# Segmentation of Telugu Touching Conjunct Consonants Using Overlapping Bounding Boxes

J. Bharathi
Associate Professor, Department of Electronics and Communications Engineering
Deccan College of Engineering and Technology
Hyderabad, India
jagashettyb@gmail.com

Dr. P. Chandrasekar Reddy
Professor, Department of Electronics and Communications Engineering
JNTU College of Engineering
Hyderabad, India
drpcsreddy@gmail.com

*Abstract*— **Telugu is an ancient historic language. It is spoken by about 84.6 million people of Andhra Pradesh. The script has circular orthography with few horizontal and slant strokes. Huge literature exists for this language in printed form which needs to be preserved by scanning and converting it into editable form. Segmentation of touching characters is a major issue in any OCR system. Segmenting the words into individual glyphs by Connected Component Analysis yields poor results due to touching characters. Touching conjunct consonants is the major component which needs to be properly addressed for improving the accuracy of an OCR system. In this paper an overlapping bounding box approach is presented for segmenting the conjunct consonants along with an algorithm for identifying the correct touching location. An accuracy rate of 91.27% is achieved.**

***Keywords- Conjunct Consonants, Overlapping Bounding Boxes, Connected Components, Telugu OCR, Touching Point Identifier, Partial Projection Profile.***

## I. INTRODUCTION

### A. Basic Telugu Character Set

India is a multi lingual country. About 150 different languages are spoken in India. Out of these, 18 major languages have been constitutionally recognized. These languages have been divided into Indo-Aryan, Dravidian, Tibeto-Burman and Austro-Asiatic families. Telugu falls into Dravidian family and is spoken in the State of Andhra Pradesh. This is derived from Brahmi script.

A syllable is a unit of organization for a sequence of speech sounds. The Telugu language is syllabic in nature. Phonetic correspondence exists between the spoken and the written syllables. This avoids the elaborate spelling rules which is the case with Roman Script. However the syllables and the language is governed by script grammar.

The units of writing are known as Akshara or Character. There are 18 vowels (Achulu) [Fig.1] out of which two are not in use and 38 consonants (Hallulu) [Fig.2]. All the primary consonants have an implicit /a/ vowel combined in them. A “halant” has to be combined with the consonants to remove the implicit /a/ sound. A consonant without an implicit vowel is said to be in pure form.

అ ఆ ఇ ఈ ఉ ఊ ఋ ౠ ఎ ఏ ఐ ఒ ఓ ఔ అం అః

Figure 1. Vowels (Achulu) (7th and 8th characters are not in use)

క ఖ గ ఘ ఙ
చ ౘ ఛ జ ౙ ఝ ఞ
ట ఠ డ ఢ ణ
త థ ద ధ న
ప ఫ బ భ మ
య ర ల వ శ ష స హ ళ క్ష ఱ

Figure 2. Consonants (Hallulu)

## *B. Combined characters*

Telugu script is written from left to right. The vowels and consonants are combined according to a set of rules to obtain written script. The vowel modifiers (vowel matras or guninthalu) are added to the pure form of consonants to produce consonant-vowel combinations [Fig.3]. These shapes bear more resemblance to the preceding consonant than the following vowel which modifies the consonant shape either at top, middle or bottom [Fig. 4].

క కా కి కీ కు కూ కృ కౄ కె కే కై కొ కో కౌ కం కః

Figure 3. Vowel modifiers (vowel matras) attached to consonant ka – (Gunithalu)

| | | |
|---|---|---|
| క్ + అ = క | వ్ + ఈ = వీ | క్ + క = క్క |
| క్ + ఇ = కి | క్ + ఉ = కు | వ్ + య = వ్య |
| క్ + ఎ = కె | క్ + ఐ = కై | ర్ + త = ర్త |
| **c + v = cv** | **c + v = cv** | **c + cv = c(cv)** |

Figure 4. Formation of consonants

The consonants have secondary forms [Fig.5], which in combination with primary consonants form conjunct consonants. These have similar shapes as primary consonants except for eight consonants [Fig.6].

