Learning Segmentation of Documents with Complex Scripts

K.S. Sesh Kumar, Anoop M. Namboodiri, and C.V. Jawahar

Centre for Visual Information Technology,
International Institute of Information Technology, Hyderabad, India.

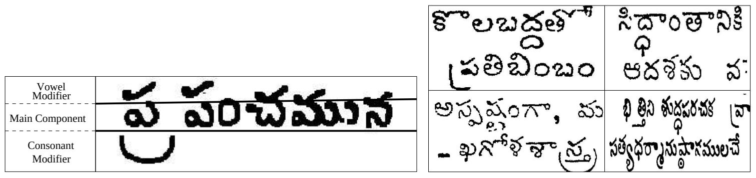
Abstract. Most of the state-of-the-art segmentation algorithms are designed to handle complex document layouts and backgrounds, while assuming a simple script structure such as in Roman script. They perform poorly when used with Indian languages, where the components are not strictly collinear. In this paper, we propose a document segmentation algorithm that can handle the complexity of Indian scripts in large document image collections. Segmentation is posed as a graph cut problem that incorporates the apriori information from script structure in the objective function of the cut. We show that this information can be learned automatically and be adapted within a collection of documents (a book) and across collections to achieve accurate segmentation. We show the results on Indian language documents in Telugu script. The approach is also applicable to other languages with complex scripts such as Bangla, Kannada, Malayalam, and Urdu.

1 Introduction

Document image understanding algorithms are expected to work with a document, irrespective of its layout, script, font, color, etc. Segmentation aims to partition a document image into various homogeneous regions such as text blocks, image blocks, lines, words etc. [1]. Page segmentation algorithms can be broadly classified into three categories: bottom-up [2, 3], top-down [4, 5], and hybrid [6] algorithms. The classification is based on the order in which the regions in a document are identified and labeled. The layout of the document is represented by a hierarchy of regions: *page*, *image* or *text blocks*, *lines*, *words*, *components*, and *pixels*. The traditional document segmentation algorithms give good results on most documents with complex layouts but assume the script in the document to be simple as in English. These algorithms fail to give good results on the documents with complex scripts such as African, Persian and Indian scripts.

1.1 Challenges in Segmentation of Indian Language Documents

In the recent past, the number of document images available for Indian languages has grown drastically with the establishment of Digital Library of India [7]. The digital library documents originate from a variety of sources, and vary considerably in their structure, script, font, size, quality, etc. Of these, the variations in the structure of the script are the most taxing to any segmentation algorithm.



(a) (b)

Fig. 1. Complexities of Telugu Script: (a) spatial distribution of connected components, (b) Non-uniform spacing between lines and components.

In this paper we deal with documents in Indian languages such as Telugu, Tamil, Bangla, and Malayalam, which have similar script structure. The complexity of these scripts lie in the spatial distribution of the connected components. Unlike English, most characters in Indian scripts are made up of more than one connected component. These connected components do not form meaningful characters by themselves, but when grouped together, form different characters in the alphabet. The components of a character can be classified into:

Main Component: It is either a vowel, a consonant or a truncated consonant. The main components of characters within a line are nearly collinear.

Consonant Modifier: In the above scripts, a character could be composed of two consonants, the main component and a *consonant modifier* or *half consonant.* Spatially, the consonant modifier could be to the left, right or bottom of the main component, and hence lie within a line, or below it.

Vowel Modifier: A character also can have a vowel modifier, which modifies the consonant. When the vowel modifier does not touch the main component, it forms separate component, which lies above the main component.

Figure 1(a) shows the spatial distribution of components in Telugu script. Due to variations in spatial distribution of the components within a the line structure is non-uniform. This is the primary reason for the failure of many traditional segmentation algorithms. Due to the positional variation of a modifier component, the task of assigning it to a line above it or below it is ambiguous. Heuristics such as assigning a component to its nearest line might fail because the distances between the components vary depending on the font style, font size and typeset as shown in figure 1(b).

