CHAPTER 5

SPATIAL FEATURES BASED SCRIPT RECOGNITION
FROM PRINTED BILINGUAL DOCUMENT IMAGES

5.1 INTRODUCTION

Script Recognition is a crucial step which arises in document image analysis when the given document image is multi script in nature. Numerous Script recognition techniques reported in the literature for the Indian scripts (Pal and Chaudhury 1997, 1999) would not suit for the differentiation of Tamil and English scripts since they are script dependent.

The discrimination of Scripts among English and Tamil documents attempted in the literatures at the word level such as Aspect ratio (Tan et al 1999), Spatial spread (Dhanya et al 2002), Water Reservoirs, Structural Features exhibits an average performance, since they rely on the parameters such as the width of the word (Tan et al 1999), reservoirs (open holes), Presence of ascenders and descenders (Dhanya et al 2002) and these fail to exploit the detailed nature of the script. A problem arises in many situations when the script does not conform to these parameters. For instance, ascenders and descenders could be obtained from the word images only when the entire line in the document image produces uniformity. When the word images in a line do not possess uniformity, assumptions of an ascender and descender would cause total degradation. Therefore, the recognizer must be able to identify the script at the word level without depending on a single parameter and by exploiting the complete nature of the script.
The above shortcomings motivated this research towards analyzing the density of the characters, (black spread of the character over its total area), since it differs from character to character based on the shapes, and could well exploit the nature of the scripts. As a result, we proposed a new script recognizer called the Spatial Features Based Script Recognizer (SFBSR), which adopts the local approach, to recognize the script of words in Bilingual document images (Tamil, English) at the character level (recognizing the script of the word image using its initial character). The SFBSR analyzes the character images based on their spread of their densities, which is insensitive to variations (across various font faces and sizes). This recognizer models every character as Tetra Bit Values (TBV), which corresponds to the spatial spread or density, derived from the segmented grids of the character. Later, a Decision Tree Classifier (DTC) is employed for the classification of the script on the patterns generated from the TBV. The SFBSR has been trained and tested with a corpus of bilingual document images, consisting of various Tamil and English words, to show its effectiveness in script recognition.

With this, the major benefits addressed by our SFBSR are as follows:

- Script Recognition of the word image using its initial character across various font faces and sizes.
- Works well across the quality of images.
- Script Recognition rate better than that of the existing approaches.

Since the SFBSR is intended to identify the script as either Tamil or English using its character set, an overview of the Tamil and English scripts and their character set has been explained in the subsequent section.
5.2 BACKGROUND

This section analyses the nature and characteristics of bi-scripts, Tamil and Roman script (English), with the perspective of the densities of the characters in both the scripts, before feature extraction.

5.2.1 Roman (English) Script

The Roman script consists of 52 characters, 26 each of upper case and lower case letters. The structure of the Roman alphabet contains more vertical and slant strokes. Roman alphabets are mainly made up of vertical lines (limited number) rather than curves, circles and arcs. Moreover, while dealing with the density of Roman characters, it is observed that most of the alphabets (except a few) are less dense or not uniformly spread over their character area (Spatial distribution not proportionally well over the total area of the character). The characters of the Roman script are shown in Figure 5.1.

\[
\begin{align*}
\text{a} & \quad \text{b} & \quad \text{c} & \quad \text{d} & \quad \text{e} & \quad \text{f} & \quad \text{g} & \quad \text{h} & \quad \text{i} & \quad \text{j} & \quad \text{k} & \quad \text{l} & \quad \text{m} & \quad \text{n} & \quad \text{o} & \quad \text{p} & \quad \text{q} & \quad \text{r} & \quad \text{s} & \quad \text{t} & \quad \text{u} & \quad \text{v} & \quad \text{w} & \quad \text{x} & \quad \text{y} & \quad \text{z} \\
\text{A} & \quad \text{B} & \quad \text{C} & \quad \text{D} & \quad \text{E} & \quad \text{F} & \quad \text{G} & \quad \text{H} & \quad \text{I} & \quad \text{J} & \quad \text{K} & \quad \text{L} & \quad \text{M} & \quad \text{N} & \quad \text{O} & \quad \text{P} & \quad \text{Q} & \quad \text{R} & \quad \text{S} & \quad \text{T} & \quad \text{U} & \quad \text{V} & \quad \text{W} & \quad \text{X} & \quad \text{Y} & \quad \text{Z}
\end{align*}
\]

Figure 5.1 Roman Script

5.2.2 Tamil Script

The Tamil Script has been derived from the Brahmi Script. A shape based character set of Tamil script has been shown in figure 5.2. The Tamil Script has 12 vowels (indicated in the first row of Figure 5.2) and 18 consonants (shown in the third row of Figure 5.2). These are combined with each other to yield 216 composite characters and 1 special character (aayutha ezhuthu, first row, the last character of Figure 5.2) accounting to a total of (12+18+216+1) 247 characters.
There is a dominance of both horizontal and vertical strokes in the Tamil script. Characters of the Tamil script are made not only of lines and strokes but also circles, kems and curves (more complications). Moreover, characters in the Tamil script are denser (except a few) and their density gets distributed uniformly over the total area of the character cell (where spatial distribution is proportionally equal over the total area of the character, irrespective of the font face and size).

Based on the characters shown in Figures 5.1 and 5.2 and their densities, we proposed the SFBSR for bilingual images (Tamil and English) in this thesis, which analyzes the density of the initial character of the word image and generates the TBV (Patterns) and identifies the script using DTC based on these patterns. The architecture of the SFBSR has been detailed in the next section.

![Tamil Script](image)

**Figure 5.2 Tamil Script**

### 5.3 SCRIPT RECOGNITION ARCHITECTURE

The SFBSR (Script Recognition) consists of two phases namely, the Training phase and the Testing phase. The involvement of the Training phase arises here to extract and analyze the densities of the characters of bi-scripts (from the bilingual document image corpus) constituting various font faces and sizes, and to train the classifier to classify the script across variations.
In the Training phase, a corpus of bilingual document images (Web and scanned book images) undergoes preprocessing to eliminate noises, and word images are segmented from it. In the word images, tetra bit values (patterns) have been generated for the initial character of the word image (both English and Tamil) based on its density spread. Later, these patterns (training samples in both the scripts) belonging to various document images are fed as input to the DTC, which learns the knowledge from the presented patterns, to predict the script.

During the testing phase, when the user supplies the query bilingual document image as input, words and their initial characters get bounded. The bit generation process extracts the density and transforms it into tetra bit values. The decision tree classifier which is trained offline on synthetic training samples (prepared in various font faces and sizes) predicts the script of the word images properly using their initial characters. The SFBSR process is depicted in Figure 5.3.
The Processes involved in the Training and Testing phases of Script Recognition architecture have been explained in the following sections 5.4 and 5.5.

5.4 TRAINING PHASE OF SCRIPT RECOGNITION

In the Training Phase the bilingual document image corpus (mixture of Tamil and English words in document images) acquired from various sources such as the Web and scanners, has been analyzed and trained to recognize the script across variations. This phase includes the preprocessing of document images, word and character image segmentation, Tetra Bit generation, Classification selection, Decision tree induction (involving a training set of patterns), Rule Extraction and Sub-classification technique.

