5 SEGMENTATION

The scanned text document is input to the preprocessing phase of OGHTR system transformed to binary image which is ready for segmentation process. To segment Gujarati text documents into isolated units i.e. core characters and diacritic marks is one of the objective of segmentation phase of OGHTR system and this research.

Segmentation subdivides an image into its constituent regions or objects. The level of detail to which the subdivision is carried depends on the problem being solved [65]. Segmentation accuracy determines the eventual success or failure of computerize analysis procedures. A rugged segmentation procedure brings the process a long way toward successful solution of imaging problems that required objects to be identified individually [65,72,92,120]. For this reason considerable care should be taken to improve the probability of accurate segmentation.

Based on literature survey, handwritten text line segmentation approaches can be categorized according to the different strategies used. These strategies are projection based, smearing, grouping, Hough-based and other approaches. Related work can be found in [21,75,90,94,96,129-132].

The projection based approaches are very much effective for machine printed documents as it generates clear mountain area in histogram due to proper space between lines. But in case of handwritten documents humans tend to write in curved and skew lines especially while writing on paper without lines. The arrangements of text lines in handwritten documents are not in proper format as a result segmentation method based on projection not succeeded.

The Hough-based methods are good but required high computation also it is not good with skew of text lines varies along its width [95,133]. The grouping/connected component based method is effective as it segments all components as separate unit but require great care for assignment of component [21,57,100], especially for Indian script due to large character set and modifiers.

It is clear from the discussion given in section 1.5 and 2.1 that it is mandatory to find appropriate approach for segmentation of Gujarati text document. Based on discussion given in section 1.4, the Gujarati text lines can be segmented by tracing characters. The OGHTR system need algorithm which is suitable for given pattern i.e. handwritten Gujarati text.

The segmentation phase of OGHTR divides each group of segmented units into core characters and diacritic marks, so that it can process separately for recognition. Based on above discussion the designed segmentation algorithm fulfils following three objectives:

1. Segmentation of text into isolated unit i.e. component.
2. Finding beginning of each line so that line can be traced.
3. Classification of isolated unit i.e. core characters, diacritic marks or conjuncts.

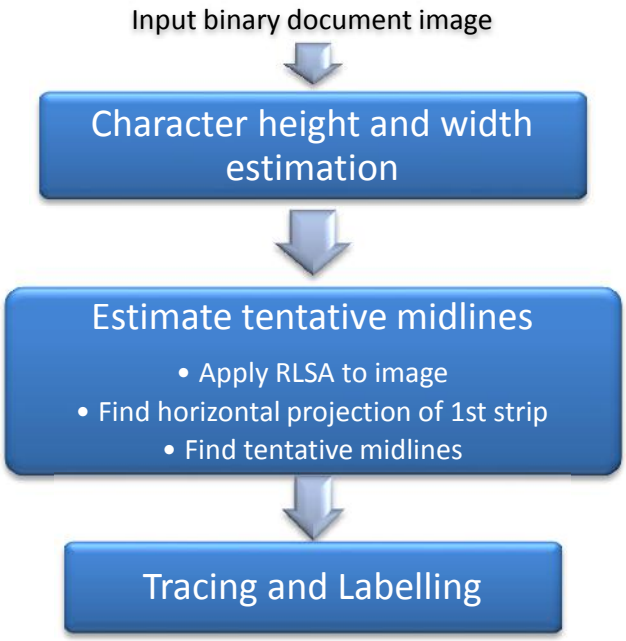


Figure 19 Block diagram of segmentation process

To satisfy above stated objectives combination approach is used for segmentation in OGHTR system. To satisfy first objective connected component analysis method is used. To fulfill second objective combination approach of

