

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/323925357>

Improved Symbol Segmentation for TELUGU Optical Character Recognition

Chapter · March 2018

DOI: 10.1007/978-3-319-76348-4_48

CITATIONS

2

READS

425

4 authors, including:



Amit Patel

Rajiv Gandhi University of Knowledge Technologies, Nuzvid, India

12 PUBLICATIONS 42 CITATIONS

[SEE PROFILE](#)



Chakravarthy Bhagvati

University of Hyderabad

84 PUBLICATIONS 479 CITATIONS

[SEE PROFILE](#)



Atul Negi

University of Hyderabad

152 PUBLICATIONS 1,105 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Machine Learning Projects [View project](#)



Robust OCR DIT [View project](#)

Improved Symbol Segmentation for TELUGU Optical Character Recognition

Sukumar Burra¹, Amit Patel², Chakravarthy Bhagvati¹ and Atul Negi¹

¹ School of Computer and Information Sciences, University of Hyderabad,
Hyderabad-500046, Telangana, India

² RGUKT IIIT Nuzvid, Krishna - Andhra Pradesh, India
sukumar.leo@gmail.com, amtpt193@gmail.com
chakcs@uohyd.ernet.in, atulcs@uohyd.ernet.in

Abstract. In this paper, we propose two approaches to improving symbol or glyph segmentation in a Telugu OCR system. One of the critical aspects having an impact on the overall performance of a Telugu OCR system is the ability to segment or divide a scanned document image into recognizable units. In Telugu, these units are usually connected components and are called glyphs. When a document is degraded, most connected component based algorithms for segmentation fail. They give malformed glyphs that (a) are partial and are a result of breaks in the character due to uneven distribution of ink on the page or noise; and (b) are a combination of two or more glyphs because of smudging in print or noise. The former are labelled *broken* and the latter, *merged* characters. Two new techniques are proposed to handle such characters. The first idea is based on conventional machine learning approach where a Two Class SVM is used in segmenting word into valid glyphs in two stages. The second idea is based on the spatial arrangement of the detected connected components. It is based on the intuition that valid characters exhibit certain clear patterns in their spatial arrangement of the bounding boxes. If rules are defined to capture such arrangements, we can design an algorithm to improve symbol segmentation. Testing is done on the Telugu corpus of about 5000 pages from nearly 30 books. Some of these books are of poor quality and provide very good test cases to our proposed approaches. The results show significant improvements over developed Telugu OCR (Drishti System) on poor-quality books that contain many ill-formed glyphs.

Keywords: Bounding Boxes, Optical Character Recognition, Symbol level segmentation, Support Vector Machine, Telugu OCR

1 Introduction

Optical Character Recognition (OCR) systems deal with the recognition of printed or handwritten characters. OCR System is the one which converts a scanned document image into editable text. Any OCR system implementation involves the following basic steps: Preprocessing which involves binarization,

skew detection and correction. Symbol Segmentation i.e. Line, word and character segmentation. Feature extraction of a symbol, actual Recognition using classifier and finally, Conversion of recognized symbol into text format(UNICODE). Symbol Segmentation is the process of decomposing lines into words, then words into characters or recognizable units called **Glyphs**. Glyph is a basic orthographic unit in a scripture of a language. English and other European language scriptures contain small set of alphabets which are combined to form words. So segmenting and recognizing these characters is not as big an issue as segmenting Indic script characters.

Where as an Indian languages shows huge diversity in their orthography. Especially south Indian languages like Telugu, Kannada etc. are of complex orthography with a large number of distinct character shapes which are combined to form compound characters. In order to reduce the complexity, the characters are generally broken into basic glyphs to recognize them. In most of the OCR systems this segmentation is done by extracting the connected components.

Developing the OCR systems for Indian language scripts is challenging task. Telugu is a phonetic language, written from left to right, with each character representing generally a syllable[1]. There are 52 letters in the Telugu alphabet: 16 *Achchulu* which denote basic vowel sounds, 36 *Hallulu* which represent consonants. In addition to these 52 letters, there are several semi-vowel symbols, called *Maatralu*, which are used in conjunction with *Hallulu* and, half consonants, called *Voththulu*, to form clusters of consonants.