5.4.1 Preprocessing

An input is a gray scale image obtained by scanning the bilingual document image (English and Tamil). Here, the input area is assumed to contain a text and is thus free from graphics, figures, maps and tables. Variations in the quality of document images such as noise, poor image resolution and document degradation make the script identification quite difficult. All these kinds of degradations must be compensated before the density analysis. In order to minimize the effect of variability in document images related to different noises, we take some preprocessing steps such as Noise Filtering, Binarization and Extra pixel removal process.

Initially, noises are suppressed in the document image by using a median filter, since median filters smear the character image strokes (Lu and Tan 2008). Subsequently, binarization has been performed. Binarization is a technique by which color and gray scale images are converted into binary images as stated in Section 4.2.1.
Since the character strokes suffer from single pixel holes, concavities, and convexities along the boundaries of characters, may affect the density analysis of characters. Therefore, these single pixel defects have been removed here using logical operators (Lu and Tan 2008) according to their neighborhood patterns.

5.4.2 Word Image Segmentation and Character Bounding

Since this SFBSR adapts the local or statistical approach for Script recognition, it involves the segmentation process. In this thesis, the top down segmentation approach has been adopted, which breaks the document image into lines, words and characters. Here, we segment the word image first and its initial character gets bounded.

The algorithm for segmentation is described below:

1. The binarized image is checked for inter line spaces.
2. Each line in the paragraph is identified through a Horizontal Projection Profile as defined in equation (5.1)

\( f(x, y) \) is the 2D array pixel of image consisting of text block.
\( x \) corresponds to width of text block.
\( y \) corresponds to height of text block.
\( 0 \) – black pixel \( 1 \) – white pixel

\[
\text{for each } y \text{ in } f(x, y) \\
ss_i(a, b_1^1) = f(0, y) \text{ if } ((x_i \& y = 0) \text{ and } (x_i \& y - 1 = 1)) \text{ where } (0 \leq i \leq x) \\
se_i(a, b_2^2) = f(w, y) \text{ if } ((x_i \& y = 1) \text{ and } (x_i \& y + 1 = 0)) \text{ where } (0 \leq i \leq x) \\
\text{next}
\]

(5.1)
where \( L_i \) corresponds to 2-D pixel array of \( i^{th} \) line

\[ \text{ss}_i(a,b_1) \] — Starting coordinates of \( i^{th} \) line

\[ \text{se}_i(a,b_2) \] — Ending coordinates of \( i^{th} \) line

3. Once the line bounding is over, with the help of vertical projection, word images are identified as in equation (5.2)

\[
\text{for each } x \text{ in } L_i(x,y) \\
\text{ws}_i(a_1,b_1) = f(x,b1) \text{ if } ((x \land y \_j = 0) \text{ and } (x - 1 \land y \_j = 1)) \text{ where } (b1 \leq j \leq b2) \\
\text{we}_i(a_2,b_2) = f(x,b2) \text{ if } ((x \land y \_j = 1) \text{ and } (x + 1 \land y \_j = 0)) \text{ where } (b1 \leq j \leq b2) \\
\text{next}
\] (5.2)

\[ W_i(x,y)=\text{ws}_i(a_1,b_1)...\text{we}_i(a_2,b_2) \]

where \( W_i \) corresponds to 2-D pixel array of \( i^{th} \) word

\[ \text{ws}_i(a_1,b_1) \] — Starting coordinates of \( i^{th} \) word

\[ \text{we}_i(a_2,b_2) \] — Ending coordinates of \( i^{th} \) word

Once the word gets identified, the initial character of the word image gets bounded further using equation (5.2) for density analysis. The density of the character over its total area is analyzed using the tetra bit values generated for the character image.

5.4.3 Tetra Bit Generation

An analysis of bi-scripts across variations in font faces and sizes shows that (as stated in the Section 5.2), characters in the Tamil script are denser than the characters of the English script. In Tamil script, density or
pixel concentrations of the characters are almost spread over its width and height uniformly (except few zones such as top left, bottom right etc.). It is interesting to find that in the English script most of the characters except few (such as $M$, $W$, $w$, $m$, $S$ etc...), are not equally spread over the character area; otherwise, the density (pixel concentration) is not uniformly spread over the total area of the character (total area is referred as the height*width of the character). In contrast, densities of the Tamil characters are uniformly spread over their total area (denser) except for a set of characters.

In this scenario, the problem arises with the discrimination of the characters, which are not denser (uniformly spread) in both the scripts. Further analysis leads to the evidence that, the pixel concentration of the characters is less denser in a few regions (limited) of the total character area for Tamil script whereas it is more for the English Script. Less density over the total area in few regions correspond to either top left corner, top right corner, bottom left corner, bottom right corner or corner of middle regions etc. over the total area of character cell (height*width).

In both the scripts, discrimination of less dense characters based on their zones could very well distinguish the script. In order to analyze these less dense regions, if two horizontal cuts or two vertical cuts are performed over the character, it yields three zones and results in poor classification (because this is not sufficient to classify the density). Alternatively, if two horizontal cuts and one vertical cut has been made over the character ($2*3$), a total of six regions would result. (Top left, Top right, Middle left, Middle Right, Bottom Left and Bottom Right). Again, the density analysis, which results from this grid formation, is not satisfactory.

Partitioning of six regions over the characters is able to identify 47 shapes in The Tamil script and 21 shapes in the English script distinctly, whereas, this fails for the character set such as $\hat{U}$, $\hat{Y}$, $\hat{P}$, $\acute{a}$, $\ddot{a}$, $\ddot{ae}$, $\circ$, $\|$, etc.
in Tamil and a, g, s, w, x, B, M, R, W, etc. in English since the density remains the same in all the six zones. Therefore, increasing the number of partitions over the characters for density analysis would produce better discrimination among the scripts.

Based on these problems, we proposed a technique, which bounds the initial character of the word image (using equation 5.2) and divides it into nine zones (3*3) to analyze its density or spatial spread. Therefore, grid formation over the character cell divides the total character area into nine zones (Top left, Top middle, Top Right, Left Middle, Central, Right Middle, Bottom Left, Bottom Middle and Bottom Right). Since, most of the characters could be discriminated over these nine regions, a 4*4 partition over the characters is unnecessary (also tedious). As a result, we adopted a 3*3 partition based density analysis of characters in this work.

5.4.3.1 Grid Formation

In this thesis, the character images of I*J pixels are divided into M*N grids (M=N=3) as shown in figure 5.4 by applying two horizontal projections and two vertical projections over the character area. The Horizontal projections depict the horizontal cuts or partitions made over the total height of the character cell. Here, two horizontal partitions have been employed, which divide the total length of the character into three distinct regions. (Therefore M=3). Horizontal projections are applied over the height of the character cell as represented in equations (5.3) and (5.4).

\[ H1c = f(x, y), (0 \leq x \leq wc), (0 \leq y \leq hc/3) \]  
\[ H2c = f(x, y), (0 \leq x \leq wc), (0 \leq y \leq 2*hc/3) \]

Where \( f(x, y) \) = 2-D array pixels of image
H1c = First Horizontal cut
H2c = Second Horizontal cut
wc = Width of the Character
hc = Height of the Character

Similarly, Vertical projections depict the vertical cuts or partitions made over the total width of the character cell. In this thesis, two vertical partitions are applied, which divide the total width of the character into three regions. Vertical projections are applied over the width of the character cell as represented in equations (5.5) and (5.6)

\[
V1c = f(x, y), (0 \leq x \leq wc/3, 0 \leq y \leq hc)
\]

