# Chapter 5

# Character segmentation

## 5.1 Background

In Chapter 4, we have proposed method for segmentation of text lines from Indus documents. To recognize the character, it is necessary to segment character from text lines. This chapter focuses on character segmentation by introducing watershed model. This model works based on fact that when there is a space between characters, water flows in linear passion and when there is a touching between the characters, water collects as basin.

## 5.2 Proposed Methodology

Proposed method considers text lines extracted from Indus document images. We use the same combination of the Sobel and the Laplacian as discussed in previous chapter at text line level rather than at image level to enhance the edge. Thus we apply morphological operations to obtain smoothed image. The smoothed image is given as input for watershed model for character segmentation. The flow of the proposed method for character segmentation in Indus document images is shown in Fig. 5.1.

The proposed methodology is divided into two subsections. In first subsection, we discuss about the enhancement of the text lines. In second subsection, we introduce watershed model for segmentation of characters.

Some parts of the material of this chapter appeared in the following research papers:

1. "A New Watershed Model based System for Character Segmentation in Degraded Text Lines", International Journal of Electronics and Communications(AEU),Vol 71,2017, 45-52.

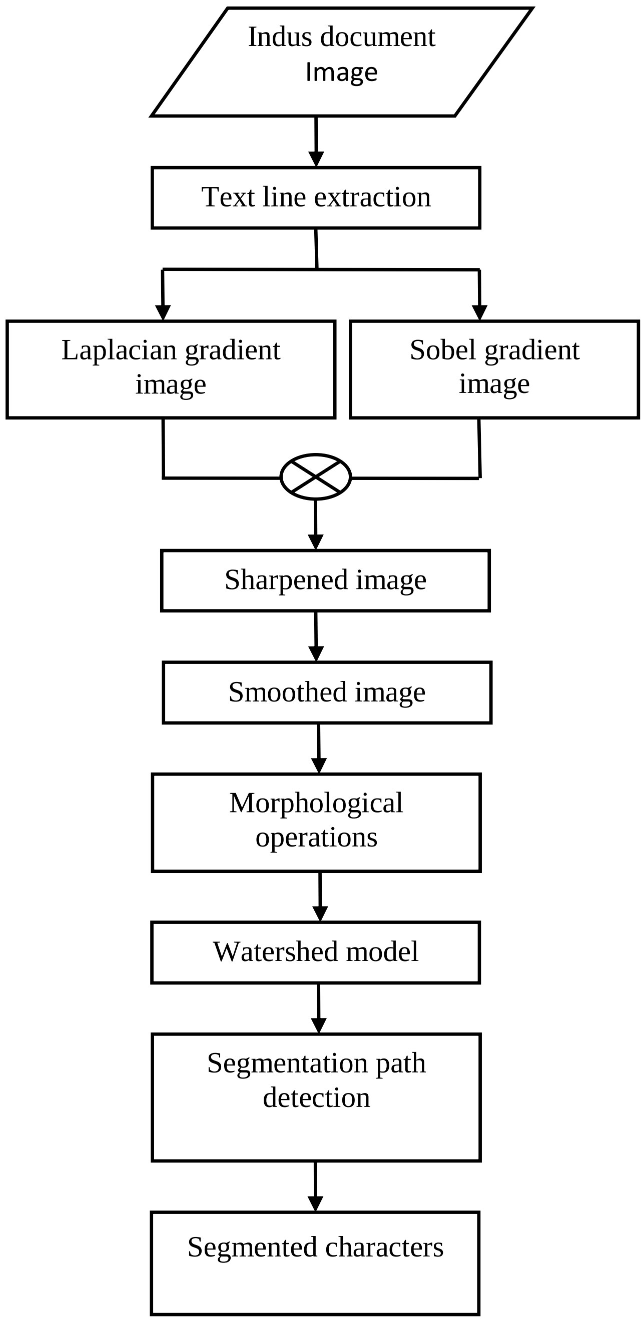


**Fig. 5.1. Proposed character segmentation**

## 5.2.1 Character Enhancement

For enhancement, text lines are extracted using the procedure discussed in chapter 4. It is a fact that Sobel gradient responds to high contrast pixels as shown in Fig. 5.3(a) resulting in thick double and sharp edges. Sobel gradient, computes first derivative measurement of an image $\nabla I(x,y)$ finds gradient in x and y direction as shown in equation 5.1.

$$\nabla I(x,y) = \sqrt{(G_X + G_y)} \tag{5.1}$$

$$\approx \nabla I(x,y) = |G_X| + |G_Y| \tag{5.2}$$

Laplacian of an image $\nabla^2 I(x,y)$computes a second derivative measurement shown in equation 5.3, is sensitive to noise as seen in Fig. 5.3(b). In order to obtain pixels which provide edges of text components, we convolve the gradients $G_X\ and\ G_Y$ images as discussed in section 4.2.1 of chapter 4. It is found convolution of Laplacian with Sobel gradient image gives the approximated smoothed image.