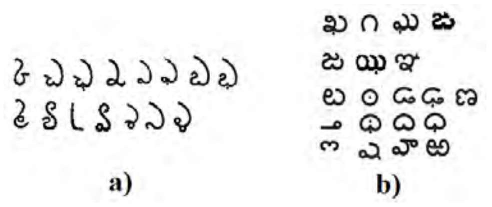

Figure 5. Secondary form of consonants – a) written in bottom and middle zones b) written in bottom zones

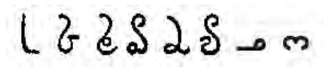

Figure 6. Secondary form of consonants which do not resemble their primary form

The secondary form of consonants in Fig.5a occupy both middle and bottom zones and the forms in Fig.5b are written in smaller size and occupy only bottom zone.

The syllables are CV combinations and the above can be represented as C*V, where C* denotes number of consonants [1],[2] [Fig.4,6]. In practice the consonant cluster at the most have 5 consonants [3] [Fig.7], the most common case being either one or two.

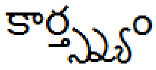

Figure 7. A long Consonant cluster - kaartsnyam

In the case of vowel modifiers, the modifiers are attached to the consonants. The secondary form of consonants have separate shapes written after the consonants (only in a few alternative forms, these are written before or to the left of the consonants e.g. 'ra' vattu) in comparison to vowel modifiers which are attached to consonants.

## *C. Telugu OCR*

The shapes of aksharas are complex and have segments of circular shapes, slant and horizontal strokes. They are formed by the combinations of glyphs, which have independent shapes. It is to be noted that there is no standard for glyphs. Broadly a connected component is considered as a glyph which can be used as basic unit for the purpose of pattern recognition.

The aksharas are written non-linearly i.e., the width of aksharas is not constant. The vowel modifiers are attached to the consonants either at top, bottom and middle zones. This increases the number of characters to be recognized.

The Telugu script lines may be divided into top, middle and bottom zones [Fig.8]. Most of the aksharas occupy the middle zone and some parts of aksharas extend into top and bottom zones [Fig.9].

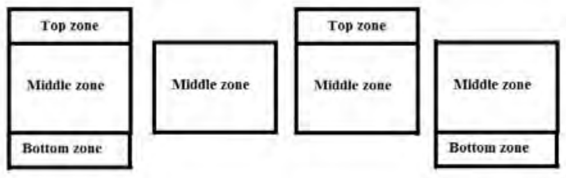

Figure 8. Character Model based on zones

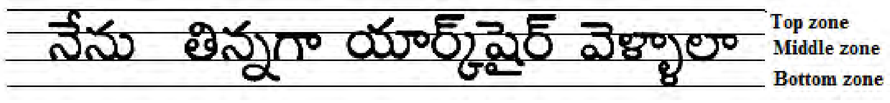

Figure 9. Characters extending into top, middle and bottom zones

Telugu script has no separate capital and small case characters. This reduces the number of characters to be recognized in the recognition stage.

In the above Fig.8 top, bottom zones usually contain vowel modifiers (Guninthalu) as well as the secondary form of consonants as conjuncts. Unlike in Devanagari, some of the vowels and consonants extend into top and bottom zones also. The knowledge of the script level grammar resolves some of similar and confusing characters enhancing the recognition rate.

A set of unique 386 glyphs have to be recognized based on this connected component unit concept. Many researchers have adopted this approach for the OCR engines [4,14].Poor quality paper, ink leads to characters touching at various places in a word. In Samyukthaksharas or consonant clusters, this touching renders recognition difficult leading to poor recognition rates.

Due to improper binarization, poor print quality, low grade paper, smear and seeping of ink the characters touch with the neighboring characters at various places. The unique orthography of Telugu script and script rules make the conjunct consonants prone to touching and fusing [Fig.10]. Among the touching characters, touching conjunct consonants account for about half of them. The segmentation becomes more challenging and needs extra effort as the bounding boxes of the first and second character overlap.