(5.5)

\[
V2c = f(x, y), (0 \leq x \leq 2*wc/3, 0 \leq y \leq hc)
\]

(5.6)

Where 
\[f(x, y) = \text{2-D array pixels of image}\]
V1c = First Vertical cut
V2c = Second Vertical cut
wc = Width of the Character
hc = Height of the Character

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 5.4 Tetra zones for a character**

The Horizontal and Vertical projections applied together over the total area of characters, yield nine regions (3*3) or nine grids, which are also called as Tetra zones, as depicted in Figure 5.4. The formation of nine grids for sample Tamil and English characters is depicted in the Figure 5.5.
Figure 5.5 Grid formation for sample characters

5.4.3.2 Tetra Bit Values (TBV)

Since the SFBSR performs script identification by analyzing the density or spatial distribution over the total area (nine grids) of the character, the black and white densities of each grid in the character cell have been analyzed to record its distribution in that area. The Black density corresponds to the ratio of black pixels to the total area of the grid, whereas the white density corresponds to the ratio of white pixels to the total grid area. The presence of Black and White pixels in each grid is computed as indicated in equations (5.7) and (5.8). After investigating the presence of black and white pixels in each grid, The Black and White density of each grid has been computed as represented in equations (5.9) and (5.10).

\[
bpc(zn(i,j)) = \begin{cases} 
  bpc = bpc + 1 & \text{if } zn(i, j) = 0 \\
  bpc & \text{if } zn(i, j) = 1 
\end{cases} \\
\]

where \( zn - n^{th} \) zone \( n = 1..9 \)

\[
\text{where } (i, j) \in (m,n) \ (0 \leq m \leq M, 0 \leq n \leq N) \]

\[
bpc = \text{Black Pixel count} \\
M = \text{Total number of rows} \\
N = \text{Total number of columns} \\
\]

\[
wpc(z\alpha(i, j)) = \begin{cases} 
  wpc = wpc + 1 & \text{if } z\alpha(i, j) = 1 \\
  wpc & \text{if } z\alpha(i, j) = 0 
\end{cases} \\
\]

\[
\text{where } (i, j) \in (m,n) \ (0 \leq m \leq M, 0 \leq n \leq N) \]

\[
\text{where } (i, j) \in (m,n) \ (0 \leq m \leq M, 0 \leq n \leq N) \]
\[ wpc = \text{White pixel Count} \]

\[ \text{bpt}(z_n) = \left( \frac{\text{bpc}(z_n)}{\sum (\text{bpc}(z_n) + \text{wpc}(z_n))} \right) \times 100 \]  \hspace{1cm} (5.9)\]

\[ \text{wpt}(z_n) = \left( \frac{\text{wpc}(z_n)}{\sum (\text{bpc}(z_n) + \text{wpc}(z_n))} \right) \times 100 \]  \hspace{1cm} (5.10)\]

where \( z_n \) - \( n^{th} \) zone \( n = 1..9 \)

\[ \text{bpt} = \text{Black Density} \]

\[ \text{wpt} = \text{White Density} \]

Later, a suitable threshold (experimentally fixed as 0.25) is chosen to determine the dominance of ink distribution in each grid (the evaluation of different thresholds and their performance has been recorded in the ‘Results’ section). A Grid is assumed to be Black dominant (dense) when its black density exceeds the threshold otherwise it is assumed to be White dominant (less dense). Here, the black dominance represents the spread of density (black pixels) in a particular area (grid) of the character cell, whereas the white dominance represents the absence of black density in that area. Since some parameters are required to indicate the dominance of that grid, bit ‘1’ has been assigned to the grid if it is White dominant and bit ‘0’ has been assigned to the grid for Black dominance. Consequently, bit values get generated for each grid (totally nine grids) based on the black and white density as represented in equation (5.11) which results in a vector of nine dimensions corresponding to nine zones (Tetra bits) of a character. Tetra bit values generated for a set of sample English and Tamil characters have been represented in Table 5.1.

\[ \text{Dom}(zn) = \begin{cases} 
1 & \text{if } \text{wpt}(zn) > th \\
0 & \text{if } \text{bpt}(zn) > th 
\end{cases} \hspace{1cm} (5.11) \]

\[ zn - n^{th} \text{ zone } n = 1..9 \]
Where, \[ \text{Dom} = \text{Dominance of } n_{th} \text{ zone} \]
\[ th = \text{Threshold} \]

Table 5.1 Sample Tetra Bit Values

<table>
<thead>
<tr>
<th>Zones (Binary form)</th>
<th>C</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{LI} )</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( \text{GR} )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( \text{P} )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Since the density or spatial distribution of the characters has been indicated using nine bits, this clearly depicts the spread over of the characters across their width and height. From the above table, it is evident from the bits of the first character, that it is not equally spread over its top middle and central zone along the total character area. Also, it is observed that the second character has a uniform pixel spread over its total area, but the pixels are not spread over the bottom right corner area in the case of the third character. Therefore, bits generated for the characters of bi-scripts make the assumption to be true.

In this manner, tetra bits have been generated for various training samples of characters constituting various font faces and sizes. Moreover, a proper discriminator is required to recognize the origin of the script based on the tetra bits of the characters.

5.4.4 Classifier Construction

The task of discrimination can be carried out successfully by a supervised learner, because Supervised Classification is a machine learning
technique which learns a function from the training data and produces an output based on the training set (consisting of input objects and desired outputs). It is a two-way classification process (Han and Kamber 2006) in which a classifier is built during the first step using a pre-determinant set of data classes or concepts. This is the learning step (or training phase), where a classification algorithm builds the classifier by analyzing or learning from a training set made up of database tuples and their associated class labels. In the second step, the learned model is used for classification. Therefore, to classify the tetra bit values, the supervised learner (classifier) has been adopted.

Even though different supervised classification algorithms exist, the Decision Tree Classifier (DTC) has been employed in this thesis, since it supports multi-stage decision-making by testing the samples against certain subsets of classes rather than the conventional single stage classification, and this would suit the training set. The Design of a Decision Tree Classifier depends mainly depends on the choice of feature subsets or feature classes to be used as internal nodes of the tree and the choice of decision rules used at each internal node for classification. In this thesis, since tetra bits have been generated for each character, nine feature classes are used as internal nodes in the design of the DTC.

As a result, in this work, nine feature classes (corresponding to every bit of TBV) or subsets would be available to train the decision tree classifier where the nine feature classes correspond to nine tetra bit values. There is a possibility of tetra bits being less dense in 1 grid, 2 grids, 3 grids (presence of ‘1’ bits in TBV) and so on, up to 9 grids. But the possibility of tetra bits being less dense in more than six grids would never arise but the feature reduction or feature selection cannot be applied over these nine feature classes directly for classification, since the less dominant zones can occur
anywhere at random among the nine zones, and the features appear to be ‘0’ and ‘1’ in a dispersed manner.

This leads us to the conversion of tetra bit values into a suitable pattern representing the position of the grid (grid number), where the density remains sparse. Obviously, the zone numbers where the less dominance appears in tetra bits must be accounted for processing. Based on these patterns (position of the grid), inputs for the nine feature classes are fed, so that the feature reduction is possible before classification.

