Chapter 3

Zone Boundary Identification : Slope based approach

It is clear from the justification given in the previous chapter that it is mandatory to recognized components in the three zone separately. Hence, we need a robust algorithm which can determine the presence of a glyph in a particular zone and accurately locate the zone boundaries.

In this chapter we justify the need of a novel algorithm for documents of Gujarati script and present one of the two algorithms that has been formulated as part of this research work.

3.1 Mathematical Preliminaries

In the method proposed during this research work we have used concept of slope of a line to locate the zone boundaries.

Definition 3.1.1 *Let (x_1,y_1) and (x_2,y_2) be two points on a line l then the slope of the line, say S, is given by*

$$S = \frac{(y_2 - y_1)}{(x_2 - x_1)}$$

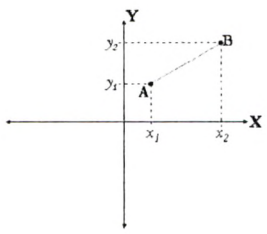


Figure 3.1: Slope of a Line

Figure 3.2 shows a text line and the coordinates of its pixels in image coordinate.

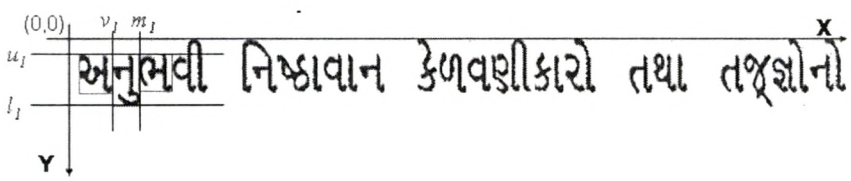


Figure 3.2: Text Line Coordinate System

We know that in the image coordinate system each pixel is referred by its row and column number. For example, (u_1,v_1). It should be noted from Figure 3.2 that when we superimpose this image coordinate system on to cartesian coordinate system, the row coordinate in image coordinate system becomes Y coordinate and column coordinate serves as X coordinate in the new frame work.

3.2 Zone Boundary Identification : State-of-the-Art

Gujarati is not the only script for which zone boundary detection is mandatory. Other Indo-Aryan scripts like Devanagari, Bangla and Gurmukhi also require zone identification for reducing the number of symbols to be recognized.
In [6], Veena Bansal et. al. showed an approach for identifying zone boundary for Devagnagari script. [10] documents efforts of B B Chaudhury and U. Pal for carrying out this task for Bangla. Both of these approaches use maxima in the horizontal projection [6] of a text line to decide the zone boundary. Mathematically, their logic tries to find out the cut around the pixel row(s) r such that,

$$hp[r] = \max_i hp[i] \quad \text{where hp[i] is horizontal projection.}$$

It is clear that the maxima is always located at the pixel row(s) corresponding to *shirorekha* in these scripts (see Fig. 3.3(a)).It may be noted from the Fig.

3.3(b) that horizontal projection of the Gujarati text does not have any prominent peak. Hence, it is not possible to apply the algorithms proposed for zone boundary identification for scripts like Devanagari and Bangla to Gujarati in the same form.

(a) Horizontal projection of Devanagari Text

(b) Horizontal projection of Gujarati Text

Figure 3.3: Horizontal Projection

It could be argued here, that instead of a peak at the header line, a trough in the horizontal projection may be used to detect the upper zone boundary. However. following are some of the instances in which this may not work to the required accuracy:

1. In the case where the number of modifiers is significantly large this trough will not be very prominent.
2. A slight misalignment of text may lead to cutting off of a significant part of a glyph in the middle zone (eg. a small arc at the top left corner of the letter /ya/) which can affect the recognition accuracy(in case of previously sited example of /ya/ the shape after removal of that curvature would be similar to alphabet /va/).
3. When the number of modifiers is less, the trough is shallow and hence almost undetectable.

Before this work started there were no documented efforts for detecting zone boundaries in documents printed in Gujarati text. It is also clear that we need to devise a novel algorithm to handle this task.