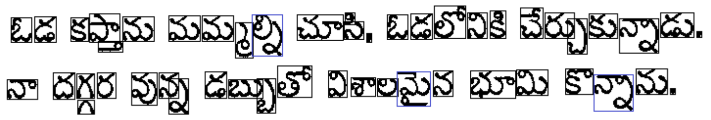

Figure 10. Touching characters having single bounding box shown in blue color

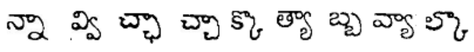

Figure 11. Few touching conjunct consonants

## II. LITERATURE SURVEY

Many segmentation algorithms are described by R.G. Casey and Eric Licolinet [5]. K.Ohta et al considered a profile based approach for segmenting a line. These approaches are effective for non cursive writing systems and hand written systems. The ratio of second derivative of the profile based patterns to its height is used for identifying segment separation in [5].

Y. Lu proposed a peak to valley function by extracting valleys between successive peaks. Touching characters are identified with an average function [5].

Su Liang et. al [6] proposed segmentation algorithm for touching characters in printed English documents based on both pixel and projection profiles. A discrimination function is used to identify the potential cutting candidates. The dynamic recursive algorithm cuts at the potential candidate points and classifies the resulting characters until all the characters are classified. Contextual information and spell checking is used to correct the errors in the segmentation of combined characters having more than two touching characters.

Min-Chul Jung and others [7] used side profiles to segment the touching characters. A Line Adjacency Graph analysis is preformed on words, and then typographical zone analysis is done. Four side profiles (right, left, upward, downward respectively) of the touching characters are extracted. Histograms are produced for all the directional profiles. The histograms have character width or height on horizontal axis and number of white pixels on vertical axis. The distances of the histogram are matched with the histograms of the prototypes normalizing the width or height of the character for font independency.

$D[p,q] = \Sigma |x_{ip}-x_{iq}|$

Where p = prototype

q = query

$x_{ip}$ = value of vertical axis in histogram of prototype

$x_{iq}$ = value of vertical axis in histogram of query

Akihiro Nomura et al., detected and segmented touching characters in mathematical expressions from several scientific journals [8]. The segmentation technique is a recognition based segmentation method. A single character is matched to one of the four corners of the touching character. After matching character is found, the residual is created and once again it is also identified as above. The two individual characters are again recognized for verification. The computational efficiency is improved by a simple clustering technique.

North Indian scripts are hybrid in nature with stroke being dominated by curves. Segmentation of this is achieved by exploring the statistical properties of zone information using profile based methods. [9,10,11,12].

Veena Bansal and RMK Sinha [11] presented a two pass algorithm for the segmentation and decomposition of Devanagari composite characters or symbols into their constituent symbols. This algorithm uses structural properties of the script and in the first attempt the words are segmented into separate characters or composite characters and based upon the statistical properties, the height and width of the character is determined and as a second step the characters are further segmented.

M.K.Jindal and others [12] have proposed segmentation of touching characters of Gurmukhi characters in the upper zone by using the structural properties of the script.

L.P Reddy and others [13] have proposed a canonical syllable method consisting of base symbol in middle zone. Superscript and subscript in top and bottom zone respectively. The basic symbol is preceded and succeeded by half or full form extending the middle and bottom zones. The touching point is identified using the vertical split threshold in the lower profiles based on the topological properties.

Santanu Choudary in his report [15] has enlisted the issues in segmenting the touching characters of Indic Scripts.

## III. PRESENT METHOD

### A. *Overlapping Bounding box approach*

The proposed method is motivated by the observation that each individual character in the conjunct consonant can be bounded by a rectangle, but with overlapping areas. The overlapping bounding boxes can be identified from the image using the projection profiles of the individual syllables or the partial projection profiles of the touching conjunct consonant.