DRISHTI is a complete OCR system for Telugu, which was developed at University of Hyderabad[2]. DRISHTI is the first comprehensive OCR system for Telugu. This system was initially described[1], and then improved subsequently[3]. It was reported[1] that DRISHTI system gives better recognition accuracy at the glyph level up to 97% on good quality Telugu document images, which are scanned at 300 dpi. It also resulted in reasonable accuracy over different kinds of inputs like novels, newspapers and laser printed text. A significant improvement were reported in[4] using inverse fringe.

DRISHTI system uses connected component extraction to segment the characters. But sometimes, imperfections in scanning, poor quality printing and binarization leads the characters to break or merge with each other. In such cases connected component extraction results in touching and broken glyphs. In this paper, we have proposed the methods to handle this situation in order to improve the symbol segmentation. We have proposed two level Segmentation in order to reduce the broken glyphs and proposed an approach to merge the broken glyphs.

The content of paper is organized as follows: apart from introduction section 2 provides overview of proposed approaches. Two level Symbol Segmentation approach was described in section 3 . Approach to merge the broken glyphs is discussed in section 4, Section 5 discusses about experimental results and analysis of it and Finally, section 6 gives Conclusion.

2 Proposed methods

As we discussed earlier, connected component labelling algorithm on word images may result in broken and touching glyphs, which are recognized wrongly most of the time. So there is a need to handle these touching and broken glyphs in order to get better recognition. And also we need to segment the symbols in a better way to avoid broken and touching components.

We have proposed following approaches in order to improve the symbol segmentation:

2.1 Two Level Symbol Segmentation

In order to reduce the broken glyphs, We have proposed an approach to segment the words in two levels into connected components. If we observe the glyphs in Telugu, almost all of them are round in shape and with curves. And, there are no vertical strokes in Telugu script. So, vertical projection profile of a word will help in segmenting the connected glyphs correctly, though they may have breaks at certain locations and angles.

Segmenting the words vertically does not result in single connected components over the words having the compound characters. So, we need to segment these compound characters into connected components in order to send it to the recognition step. For this, we have used a classifier to classify the component which is obtained after Vertical cut segmentation, into a single connected component(SCC) and Compound connected component(CCC). We will send single connected component to the recognition directly. On the Compound connected component, we apply connected component approach[10] to extract the connected components and then send to the recognition step. We have briefly discussed all the steps in Two level symbol segmentation approach in section 3.

2.2 Merging Broken Glyphs using Bounding Box Analysis

Two level symbol segmentation is able to extract most of the broken Single-CCs correctly. But, whenever broken Compound-CCs having are subjected to Connected component labelling, they result in broken glyphs.

To handle these broken Compound-CCs, we have computed bounding boxes for each connected component resulted from the segmentation. Bounding box for a connected component is the Minimum Bounding Rectangle (MBR). Later, we analysed MBRs of CCs in different Telugu document images. This analysis helped us to identify the possible MBRs of connected components which can be generated only by valid Telugu characters.

Along with this analysis, we have used the zone information[5] of a line for framing the rules to merge the MBRs of broken glyphs. We have described these details in section 4.

3 Two Level Symbol Segmentation

As we mentioned in section 2.1, in Two level symbol segmentation, first segment the words into components using Vertical Cut Segmentation algorithm, which is designed based on vertical projections. We refer these components as **Vertical Cut Components**. Later, we classify these vertical cut components into single and compound connected components using Two class SVM. Finally, connected component segmentation algorithm was applied on compound connected components to extract CCs.

3.1 Vertical Cut Segmentation

Vertical Cut Segmentation algorithm splits the word image into vertical cut components. These components are segmented by identifying the vertical boundaries for each component in a word. This is done by using vertical projection profile[6] of the word i.e. number of black pixels in each column of the word. The Vertical Cut Segmentation on the word image shown in Fig 1(a), results in the components shown in Fig 1(b). Connected component segmentation on this word image will definitely result in broken glyphs at third and fourth component in Fig 1(b).