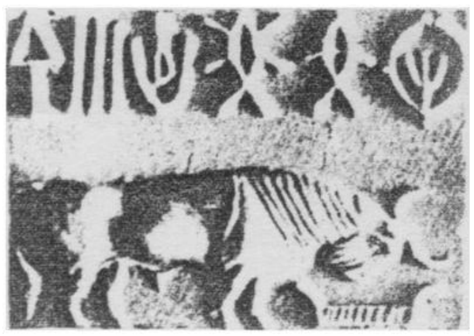
Laplacian of Image with pixel intensities $I(x,y)$ is given by,

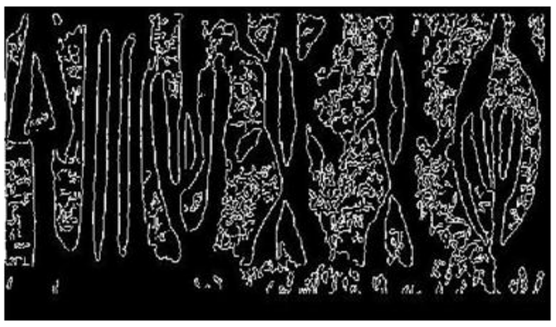
$$\nabla^2 I(x,y) = \frac{\delta^2 I}{\delta x^2} + \frac{\delta^2 I}{\delta y^2} \tag{5.3}$$

$$\nabla^2 I(x,y) \approx 0 \tag{5.4}$$

Convolution $C(I,J)$ is performed by sliding the Laplacian image over the Sobel image for finding the intersection of these two images as in equation 5.5.

$$C(I,J) = \nabla I(x,y) \circ \nabla^2 I(x,y) \tag{5.5}$$


(a) Indus Document


(b) Edge image of text line

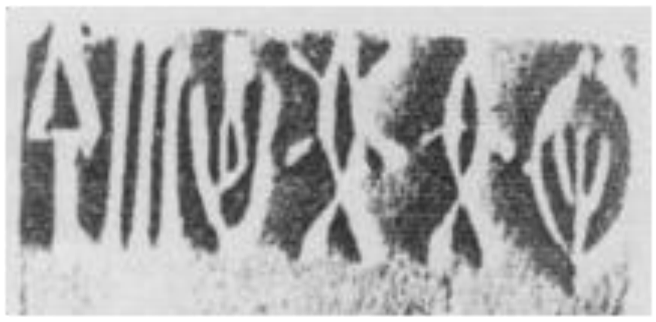
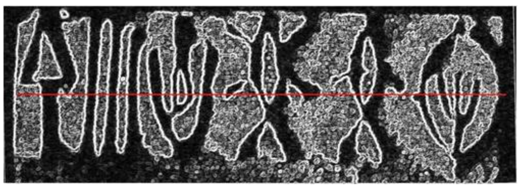
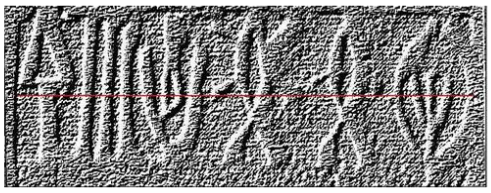

(c) Text line

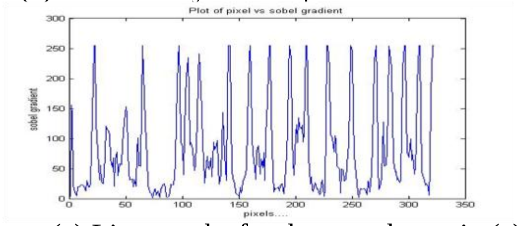
**Fig. 5.2. Text line segmentation from Indus document image**

When we look at the line graphs Fig. 5.3(c) drawn for the results in Fig. 5.3(a) it is found more number of valleys than the number of characters. Similarly, when we look at the line graphs Fig. 5.3(d) drawn for the results in Fig. 5.3(b) it has more number of peaks than the number of characters. Lines to peaks are more cursive. This shows intensity changes are not even and there are more noisy pixels. When we look at line graphs Fig. 5.3(f), drawn for the results in Fig. 5.3(e), better sharp peaks at edges and low peaks at spaces. Lines are not cursive compared to the line graphs in Fig. 5.3(c) and Fig. 5.3(d). However, Fig. 5.3(e) still contains a few background noises. To remove such small noise pixels, the proposed method uses connected component analysis to remove small components for the output of enhancement step as shown in Fig. 5.3 (g), where noise pixels are removed and we can see clear spaces between the character components. The line graphs drawn in Fig. 5.3(h) for the image in Fig. 5.3(g) shows that the enhancement step removes background noises clearly.