Variations in scanned books are also introduced due to the change in writing style of certain character over time, which need to be taken into account while segmenting a document. Most of the old books are typeset by human and not machine and hence it is difficult to specify a consistent distance between the components. We introduce a *Spatial Language Model* that encapsulates the local variations in component distribution and use it to perform segmentation.

2 Document Segmentation using Graph cuts

Segmentation is the process of partitioning an image into regions with homogeneous properties. Ideally, pixels forming a semantically meaningful object should be grouped together. Traditional image segmentation approaches usually group pixels using low level cues such as brightness, color, texture, and motion. However, the end goal is to have segments that correspond to a particular object, which is often measured using mid and high level cues such as symmetry of objects and object models. This warrants a framework, where both low level cues and high level cues are integrated to segment an image. For document images, this would mean the use of higher level structure of the document in determining the grouping of connected components to form characters, words, and lines.

Segmentation can thus be posed as an optimization problem, where low level cues are used to group the pixels into regions that represent an object, which can be analyzed using high level cues. A variety of methods have been proposed that pose segmentation as an optimization problem. Graph cuts is one of the methods used to perform image segmentation. It promises a near optimal solution; i.e., a solution at a known distance from the global optimum. To apply graph cuts to document images, a graph is built using either the pixels or the connected components of the image as nodes, which are linked to its neighbors through edges. During segmentation, a cut is defined on the graph, which labels the pixels or components on either side of the cut as belonging to different segments. Boykov et al. [8] proved that minimizing an energy function is equivalent to minimizing the cost of cut on the graph.

Graph cuts form an effective combination of top-down and bottom-up approaches for the segmentation [9, 10] problem. They also provide a framework for learning the shape priors [11] and use them to perform effective segmentation. The traditional page segmentation algorithms do not provide the ability to learn from or adapt to the nature of images, given a large collection of data. We pose the page segmentation problem as an optimization problem, which minimizes the energy calculated using graph cuts. This provides the ability to learn the layout parameters in an incremental fashion.

We propose a segmentation algorithm that partitions a document with complex scripts. We initially assume that the layout not very complex, contains only text, and the document is skew corrected [12, 13]. Later, we show how to extend the algorithm to work with complex layouts also.

Each connected component forms a node that is linked to its k nearest neighbors, where the nearest neighbors are calculated using the Euclidean distance between the centers of their bounding boxes. The value of k is selected such that each component is connected to all the components surrounding it spatially. For Telugu, the value of k that yielded the best results was 8. Once the neighbors are identified, a graph is constructed, and an initial estimate of the lines is obtained using the horizontal projection profile. The initial cut ensures that a majority of the components that belong to a particular line remain connected. Later we replace the projection profile based method with simple heuristics on components to arrive at a robust initial segmentation of documents with complex layouts.

2.1 Energy Function

Our goal is to assign a line number (label) to every connected component within a document. All the connected components that belong to a document ($\mathcal{C}$) need to be partitioned into mutually exclusive and collective exhaustive subsets, $\mathcal{C}_i$, where i denotes the line number. The goal is to find a labeling f that labels each component $c \in \mathcal{C}$ a label $f_p \in \mathcal{L}$, where $\mathcal{L}$ denotes the label set (here the line numbers). The labeling should be done in such a way that it is piecewise smooth. In this framework, a labeling f is computed so as to minimize the total energy:

$$E(f) = E_{smooth}(f) + E_{data}(f), \tag{1}$$

where E_{smooth} gives the measure of the smoothness of the labeling, while E_{data} gives a measures of the consistency of labeling with the observed data.

Smoothness Term: If a component along with k nearest neighbors belong to a single line, the labeling is considered extremely smooth and the contribution due to the component to the E_{smooth} term decreases.

$$E_{smooth}(f) = \sum_{(c,c') \in \mathcal{N}} V_{(c,c')}(c, c'), \tag{2}$$

where $\mathcal{N}$ denotes the set of neighboring connected components within a document. We will require the term $V_{c,c'}$ to be a metric for the expansion algorithm to give near optimal solution using the graph cut.