The vertical profile will not show the zero valley in the vertical profile, because of the overlapping orthography of the conjunct consonant. The connected component analysis will yield the separated components, if they are not touching with each other. However if the glyphs are touching, the connected component analysis fails to segment the glyphs.

### B. *Isolation of bounding box*

The first character of the touching Samyukthakshara (consonant cluster) is identified using the partial vertical profile and the partial horizontal profile. In Figure13 the bottom and the right side of the first bounding box need to be identified for segmenting the first character.

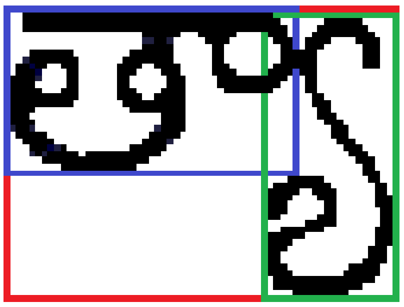

Figure 12. Overlapping bounding boxes of the touching conjunct consonant cluster

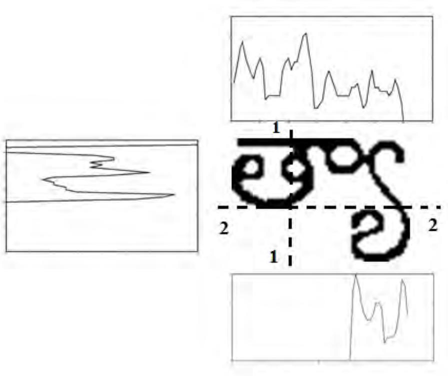

Figure 13. Partial projection profiles. Left side: left side profile at 1-1. Top: top vertical profile at 2-2. ottom: bottom vertical profile at 2-2.

A partial projection profile is defined as the projection profile of a part of the character image. The left side partial horizontal projection profile that includes the bottom most pixels of the first character needs to be calculated at ‘α’ times the width of the combined character. It is observed that a suitable value of ‘α’ is found to be 0.3 (at section 1-1 in Fig.14).

$$php_l = \sum_{j=1}^{\propto w} I_b(i,j)$$

Where w = width of the image

$I_b(i, j)$ = binary image

α = 0.30

The partial horizontal projection profile has zero height in the lower part and non-zero height in the upper part. The bottom line of the bounding box is the line dividing the upper non-zero part and lower zero part of partial horizontal projection profile.

The right side of the first bounding box is found by using the partial vertical projection profile of combined character at section 2-2 in Fig.14.

$$pvp_t = \sum_{i=1}^{bb1\_h} I_b(i,j) \forall j$$

Where bb1_h = height of bounding box 1

$I_b(i,j)$ = binary image

It is observed in the partial vertical projection profile, a valley point occurs at the touching point of the two characters. The width of touching pixels is more than twice the stroke width of the character. However, because of small fonts and binarization inaccuracies vertical profile shows valley points at locations other than the touching point. A novel touching point algorithm is proposed to identify the correct location.

## C. *Touching point location identifier algorithm*

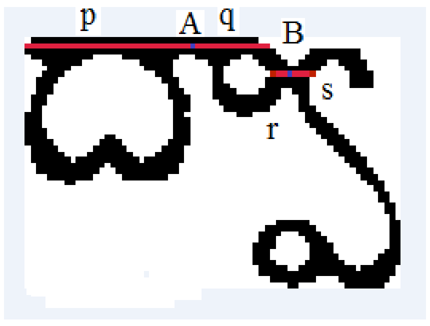

Figure 14. Location of correct touching point

In Telugu script few characters have horizontal strokes. The width of these horizontal strokes is not uniform after binarization and may have neck regions in them, which result in valley points in vertical projection profiles. Thus they are potential points for false touching while searching valley points for touching locations. These locations need to be eliminated for correctly identifying the touching point.