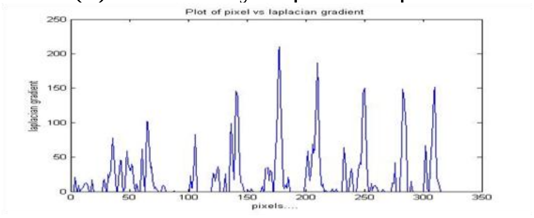
In this work, we prefer to use the above combination rather than popular canny edge operator because canny operator which is highly sensitive and outputs thinned edge images which are spurious in nature. Also we can see sobel and canny edge images appear with double lines. We can see that Sobel operator does not produce much spurious edges as in Fig. 5.3(i) compared to canny operator as in Fig. 5.3(k). This is because canny operators are very sensitive to Image pixels. The same thing can be confirmed from the line graphs in Fig. 5.3(j) and Fig. 5.3(l) where we can see more spikes in case of canny compare to sobel.Hence we decide that images are not smoothened using combination of sobel and canny as in Fig. 5.3(m). Due to this, intersection of these two operations will not yield good result in case of degraded documents such as Indus. Thus when compare to intersection of gradients as in Fig. 5.3(e) and Fig. 5.3(m) edges of gradient are very much sharpen compare to combination of sobel and canny. It can also be noticed in Fig. 5.3(f) some of the pulses are separated with certain wavelength whereas in Fig. 5.3(n) it is found more sharp spikes which are closed to each other. This signifies number of pixels in gradient image is reduced comparative to edge image. Other thing what we notice is, convolution of gradient image reduces discontinuities thus giving strong edges compare to sobel and canny edge combination. Thus, noisy pixels in gradients combination are also less. Double edges in combination of sobel and canny edge images decrease the spacing between the characters. Therefore, it can be concluded that the Sobel and the Laplacian combination is better for the Indus document.


(a) Gradient by Sobel operator

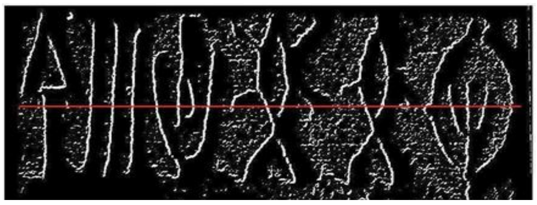

(b) Gradient by Laplacian operator

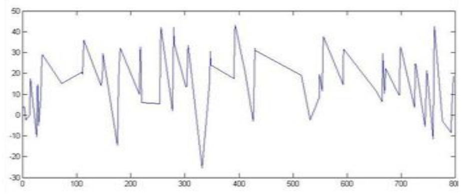


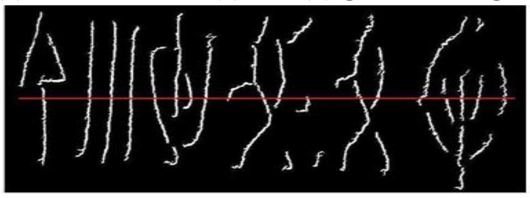
(c) Line graphs for the row shown in (a)

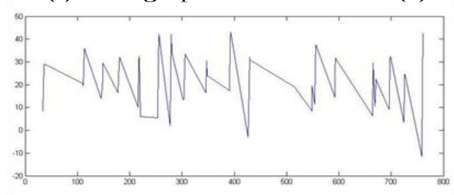


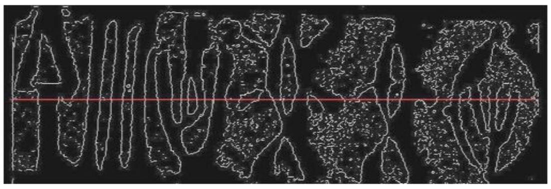
(d) Line graphs for the row shown in (b)

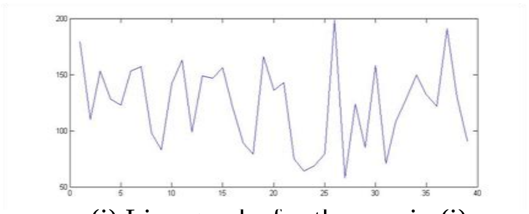

(e) Intersection of (a) and (b) gradient images

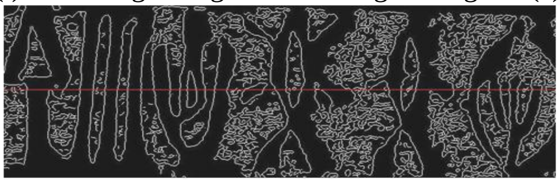

(f) Line graphs for the row in (e)