Data Term: The data term is one of the most important measure in the calculating the energy of the segmentation algorithm. This term enables the improvement of the segmentation algorithm by using apriori information, which is available in the form of *spatial language models*. This term gives a measure of disagreement of labeling of a connected component to a line above it or to the one below it.

$$E_{data}(f) = \sum_{c \in \mathcal{C}} D(f(c)) \tag{3}$$

The calculation of $D(f(c))$ denotes the disagreement of the observed data and the apriori information available. For instance if a particular type of connected component, according to the apriori information available through the spatial language model, belongs to the line above it in spite of being nearer to the line below it, the Disagreement of labeling the connected component to the line above it is less than the labeling of the connected component to the line below it. Thus we attain better segmentation with the availability of good spatial language models of the document. However, if the E_{data} term is a constant function for any assignment of labels to a connected component c, the graph cut is same as a distance based heuristic, i.e., assign the connected component to the line that it is nearest to.

2.2 Graph Construction

The problem of segmenting a text block into lines can be viewed as the grouping of connected components into clusters. Each of the cluster of connected components define a line. Hence, the first and the foremost step is to find the number of clusters. The number of lines within a text block can be calculated using the projection profile formed by the horizontal projection of foreground pixels. The number of peaks within the profile gives the number of lines, n_l into which the text block is to be segmented.

Graph cuts need an initial labeling of the connected components. The text block is first segmented into n_l lines using the projection profile based approach. There exist connected components that lie between two lines. All these components are assigned to the lines that lie below it. Thus the text blocks are segmented into lines, where there are chances that some of the component assignments could be wrong. However, an initial labeling of the components is achieved through this process. Now the cost of graph cuts could be used to perform changes of labeling such that the cut is minimal and the energy calculated using the labeling is minimized.

We start with the initial set of labels for all the connected components computed from the projection-based cut. A graph is constructed using these initial set of labels. We know all the components that belong to two consecutive lines. A graph is constructed for every pair of consecutive lines with two extra nodes, (α, β), representing the labels of the nodes. Every pair of nodes that represent neighboring components (c, c') are linked by and edge, $e(c, c')$ with the weight $V_{(c,c')}(c, c')$, a metric in the label space (i.e. the distance defined between the labels of the two components). It follows a Potts model defined by:

$$V(\alpha, \beta) = K.T(\alpha \neq \beta) \tag{4}$$

where $T(.)$ is 1 if its argument is true, and 0 otherwise.

Each of the components is also linked to one the two nodes, α, and β, which represent the labels of the lines. The weights of these edges, denoted by t_c^α and t_c^β, are calculated using the following equation:

$$t_c^\alpha = D_c(\alpha) + \sum_{q \in \mathcal{N}_p; q P_{\alpha\beta}} V(\alpha, f_c), \tag{5}$$

where $D_c(\alpha)$ is the distance of the connected component to the nearest component of the line with the label α, which is calculated using the spatial language models. t_c^β is also calculated in a similar manner.

Now we have a graph that is constructed using the connected components as nodes, along with two extra nodes with labels of the two lines. The α-β swap algorithm, proposed by Boykov et al. [8], is used to perform the graph cut. If the cut separates the node representing the component c with the node α, it given the label α, which specifies the line it belongs to. The $\alpha - \beta$ swap algorithm tries to swap the labels of the nodes in such a way that the energy calculated using the configuration of the graph is minimum. This graph cut algorithm iteratively swaps the labels in such a way that the local minima of the energy function calculated using equation 1 attained for the particular labeling.

3 Spatial Language Models

A large number of document collections are available in the digital library of India that belong to the different Indian languages. There is no ground truth available for these documents. Most of the documents available are from a large number of books that are scanned. The books with same script that are published by a same publisher may have same font size, font style and similar typeset. Hence the Spatial Language Models of all these books will be the same. There could be Spatial Language Models that could be built for a book, a publisher, etc.

Each component can be classified into one of the classes: main component, vowel modifier and consonant modifier. The main component falls within a line, the vowel modifier falls above a line, and the consonant falls below a line. Hence a vowel modifier has higher affinity towards the line below it and a consonant modifier has higher affinity to the line above it. The affinities can be changed appropriately by changing the metric (distance between two components), such that the a vowel modifier falls into a line below it and the consonant modifier falls into a line above it even if they are farther from the line to which they belong.

