_R(i) + P(N_R(i))) \), where \( P(N_R(i)) \) is the relative position of the \( i^{th} \) neighborhood with respect to the pixel R.

The eight point pairs are formed by the pixel P and 8 neighbors of the pixel R, which are denoted as \( (Q, Q_1) \)

Apply the connectivity criteria for \( (Q, Q_1) \):

\[
D^2 - |Q_1 - Q|^2 \geq \delta \quad \text{and} \quad |Q_1|^2 - |Q|^2 \leq \max(X(Q_1 - Q), Y(Q_1 - Q))
\]

where \( X(.) \) and \( Y(.) \) represents the x and y coordinates of the pixel.

If one of the points \( (Q, Q_1) \), satisfies the connectivity criterion, then the pixel R is a contour point.

until the last foreground pixel of an image is checked.
If we start with foreground pixel then the above algorithm results in discontinuity in the contour near the starting pixel. This is the reason for the selection of a background pixel as a starting point.

4.6.2.2. Result and Discussion

The proposed contour extraction algorithm is implemented with MATLAB. We carried out experiments with different threshold values $\delta$. When the value of the threshold $\delta$ increases that is $\delta \geq 6$, the algorithm plots contour with discontinuity (Figure 4.27(c)). This is because many contour points do not satisfy the connectivity criteria with $\delta \geq 6$. When the value of the threshold is $\delta \leq 2$, the algorithm plots all foreground pixel in images as contour points (Figure 4.27(a)) that is all foreground pixels now satisfies the connectivity criteria. Hence in this work $\delta$ is taken as 3.

![Figure 4.27](image.png)

**Figure 4.27** The contour using different threshold $\delta$ with an image

Table 4.4 shows the output of different contour tracing algorithms applied on five Tamil handwritten characters. From the Table 4.5 it is clear that the proposed method never loses the connectivity. A comparison of execution time is given in Table 4.6. As it is evident the time taken by the proposed method is considerably lesser when compared to the other two methods. Figure 4.28 shows the execution time of three contour tracing methods applied on five handwritten Tamil characters.
Table 4.5 Results of different contour tracing algorithm applied on Tamil handwritten characters

<table>
<thead>
<tr>
<th>Samples</th>
<th>Ch1</th>
<th>Ch2</th>
<th>Ch3</th>
<th>Ch4</th>
<th>Ch5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original image/method</td>
<td><img src="image1" alt="image" /></td>
<td><img src="image2" alt="image" /></td>
<td><img src="image3" alt="image" /></td>
<td><img src="image4" alt="image" /></td>
<td><img src="image5" alt="image" /></td>
</tr>
<tr>
<td>Moore</td>
<td><img src="image6" alt="image" /></td>
<td><img src="image7" alt="image" /></td>
<td><img src="image8" alt="image" /></td>
<td><img src="image9" alt="image" /></td>
<td><img src="image10" alt="image" /></td>
</tr>
<tr>
<td>Canny</td>
<td><img src="image11" alt="image" /></td>
<td><img src="image12" alt="image" /></td>
<td><img src="image13" alt="image" /></td>
<td><img src="image14" alt="image" /></td>
<td><img src="image15" alt="image" /></td>
</tr>
<tr>
<td>Proposed</td>
<td><img src="image16" alt="image" /></td>
<td><img src="image17" alt="image" /></td>
<td><img src="image18" alt="image" /></td>
<td><img src="image19" alt="image" /></td>
<td><img src="image20" alt="image" /></td>
</tr>
</tbody>
</table>

Table 4.6 Execution time of the selected contour tracing algorithm (in seconds)

<table>
<thead>
<tr>
<th>Methods/samples</th>
<th>Ch1</th>
<th>Ch2</th>
<th>Ch3</th>
<th>Ch4</th>
<th>Ch5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moore</td>
<td>0.901</td>
<td>0.943</td>
<td>1.115</td>
<td>0.928</td>
<td>0.916</td>
</tr>
<tr>
<td>Canny</td>
<td>0.18</td>
<td>0.158</td>
<td>0.154</td>
<td>0.173</td>
<td>0.182</td>
</tr>
<tr>
<td>Proposed</td>
<td>0.071</td>
<td>0.087</td>
<td>0.089</td>
<td>0.077</td>
<td>0.102</td>
</tr>
</tbody>
</table>
4.7. Summary

This chapter describes different preprocessing steps used in this research work. To bring about size uniformity, the character image samples are normalized to 72x72 pixel sizes using nearest neighbor interpolation method. The normalized image is converted into binary image using Otsu’s threshold selection technique. Thinning of binary character image is an important preprocessing step, which reduces the size of values to be processed considerable. An efficient thinning algorithm is a requirement of any handwritten character recognition system. A concise description of six popular thinning algorithms is included followed by the proposal of Modified Stentiford Thinning (MST), a novel thinning algorithm. The algorithm are implemented and compared. The proposed method outperforms others in terms of quality of output. Finally a fast, efficient and accurate contour extraction method using eight-point sequential Euclidian distance map and connectivity criteria based on maximal disk is proposed. The connectivity criterion is based on a set of point
pairs along the image boundary pixels, which are the nearest point under consideration and its neighbours. The proposed method is implemented and compared to the two standard contour tracing algorithms, the Moore method and the Canny edge detection method. The results established the efficacy of the proposed algorithm as it produce better output and taking lesser computing time.