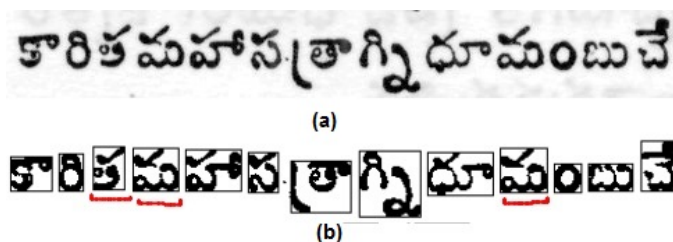


Fig. 1. Sample Word image with Segmented Vertical Cut Components

3.2 Classification of Single and Compound Connected Components

The components obtained from the vertical cut segmentation algorithm are either Single connected components or Compound connected components. Only, the compound connected components are to be segmented further, in order to get Single connected components. Otherwise, the SCCs which having breaks in them will be segmented further, and results in broken glyphs. So, we have used a binary classifier i.e. Two Class SVM, in order to classify which components are to be sent directly to the recognition and which are to be segment further.

Features We have scaled the components obtained from Vertical Cut Segmentation, into the size 32×32 . We have stored pixel intensities of entire 32×32 sized components in 1024 sized array row by row. These 1024 valued array is taken as a feature vector for each component. Since the components are binarized, these 1024 values are either 0 (i.e. black pixel) or 255 (i.e. white pixel).

Classifier - (Two Class SVM) We have used **Two Class-Support Vector Machine** to classify the components into two classes. Here, we consider the first class as Single CC and the second class is Compound CC. As we know that, TC-SVM learning constructs a classification hyperplane to maximally separate the two classes in the feature space. We have used ν -Support Vector Classification(ν -SVC) implementation of Two class SVM[7].

Here, we have implemented Two Class SVM by using CvSVM[8], which is in-built OpenCV library class for Support Vector Machines. This CvSVM class implementation in OpenCV is based on LibSVM[9].

Training and Testing: We have collected 1000 samples of vertical cut components for each class i.e. 1000 Single-CCs and 1000 Compound-CCs. These 2000 samples are extracted from different the Telugu books, which include good quality printed books as well as poor quality old printed books.

As we mentioned earlier, Single-CC class samples are labelled as Class-0. Class-0 includes the single connected components which contains breaks too. These trained, broken single-CCs in this class are responsible for segmenting single-CCs correctly, though they contain breaks.

The Compound-CC class samples are labelled as Class-1. In this class, all samples are compound characters, which contain more than one CC. We have included some of basic glyphs like *sa,pa*, because they contain 2 CCs. We have also included the components having multiple basic glyphs, which we are not able to segment using our vertical cut segmentation algorithm.

Trained TC-SVM builds a model, which is able to classify the test sample i.e. a vertical cut component as either Class-0 or Class-1.

For testing, we collected 1000 samples of single-CCs and 1000 samples of Compound-CCs, other than 2000 training samples. TC-SVM is able to classify 940 samples as Class-0, out of 1000 class-0 test samples, which gives accuracy 94% for Class-0. And, it classified 898 samples as Class-1, when it is tested over 1000 Class-1 samples. The accuracy for class-1 is 89.8%.

3.3 Experiments and Observations

We have experimented Two-level Symbol Segmentation over a different type of document images which are good as well as poor quality printed.

The Vertical Cut segmentation output of a input line image in Fig 2(a) will be shown in Fig 2(b). Then, the Compound-CCs in Fig 2(b) subjected to Connected component segmentation. The final output of Two level segmentation on input image i.e. 2(a) is shown in Fig 2(c). Here, it can be observed that, current approach is able to segment broken glyphs as a combined and able to break Compound-CCs into valid glyphs.