3.3 Proposed Approach

In order to overcome the problems posed by the above-mentioned situations, we devised a new algorithm to determine the zone boundaries.

3.3.1 Using Slopes of Lines Joining Top Left (Bottom Right) Corners of Connected Components

The pseudo code of the algorithm for detecting the zone boundaries is described below. It is observed that in most of the cases a line / word has more

number of base line component without upper / lower modifier than those without modifiers. This is used as basic assumption in this algorithm.

In our algorithm first we consider all the potential connected components (CC) within a text line and compute the slopes of all the imaginary lines that join top left corners of all possible pairs of CC. The row coordinate of the CC that are end points of the maximum number of lines having the least slope would identify the row of separation between the upper zone and the middle zone. Similar procedure for the bottom right coordinates of the connected components would indicate the location separating the middle and the lower zones. **Algorithm 3.1** gives step by step procedure for the same.

Algorithm 3.1 To Find Zone Separation Boundary

Input: Binarized Image of a line of Gujarati text

Output: Row numbers of the two lines that separate upper and lower modifiers from the middle zone.

Step 1: Extract the connected components in the line with the information about their bounding boxes,as in Fig. 3.4.

Step 2: For each pair of distinct connected components, compute the following:

1. Identify the Coordinates (u_1, v_1) and (u_2, v_2) of the top left corners of the bounding boxes of the two components. (see Fig. 3.4)
2. Identify the Coordinates (l_1, m_1) and (l_2, m_2) of the bottom right corners of the bounding boxes of the two components.
3. Find the absolute values S1 and S2 of the slopes of the lines connecting (u_1, v_1) to (u_2, v_2) and (l_1, m_1) to (l_2, m_2) (see Fig. 3.5)

$$S1 = \frac{|u_2 - u_1|}{|v_2 - v_1|} \tag{3.1}$$

$$S2 = \frac{|l_2 - l_1|}{|m_2 - m_1|} \tag{3.2}$$

Step 3: Identify the lines that give the minimum of slopes $S1$. Those lines that fall in the region between 15% and 40% of line height below the top of the text line are candidates for being considered as separators of upper zone from the middle zone. If there is more than one line that satisfies this criterion, choose the line that occurs maximum number of times as the zone separator.

Step 4: Identify the lines that give the minimum of slopes $S2$. Those lines that fall in the region between 15% and 40% of line height above the bottom of the text line are candidates for being considered as separators of lower zone from the middle zone. If there is more than one line that satisfies this criterion, choose the line that occurs maximum number of times as the zone separator.

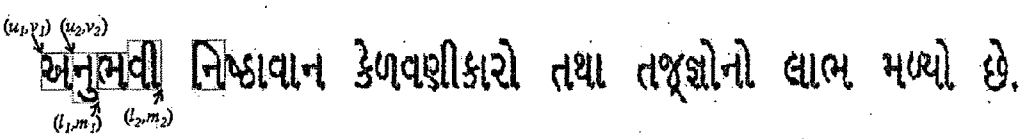


Figure 3.4: Bounding Box with Coordinates

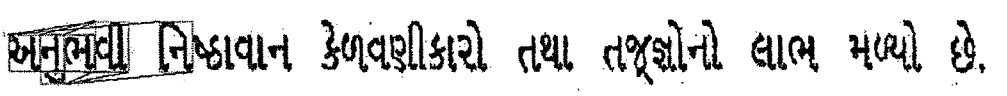
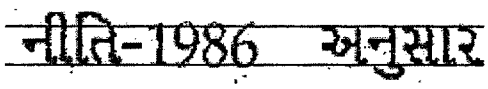


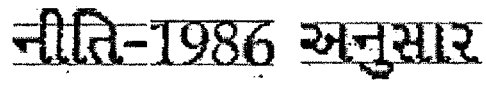
Figure 3.5: Line Segment Joining Top Left (Bottom Right) Corners of Bounding Boxes

Doing this process only at line level may not be sufficient, because it still leaves the possibility of errors in the situations where words are not horizontally aligned. Fig.3.6(a) illustrates this case for a sample line of text.