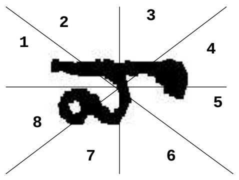


Fig. 2. Zones of a Connected Component.

The region surrounding a component is divided into 8 equi-angular regions labeled in a clockwise direction as shown in the Figure 2. The affinities between two components is represented by an $n \times n \times 8$ matrix denoted by $\mathbf{K}$, where n is the number of classes of the component recognition system. The prior information is fed into the system by initializing the matrix based on language information. As noted before, the n classes have three types of components: the main components, the vowel modifier and the consonant modifier. The affinities that belong to the classes of the main components are initialized to 1. However if the components are vowel modifiers, they have higher affinity to lines below it. Hence the zones 5,6,7 and 8 of the particular vowel modifier has lower values. If the component is a consonant modifier, it has higher affinity to lines above it. Hence the zones 1,2,3 and 4 of the consonant modifier are initialized to lower values.

$$\mathbf{K}_{i,j}^{k} = \begin{cases} 0.75 & \text{if } i \text{ is a vowel modifier and} \\ & j \text{ takes all values from 1 to n and } k \text{ ranges from 5 to 8} \\ 0.75 & \text{if } i \text{ is a consonant modifier} \\ & \text{and } j \text{ takes all values from 1 to n and } k \text{ ranges from 1 to 4} \\ 1 & \text{remaining } i,j \text{ and } k \end{cases} \tag{6}$$

3.1 Distance based Graph

The connected components attained by connected component labeling are first classified using a naive classifier. We define a metric to calculate the distance between two components based on the affinity between them as follows.

$$w_{ij} = \mathbf{K}_{i,j}^{k} * d_{ij}, \tag{7}$$

where d_{ij} denotes the nearest distance between the components, i denotes the class the first component belongs to, j denotes the class the second component belongs to and k denotes the zone of the first component through which the line joining both the component passes through.

A graph is created with the components as the nodes the distance between the components as the weight of the edge joining the components. The distance between the components is calculated using the equation 7. The components that belong to a line are within a particular distance threshold. Thus line segmentation of the document image is performed.

3.2 Learning to Segment

The matrix $\mathbf{K}$ denotes affinity of a particular component to another component in a particular direction. In a particular book a modifier component are generally placed at a constant distance from a particular main component in the complete book. When a component that belongs to a particular line is encountered, which is still at a large distance from the main component, we reduce the distance between the components by reducing the value of $\mathbf{K}_{i,j}^{k}$. Here, i denotes the class label of the main component, j denotes the class label of the modifier component, and k is the zone of the main component, where the second component lies.

The segmentation of the documents can be learnt over time across documents. The values of the matrix $\mathbf{K}$ are initialized as shown in Equation 6. A graph is built using the components in the document image as shown in section 3.1. The line segmentation is performed using graph cut proposed in section 2 and the quality of segmentation is calculated using the segmentation quality metric equation 8 proposed in [14].

$$J_l(.) = \frac{1}{1+\sigma_1} + \frac{1}{1+\sigma_2} - BLD + ILD, \tag{8}$$

where σ_1 denotes the variance of height of the lines, σ_2 denotes the variance of distance between the lines, BLD density of black/foreground pixels between two lines and ILD the density of black/foreground pixels within a line. $J_l(.)$ takes values between $\{-1, 3\}$.

This gives an estimate of the performance of line segmentation. If the performance is low i.e the value of $J(.)$ calculated is less than 2 then we perform the line segmentation . The segmentation is projection profile based, and has a tunable set of parameters. The best segmentation results are achieved by learning the best set of values for the parameters using the segmentation quality metric in equation 8.