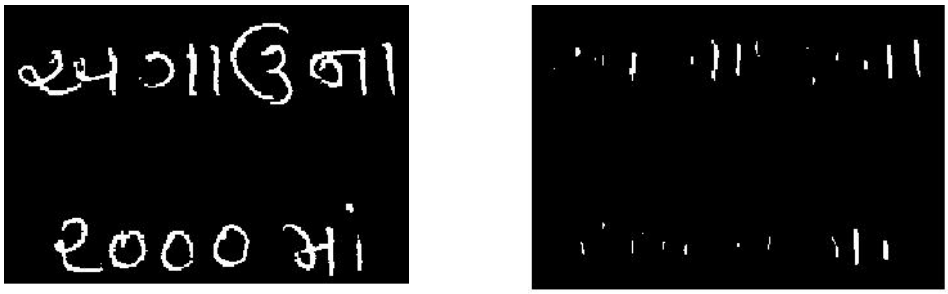
histogram projection method and RLSA is used to locate line, which is used to determine first component. And to categorize component parameters are decided based on various parameters like character width, height and location is used.

The segmentation phase of OGHTR is divided into three steps i.e. 1) character height and width estimation, 2) estimate tentative midline and 3) tracing and labeling.

The first two phases consist of finding parameters which are character height, character width, number of lines and its related parameters. In the third phase connected components (CCs) belongs to line are traced and assigned label as core character, diacritic mark, or conjuncts using parameters. The major steps of segmentation phase are shown in Figure 19. The next successive sections describe all the steps in the detail.

5.1 CHARACTER HEIGHT AND WIDTH ESTIMATION

In the first step of segmentation phase, connected components (CCs) of binary image are extracted. The connected component labelling method is used to give unique label to connected group of pixels in an image so that analysis can be carried out for each group.



(a) **(b)**

Figure 20 Effect of erosion with vertical line structuring element

After extracting CCs, the ACH and ACW are calculated based on average height and width of the CCs. For better estimation small CCs which are fewer than stroke width (SW) pixels are not considered for calculating height and width.

The stroke width is used to give estimation of stroke in terms of number of pixels. The stroke width is depends on pen tip and ink which is used to write. The CCs with less than stroke width in terms of height and width considered as noise. Hence, they are not considered in estimation as well as ignored during recognition process.

The image morphology is used to find value of SW. As we want to extract line shapes from the text document image the vertical line structuring element is used to erode an image. To find SW, after applying connected component extraction, the average width of CCs are calculated. A part of binary image is shown in Figure 20 (a) and effect of erosion using line structuring shown in Figure 20 (b). The step for finding SW is given below:

Algorithm:	getStrokeWidth
Input:	Binary Image
Output:	SW

```
1.  width=0, count=0
2.  Erode binary image with vertical line structuring element
3.  Apply connected component analysis on eroded image
4.  FOR each component(CC) in labeled image
5.          width= width + CC_width
6.          count=count+1
7.  END FOR
8.  SW = width / count
9.  END
```

For finding ACH and ACW, connected component (CCs) labelling is applied on preprocessed binary image. The CCs are then extracted from labelled binary image and analyse for their size. Each CCs are compared with stroke width (SW) and CCs size greater than SW are considered for average calculation. The step for estimating ACH and ACW is given as below:

Algorithm:	EstimateCharacterSize
Input:	Binary Text Image, SW
Output:	ACH, ACW

```
1.  Label the binary image
2.  Initialize CH and CW to zero
```

```
3.    Apply connected component analysis on binary labeled image
4.    FOR each component(CC) in labeled image
5.          IF CC_height > SW AND CC_ width  > SW THEN
6.                count=count + 1
7.                CH = CH + CC_height
8.                CW = CW + CC_width
9.          END IF
10.   END FOR
11.   ACH = 0.9*(CH / count)
12.   ACW = 0.9*(CW / count)
13.   END
```

5.2 ESTIMATE TENTATIVE MIDLINES

The segmentation approach of OGHTR system is based on connected component, we need to assign each CCs to its respective line. To assign CCs to its line require location of each line. In this step of segmentation phase the middle Y location of each line is estimated which is called midline. Once midline is available it is used in next step of segmentation for tracing and labelling CCs to text line.

For finding midline of each line the text document image is virtually visualised into N number of strips. Out of these N strips, horizontal projection is computed for first strip for line detection. The selection of N is based on text data area. Generally, a Gujarati handwritten document contains average 5 to 6 words in a line and average characters in word is 4 [82]. By empirical study we decided the value of N is as 4 so at-least 1 word participate in the first strip.