Due to this problem, the process mentioned above for a line, needs to be repeated for the connected components of individual words also, for determining the zonal boundaries. Here, there is a possibility of disagreement between the boundaries detected at word level and line level. In such cases, if a new location for any of the two separators is detected then those are considered to be valid, but if the word level execution removes a zone detected in the line level execution then the line level decision is considered to be valid. Fig.3.6(b) shows the correction after calculating the zone boundaries at word level.



(a) Line level zone detection(Notice over segmentation)



(b) world level improvement

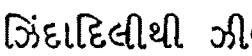
Figure 3.6: Zone boundary detection

Following table gives details about the testing :

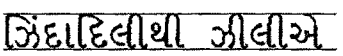
Table 3.1: Result of zone boundary detection using slope of imaginary lines

No. of Books	No. of Pages	No. of Lines	Success(No. of lines)
5	43	935	858(91.76%)

The failure case analysis revealed the fact that the basic assumption that a line would have more baseline component without modifier than those with modifiers on upper and lower zones is not valid in general and there are significant number of cases which do not satisfy this (see Fig. 3.7(a)). Hence, there is a need to improve this algorithm.



(a) Line Failing to Satisfy Assumption in Algorithm 3.1



(b) Result of Applied Algorithm 3.1

Figure 3.7: Limitation of Algorithm 3.1

3.3.2 Improved Zone Boundary Identification

In order to find root cause of the problem we took out cases where it was failing. They found to be satisfying one of the following :

- There is only one word in the line and all the base line components have modifiers in upper (lower) zone connected to it.(Fig. 3.8)

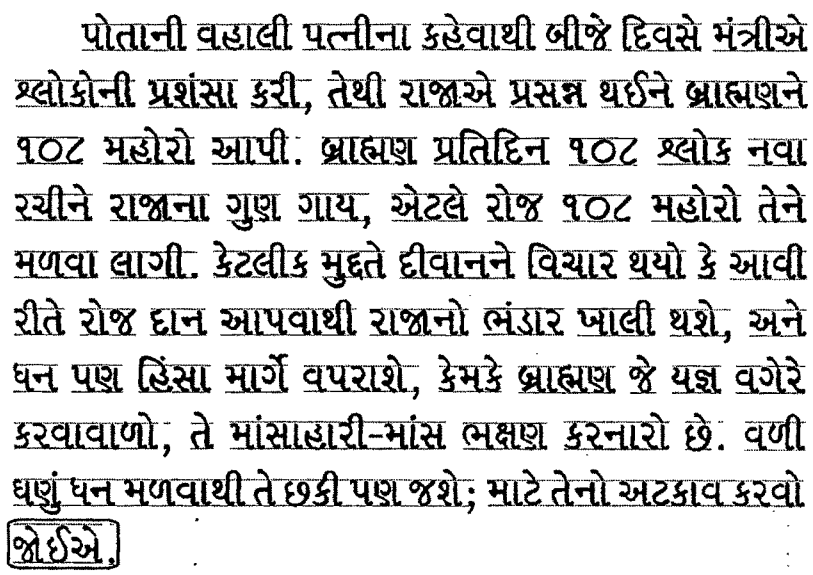


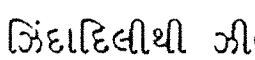
Figure 3.8: Failure Due to Only One Word

- None of the base line glyph has modifier and it also contains more than

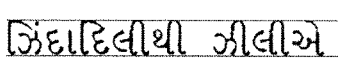
Table 3.1: Result of zone boundary detection using slope of imaginary lines

No. of Books	No. of Pages	No. of Lines	Success(No. of lines)
5	43	935	858(91.76%)

The failure case analysis revealed the fact that the basic assumption that a line would have more baseline component without modifier than those with modifiers on upper and lower zones is not valid in general and there are significant number of cases which do not satisfy this (see Fig. 3.7(a)). Hence, there is a need to improve this algorithm.