The line segmentation algorithm detects line boundaries that gives the set of edges in the graph that cross the line boundaries. The weights of all such edges are increased by changing the values of $\mathbf{K}_{i,j}^{k_1}$ and $\mathbf{K}_{j,i}^{k_2}$ as shown in equation 9. The weights of all the missing edges within a line are reduced by changing the values of $\mathbf{K}_{i,j}^{k_1}$ and $\mathbf{K}_{j,i}^{k_2}$ as shown in equation 10.

$$\mathbf{K}_{i,j}^{k_1} = \mathbf{K}_{i,j}^{k_1} + 0.05; \qquad \mathbf{K}_{j,i}^{k_2} = \mathbf{K}_{j,i}^{k_2} + 0.05 \tag{9}$$

$$\mathbf{K}_{i,j}^{k_1} = \mathbf{K}_{i,j}^{k_1} - 0.10; \qquad \mathbf{K}_{j,i}^{k_2} = \mathbf{K}_{j,i}^{k_2} - 0.10, \tag{10}$$

where i and j gives the class labels of the components, k_1 denotes the zone that component j belongs to with respect to i, and k_2 denotes the zone that the component i belongs to with respect to j. The weights of the graph are recalculated using the above equation and the lines segmentation is again performed as suggested in section 3.1. The segmentation quality metric is then recalculated. This is performed iteratively until the segmentation quality metric improves to an acceptable value. This gives the learning phase of the document, where the corrections made by a segmentation algorithm automatically is learnt and stored in the form of the matrix $\mathbf{K}$. The procedure to perform segmentation iteratively on a document is give in Algorithm 1. To improve the performance across documents the $\mathbf{K}$ is retained for further documents.

Algorithm 1 Learning Spatial Language Models and Segmentation.

```
1: Initialize the Spatial Language Models K
2: while Change in K do
3:    Find components with ambiguous Orientation(TOP/BOTTOM)
      and initialize with arbitrary orientation
4:    Calculate the distances between components using K
5:    Create a graph and calculate the edges of the graph using the distances and
      initial labels
6:    Perform Graph Cut to give new labels to the components
7:    if Errors in segmentation then
8:       Correct them and update K
9:    end if
10: end while
```

4 Results and Discussions

There are large number of Indian language documents available at the Digital Library of India. In this work, we used the documents that were scanned from a Telugu book titled "Aadarsam", which was printed in the year 1973 and contains 256 pages. The goal is to learn the spatial language model of a particular book from a few initial pages and use it to segment the remaining pages in the book. There are two important steps involved in the process: i) adaptation of segmentation to a single document, and ii) learning the spatial language model from the sample documents.

Adaptation We take a sample page from the above mentioned book. This sample page is segmented without using any apriori information. This leads to poor results of segmentation. When the segmentation is corrected either manually, or using ground truth, or using another segmentation algorithm, the spatial language models that belong to the particular page are updated. On segmenting the same page with the updated spatial language models, it is observed that the performance of the segmentation is improved. This is done iteratively to improve the accuracy of the segmentation.

Figure 3 shows that the segmentation of the document improves over iterations because spatial language models are adapted to the document with iterations. The components marked by the oval are the dangling components that need to be assigned an appropriate line number. It can be observed that all the similar looking components that were assigned a wrong line initially, are assigned to the correct lines over iterations. The affinity of a component in a particular direction is learnt during segmentation correction of the previous iterations. Figure 4(a) shows the improvement in performance of segmentation on a particular document over iterations. The performance of segmentation is calculated using the Equation 8. From Figure 4(b) it can also be observed that the number of corrections that need to be made on the document decreases over iterations.

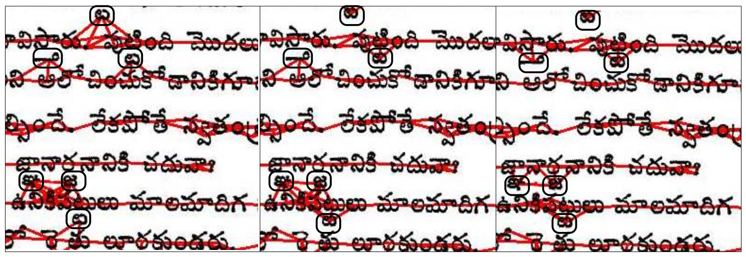


Fig. 3. Segmentation of a single document after 1st, 24th and final iterations.