5.4.4.1 Patterns for Feature Subsets

Tetra bit values generated in the above section gets transformed into a suitable pattern which act as an input for nine feature classes C1, C2, C3,C4..C9, with a subsequent decision as output (analyzed from the training samples). For example, if the training TBV is less dominant in the 2nd, 5th and 8th zone, pattern would be “258” and its probable decision would be “English”. To generate patterns for the input, the total number of less dominant bits in TBV and the position of each less dominant bit have been calculated as indicated in algorithm 5.1.

Algorithm 5.1: Finding Bit positions

Declare

n - Number of Zones
Z_n - n^{th} Zone where (1<=n<=9)
TBV - Tetra bit values
nb - number of less dominant bits in TBV
P_{nb} - Position or zone number of the less dominant bit where, (0<=nb<=9)

Procedure Find_Position ()
1. Start
2. Initialize nb as zero
3. For n = 1..9
   1) If \((Z_n\) equals 1) in \(TBV(Z_n)\)
      \(nb++;
      P_{nb} = n;
4. Next
5. End

Once the total number of less dominant bits and their position has been identified, they are transformed as input into the nine feature classes as defined in the definitions 5.1 and definition 5.2.

**Definition 5.1:** Generate nine input values for the DTC, if the total number of bits (nb) equals ‘n’ (where \(n \neq 0\)), with positions corresponding \(p_1..p_n\), assign each position(grid number) \(p_1, p_2..p_n\) to the predefined classes \(C_1, C_2..C_n\) and null to the remaining classes \(C_{n+1}..C_9\) with its probable output. This definition is explained with the help of different cases.

Case 1: If the number of bits equals ‘1’ and the position corresponds to ‘5’, nine values (input values) get generated by assigning ‘5’ to class \(C_1\) and null to the remaining classes \(C_2..C_9\), with a suitable decision as output.

Case 2: If the number of bits equals ‘2’ and the position corresponds to ‘5’ and ‘8’, nine values (input values) get generated by assigning ‘5’ to class \(C_1\), ‘8’ to \(C_2\) and null to the remaining classes \(C_3..C_9\), with a proper output. Similarly, input values get generated for the other combinations also.
**Definition 5.2:** Generate nine input values for the DTC, if the total number of bits (nb) equals ‘0’, with positions corresponding to ‘null’, assign ‘0’ to all the predefined classes $C_1$, $C_2$, $C_3..C_9$ and the decision result as “Sub-classification (HT)”.

Table 5.2 represents the training set of class labeled tuples randomly selected from the training set of images (out of 150 tuples). The tetra bit values have been transformed into nine predefined attributes (input objects) or columns $C_1$, $C_2$, $C_3..C_9$. The class label attribute, “Decision” (output), has three possible values namely “Tamil”, “English” and “HT”, where HT corresponds to the sub-classification (Section 5.4.4.4). The decision attribute has been defined based on the knowledge (domain level) analyzed from the training set.

<table>
<thead>
<tr>
<th>$C_9$</th>
<th>$C_8$</th>
<th>$C_7$</th>
<th>$C_6$</th>
<th>$C_5$</th>
<th>$C_4$</th>
<th>$C_3$</th>
<th>$C_2$</th>
<th>$C_1$</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>3</td>
<td>$T$</td>
</tr>
<tr>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>6</td>
<td>$T$</td>
</tr>
<tr>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>8</td>
<td>$E$</td>
</tr>
<tr>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>2</td>
<td>$T$</td>
</tr>
<tr>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>5</td>
<td>$HT$</td>
</tr>
<tr>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>4</td>
<td>$E$</td>
</tr>
<tr>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>1</td>
<td>$T$</td>
</tr>
<tr>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>2</td>
<td>$E$</td>
</tr>
</tbody>
</table>

Since all the nine input objects cannot be considered for classification, feature selection is needed to deduce the features for classification. Wikipedia states that, ‘Feature selection, (also known as variable selection, feature reduction, attribute selection or variable subset selection), is a technique (Feature Selection 2009), commonly used in machine learning, of selecting a subset of relevant features for building robust learning models’. 
For Subset selection, search approaches are popular which include Exhaustive, Best first, simulated annealing, Greedy forward selection and Greedy backward selection techniques. Exhaustive search is impractical and many applications make use of greedy techniques which consist of Sequential Forward Selection (SFS) and Sequential Backward Selection (SBS). SFS starts from the empty set and sequentially adds the feature X that results in the highest objective function when combined with the features Y that have already been selected. On the other hand, SBS starts from the full set and sequentially removes the feature X, which results in a negligible decrease in the objective function. SBS works best when the optimal feature subset has a large number of features.

In this thesis, SBS has been selected for feature selection with Information Gain (Han and Kamber 2006) as the objective function since this suit well for large feature sets well. Based on the Information Gain values obtained through SBS, input classes C6, C7, C8 and C9 have been removed from the list since it exhibits null. Feature classes C1, C2, C3, C4 and C5 have been considered for the construction of a Decision Tree Classifier and this involves decision tree induction

5.4.4.2 Decision Tree Induction

Decision Tree induction is the learning phase of decision trees from class-labeled training samples. A decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node holds a class label (Han and Kamber 2006). The topmost node in a tree is the root node and building the decision tree hierarchy needs attribute selection.

An attribute selection measure is an important heuristic for selecting the splitting criterion that “best” separates a given data partition, D,
of class-labeled training tuples into individual classes in order to build a decision tree (The root node and its hierarchy have been decided based on the attribute selection). In this thesis, information gain (Han and Kamber 2006) has been selected as an attribute measure for the construction of the decision induction tree.

From the above Table 5.2, columns C1, C2, C3, C4 and C5 are selected as features and considered for information gain computation. Here class C1 produces the information gain as 0.2690, C2 as 0.1982, C3 as 0.1941, C4 as 0.0877, and C5 as 0.0234 respectively. The result proves that the highest information gain valued C1 can be considered as the root node, C2 follows C1 and C3 follows C2 and C4 follows C3 and C5 further follows C4. The final decision tree consists of five different levels; some lateral views of the decision tree have been shown in Figures 5.6, 5.7, 5.8, 5.9 and 5.10.

![Decision Tree depicting Root Node](image-url)
Figure 5.7 Sample Decision Tree representing Branch A
Figure 5.8 Sample Decision Tree representing Branch C

Figure 5.9 Sample Decision Tree representing Branch D
The typical decision tree shown in the above figures represents the classification decision in the leaves i.e. whether the particular character in the document image belongs to either Tamil or English or HT. HT means that the training sample must undergo a sub classification procedure for further discrimination (stated in sub classification section 5.4.4.4).

5.4.4.3 Rule Extraction from Decision Tree

The decision tree can be automatically converted into classification IF-THEN rules by tracing the path from the root node to each leaf in the tree. Among the rules extracted from the decision tree (Figures 5.6 to 5.10), sample rules are shown in the algorithm 5.2.