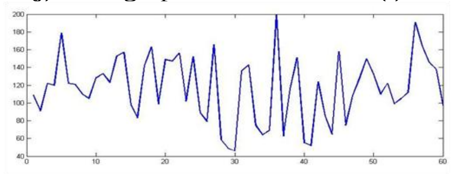

(g) Final cleaned image

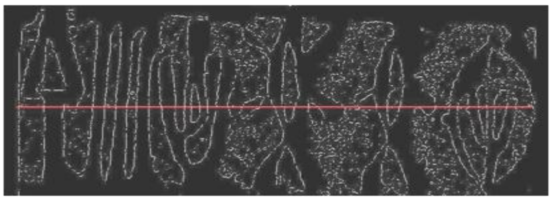

(h) Line graphs for the row in (g)


(i) Sobel edge image for the image in Fig. 5.2(a)


(j) Line graphs for the row in (i)


(k) Canny edge image of the input image in Fig.5.2 (a)


(l) Line graphs for the row in


(m) Intersection of Sobel and Canny edge images

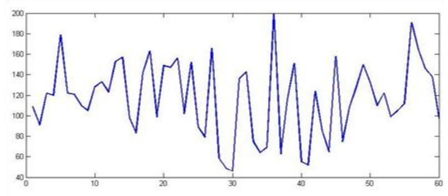
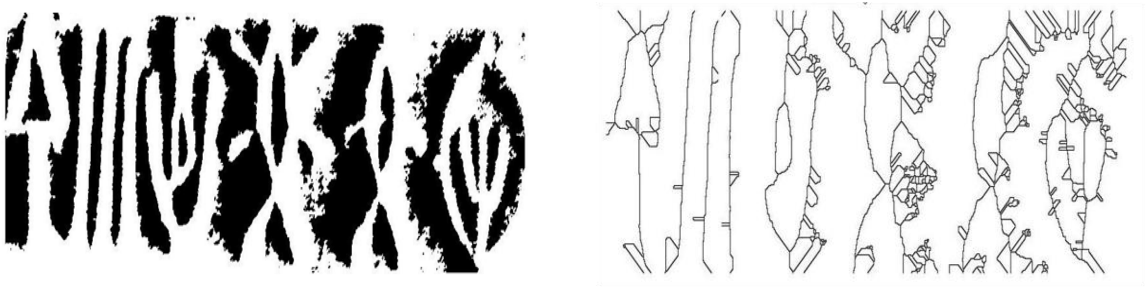

(n) Line graphs for the row in (m)

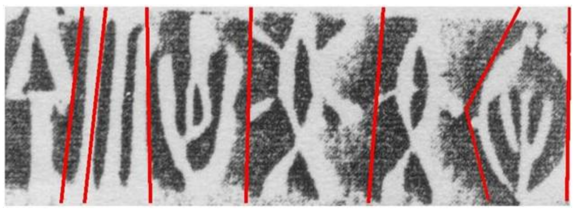
**Fig.5. 3. Illustration of the proposed enhancement**

## 5.2.2 Watershed Model for Character Segmentation

We know that water flows from maximum region to minimum region and stores in region called catchment basin. As observed in Fig. 5.3(g) there are many disconnected components in each of the text character. As water flows from maximal region to minimal region there are chances of water entering through these disconnections and forming the watersheds. Thus it is found necessary to group these disconnected components in each of the character. Thus we perform morphological operations as shown in Fig. 5.4(a) to group the disconnected components. It is noticed that edges of text pixels are cursive thus it is found pixels of single column not possible to segment characters. However water can be flowed at these cursive exterior edges of character. Hence this observation, inspired to use the watershed algorithm to find the boundaries of text characters. The watershed algorithm finds water flow and high volume of collection of water where there is a space between the character components. These two properties work well even if any touching exists between the character components. In this way, watershed algorithm helps in segmenting characters from Indus text line.



(a) Morphological operations to the image in Fig.5. 3(g)    (b) Water shed image for the image in (a)



(c) Character segmentation

**Fig. 5.4. Steps of the Proposed Character Segmentation**

Watershed image for Fig. 5.4(a) is shown in Fig. 5.4(b).Watershed image shows some of the unclosed white regions where water can be stored. We know water flow from upward towards downward. It can be noticed there are small regions within each of the character component where small amount of water can be stored. There also exist regions between the text characters where large amount of water can be stored. These high amount storage regions clearly indicate spacing between characters. These region points are extracted and works well for segmentation of characters as shown in Fig. 5.4(c). Fixing bounding box for each of the connected components leads to overwhelming of boundaries as shown in Fig. 5.5. Hence the method finds the point where maximum water is stored. Further to detect the segmentation path we consider the region between the characters and count the number of white pixels between the extreme black pixels. We then compute the position of midpoint at the top, middle and bottom of the region called as segmentation path markers. Joining of these markers is the segmentation path as shown in Fig. 5.6.Thus input image of Fig. 5.2 (c) segmented using this proposed method is as shown in Fig. 5.4 (c).