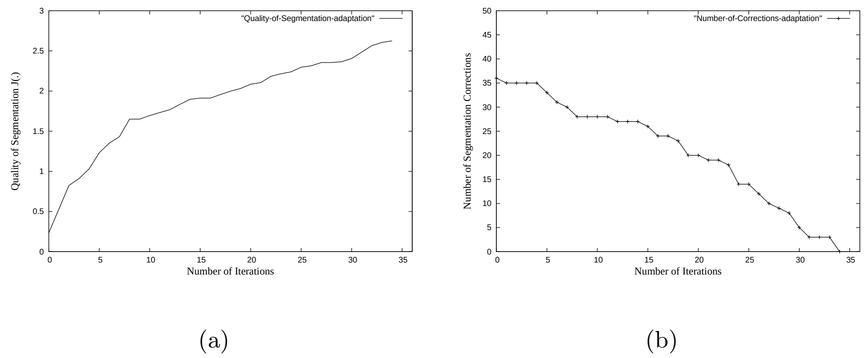


Fig. 4. (a):Performance of segmentation on a particular document over iterations, (b): Number of corrections made after each segmentation.

Learning: We take all the pages of the book and perform segmentation on each page iteratively adapting the spatial language models to the initial pages of the book. After a few pages, there is no necessity to adapt the documents because the spatial language models have completely learnt the characteristics of the book and it can be used directly to perform segmentation on the remaining pages of the book. Figure 5 shows that the algorithm can learn the spatial language model of documents with different styles and give good results on segmentation.

Figure 6(a) shows that the performance of the segmentation eventually improves on every page as information gathered from every page is used to perform segmentation on later pages of the book. Figure 6(b) shows that the number of segmentation corrections that need to be made on a new page is close to zero after adapting to the first 44 pages of the book. The remaining 212 pages could be segmented without any errors for the book under consideration.

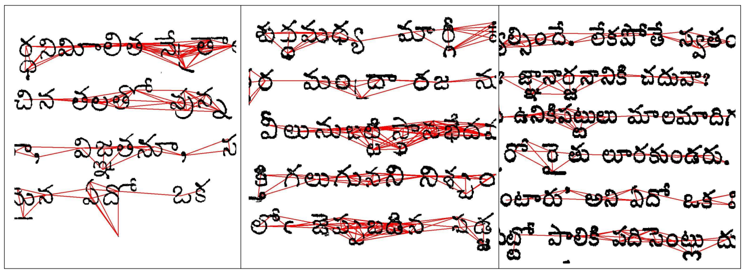
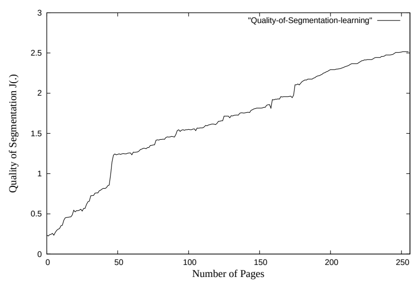
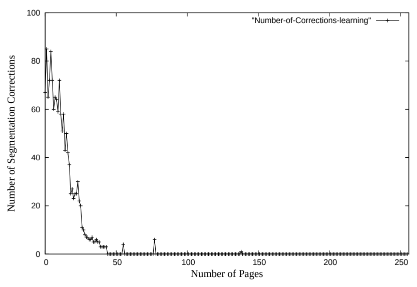


Fig. 5. Segmentation results on three documents of different styles.

The algorithm can also handle text with arbitrarily complex layouts as long as an initial estimate of lines can be found. We use the component classifier



(a)



(b)

Fig. 6. (a): Performance of the algorithm over the pages in the book, (b): Number of segmentation corrections made for each document.