The reason to process for first strip only is due to Gujarati text lines written from left to right direction and at the beginning skew in the lines is rare. Also, we are interested in line location at the beginning.

However, the horizontal projection of first strip contains few characters; sometimes it does not produce clear mountain and valley points, which is used to detect lines. Therefore, to strengthen the horizontal histogram projection, RLSA is applied to image before the computation.

The RLSA is applied to binary image in horizontal and vertical direction. The effect of applying run-length is it increases foreground points if another foreground point presence within the given run-length. After applying horizontal run-length to an image, increases pixels in row and vertical run-length increases pixels in the column.

The overall effect of applying horizontal run-length is it connects on pixels in the horizontal direction which causes connection between the characters. The size of horizontal run-length is taken as ACW due to a character is followed by other character within character width space.

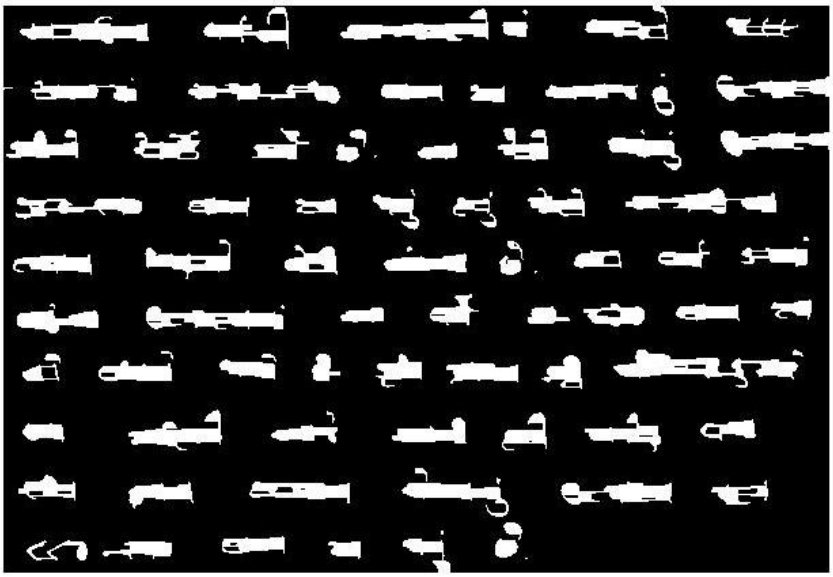


Figure 21 Document image after horizontal and vertical run-length

The effect of applying vertical run-length is it connects pixels in the vertical direction which cause connection of diacritic mark with the character. Increase in vertical run-length size may connect lines which is not required. Hence, the size of vertical run-length is decided less as one fourth of ACH. The binary document image is shown in Figure 21 after applying horizontal and vertical run-length.