Consider the horizontal stroke in the Fig.15. The neck region at point A is a false touching point and the midpoint B is the correct touching point. A horizontal line at midpoint of the neck region A and at B are extended on the both ends of the text pixels. The lengths of the line 'p' and line 'q' (count of pixels) at point A and lengths of lines 'r' and 's' at point B are compared. The lengths 'p' and 'q' are more than twice the stroke width whereas at least one of lengths 'r' and 's' is less than twice the stroke width. This logic is incorporated into the algorithm for identifying the correct touching points.

### *D. Algorithm for segmentation of touching conjunct consonants*

The vertical projection profile for the partial image at section 2-2 is computed. Assuming that the lowest valley point is the correct touching point, the pixels contributing to the vertical projection profile are identified. At the midpoint of the pixels a scan line is passed and the number of continuous text pixels on the left side and on right side of the scan line is counted. If the count on each side is more than twice the stroke width it is termed as false point and next lowest valley is considered as the correct touching point. The above steps are repeated until correct point is located. If the count on the one side is less than twice the stroke width and the count on the other side is more than this, it is termed as correct touching point and the search is stopped.

1. Read the image

   $I \xleftarrow{read} image\ file$

2. Convert to gray scale image

   $I_g \xleftarrow{convert} I$

3. Binarize the image

   $I_b \xleftarrow{binarize} I_g$

4. Calculate partial horizontal projection profile of left side of the image divided at α times the width image

   $php_l = \sum_{j=1}^{\propto w} I_b(i,j)$

   Where $w$ = width of the image

   $I_b(i,j)$ = binary image

   $\alpha = 0.30$

5. Find *hbl*, the width of zero height of $php_l$ at bottom of the profile

   $php_l(i) = 0;\ i = h\ to\ hbl$

   $bb1\_h = h - hbl$

   Where *bb1_h* = height of bounding box 1

6. Calculate $pvp_t$, the partial vertical projection profile of image from top to *bb1_h*

   $pvp_t = \sum_{i=1}^{bb1\_h} I_b(i,j) \forall j$

   Where *bb1_h* = height of bounding box 1

   $I_b(i,j)$ = binary image

7. Remove width of zero height at end of $pvp_t$, then find the lowest valley point in the $pvp_t$

   $tp = argmin\big(pvp_t(i)\big)$

8. Evaluate the touching point criteria for the point tp

   Evaluate ***touching point identifier algorithm***

   tp criteria = true

   where *tp* = touching point

9. Calculate $pvp_b$, the partial vertical profile of image bottom from *bb1_h* to *h*

   $pvp_b = \sum_{i=bb1\_h}^{h} i_b(i,j)\ \forall\ j$

10. Find *w_bb2,* the width of non-zero height of $pvp_b$ at the end of the profile

    $pvp_b(i) > 0;\ i = w\ to\ w\_bb2$

11. Segment the bounding box 1

    Height = 1 to (*h* – *hbl*)

    Width = 1 to *tp*

12. Replace the pixels of bounding box 1 with white pixels

```
13. Segment the bounding box 2
        Width = $w$-$w\_bb2$ to $w$
14. Find the horizontal profile of bounding box 2 and remove width of zero height at top
```

*E. Touching point identifier algorithm*

```
1. Initialize $i$ to the lowest valley point
       $i = 1$
2. Identify the starting point of the text pixels along the vertical line at $i^{th}$ valley point
       $y_k = start\ of\ text\ pixel$
3. Compute half depth of the valley in the partial vertical projection profile
       $v_i = i^{th} valley$
       $tp_{yi} = y_k + v_i/2$
4. Move left side up to white pixels and count the text pixels
       $count_{left} = tp_{xi} - a_{xi}$
5. Move right side from $tp_i$ up to white pixels and count the text pixels
       $count_{right} = b_{xi} - tp_{xi}$
6. If $count_{left}$ and $count_{right} > 2*stroke\_width$
       Not a correct location,
       Go to next valley point, $tp_{i+1}$, go to 1
   Else if $count_{right} < 2*stroke\_width$ and $count_{left} > count_{right}$
       Correct location, exit
   Else if $count_{left} < 2*stroke\_width$ and $count_{right} > count_{left}$
       Correct location, exit
```

*F. Stroke width*

The stroke width of a character image is usually not uniform and an average width is to be calculated. A simple procedure is described below.