Algorithm 5.2: Rule based Classification

Start
if (C1 exists)
    if (zone3 equals ‘1’) Decision=’T’
    else if (zone9equals ‘1’) Decision=’T’
    else if (zone1equals ‘1’) Decision=’T’
    if (zone4 equals ‘1’) Decision=’E’
etc…

else if (C1 & C2 exists)
    else if ((z6 △ z9) equals ‘1’) Decision = “T”
    else if ((z3 △ z6) equals ‘1’) Decision = “HT”
    else if ((z4 △ z6) equals ‘1’) Decision = “E”
    if((z7 △ z8) equals ‘1’) Decision = “HT”
    if ((z5 △ z6) equals ‘1’) Decision = “E”
    etc…

else if ( C1 & C2 & C3 exists)
    if ((z3 △ z6 △ z9) equals ‘1’) Decision = “T”
    else if ((z1 △ z6 △ z9) equals ‘1’) Decision = “T”
    else if ((z4 △ z5 △ z8) equals ‘1’) Decision = “E”
    else if ((z3 △ z5 △ z6) equals ‘1’) Decision = “HT”
    etc…

else if (C1 & C2 & C3 & C4 exists)
    if ((z4 △ z6 △ z7 △ z9) equals ‘1’) Decision = “E”
    else if ((z1 △ z6 △ z9) equals ‘1’) Decision = “E”
    else if ((z4 △ z5 △ z8) equals ‘1’) Decision = “E”
    else if ((z1 △ z3 △ z7 △ z8) equals ‘1’) Decision = “T”
    etc..

else if (C1 & C2 & C3 & C4 & C5 exists)
    if ((z2 △ z4 △ z6 △ z8 △ z9) equals ‘1’) Decision = “E”
    else if ((z2 △ z3 △ z7 △ z8 △ z9) equals ‘1’) Decision = “T”
    etc..

End

Apart from the classification decision ‘Tamil or English’, one more decision called as “HT”- Horizontal Transition (sub classification technique) exists which goes through the procedure stated in the subsequent section for a further classification of the script. Some training samples undergo HT, since a
set of characters in both Tamil and English scripts possess a uniform distribution in their density. This kind of complication cannot be solved by DTC and it is further discriminated using a sub-classification technique called as “Horizontal Transition”. Examples for all kinds of cases (C1, C2, C3, C4 and C5) which could be solved by DTC have been explained below.

Case 1: (Example representing a single feature C1)
- Consider the letter ந. Tetra Bit value is 1 0 0 0 0 0 0 0 0. It means the letter is distributed in all zones except the first. Based on the decision tree classification rule, the letter which is not in zone 1, will be identified as “Tamil”. So the script of the letter ந is Tamil.

Case 2: (Example representing Two features C1 and C2)
- Consider the letter த. Its Tetra Bit Value is 1 0 0 0 1 0 0 0 0. It means the letter is distributed in all zones except 1 and 5. Based on the decision tree classification rule, the letter which is not in zones 1 and 5 will be identified as “Tamil”. So the script of the letter த is Tamil.
- Consider the letter ன. Its Tetra Bit Value is 0 1 1 0 0 0 1 0. It means the letter is distributed in all zones except 2 and 3. Based on the decision tree classification rule, the letter which is not in zones 2 and 3, will be identified as “English”. So the script of the letter ன is English.

Case 3: (Example representing Three features C1, C2, and C3)
- Consider the letter ர. Its Tetra Bit Value is 0 0 0 0 1 1 0 1. It means the letter is distributed in all zones except 6, 7 and 9. Based on the decision tree classification rule, the letter which
is not in zones 6, 7 and 9, will be identified as “Tamil”. So the script of the letter  ông is Tamil.

Case 4: (Example representing Four features C1, C2, C3, and C4)

- Consider the letter j. Its Tetra Bit Value is 1 1 0 1 1 0 0 0 0. It means the letter is distributed in all zones except 1, 2, 4 and 5. Based on the decision tree classification rule, the letter which is not in zones 1, 2, 4 and 5, it will be identified as “English”. So the script of the letter j is English.

Case 5: (Example representing Five features C1, C2, C3, C4, and C5)

- Consider the letter y. Its Tetra Bit Value is 0 1 0 1 0 1 0 1 1. It means the letter is distributed in all zones except 2, 4, 6, 8, 9. Based on the decision tree classification rule, the letter which is not in zones 2, 4, 6, 8 and 9, it will be identified as “English”. So the script of the letter y is English.

Training samples which could not be solved by Decision Tree Classifier undergoes Horizontal Transition for discrimination.

5.4.4.4 Sub-Classification (Horizontal Transition)

The Horizontal Transition rate detects the horizontal disposition rate, in the middle grids along the width w of the bounded character. The Horizontal disposition rate is recorded by accounting every black to white and white to black dispositions over the middle zone. It is also observed from the training samples that the transition rate of middle zones lies between 1 and 2 for English characters (experimentally fixed as 2 for English) and more than 2 for the Tamil Script due to its spatial spread. Horizontal Transition could very well discriminate the characters through its black runs if density remains same
in both the scripts. The Horizontal Transition algorithm and the different cases of characters requiring horizontal transition are explained below.

**Algorithm 5.3: Horizontal Transition identification**

Declare

ht – horizontal transition rate.
ht – Total number of Horizontal Transitions in a character.

**Procedure HT_Iden ()**

1. Start
2. h = Middle zones (z4, z5, z6)
3. ht1 = ∅
4. flag = false
5. for each w_i of h in W
   i. if (pixint (w_i,h) = 0) ∧ (flag = true)
      1. set flag to false
      2. ht = ht +1
   ii. if pixint (w_i,h) = 1
      1. set flag to true
   iii. ht = max (ht, ht_i)
6. if (ht <=2) decision = “English”
   i. else if (ht >2) decision = “Tamil”
7. End.

**Case 1:** Characters not classified by DTC - Consider the letter ». Its Tetra Bit value is 0 0 0 0 0 0 1 1 0. It means the letter is distributed in all the zones except 7 and 8. The Decision tree classifier cannot identify the script of the letter which is not distributed in zones 7 and 8, since there are some English letters such as q which is also having the same TBV. Therefore, the horizontal transition method is applied to recognize the given
letter. The number of horizontal transitions of the letter is more than 2. So the script of the letter » is Tamil whereas it is less than 2 in the case of q.

**Case 2:** Characters in which the density is distributed all over the zones - Consider the letter á. Its Tetra Bit value is 0 0 0 0 0 0 0 0 0. It means the letter is distributed in all the zones. The Decision tree classifier cannot identify the script of the letter which is distributed in all zones, since there are some English scripts such as E which are also having the same TBV. Therefore, the horizontal transition method is applied to the given letter. The number of horizontal transitions of the letter is more than 2. So the script of the letter á is Tamil.

Training samples of bilingual document images (corpus) acquired from multiple sources (Scanners and Web images), undergo all the processing reported above, to extract the features and to train the classifier with those features. After training, the classifier is able to recognize the script of the word images and the efficiency of the script recognizer has been tested in the testing phase.

### 5.5 TESTING PHASE OF SCRIPT RECOGNITION

When a test document image is provided to the system, words and characters are segmented, tetra zones are formed for the initial character, and in turn the features of every zone get extracted. These features are then transformed into tetra bit values as explained above. The Tetra bit values get transformed into patterns based on the position of the less dominant zones. The generated pattern goes through the above rule-based classifier to classify the values as either “Tamil” or “English” or “HT”. If HT is crossed, again it comes through a Horizontal transition procedure to classify the script as Tamil or English.
For instance, the Tamil word represented in figure 5.11 is properly classified as Tamil. Initially, the first character gets segmented and the bits generated for that character are 000000000. Here the classification decision is made as “HT”. The Horizontal transition procedure clearly discriminates this as Tamil.