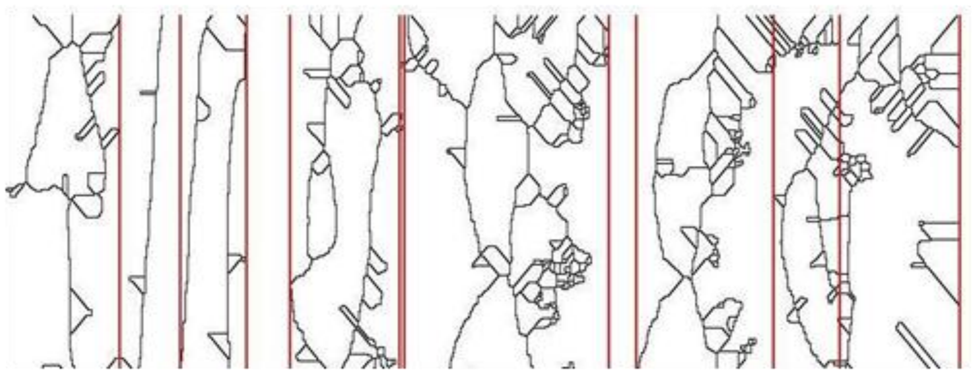


**Fig. 5.5. Over segmentation**

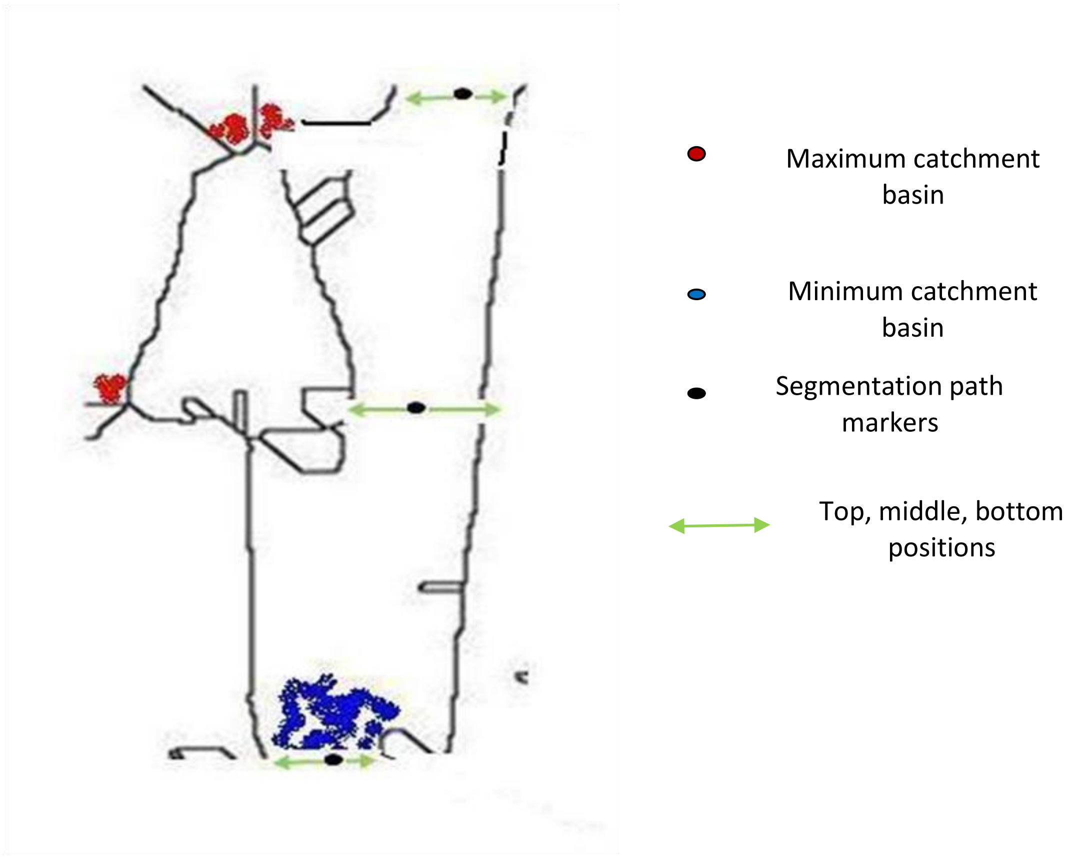
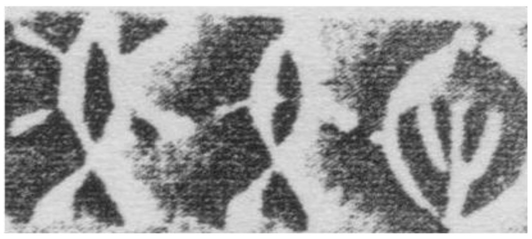
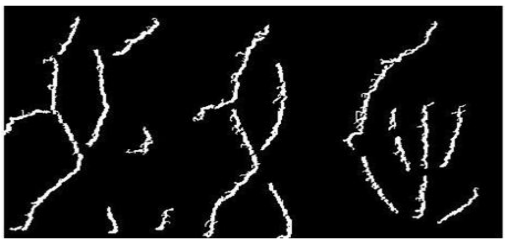
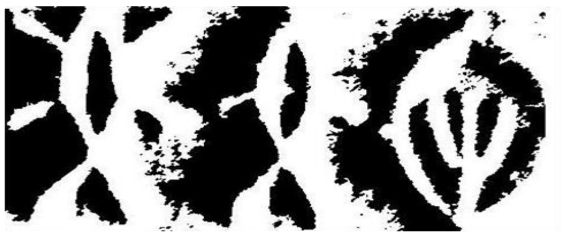