The character image is thinned using morphological operations [Fig.16. a,b]. The length of the character is calculated by counting the one pixel thin character. The total number of pixels in a character is also counted. The average stroke width is the number of pixels divided by the length rounded to the nearest integer.

Average Stroke width = Number of pixels in the character/length of thinned character.

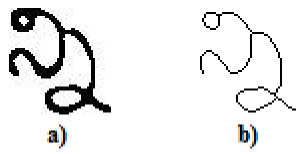

Figure 15. Touching character a) before thinning b) after thinning

*G. Algorithm for calculation of average stroke width*

```
1. Read the binarized character image
       $I \xleftarrow{read} Image\ file$
2. Thin the image using morphological operations
       $I_{thin} \xleftarrow{thin} morph\ op(I_{thin})$
3. Calculate the length of the character
       $length = count(I_{thin})$
4. Calculate total number of text pixels in the image
       $total\ pix = count(I)$
5. Average stroke width is total number of pixel divided by the length of character
       $stroke\ width = total\ pix/length$
```

*H. Dataset*

A collection of books and documents is created to test our algorithm. Three books which were already scanned, binarized and de skewed are downloaded from Digital Library of India. Children's books, story books and normal documents are scanned at 300 dpi, binarized and de skewed. These documents were segmented for lines and characters. The fonts are Anupama, Priyanka, Goutami and Hemalatha; and the font sizes are 10, 12 and 14. A total of 187,084 characters are obtained and these consisted 1,776 touching conjunct consonants.

## IV. Experiments and results

The touching character segmentation algorithm proposed is verified with 188 documents consisting of children's books, short stories, fine printed Telugu books and documents from Digital Library of India. The scanned document images are binarized and the skew is removed. The lines and words are segmented using horizontal and vertical profiles respectively. The characters are segmented using connected component approach.

TABLE I. Results

| | |
|---|---|
| Total documents | 188 |
| Total characters | 187,084 |
| Total touching characters | 3,149 |
| Conjunct consonants | 1,776 |
| % of conjunct consonants | 56.40% |
| Correctly segmented | 1,621 |
| % of success | 91.27% |

All the touching characters are grouped manually. These touching characters are further grouped into conjunct consonants and others. The conjunct consonants are segmented using the present proposed algorithm.

We achieved an appreciable success rate of 91.27%. The failures are found to be some characters which have projections into bottom zones and second characters projecting into middle zone. The bounding boxes for these cases could not be properly identified using the partial horizontal projection profiles at 0.3 times the width. At present these characters need specific algorithms to segment them properly taking the specific shape into consideration. Some of the rejects also include the alternative forms of the conjunct consonants.

## V. Conclusions

The proposed algorithm has achieved high success rates of 91.27% for the touching consonant conjuncts of the Telugu script. Some of the rejects need shape specific algorithms for segmenting them.

## References

[1] Atul Negi, Kavi Narayana Murthy, 2004. "Issues of Document Engineering in the Indian Context", University of Hyderabad, India, Research Report supported by University Grants Commission under the Project "Language Engineering Research" under UPE scheme, pp. 1-11.

[2] Wong, K. Y., Casey, R. G., Wahl, F. M. 1982. "Document Analysis System", IBM Journal of Research and Development, Vol. 26, No. 6, pp. 647–656.

[3] Edward Hill, C. 1991. "A Primer of Telugu Characters," Manohar Publications, New Delhi. Can be viewed online at Digital South Asia Library (DSAL), University of Chicago.