(a) Line Failing to Satisfy Assumption in Algorithm 3.1



(b) Result of Applied Algorithm 3.1

Figure 3.7: Limitation of Algorithm 3.1

3.3.2 Improved Zone Boundary Identification

In order to find root cause of the problem we took out cases where it was failing. They found to be satisfying one of the following :

- There is only one word in the line and all the base line components have modifiers in upper (lower) zone connected to it.(Fig. 3.8)

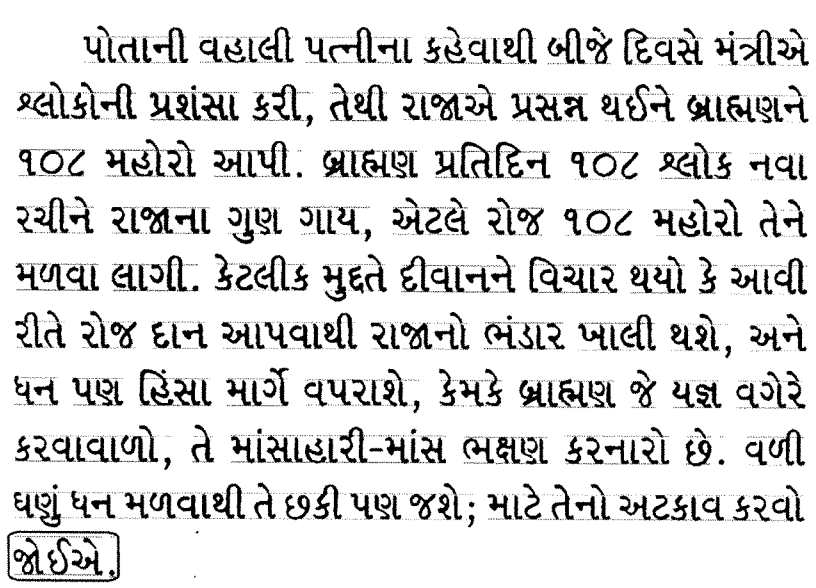


Figure 3.8: Failure Due to Only One Word

- None of the base line glyph has modifier and it also contains more than

one alphabets like /ga/, /Na/ etc. , where row coordinate of one of the components satisfy zone constraints mentioned in the algorithm above (Fig. 3.9)

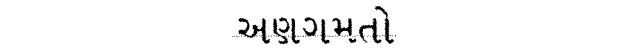


Figure 3.9: Failure Due to glyphs like /ga/ and broken glyph

- There are broken glyph(s) giving rise to the connected components such that it results in to situation described in second point.(Fig. 3.9)

It is also noticed that horizontal projection profile based algorithm for detecting zone boundary is performing much better in such cases. Therefore, we utilized the complimenting nature of these two algorithms to overcome these failures to a significant extent and devised a hybrid algorithm.
Following are the broad steps of the new procedure for finding zone boundary between upper and middle zones. The details are given after that. Same can be repeated for finding the boundary between middle and lower zones.

1. Analyse line / word for deciding number of base line connected components with modifiers in upper zone. Let that number be NBC_u and NBC be total number of base line components.

2. If $NBC_u = 0$ or $NBC_u > NBC/2$, then use projection profile based **Algorithm 3.2** for zone detection else use **Algorithm 3.1**.

The challenge here is to find out how many base line glyphs in a text line / word have modifiers attached to them. Here again we make use of an observation that if $NBC_u = 0$ or $NBC_u > NBC/2$ then more than half of the imaginary lines passing through top left corners of all possible pairs that have minimum slope will be near the the first pixel row of line / word (Fig. 3.10).

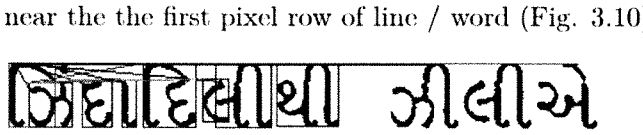


Figure 3.10: Location of Imaginary Lines with Minimum Slopes

Hence, the algorithm mentioned above can be verbally expressed in more elaborate way as follows :

1. Find out slopes of the lines joining the top left of all possible pairs of potential connected components of lines / words.