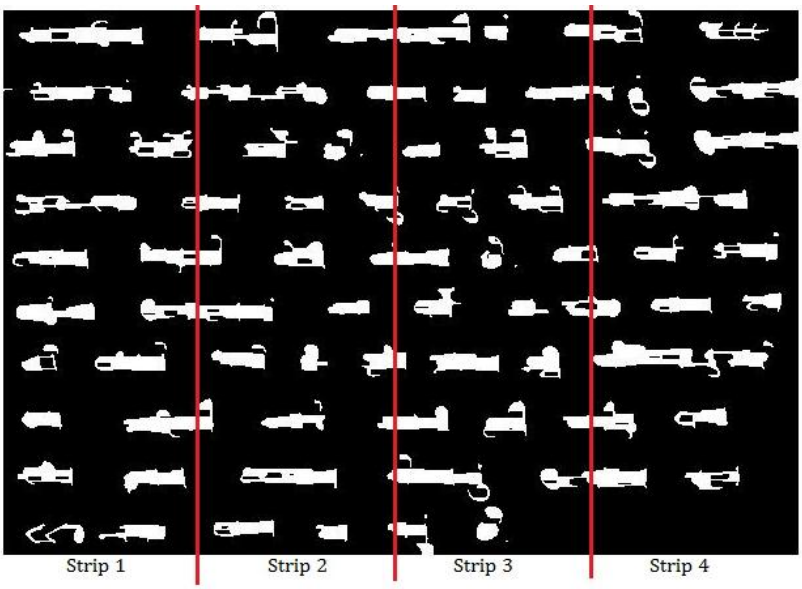
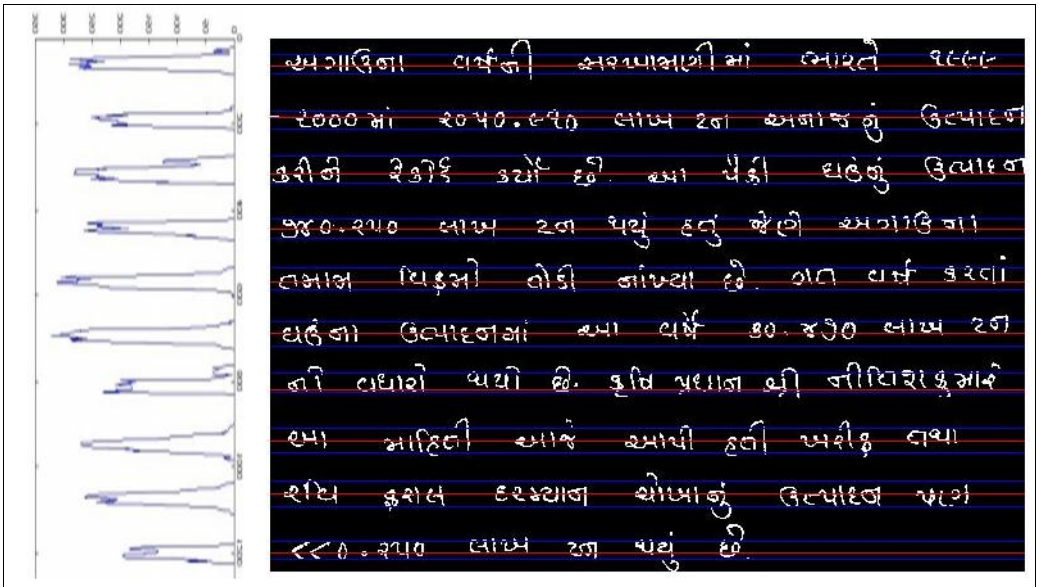


Figure 22 Document image divided into 4 strips

After applying RLSA in both the direction the binary image is divided into N (N=4) strips as shown in Figure 22. Next, out of N strips, horizontal projection histogram of first strip is calculated. In horizontal histogram mountain region represents location of each line. The start point, stop point and peak of mountain are recorded.



(A) (B)

Figure 23 (A) Horizontal Projection histogram of 1st strip (B) Tentative line detection

The Figure 23 (a) shows horizontal projection histogram of first strip and detection of midline in Figure 23 (b). The midline location is calculated using peak of the mountain where maximum pixel frequency is obtained. The midline location is shown in red colour horizontal line in Figure 23 (b). The difference between stop point and start point gives height of line. The height of line is compared with ACH to retain line, otherwise rejected. In Figure 23 (b) blue line shows begin and end of line location.

Note that segmentation algorithm required only beginning of line to determine first CCs belongs to that line. Rest of the CCs are going to labelled based on tracing CCs one followed by other. The algorithm for estimating tentative line is given below, based on above discussion.

Algorithm:	EstimateLines
Input:	Binary Image, CH, CW
Output:	Array containing lines parameter

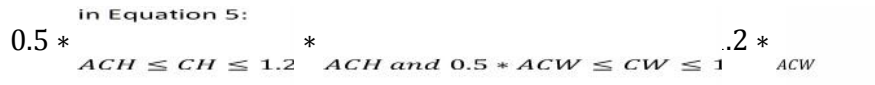
1. Apply RLSA in horizontal direction to binary image with run-length of ACW
2. Apply RLSA in vertical direction to binary image with run-length ¼ of ACH
3. Calculate horizontal projection of 1st strip out of 4 strip of image
4. Find peak, valley points and height of each mountain
5. Use threshold of ACH to select tentative lines and store its peak and valley points
6. END

5.3 TRACING AND LABELLING

In the third step of segmentation phase of OGHTR system, objective is to assign label to each component (CCs) with line and defining category as core character, and diacritic mark. The parameter obtained by previous two steps taken as input to this step. These parameters are SW, ACH, ACW and line points vector containing midline location for each line.