Two level symbol segmentation is capable of handling the breaks in single-CCs and extract them as a single components. But, the broken glyphs due to the segmentation of Compound-CCs are still remain. In the next section, we have proposed an approach to merge the broken glyphs arises in Compound-CCs.

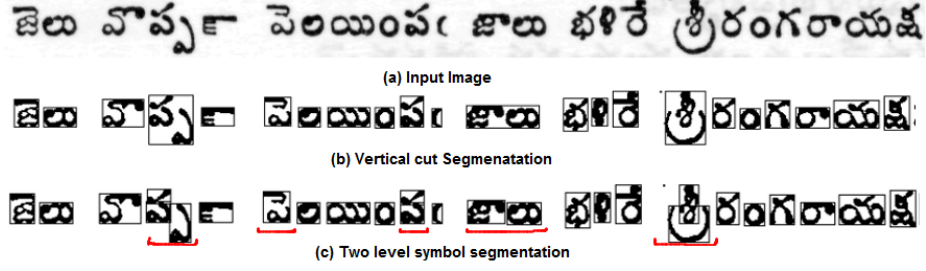


Fig. 2. (a) Input test line image (b) Output of Vertical cut Segmentation on line image (c) Output of Two level Segmentation on line image

4 Merging Broken Glyphs using Bounding Box Analysis

4.1 Bounding Boxes

Bounding box of a connected component is a representation of its maximum extents in 2D(X-Y) co-ordinate system. It's also referred as Minimum Bounding Rectangle(MBR). MBR is the rectangle formed by $min(x)$, $max(x)$, $min(y)$ and $max(y)$ of a connected component. Fig 3 shows the word and its bounding boxes.



Fig. 3. Word Image and its Bounding boxes

4.2 Zone Information of a Line

Most of North Indian language scripts like Devanagari have Shirrekha. There is no Shirrekha in Telugu script. It is difficult to identify the top and lower zones. But, most of the Telugu base characters are in middle zone. This will be used to divide the line into three zones i.e. top, bottom and middle zone.

To divide a line into 3 zones, 4 boundaries are required. Two boundaries came to be known though the line MBR i.e. top and bottom. To identify other two boundaries, horizontal projection profile of a line is computed. Then two peaks are identified. The position at which the peak is identified in the region from top boundary to middle point of the line is considered as *shirrekha* and the region from middle point to bottom boundary is considered as *baseline*. Fig 4 (a) depicts the top, middle and bottom zones along with its boundaries, of a line image shown in Fig 4 (b)

4.3 Rules to merge the Broken Glyphs

In this approach, we have framed some rules, in order to combine the broken glyphs in the Compound-CCs by using their bounding boxes i.e. MBRs. We have also used Zone information of the line, word MBR and Compound-CC MBR to build these rules. The midpoints of Compound-CCs and component CCs of it are also computed. These are used for knowing to which zone of the line, the given CC belongs.

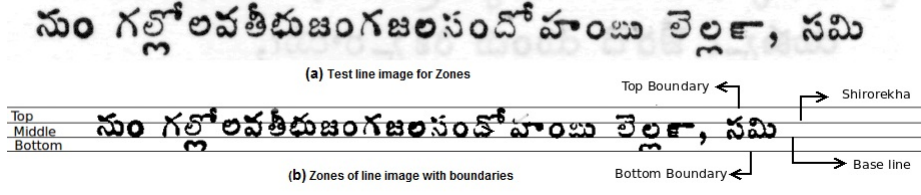


Fig. 4. Test line image with zone boundaries

Broken Voththulu (At the right and left) Consider a Compound-CCs having a broken voththulu which is joined at right or left to the hallu. The Connected component labelling on it leads to break that voththulu. This type of breaks in voththulu are frequent and can be observed in Fig 5.

Fig 5(A) shows a Compound-cc obtained after Vertical Cut algorithm. The connected component segmentation on it results in CCs that can be shown in Fig 5(B). The bounding boxes of CCs in that Compund-CC, can be seen in Fig 5(C). Fig 5(D) depicts the top,bottom boundaries,shirorekha and base line for the given Compound-CC.