Figure 5.11 Sample Tamil word image

For example, the English word “Pulse” gets classified as English. After segmenting the first character, the system produces “000000011” as nine bit values. The Classifier classifies this as English. The Script of the word pulse has been predicted using its first character “P”.

The results of the SFBSR in the testing phase are depicted in Figures 5.12 and 5.13. Figure 5.12 shows the sample document image and Figure 5.13 shows the reported script using its initial character. Further, the test results of the proposed system and its efficiency over the existing system have been analyzed in the subsequent section.

Figure 5.12 Input Image for Script Recognition
5.6 RESULTS AND PERFORMANCE ANALYSIS

In this thesis, a data set has been acquired from 1000 bilingual document images (web) each of which contain about 50-150 training samples of word images. Out of this, we considered 7000 Tamil word images and 4000 English word images for training the system. Scanned images as well as the images downloaded from the web consisting of newspaper and magazine articles etc., with various font faces and sizes are considered, assuming that no skew and italic style exists. In the testing phase, 2000 word images, which contain 1000 samples for each script, are tested using the 2.4 GHz system having 512MB RAM and various metrics have been evaluated and the results are tabulated in the following sections.

The metrics used to evaluate the performance of the system are Accuracy, Script Recognition Rate and F-Score at the word level. They are defined in equations (5.12), (5.13) and (5.14) as follows:
Accuracy is defined as the ratio of the number of scripts identified correctly, to the total number of scripts identified as represented in equation (5.12). Script Recognition Rate is defined as the ratio of the number of relevant scripts identified by the algorithm, to the total number of word images with their relevant scripts as represented in equation (5.13), and the F-Score of Script Recognition has been identified as represented in equation (5.14).

\[
\text{Accuracy} = \frac{\text{No of relevant Scripts identified}}{\text{No of total Scripts identified}}
\]  
(5.12)

\[
\text{Script Recognition Rate (SRR)} = \frac{\text{No of relevant Scripts identified}}{\text{No of relevant Scripts exists}}
\]  
(5.13)

\[
F - \text{Score} = \frac{2 \times \text{Accuracy} \times \text{SRR}}{\text{Accuracy} + \text{SRR}}
\]  
(5.14)

The performance of this system has been evaluated for the issues reported, using the test results obtained. The issues reported are as follows:

- Accuracy and Script Identification Rate of Word images using their initial character with adaptability to various font faces and sizes.
- Performance Evaluation of the SFBSR with respect to the various Threshold settings.
- Better Discrimination Rate than the existing approaches available for Bilingual (Tamil, English) script recognition.
- Better performance even with noisy images.
5.6.1 Accuracy and Identification Rate of SFBSR

From the experiment on the data set defined above, we noted that the overall accuracy of the system is about 96% for the Tamil script and around 93% for the English script. The distribution of the results of the English and Tamil scripts based on accuracy has been shown in Table 5.3 and Table 5.4 shows the distribution results based on the script identification rate.

From Table 5.3, it is observed that the overall accuracy of the script recognizer as termed in equation 5.15 (Tamil and English) is about 94.5%, since the sensitivity (True positive rate - ratio of the number of true positives to the number of true positives and false negatives) of the Tamil and English scripts remains 96.3 and 92.8 respectively and the specificity (True Negative rate - ratio of the number of true negatives to the number of true negatives and false positives) of the Tamil and English script remains vice versa respectively.

Table 5.3 Accuracy of Tamil and English word images

<table>
<thead>
<tr>
<th>Actual</th>
<th>Tamil</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tamil</td>
<td>96.3%</td>
<td>3.7%</td>
</tr>
<tr>
<td>English</td>
<td>7.2%</td>
<td>92.8%</td>
</tr>
</tbody>
</table>

Overall Accuracy = Sensitivity/2 + Specificity/2  \hspace{2cm} (5.15)

From Table 5.3 it is also noted that the highest accuracy which is about 96.3% is obtained for the Tamil script. This is because the TBV shows a distinct behavior in the Tamil script and a few less dense characters in Tamil get identified as English. Errors came on the English script since some of the higher dense characters are identified as Tamil.
Table 5.4 Script Recognition Rate of Tamil and English word images

<table>
<thead>
<tr>
<th>Word Images</th>
<th>Hit Rate</th>
<th>Miss Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tamil</td>
<td>93.8%</td>
<td>6.2%</td>
</tr>
<tr>
<td>English</td>
<td>90.4%</td>
<td>9.6%</td>
</tr>
</tbody>
</table>

In Table 5.4, it was found that the recall rate of Tamil and English gets variations, since some of the TB values in the English (M, W, w, m) and Tamil (i, ṭ, £, etc...) scripts cannot get discriminated from one another. Moreover, there are some testing samples, which produce TBV apart from the training set and reduce the identification rate. Some testing samples rise problems due to touching characters and their density gets recorded as Tamil. Since M, W, w, m and touching characters are identified as Tamil, the recall rate of English gets reduced to 90%. Due to the existence of a few sparsely dense characters in Tamil and they are being identified as English, the recall rate of the Tamil script is around 94%.

5.6.2 Performance of SFBSR with Respect to various Threshold Setting

This section analyzes the performance of the SFBSR with respect to the various thresholds and heuristics applied in Tetra bit generation and Horizontal Transition Rate. In this thesis, the threshold has been fixed as 0.25 experimentally in Tetra bit value generation (Section 5.4.3) and the Horizontal Transition rate has been fixed as 2 for English and greater than 2 for Tamil based on the experimental results. (Sub classification – Section 5.4.4.4). Also this section analyzes the performance of the various classifiers used in the SFBSR.
The SFBSR achieves a script recognition rate ranging from 86 to 98% for the Tamil script and 82 to 90% for the English script, depending on the threshold chosen in Tetra bit value generation as shown in Figure 5.14. It is obvious from the Figure 5.14 that the higher threshold is not suitable and a lower threshold would work well. In order to achieve a good script recognition rate in both the scripts, the threshold has been fixed as 0.25 in TBV generation, after experimental results. The higher threshold is not suitable to identify some of characters such as $\mathfrak{B}$, $\mathfrak{g}$, $\mathfrak{c}$, etc. and the lower threshold would not be suitable for some of the characters such as $\mathfrak{d}$, $r$, $f$, $t$, etc.

![Script Recognition in Various Thresholds](image)

**Figure 5.14 Script Recognition Rate at various Thresholds**

Next to TBV, the threshold has been fixed as 2 in the Horizontal Transition rate of English and more than 2 for the Tamil script. The Script Identification Rate for both English and Tamil scripts using the SFBSR has been tabulated in Table 5.5 and Figure 5.15 depending on different Horizontal Transition runs.
Table 5.5  Script Recognition Rates based on various Horizontal Transition (HT) Runs

<table>
<thead>
<tr>
<th>HT threshold</th>
<th>Tamil(124)</th>
<th>English(52)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;1</td>
<td>124</td>
<td>17</td>
</tr>
<tr>
<td>&gt;2</td>
<td>120</td>
<td>47</td>
</tr>
<tr>
<td>&gt;3</td>
<td>65</td>
<td>52</td>
</tr>
<tr>
<td>&gt;4</td>
<td>24</td>
<td>52</td>
</tr>
</tbody>
</table>

Figure 5.15  Script Recognition Rate based on various Horizontal Transition Runs

A lower transition rate as the threshold allows a higher recall of the Tamil script but at the expense of the English script. The reverse holds true for a higher threshold. Therefore, to have an ideal script recognition rate, the horizontal transition rate has been chosen as greater than 2 for Tamil and lesser for English.
Since the SFBSR employs two kinds of classifiers, the Decision Tree Classifier (DTC-major classifier) and Horizontal Transition (HT-sub classifier) to classify the Tetra bit values generated for the characters of both the scripts, the performance of both the classifiers have been compared. Here, an individual and an integrated performance of the classifiers in recognizing the Tamil and English scripts have been analyzed and it is represented in Figure 5.16.