2. Find out row coordinates of the top left corner of the connected components through which lines with the minimum slope passes.

3. Compute number of lines that passes through each of these points.

4. If the point through which more than half of these lines passes is near the top boundary of line then $NBC_u = 0$ or $NBC_u > NBC/2$.

5. If $NBC_u = 0$ or $NBC_u > NBC/2$, then use projection profile based **Algorithm 3.2** for zone detection else use **Algorithm 3.1**.

The detailed projection profile based algorithm can be given as follows:

Algorithm 3.2 To Find Zone Separation Boundary

Input: Binarized image, IM_{ij} of a line/word of Gujarati text with m pixel rows and n pixel columns

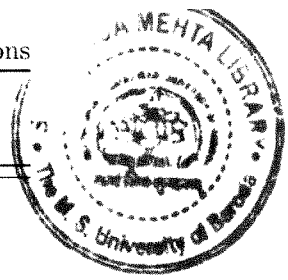
Output: Row numbers of the two lines that separate upper and lower modifiers from the middle zone.

Step 1: Compute horizontal projection

$$hp[i] = \sum_{j=1}^{n} IM[i][j], \quad \forall i = 1, 2, \ldots, m. \tag{3.3}$$

Step 2:

$isDecreasing \leftarrow false$
$cutLevelFraction \leftarrow 0.4$
$maxHPRow \leftarrow 0$
$maxHP \leftarrow 0$
for $i = 1$ to m **do**
  if $hp[i] > maxHP$ **then**
    $maxHP \leftarrow hp[i]$
    $maxHPRow \leftarrow i$
  end if
end for
for $i = maxHPRow$ to 1 **do**
  if $hp[i] > hp[i-1]$ **then**
    $isDecreasing \leftarrow true$
  else
    $isDecreasing \leftarrow false$
  end if
  if $isDecreasing = false$ **then**
    if $hp[i-1] \leq (cutLevelFraction * maxHP)$ **then**
      if $i - 1 \geq m * 0.15 \quad AND \quad i - 1 \leq m * 0.30$ **then**
        $uCutRowHP \leftarrow i$
      else
        $uCutRowHP \leftarrow 0$



```
            end if
            break
        end if
    end if
    i ← i − 1
end for
cGradient ← hp[uCutRowHP] / hp[uCutRowHP+1]
for i = uCutRowHP + 1 to m do
    nGradient ← hp[i] / hp[i+1]
    if nGradient > cGradient then
        cGradient ← nGradient
    else
        if i ≥ m ∗ 0.15  AND  i − 1 ≤ m ∗ 0.30 then
            uCutRowHP ← i
        end if
        break
    end if
    i ← i + 1
end for
```

Fig. 3.11 shows the improvement of applying **Algorithm 3.2** on image shown in Fig. 3.7(a). The separating white line indicates the zone boundary.

Figure 3.11: Result of Applying Algorithm 2

The application of this algorithm on a sample data set improved accuracy by 2%.

3.4 Conclusions

It is clear from the discussion in this chapter that zone separation algorithms that works satisfactorily for the Indo-Aryan scripts like Devanagari and Bangla cannot be used readily for detecting zone boundaries in the documents of Gujarati script. A novel algorithm (**Algorithm 3.1**)based on slope of imaginary lines passing through top left and bottom right corners of the bounding boxes of all possible pairs of potential connected components is proposed. This algorithm assumes that in a text line / word there will be more baseline glyphs

without modifiers in upper / lower zones than those with modifiers and hence it fails in the cases where this assumption is not satisfied. A method is proposed to detect the exceptional cases and a hybrid algorithm which uses both slope of imaginary lines and horizontal projection is proposed to take care of majority of the cases. However, both these algorithm performs poorly in case where line / word detection algorithm gives incorrect boundaries or the text is not aligned at word level.