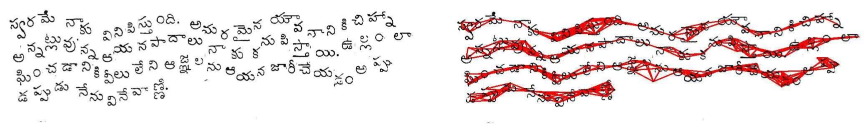


Fig. 7. Segmentation results on a page with *wavy* layout.

to identify the main components and use it to find an initial assignment of components to lines. Performance of segmentation on a wavy text can be seen in Figure 7 and semi-circular text in Figure 8.

The algorithm is also fast in practice, since the graphs that are formed using the k-nearest neighbor algorithm are sparse. However, if the graph is completely connected, the time complexity can rise up to $O(n^2)$. To segment a document with around 1200 components and 25 lines, the graph cuts takes an average time of 0.268 seconds on a PC with 512 MB RAM and 3GHz single core processor.

5 Conclusions and Future Work

We have presented a graph cut based framework for segmentation of document images that contain complex scripts such as in Indian languages. The framework enables learning of the spatial distribution of the components of a specific script and can adapt to a specific document collection, such as a book. Moreover, we are able to use both corrections made by the user as well as any segmentation quality metric to improve the quality of the segmentation. We have demonstrated, albeit on a limited set of examples, the ability of the framework to work with complex scripts, where the traditional algorithms fail completely. Currently, we are working on extending the algorithm to be able to learn a generic spatial model from a varying collection of documents to so as to give a good initial

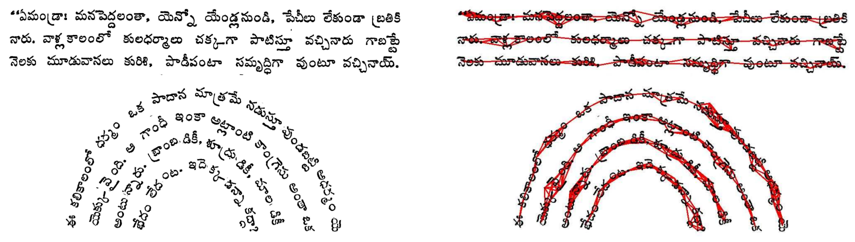


Fig. 8. Segmentation results on a page with *semi-circular* layout.

guess for a specific script. Moreover, the algorithm also needs to be extended to other region types such as words, paragraphs, etc.

References

1. Performance Comparison of Six Algorithms for Page Segmentation. In: Proceedings of the Seventh IAPR Workshop on Document Analysis Systems (LNCS 3872), Nelson, New Zealand (2006)
2. O'Gorman, L.: The Document Spectrum for Page Layout Analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence **15** (1993) 1162–1173
3. Kise, K., Sato, A., Iwata, M.: Segmentation of Page Images Using the Area Voronoi Diagram. Computer Vision and Image Understanding **70** (1998) 370–382
4. Nagy, G., Seth, S., Viswanathan, M.: A Prototype Document Image Analysis System for Technical Journals. Computer **25** (1992) 10–22
5. H.S.Baird, S.E.Jones, S.J.Fortune: Image segmentation by shape-directed covers. In: Proceedings of International Conference on Pattern Recognition(ICPR). (1990) 820–825
6. T.Pavlidis, J.Zhou: Page Segmentation and Classification. Graphical Models and Image Processing **54** (1992) 484–496
7. Digital Library of India: http://dli.iiit.ac.in/.
8. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. **23** (2001) 1222–1239
9. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **22** (2000) 888–905
10. Shental, N., Zomet, A., Hertz, T., Weiss, Y.: Learning and inferring image segmentations using the GBP typical cut algorithm. In: ICCV. (2003) 1243–1250
11. Kumar, M.P., Torr, P.H.S., Zisserman, A.: OBJ CUT. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego. (2005)
12. Baird, H.S.: The skew angle of printed documents. In: Document image analysis. IEEE Computer Society Press, Los Alamitos, CA, USA (1995) 204–208
13. Yan, H.: Skew correction of document images using interline cross-correlation. CVGIP: Graph. Models Image Process. **55** (1993) 538–543
14. Kumar, K.S.S., Namboodiri, A.M., Jawahar, C.V.: Learning to segment document images. In: PReMI. (2005) 471–476