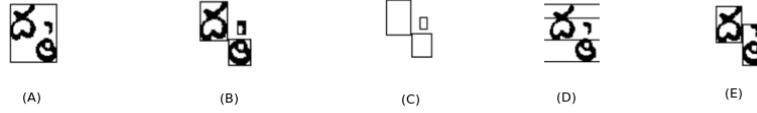


Fig. 5. Merging the broken voththulu combined with hallu

To handle this type of broken Voththulu, first the broken voththulu is to be identified in Compound-CC. Later they can be merged by combining their bounding boxes. This can be done by using following algorithm:

```

CC ← Connected Component
CCC ← Compound-CC
flag ← 0
for all  $CC_i \in CCC$  do
  if  $MBR(CC_i) \in \text{Bottom Zone}$ , and  $MBR(CC_i) \in \text{right part of } MBR(CCC)$ 
  then
    for all  $CC_j \in CCC$  do
      if  $MBR(CC_j) \in \text{Middle Zone}$ , and  $MBR(CC_j) \in \text{right part of } MBR(CCC)$ 
      then
        if  $\text{distance}(CC_i, CC_j) \leq \text{Threshold}$  then
          Break
          flag = 1
        end if
      end if
    end for
  end for
  if flag == 1 then
    Combine(  $MBR(CC_i)$ , all  $MBR(CC_j)$ s for which  $CC_i \in \text{Middle Zone}$ ,
    and  $MBR(cc_j) \in \text{right part of } MBR(CCC)$ )
  end if
end for


```



```

    end if
  end if
end for

```

This algorithm will combine the broken voththulu which can be possible right to the hallu. The combined MBR of broken voththulu can be seen in Fig 5(E). Here the voththulu is placed at the right side of hallu. The similar rule can be written for voththulu which can be placed at the left side of hallu like  voththulu.

In Top and Bottom Zones In Telugu script, a simple or compound character never contain more than one connected component in Top zone. It implies that, if two are more bounding boxes are identified in Top zone alone, they should be broken glyphs of maatralu. So, they have to be combined.

This merging is done by using following rule:

```

CC ← Connected Component
CCC ← Compound-CC
for all  $CC_i \in CCC$  do
  for all  $CC_j \in CCC$  do
    if  $MBR(CC_i) \in \text{Top Zone}$ , and  $MBR(CC_j) \in \text{Top Zone}$  then
      Combine( $MBR(CC_i), MBR(CC_j)$ )
    end if
  end for
end for
end for

```

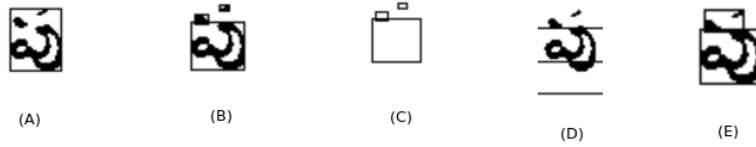


Fig. 6. Merging the broken glyphs in Top Zone

Fig 6(A) is a broken Compound-CC. The CCs can be observed in Fig 6(B). The broken bounding boxes in Top Zone, can be seen in Fig 6(C). The Fig 6(D) depicts zone boundaries. Finally, Fig 6(E) shows the combined bounding boxes in Top zone by using the rule, which was described above.

On the other hand, in Telugu script, possibility of two more CCs in the bottom zone alone is very rare. If there are more than one CC in bottom zone, they can be combined by using the similar rule, which is framed for top zone. But, these type of broken glyphs are very rare.