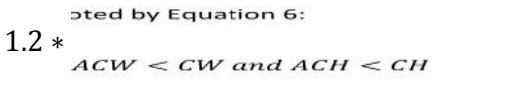
In this step labelled binary image is used for assignment of component to line. The estimated midline is used to find the first component within that line. The process of tracing lines starts with finding first CCs from the line. The detail discussion on it is in section 5.3.1.

During the tracing process each CCs is measured and subdivided into different groups. These subdivisions are based on ACW, which is denoted as core characters, and diacritic marks. The core character group expected to contain all CCs which correspond to majority of the character with size which satisfy constraint given in Equation 5:



Equation 5

Where, CW and CH denote the CCs width and height respectively. The CCs left behind are small in size belongs to diacritic mark and big in size are belongs to join characters (conjuncts and character joins with diacritic marks). The constraints for these CCs are denoted by Equation 6:



Equation 6

For grouping of diacritic mark, CCs position and its size is considered. The small CCs which are left and right of the core character belongs to this category. The CCs at the top and bottom doesn't require checking with constraints. Further discussion on diacritic mark is given in section 5.3.2 and 5.3.4 to 5.3.6.

The tracing process of CCs belongs to line is processed for all lines of the document. The number of lines is already calculated in previous section during estimate tentative lines. The tracing process starts with the first CCs of the line, marking as current CCs. As per our discussion on construction of Gujarati Akshara consists of diacritic marks at its right, top and bottom, we process for looking CCs which are at right, top and bottom of the current CCs.

After finding diacritic marks, using current CCs we can trace the next CCs using the mid of current CCs with the threshold of ACW. The detail discussion on tracing next CCs is in 5.3.7. Likewise, all CCs of lines are traced up to the end of line.

While tracing next CCs we also track the distance between the CCs. If distance between CCs is more than ACW threshold then we can recognize it as space

punctuation mark i.e. word mark. The benefit of tracing next CCs, eliminate the need of internal line skew corrections.

Now, here we need to tackle situation were first CCs of character may be left diacritic mark that is "િ – harsva-ajju". It means if we extract CCs from the left with reference to midline then it can be either left diacritic marks (િ – harsva-ajju) or core character. If it is left diacritic mark then labelled it and we can find next CCs as core character. The detail discussion on left diacritic mark process is given in section 5.3.2.

Once core character is labelled, we further process for finding bottom, right and top diacritic mark in sequence. Here, we eliminate searching of diacritic mark as per Gujarati language rules. For example, if left diacritic mark found then we need to process for only top diacritic mark which can be either anuswar(ં ṁ) or curved upward dash (half form of "ર ra rə" as first character in conjunct). Similarly, if lower diacritic mark found, then process for only upper diacritic marks. Furthermore, if right diacritic mark is ી (dīrgha-ajju) then processing upper diacritic marks only.

Here we need to consider the situation in which right diacritic mark ી (dīrgha-ajju) may attach to character as shown as in the Figure 24.

Figure 24 Joint right matra ી (dīrgha-ajju) with characters "વ va ʋə ", "ર ra rə", "બ ba bə ", and "ન na nə"

The detail discussion on segmentation of right diacritic mark is given in section 5.3.5. The sections 5.3.1 to 5.3.7 describes above tracing and labelling procedure in detail. During the tracing process once the CCs is selected thinning and cropping is applied before processing.

5.3.1 Finding first component

As given in section 5.2, we recorded line parameters which give number of line and midline of each line. The tracing CCs is performed for each of the line one by one. The line tracing starts with finding first CCs belongs to that line i.e. current line. To find first CCs, minimum Y and maximum Y value of CCs are compared with midline value.