[4] Lakshmi, C. V., Patvardhan, C., 2003. "Optical Character Recognition of Basic Symbols in Printed Telugu Text", IE(I), Journal-CP, Vol. 84, Nov 2003, pp. 66-71. 9999

[5] Richard Casey, G., Eric Licolinet, 1996, "A Survey of Methods and Strategies in Character Segmentation", IEEE Trans. In Pattern Analysis and Machine Intelligence, Vol. 18, No. 7, July 1996, pp. 690–706.

[6] Su Liang, Shridhar, M, Ahmadi, M. 1994. "Segmentation of Touching Characters in Printed Document Recognition", Pattern Recognition, Vol. 27, No. 6, pp. 825–840.

[7] Min-Chil Jung, Yong-Chul Shin, Srihari, S. N., 1999. "Machine Printed Character Segmentation Method using Side Profiles", IBM Journal of Research and Development, Vol. 26, No. 6, pp. 647–656.

[8] Akihiro Nomura, Kazuyuki Michishita, Seiichi Uchida, Masakazu Suzuki, 2003. "Detection and Segmentation of Touching Characters in Mathematical Expressions", IEEE Computer Society, ICDAR 2003, pp. 126–130.

[9] Hoffman, R. L., McCullough, J. W., 1971. "Segmentation Methods for Recognition of Machine Printed Characters", IBM Journal of Research and Development, Vol. 15, Issue 2, Mar 1971, pp. 153–165.

[10] Giovanni Seni, Edward Cohen, 1994. "External Word Segmentation of off-line Handwritten Text Lines", Pattern Recognition, Vol. 27, No. 1, Jan 1994, pp. 41–52.

[11] Veena Bansal, Sinha, R. M. K., 2002, "Segmentation of touching and fused Devanagari Characters,", Pattern Recongnition, Vol. 35, No. 4, pp. 875–893.

[12] Jindal, M. K., Sharma, R. K., Lehal, G. S., 2009. "Segmentation of Touching Characters in Upper Zone in Printed Gurmukhi Script", Compute '09, Proc. Of 2nd Bangalore Annual Compute Conference, Article 9, Jan 9,10 Bangalore.

[13] Pratap Reddy, L., Ranga Babu, T., Venkata Rao, N., Raveendra Babu, B., 2010. "Touching Syllable Segmentation using Split Profile Algorithm", IJCSI, Vol. 7, Issue 3, No. 9, Nov 2010, pp. 17–26.

[14] Atul Negi, Chakravarthy Bhagavati, Krishna, B., 2001. "An OCR System for Telugu", Sixth International Conference on Document Processing, ICDAR 2001, pp. 1110–1114.

[15] Santanu Choudary (Ed), "OCR Technical Report for the Project: Development of Robust Document Analysis and Recognition System for Printed Telugu Scripts", Project sponsored by Ministry of Communication and Information Technology, July 2008.

## Authors Profile

J. Bharathi received her B.Tech degree in Electronics and Communication Engineering from Acharya Nagarjuna Unininiversity, Guntur, India. She received her M.Tech degree in Digital Systems and Computer Electronics from Jawaharlal Nehru Technological University, Hyderabad, India. She joined as faculty member in

Electronics and Communication Engineering Department, Deccan College of Engineering and Technology, Hyderabad, India and is currently working as Associate Professor. Her research interests include Image Processing, Speech and Signal Processing.

Dr. P. Chandra Sekhar Reddy received his B.Tech. degree in Electronics and Communication Engineering from Jawaharlal Nehru Technological University, Hyderabad, India and M.E. from Bharatiyar University, Coimbatore. He received his M.Tech and Ph.D from Jawaharlal Nehru Technological University, Hyderabad, India. He joined as faculty in JNTU, Anantapur. Currently he is working as Professor Co-ordination in JNTU, Hyderabad, India. He is an author of numerous technical papers in the Fields of High Speed Networking and Wireless Networks. His research interests include Mobile and Wireless Communications and Networks, Personal Communications Service and High Speed Communications and Protocols.