Middle Zone In Telugu script, all the base characters i.e. achchulu and hallulu will be in middle zone. There is no possibility of more than one CC in middle zone alone even in Compound-CC, unless it was broken. If there are more than one CC in middle zone, they can be combined by using following rule:

```

CC ← Connected Component

```

```

CCC ← Compound-CC
for all CCi ∈ CCC do
  for all CCj ∈ CCC do
    if MBR(CCi) ∈ Middle Zone, and MBR(CCj) ∈ Middle Zone then
      Combine(MBR(CCi), MBR(CCj))
    end if
  end for
end for

```

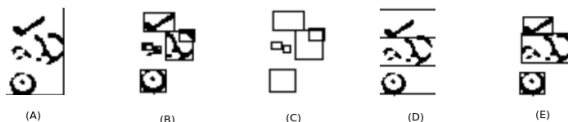


Fig. 7. Merging the broken glyphs in Middle Zone

The Compound-CC having multiple breaks in middle zone can be seen in Fig 7(A). The CCs are shown in Fig 7(B). The broken bounding boxes alone can be seen in Fig 7(C). The Fig 7(D) gives it's zone boundaries. After applying the rule mentioned above, the broken bounding boxes are combined and form a single bounding box in middle zone. This can be seen in Fig 7.

These rules are framed by observing the frequent broken glyphs. Though, these rules doesn't handle all possible broken glyphs, but still able to merge some broken glyphs. The strength of these rules is that generalization over script i.e. rules are not framed for some specific symbols.

5 Experiments and Analysis of Results

We have done the experiments over 5000 pages of Telugu document which includes, good as well as poor quality print. Here, we have used Two level symbol segmentation to extract the valid glyphs and later, we applied the rules that are proposed in section 4 to merge the broken glyphs.

Let us consider the Input image shown in Fig 8. This is a page from the book named **Vasucharithramu**, having lots of broken glyphs.

The valid glyphs which are extracted from the input image in Fig 8, using proposed approaches i.e. Two level symbol segmentation and Merging the broken glyphs, is shown in Fig 8. It can be observed that, the components are segmented well, though they have breaks in them. We can also observe, some merged broken glyphs which are underlined below.

The OCR output text for the input image shown in Fig 8, can be seen in Fig 8. For this document image, the error rate is reduced more than 10%, when compared with the OCR output of same image by the old DRISHTI system.

The Table 1 gives the comparison of the error rates for some document images computed over the old DRISHTI system and DRISTHI with proposed approaches. It can be observed that, the error rate is reduced significantly for the documents having lots of broken glyphs.