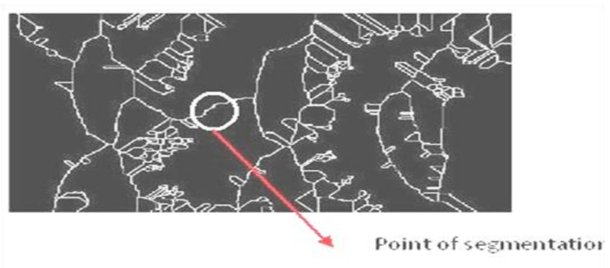
**Fig. 5.6. Segmentation path detection between first 2 text characters of Fig. 5.4(b)**

The effect of watershed algorithm can be seen in Fig. 5.4(c) that all characters are segmented correctly. One more illustration is presented in Fig. 5.7 where there is a touching exists between the characters. Fig. 5.7 shows that for the input image in Fig. 5.7(a), the proposed method obtain cleaned image by the previous step as shown in Fig. 5.7(b) and then the proposed method performs morphological operation to group the disconnected component as a single component as shown in Fig. 5.7(c). For the image in Fig. 5(c), when we apply watershed algorithm, there are chances of existing touching between character components due to complex background of Indus text line as shown in Fig. 5.7(d). In this case, when the proposed method estimates flow of water and volume of collection water, the space between the characters components has highest volume of the collection of water. Thus the markers detected guarantees the limits in which text characters exist. Therefore, the watershed algorithm segments characters components successfully for the case of touching characters as shown in Fig. 5.7(e).


(a) Input image


(b) Gradient's intersection image


(c) Morphological operations



(d) Watershed image for the image in (c)

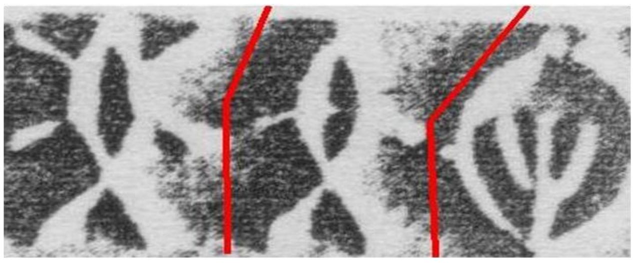

(e) Character Segmentation

**Fig. 5.7. Illustration for Proposed Character Segmentation for touching characters**

## 5.3 Experimental Results

To evaluate the effectiveness of the proposed method we first conduct experiments on 100 text lines of English and 100 cursive scripts that includeTelugu, Tamil and Malayalam scripts. Experimental results for English text lines for the proposed and existing methods are shown in Table 5.1. Experimental results for other cursive scripts (Telugu, Tamil and Malayalam) for the proposed and existing methods are shown in Table 5.2.

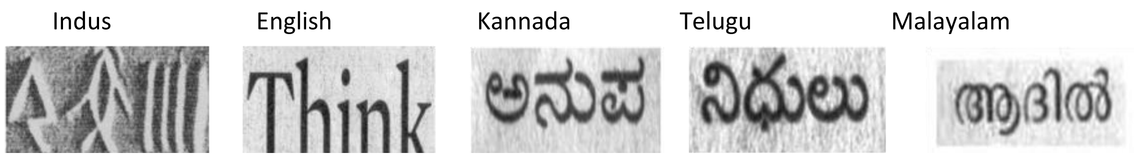
To conduct experiments on the proposed method for segmentation of characters we accumulate 500 Indus text line images segmented using method proposed in chapter 4.This is because scope of our research is limited to Indus scripts. This dataset includes a variety of text lines on different surfaces and different handwriting with different tools. As a result, this dataset is said to be complex compared. Fig. 5.4 and Fig. 5.7 show the results using the proposed method for untouching and touching characters. To measure the performance of the proposed character segmentation method, we use the well-known measures, namely, recall and precision and F-measure as defined in equation (5.6)-(5.8), where the test outcome may be in various facts such as TP (true positive) defined as the number of characters that are correctly segmented, Fp (false positive) is number of false characters detected as true, and Fn (false negative) is the number of characters which is true w.r.t false category. Experimental results for Indus text lines for the proposed and existing methods are shown in Table 5.3.