Set of CCs are collected whose minimum Y value and maximum Y value range covers midline Y value. These components are belonging to the current line. Now this set of CCs is sorted on its minimum X value in ascending order. The first CCs in the order set is the first CCs of line.

The first CCs then compared with core character constraints and labelled it as core character if satisfy. There are two cases when it does not satisfy constrains, if it is 1) left diacritic mark or 2) conjunct. The processing of left diacritic mark is discussed in next subsection. For conjunct, CCs is labelled as conjunct and further process is carried out in sequence.

The steps for finding first component are as follows:

Algorithm:	FindFirstComponent
Input:	CCs array, lines point array
Output:	First component id
1.	Initialize midline variable from lines point array for current line
2.	Find all CCs which have its row value covers midline
3.	Sort all selected components on its column value
4.	Take the first component from the sorted list
5.	END

Even though extracted CCs satisfy the condition for core character it will be check for left diacritic mark as given in next section 5.3.2. If it is left diacritic mark, further steps are skipped and next component is searched as given in 5.3.7. Also, left diacritic mark added to the recognition result with its code. Later in text generation process left diacritic mark is placed after the character.

5.3.2 Processing for left diacritic mark

The extracted component is preprocessed by applying morphological thinning followed by cropping before processing. The shape of left diacritic mark helps to identify it. The left diacritic mark shape has vertical line and curve at the top toward bottom side. The cropped component is then segmented into two parts based on ACH from the bottom. By checking the size of bottom part as vertical line and top part having bottom reservoir we can detect it is as left diacritic mark. The discussion on line detection is given in 6.2.2.4 and bottom reservoir is given in section 6.2.2.6.

If extracted component is not satisfy the constraints of left diacritic mark, it will be process further as given in next section.

5.3.3 Processing for core character

To make sure component is core character, it is checked using constraints using Equation 5. The constraint is not satisfied in two cases 1) the character is attached to diacritic mark or 2) it is conjunct character. Here if height of component is more than estimated height then component is attached with upper or lower diacritic mark. To extract lower or upper diacritic mark, we identify current character middle is above or below of the midline of the previous character. If it is above the midline than upper diacritic mark is attached otherwise lower diacritic mark is attached.

To extract upper diacritic mark, we segment character from base up to height of the estimated character height while for extracting lower diacritic mark we segments character from top up to estimated character height. Now character is labelled as core character and diacritic mark as lower diacritic mark.

As soon as component is labelled as core character it will be processed for extended line removal step and then feed in for recognition using trained model obtained by neural network. The discussion on neural network is given in section 6.2.4 and core character recognition is given in 6.2.4.4.

5.3.4 Processing for bottom diacritic mark

Bottom diacritic mark which is attached to the character is segmented during the previous step processing for core character. For detached bottom diacritic mark, the core character bottom base line is used as reference. The CCs which is near to base line in bottom part is labelled as bottom diacritic marks. There are possible three bottom diacritic marks which are ◌ુ (harsva-vararu), ◌ૂ (dirgha-vararu), and ◌ૃ (ru), which can be easily identified through its structural properties. The discussion of diacritic mark recognition is given in section 6.1.

5.3.5 Processing for right diacritic mark

To extract right diacritic mark symbol, CCs at the right side is extracted if it is within threshold value to the core character. The value of threshold is based on estimated character width value. The possible value for right diacritic marks are "◌ા (kano)", "◌ી (dirgha-ajju)" and combination of "◌ા (kano)" ("◌ા ો, ૉ" – *kāno ek mātra* and "◌ા ૌ əʊ" – *kāno be mātra*) characters. Both can be easily identified through its structural properties.

If character does not have right diacritic mark then we need to check for joint right diacritic mark ◌ી (dirgha-ajju). To identify and segment joint right diacritic mark, bottom water reservoir technique is used. The right diacritic marks attached to extreme right side of the character forms deep bottom reservoir up to character height. The only character which has deep bottom reservoir of character height at right side is character "ળ (ḷa ḷə)". To identify join right diacritic and character "ળ (ḷa ḷə)" another structural property of said character is used. That is character "ળ (ḷa ḷə)" has top reservoir of character height. Other character do not exhibit this property for extreme right deep water reservoir.