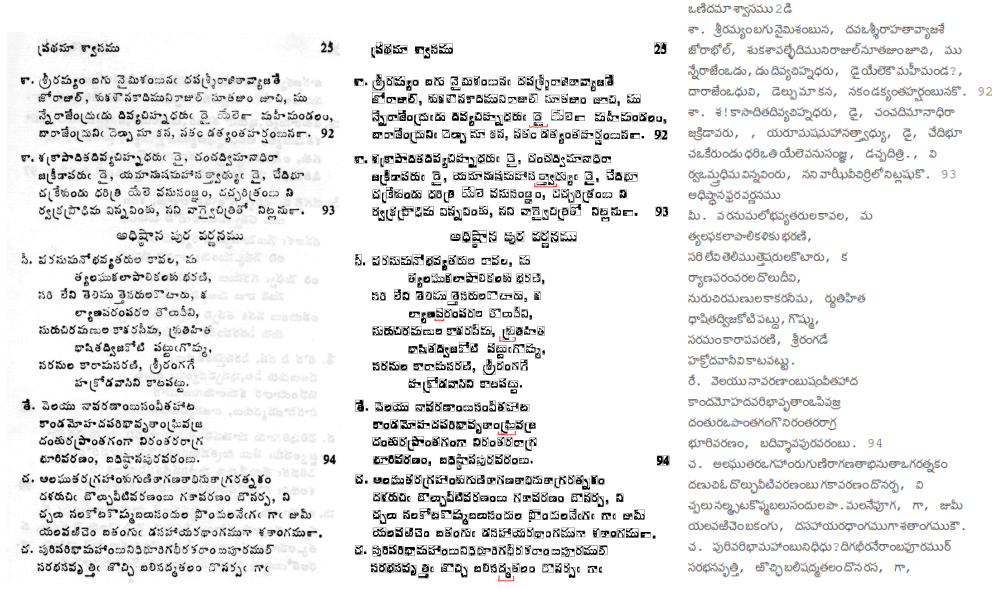


Fig. 8. Input sample image, The valid glyphs extracted from it and OCR output

S.No.	Book Name	Page No.(%)	DRISHTI System Error Rate(%)	Proposed System Error Rate(%)	Reduction in error rate
1	Ooragaya Navvindi	14	34.98	22.40	12.58
2	Vasucharitramu	40	38.69	26.41	12.28
3	Vasucharitramu	25	31.69	21.69	10.00
4	Sasaanka Vijayamu	3	33.33	24.34	8.99
5	Sasaanka Vijayamu	38	47.62	38.65	8.97
6	GVS Navalalu- Kathalu	1	25.45	16.87	8.58
7	Vasucharitramu	114	25.60	17.11	8.49
8	Sasaanka Vijayamu	122	28.17	19.85	8.32
9	GVS Navalalu- Kathalu	32	30.14	23.18	6.96
10	Ooragaya Navvindi	26	30.46	23.97	6.49

Table 1. Comparison of DRISHTI system and Proposed system error rates

We have done experiments over different Telugu books and novels with the proposed approaches. We have reported the average error rate for some books, using current approaches along with previous error rates (by DRISHTI with Connected component approach alone) in the Table 2.

From Table 2, it can be observed that, the error rate is reduced significantly in first two books. Because, in these books the print quality is so poor which causes lots of broken glyphs. These are segmented, merged and recognized correctly by using proposed approaches. We can also observe the reduction in error rate for next two books. Over running the proposed algorithm on 836 pages an average error rate reduced by 3.39 %.

S.No.	Book Name	Number of Pages	DRISHTI System Error Rate(%)	Proposed System Error Rate(%)	Difference
1	Vasucharitramu	203	27.42	22.31	5.11
2	Sasaanka Vijayamu	181	30.24	26.29	3.95
3	GVS Navalalu- Kathalu	125	21.96	18.64	3.32
4	Ooragaya Navvindi	327	18.06	16.88	1.18

Table 2. Comparison of DRISHTI system and Proposed system average Error rate over the books

6 Conclusion

In this paper, we proposed two methods to improve symbol or glyph segmentation in Telugu OCR systems. The first proposed method is Two level symbol segmentation method, which is able to extract many broken characters correctly. The second method is based on bounding box analysis. From experiments which we performed over 5000 pages of Telugu documents observed that proposed methods do result in significant improvement in OCR performance on degraded pages.

References

1. Negi, A., Bhagvati, C., Krishna, B.: An ocr system for telugu. In: ICDAR. (2001) 1110–1114
2. Vishwabharat@tdil: Journal of language technology (July 2003)
3. Bhagvati, C., Ravi, T., Kumar, S.M., Negi, A.: On developing high accuracy ocr systems for telugu and other indian scripts. In: Language Engineering Conference. (2002) 18
4. Patel, A., Burra, S., Bhagvati, C.: Svm with inverse fringe as feature for improving accuracy of telugu ocr systems. In: Progress in Intelligent Computing Techniques: Theory, Practice, and Applications. (2016) 10
5. Jignesh Dholakia, A.N., Mohan, S.R.: Progress in gujarati document processing and character recognition. *Advances in Pattern Recognition* (2010) 73–95
6. Baird, H., Kahan, S., Pavlidis, T.: Components of an omnifont page reader. In: Proc. ICPR. (Oct. 1986) 34–348
7. Schölkopf, B., Smola, A.J., Williamson, R.C., Bartlett, P.L.: New support vector algorithms. *Neural Comput.* **12**(5) (may 2000) 1207–1245
8. : OpenCV 2.4.9.0 documentation - Support Vector Machines
9. Chang, C.C., Lin, C.J.: Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.* **2**(3) (May 2011) 27:1–27:27
10. Gonzalez, R., Woods, R.: Digital Image Processing. Addison-Wesley (1993)