$$Recall = \frac{Tp}{(Tp + Fn)} \tag{5.6}$$

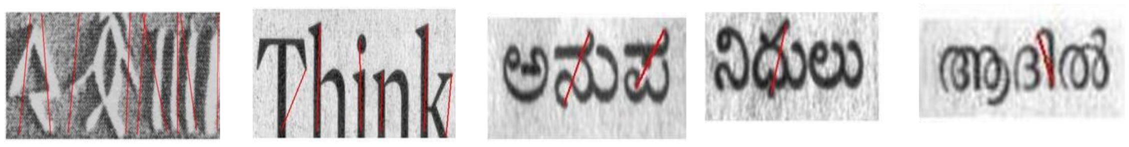
$$Precision = \frac{Tp}{(Tp + Fp)} \tag{5.7}$$

$$F\_Score = \frac{2Tp}{(2Tp + Fp + Fn)} \tag{5.8}$$

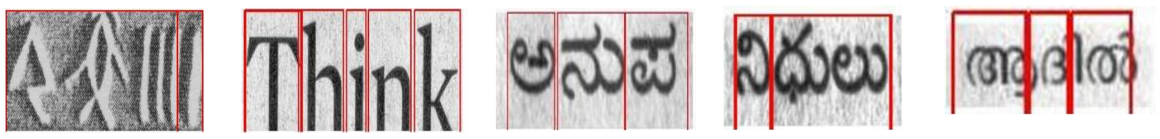
To show the effectiveness of the proposed method, we implement three existing methods for comparative studies. Sample qualitative results of the proposed and existing methods are shown in Fig. 5.8 for the different scripts text lines. Existing method Vamvakas et al.'s method does not give good result compare to propose method. Method finds segmentation points using feature points and the distances between feature points. The method gives poor results for Indus and other scripts such as English and south Indian scripts. Main reason for the poor result is that the method is not tolerant to degradation existed in Indus and scanned images of English and south Indian scripts. This is because scanned images have many disconnected pixels that are considered as true feature points in the method. Another existing method proposed by Me et al. uses connected components analysis and boundary of segmentation obtained using vertical projection profile for character segmentation. The method is tolerant to degradations to certain level. Hence compare to Vamvakas et al.'s method it gives good result for English script. Since the method finds the connected components for scripts. South Indian scripts and Indus scripts are more cursive hence there exists more number of connected components than the number of characters. This is due to background and structure of scripts. Similarly, Cheung et al.'s method does not segment characters properly because the proposed features extracted are based on the structure of a specific script. It segments using vertical projection profiles. This method gives the low values for the measures. However, there is some improvement in precision value of south Indian scripts in Table 5.2 comparatively. This is because Malayalam scripts do not have more sub components for the texts. Hence this gives more true positives comparatively. However, for English it gives low result since the text characters are almost have equal size. Thus experimental results shown for the proposed method in Table 5.1 to Table 5.4 gives good results for Indus and other script text line images because of the advantage of the enhancement step and the watershed model.



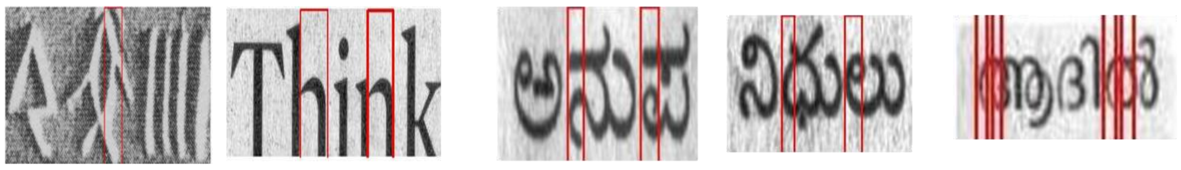
(a) Input text line images of different scripts



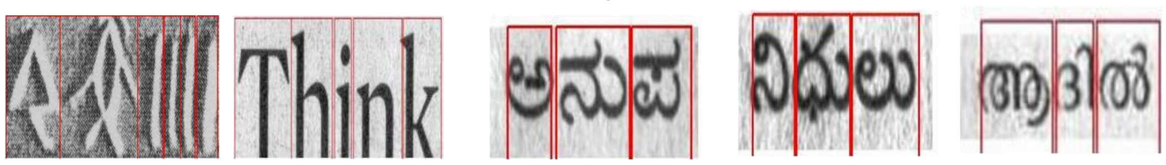
(b) Vamvakas etal.