Figure 5.16 Script Hit Rates of Classifiers

Figure 5.16 reveals that the integrated performance of the classifiers DTC and HT produces a better rate of classification than their individual performances. Decision Tree Classifier is able to classify only half the set of English characters and \(\frac{3}{4}\) of the characters in the Tamil script. This is because the same TBV has been generated for a group of characters in both the scripts (Tamil & English). Therefore, an unambiguous TBV has been generated for 74\% of the characters in Tamil script and 57\% of the characters in English script and the DTC classifies all these TBV properly. The performance of the DTC fails especially when the density remains the same in almost all the zones for both the scripts.
When ambiguity increases, a sub-classifier called the Horizontal Transition (HT) has been employed to further classify the ambiguous TBV generated. When this Horizontal Transition procedure has been implemented and tested over the images without DTC, this procedure is able to classify nearly 80% of the characters in both the scripts. Some of the characters in English (W, w, M, .) and Tamil (ð, í, ñ, #, ê, etc…) are contradictory to this assumption.

Therefore, when both the classification procedures have been employed, an excellent performance has been recorded in the experiments. Integration of both the classifiers is able to classify 97% of the characters in the Tamil character set and 90% of the character set in the English script. In the Tamil character set, characters such as ð, í, «, £, (4/124) cannot get discriminated since they are sparsely dense and their horizontal transition rate is about two, whereas in English script, characters such as M, W, w, etc, (5/52) fail the condition, since their density is spread out in almost all the zones as well as their horizontal transition exceeds two.

5.6.3 Comparison of SFBSR with Previous Local Approaches

Here, the performance of the SFBSR at the word level has been compared with the script identification rate of the existing local (spatial) approaches available for discriminating the Tamil and English scripts at the word level. The SFBSR produces a better performance than the existing local approaches.

Local approaches available for Bi Script determination between English and Tamil such as: a) Aspect Ratio (Tan et al 1999), b) Spatial spread (Dhanya et al 2002), c) Structural Features and d) Water Reservoirs have been taken into consideration for performance analysis with the SFBSR at the word level. In Spatial spread analysis, it is assumed that if the pixel concentration in
the middle and descender zones together exceeds the ascender and middle, it is treated as Tamil otherwise it is English. In Aspect Ratio, it is assumed that the width of the word images is greater in the Tamil script than in English. In Structural features (Pal and Chaudhury 1999), the number of horizontal black runs is considered more for the Tamil script and the number of vertical lines in the ascender area is considered to be more for English. In Water Reservoirs (Pal et al 2003), the number of left reservoirs is considered to be more for Tamil images than for English and this is tested at the word level here. Table 5.6 shows the comparative accuracy rate of the SFBSR with Aspect Ratio, Spatial spread, Structural features and Water Reservoirs.

**Table 5.6 Recognition Accuracies of previous local approaches**

<table>
<thead>
<tr>
<th>Actual</th>
<th>Actual</th>
<th>Aspect Ratio</th>
<th>Spatial Spread</th>
<th>Structural Features</th>
<th>Water Reservoirs</th>
<th>SFBSR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tamil</td>
<td>English</td>
<td>Tamil</td>
<td>English</td>
<td>Tamil</td>
<td>English</td>
</tr>
<tr>
<td>Tamil</td>
<td>53%</td>
<td>47%</td>
<td>73%</td>
<td>27%</td>
<td>89.5%</td>
<td>10.5%</td>
</tr>
<tr>
<td>English</td>
<td>39%</td>
<td>61%</td>
<td>29%</td>
<td>71%</td>
<td>23.7%</td>
<td>76.3%</td>
</tr>
</tbody>
</table>

It is observed from the results (Accuracy, Script Recognition rate) in Tables 5.6 and 5.7, that the Spatial spread technique gets reduced in both English and Tamil scripts due to the assumptions of the ascender, middle and descender zones. In Spatial spread, Accuracy and Recognition rate of the Tamil script gets reduced due to the existence of a group of characters (ð,£, ðα, ð, ð£, ðα, etc…) in Tamil occupying the middle and ascender zones and wrongly being classified as English. In addition, the accuracy of the English script also gets affected variably in certain groups due to the dominance of cursive, descender dominant characters (g,y, q,…). In the Aspect ratio technique, the accuracy of Tamil and English word images is poor, due to the occurrence of small word images in Tamil and wider word images in English. (The width of the word image increases proportionate to the font size).
Table 5.7 Script Recognition Rate of previous local approaches

<table>
<thead>
<tr>
<th>Actual</th>
<th>Aspect Ratio</th>
<th>Spatial Spread</th>
<th>Structural Features</th>
<th>Water Reservoirs</th>
<th>SFBSR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hit Rate</td>
<td>Miss Rate</td>
<td>Hit Rate</td>
<td>Miss Rate</td>
<td>Hit Rate</td>
</tr>
<tr>
<td>Tamil</td>
<td>63%</td>
<td>37%</td>
<td>71.4%</td>
<td>29.6%</td>
<td>89.2%</td>
</tr>
<tr>
<td>English</td>
<td>68%</td>
<td>32%</td>
<td>61.4%</td>
<td>38.6%</td>
<td>74.4%</td>
</tr>
</tbody>
</table>

Also, in Structural features, short Tamil word images having less number of transitions and long English word images having more number of horizontal transitions reduce the accuracy. Also the occurrence of Tamil characters with Vertical lines in the Ascender zone (ê¤, ì¤, ï¤, ð¤, ñ¤, ò¤, õ¤, ÷¤, ø¤, ù¤, etc…) reduces the identification rate of the Tamil script and the accuracy of the English script. In Water Reservoirs, English word images with the occurrence of more number of left reservoirs (a, x, z, s, X, S) lose the identification rate of English and the accuracy of Tamil. Also the occurrence of Tamil word images without any reservoirs such as Ü, Ŷ, ß, â, ′, Ç, ð, õ etc… leads to the loss of accuracy of English. Structural Features and Water Reservoirs could work well with the assumptions of horizontal transition and left reservoir occurrence at the line level, but accuracy fails at the word level.

It is evident from Figure 5.17 that the SFBSR has passed through all the above mentioned levels and clearly discriminates both the scripts except on a few occasions. The SFBSR technique does not rely on the heuristic which occurs for the entire line or word. Moreover, this analyzes the spatial spread out of the total character along its width and height in every corner of the character area, instead of analyzing the pixel concentration in a particular area (ascender and descender zones), irrespective of font faces and sizes.
Figure 5.17  F-Score of various Script Recognition Techniques

Even though, the performance of the SFBSR is better than that of the other two techniques, this technique fails in certain situations due to non-discriminable characters, identical characters and character segmentation and characters which possess deviated density. Since the SFBSR identifies the script of the word image using its initial character, the efficiency of character segmentation is mandatory. If a touching character occurs as an initial one, the probability of classifying it as a Tamil script is more and if an English character occurs in that position, it would be misconceived, which reduces the recognition rate of English.