To identify joint diacritic mark ◌ી (dirgha-ajju) first bottom reservoir is computed with the top extrema point for reservoir region, which can be identified by finding maximum y value point in the region. See in Figure 25 to Figure 28.

As discussed earlier the characters which have joint right diacritic mark having bottom reservoir at left side with top extrema point of character height. We can compute this by finding angle between bottom right point to top extrema point. If the angle is between given threshold value between 90˚ to 110˚ then it is considered as right diacritic marks. The value of threshold angle is decided based on experiments.

Figure 25 Character બ (ba bə) with joint *dīrgha-ajju* "બી"

Figure 26 Character ન (na nə) with joint *dīrgha-ajju* "ની"

Figure 27 Water reservoir for "બી"

Figure 28 Water reservoir for "ની"

To remove right diacritic mark from the character following steps are followed:

Algorithm:	ExtractRightDiacritic
Input:	Component image with right diacritic
Output:	Component image without diacritic

1. Perform bottom water reservoir
2. Get top extrema point
3. Compute angle between bottom right point to top extrema point
4. Crop the character upto the column of top extrema point
5. END

The result of removing right diacritic mark from the character is shown in Figure 29 and Figure 30. The other character can be appeared at right of the characters are punctuation marks. Identifying punctuation marks are not in scope so we are omitting it for recognition.

Figure 29 Character "બ (ba bə)" after removing *dīrgha-ajju*

Figure 30 Character "ન (na nə)" after removing *dīrgha-ajju*

While extracting right diacritic mark, it may possible that we can trace to left diacritic mark, which is belongs to next character. To tackle this we can check the width of CCs and based on it either retain or left for tracing in next CCs. That is if right diacritic mark width is more than SW then it is either "◌ી (dirgha-ajju)" or "િ◌ (harsva-ajju)". To identify position of top end point is compared with bottom point. The position of endpoint at left for "◌ી (dirgha-ajju)" and for "િ◌ (harsva-ajju)" at right.

5.3.6 Processing for top diacritic marks

To extract top diacritic mark symbol CCs at the top side of the core character are extracted, if they are within threshold value above the core character. The value of threshold is based on estimated character height value. The number of CCs may be more than one. In such case all CCs is processed for diacritic mark recognition. The possible value for top diacritic marks are ◌ે(ek matra), ◌ૈ(be matra), curved upward dash (half form of ◌ as ◌ˇ, i.e. conjunct ◌ ◌ે+ ◌ = ◌ઽ) and ◌ં(anusvar) characters. Further discussion on diacritic mark recognition is given in section 6.1.

5.3.7 Finding next component

After processing for diacritic mark, the midline variable change to mid row value of current core CCs. Based on midline value next CCs is extracted exactly in the

same way as we have processed for first CCs with the constraints on its column value.

Before processing further, the distance between CCs are computed. If it is more than the threshold value of ACW then marked as punctuation mark space. If next CCs not found then it is marked as new line character. Now processing again starts from the finding first CCs from the next line. The process continues till the last line in the image.

After the segmentation process document is segmented into characters and associated diacritic marks as given in below Figure 31. To demonstrate segmentation output symbols are bounded with different colours. The core characters are shown in red and green colours alternatively. The left and right diacritic marks are shown in yellow colours while top and bottom diacritic are shown in magenta colours.

Figure 31 Output of segmentation process

5.4 DISCUSSION

The preprocessed binary image is input to the segmentation phase. The output of segmentation phase is isolated segmented unit which are categorized into core character, diacritic marks and conjuncts. The segmentation phase of OGHTR is

divided into three steps 1) character height and width estimation, 2) estimate tentative midlines and 3) tracing and labelling.

The main advantage of given segmentation method is it uses knowledge of Gujarati Akshara formation and extracts symbols in the same manner. The output of segmentation phase is feed into next phase for recognition. Hence, next chapter describes recognition phase of OGHTR.