(c) Mei et al.



(d) Cheung et al.



(e) Proposed method

**Fig. 5.8. Qualitative results of the proposed method for Character segmentation**

It is found necessary to evaluate the performance of the proposed method without enhancement to show the importance of enhancement. We evaluate the performance of the proposed method without enhancement using same quantitative measures, such as recall, precision and f-measure for different types of script data, we calculate the measures for English, South Indian scripts and Indus as reported in Table 5.1-Table 5.4, respectively. It is noted from Table 5.1-Table 5.4 that the proposed method achieves better results compared to the existing methods in terms of recall, precision and f-measure. When we compare the result of the proposed method without enhancement English scripts in Table 5.1 gives good result when compare to other scripts including Indus. This is because English scripts usually have less disconnected components. Hence there exists clear spacing between the characters. Watershed images for English text lines have maximum catchment basins in between the text characters. Therefore, compare to other scripts English scripts works well without enhancement.

Proposed method with enhancement gives good results for scripts like Kannada, Telugu, and Malayalam as shown in Table 5.2. Proposed method merges the small regions that are aroused due to presence of modifiers. This avoids the free region spacing between character component and modifier component. Moreover, in scanned images there appears the image of some false pixels. Thus it is needed to enhance the gradients of text pixels. Thus enhancement step in proposed method enhance text pixels and creates the segmentation regions between the characters. Thus with enhancement the proposed method gives good result not only for English it also gives good segmentation result for cursive scripts also. Reason is proposed watershed model is good for irregular shape scripts. Proposed method performance is low without enhancement. This is due to appearance of more noisy speckles gives more ridges in watershed model. Hence this increase number of catchment basin. Thus segmentation paths found are more than the number of characters. Therefore, to achieve good results for all types of scripts, we need the enhancement step to clean background before applying the watershed model. In this way, the proposed enhancement step contributes well for achieving good character segmentation results in this work.

Table 5.1. Performance of the proposed and existing methods on English script data

| Methods | Recall | Precision | F_Measure |
|---|---|---|---|
| Mei et. al. | 0.95 | 100 | 0.97 |
| Vamvakas et. al. | 0.3 | 0.375 | 0.33 |
| Cheung et. al. | 0.18 | 0.33 | 0.235 |
| Proposed Method without Enhancement | 0.84 | 0.875 | 0.85 |
| Proposed Method with Enhancement | **100** | **100** | **100** |

Table 5.2. Performance of the proposed and existing methods on other script (Kannada, Telugu, and Malayalam) data

| Methods | Recall | Precision | F_Measure |
|---|---|---|---|
| Mei et. al. | 0.3 | 0.5 | 0.375 |
| Vamvakas et. al. | 0.3 | 0.33 | 0.315 |
| Cheung et. al. | 0.12 | 0.54 | 0.196 |
| Proposed Method without Enhancement | 0.4 | 0.5 | 0.44 |
| Proposed Method with Enhancement | **0.95** | **0.97** | **0.96** |

Table 5.3. Performance of the proposed and existing methods on Indus script data

| Methods | Recall | Precision | F_Measure |
|---|---|---|---|
| Mei et. al. | 0.016 | 0.047 | 0.024 |
| Vamvakas et. al. | 0.016 | 0.024 | 0.019 |
| Cheung et. al. | 0.006 | 0.1 | 0.0116 |
| Proposed Method without Enhancement | 0.03 | 0.285 | 0.059 |
| Proposed Method with Enhancement | **0.99** | **100** | **0.99** |

Table 5.4. Overall performance of the proposed and existing methods on the whole data

| Methods | Recall | Precision | F_Measure |
|---|---|---|---|
| Mei et. al. | 0.37 | 0.92 | 0.5 |
| Vamvakas et. al. | 0.04 | 0.36 | 0.072 |
| Cheung et. al. | 0.2 | 0.32 | 0.147 |
| Proposed Method without Enhancement | 0.3 | 0.65 | 0.41 |
| Proposed Method with Enhancement | **0.98** | **0.99** | **0.98** |

## 5.4 Summary

In this chapter, we have proposed a new method for segmentation of characters from text lines in degraded historical document images like Indus. The proposed method explores the combination of Laplacian and Sobel operations for enhancing low contrast pixels in images by suppressing background noises. The characteristics of text components in an enhanced image are studied to eliminate unwanted background noise components, which results in a cleaned image with only edges which represent text components. We have proposed the watershed model for identifying spacing between characters by exploiting catchment basin and flow of water. Experimental results and the comparisons with the existing methods show that the proposed method outperforms the existing methods in terms of recall and precision. Our future plan would be extending for multiple touching character component images in multi scales or multi oriented environments.