At the level of the training samples, the miss rate produced in this methodology is due to the identical (L, i) and non-discriminable characters (M, W, w, i, ð) in both the scripts. In non-discriminable characters, the first three letters are classified as Tamil and the rest are classified as English. This is due to the uniformity of density in both the scripts. In future, non-discrimination could be resolved if more than one character of the word is considered for recognition.
In this thesis, ANOVA, common statistical test which examines whether there exists, a significant difference between at least one of the group means has been used. Here, the ANOVA test was performed over the Accuracy and Script Identification rate of various local approaches, and the test rejected the null hypotheses ($p<0.03$) by stating that a significant difference arises between the sample of groups.

5.6.4 Comparison of SFBSR with Other Local Approaches

Here, the SFBSR has also been compared with other Script Recognition techniques available for separating the Han and Latin scripts (spatial approaches) such as Upward Concavity (Spitz 1994, 1997) and Vertical Cut Vector (Lu and Tan 2008). These two techniques have been applied to our test samples, (i.e.) Bilingual document images (Tamil and English), at word level and their efficiency of discrimination with the SFBSR has been compared.

In Upward Concavity (Spitz 1997), when two neighboring runs of black pixels on a scan line are connected below by a run, an upward concavity is formed. Based on the literature, it is assumed that the distribution of upward concavity is more concentrated at the base of each line for the English script whereas it is concentrated in the middle and the descender for the Tamil script. In the Vertical cut vector (Lu and Tan 2008), scanning from top to bottom, a vertical component cut is detected when a vertical scan line passes through the centroid of the character. To perform the vertical cut distribution, each text line is divided into three equidistant zones, namely, the upper zone U around the x line, the middle zone M around the middle line, and the lower zone L around the base line. Later, the number and positions of the vertical component cuts are characterized by using a component vector of dimension 32, where the first eight and the following 24 vector elements record the number and the position of the vertical component cuts. It is observed that the
number of vertical component cuts is more in the Tamil script when compared to English. It is also observed that more number of Vertical cuts occur in the Middle and the Ascender zone for the English script whereas for the Tamil script the vertical cuts occur in the Middle and Descender zones.

Table 5.8 shows the performance of SFBSR with Upward concavity and Vertical cut vector techniques in discriminating Tamil and English scripts.

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Tamil</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upward Concavity</td>
<td>82%</td>
<td>68%</td>
</tr>
<tr>
<td>Vertical Cut Vector</td>
<td>84%</td>
<td>71%</td>
</tr>
<tr>
<td>SFBSR</td>
<td>96%</td>
<td>93%</td>
</tr>
</tbody>
</table>

Both the techniques can work well for the separation of the Han and Latin script but this is script dependent. It is obvious from the Table 5.8 that the Upward concavity and Vertical Cut vector technique cannot be directly applied for the discrimination of scripts in bilingual document images (Tamil & English). In upward concavity, accuracy falls considerably for English since many of the concavities tend to be found at the baseline for Tamil. (Many Characters such as ã, ê, ñ, ç, ð etc. have concavities at baseline). The accuracy of Tamil falls due to the existence of upward concavity for the English letters in the descender zones such as g, j etc. Another problem arises when the characters in both the scripts do not have upward concavity. In the Vertical cut vector, since some of the Tamil words possess the same number of Vertical cuts as in English and vice versa, the accuracy of Tamil and English scripts gets reduced.
5.6.5 Performance Over Noisy Images

In this thesis, the various qualities of document images has been considered for testing, such as images affected by Salt and Pepper noise and Distorted images (letters distorted due to low resolution) apart from the good quality images, to evaluate the Script Recognition rate and it has been recorded in Table 5.9. This shows that the SFBSR shows a good performance even when the quality of the image degrades.

Table 5.9 Script Recognition Rate across the quality of images

<table>
<thead>
<tr>
<th>Quality</th>
<th>SFBSR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tamil</td>
</tr>
<tr>
<td>Images of Good Quality</td>
<td>94%</td>
</tr>
<tr>
<td>Images with Salt and Pepper Noise</td>
<td>94%</td>
</tr>
<tr>
<td>Distorted Images (Low Resolution)</td>
<td>93%</td>
</tr>
</tbody>
</table>

Not many variations arose when the images with salt and pepper noise are introduced, since it is removed through median filters properly. But variations in performance arise considerably when distorted images are tested (low resolution with more noises between the characters). Performance degradation is negligible for the Tamil script because the presence of noise between the characters does not affect the assumptions about the nature of the script. But in the case of the English script, the presence of noise inside the character and between two characters makes the characters to touch each other and get reported as a Tamil script.

5.6.6 Extension to Other Scripts

The SFBSR could also be extended to other scripts with a few assumptions or heuristics about the nature of the script. When this is applied
to numerals, the TB values get generated without creating a conflict with the English and Tamil characters. Therefore, to recognize the Hindi script, as an extension of the SFBSR, horizontal orientation of black pixels in the first three zones has been considered in this thesis. It is concluded that if the first three zones occupy black density in a horizontal orientation, it would be a Hindi script, since the Hindi letters contain a headline. The performance of the Hindi script recognition along with Tamil, English and Numerals recognition at the word level for sample word images has been recorded in Table 5.10.

### Table 5.10 Performance of SFBSR in Tri Scripts

<table>
<thead>
<tr>
<th>Script</th>
<th>Number of Words Tested</th>
<th>Tamil</th>
<th>English</th>
<th>Hindi</th>
<th>Numerals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tamil</td>
<td>500</td>
<td>482</td>
<td>6</td>
<td>12</td>
<td>-</td>
</tr>
<tr>
<td>English</td>
<td>300</td>
<td>10</td>
<td>285</td>
<td>-</td>
<td>5</td>
</tr>
<tr>
<td>Hindi</td>
<td>200</td>
<td>10</td>
<td>-</td>
<td>190</td>
<td>-</td>
</tr>
<tr>
<td>Numerals</td>
<td>50</td>
<td>-</td>
<td>6</td>
<td>-</td>
<td>44</td>
</tr>
</tbody>
</table>

From Table 5.10, it is observed that a few of the Tamil characters (ā, £, œ..) which have higher density in the first three zones get misclassified as Hindi. In the Hindi script, characters which do not have a headline in the initial zones but with equal spread over the character area get misclassified as Tamil. In addition to this, a certain amount of conflict arises between English and numerals such as the numeral 6, 8 and 0 getting classified as English, (due to the generation of same TBV and HT as with English characters) and the English letters “I” gets classified as numeral and M, W being classified as Tamil. The SFBSR has also been tested with Malayalam scripts with an additional criterion. It is assumed that if character holes and reservoirs exist in both the leftmost and rightmost zones of the characters, then the script would
be Malayalam and this criterion produced promising results in Script Recognition.

5.7  SUMMARY

This system provides a spatial features based script recognition framework with a novel approach, to identify the script of the word images in bilingual document images (Tamil and English) using the initial character of the word image. SFBSR produces better discrimination rates than those of the existing approaches. The next chapter discusses about the understanding of the word images whose script has been identified and the retrieval framework which enables the user to retrieve the desired documents.