2001

Adaptive Segmentation of Document Images

Don Sylwester
Concordia University - Seward, NE, Don.Sylwester@cune.edu

Sharad C. Seth
University of Nebraska-Lincoln, seth@cse.unl.edu

Proceedings of the Sixth International Conference on Document Analysis and Recognition, 2001.
doi: 10.1109/ICDAR.2001.953903

Adaptive Segmentation of Document Images

Don Sylwester, Concordia University, Nebraska, syl@seward.cune.edu
Sharad Seth, University of Nebraska, Lincoln, seth@cse.unl.edu

Abstract

A single-parameter text-line extraction algorithm is described along with an efficient technique for estimating the optimal value for the parameter for individual images without need for ground truth. The algorithm is based on three simple tree operations, cut, glue and flip. An XY-tree representing the segmentation is incrementally transformed to reflect a change in the parameter while intrinsic measures of the cost of the transformation are used to detect when specific tree operations would cause an error if they were performed, allowing these errors to be avoided. The algorithm correctly identified 98.8% of the area of the ground truth bounding boxes and committed no column bridging errors on a set of 97 test images selected from a variety of technical journals.

1. Introduction

Accurate identification of text lines is a critical component of the physical layout analysis step in document image analysis. This step follows the scanning of the page image into a two-dimensional array of pixels and is typically preceded by binarization, deskewing, and noise removal. The output from the physical layout analysis is fed to OCR to recover the text and then to logical layout analysis to identify the logical components of the page, including paragraphs, headings, titles, and non-text blocks [6].

Our objective is to capture the columnar structure of the page image in an XY-tree [10] in which the leaf nodes are the text lines found on that page. Our algorithm retains the key simplifying idea of our earlier work [15] - the use of a single primary parameter to control the segmentation decisions - but extends it many significant ways. The single-pass of the earlier algorithm has been replaced by a deliberate over-segmentation of the page image using a low parameter value, followed by an aggressive horizontal merging of segments, using a sequence of higher parameter values that are chosen independent of the page image. The application of the parameter to a specific instance of the merging of two fragments is governed by *intrinsic* measures of the effect of this merge, allowing us to effect maximal merging of line fragments while avoiding bridging of columns. We have developed several mechanisms that leverage the use of a single parameter to provide near ground-truth quality line location on individual images without comparison with ground truth. This may be contrasted with the training required by other algorithms (e.g. [1, 8, 12]) to set the parameters.

Notably, only the initial "cutting" of the page image using the low parameter value works at the pixel level. Subsequent steps of merging are carried out at or above the level of characters hence are computationally quite efficient. It is therefore feasible to experiment with many different parameter values in order to determine an optimal one.

The effectiveness of our algorithm is demonstrated on page images from UW-III database [14] and others [5], all of which come from technical journals. However, the algorithm does not make any specific assumptions about this domain therefore it should apply equally well to pages that are primarily text and can be segmented into an XY-tree. This is verified by our limited experiments with telephone book white pages.

A rich variety of page segmentation algorithms have been proposed in the literature. A detailed review of these algorithms is not possible here because of limitation of space. The interested reader may refer to the several excellent tutorials and surveys [2,11,13]. All algorithms use one or more parameters that are determined by training. A recent paper discusses an automatic way of choosing the parameters by posing this as a multivariate non-smooth function optimization problem [9].

The rest of the paper is organized as follows. During the segmentation, the XY-tree is dynamically modified. The tree modifications can be captured in terms of three basic operations that are discussed in Section 2. The availability of ground truth for benchmarks makes it possible to accurately measure and report the performance of page segmentation algorithms. The specific error measures we use are described in Section 3. The next two sections comprise the main contributions of this paper. Our base algorithm, without adaptation, is described in Section 4 and evaluated by setting the parameter values by training. In Section 5, we discuss the adaptation

mechanisms that allow parameter estimation without training. In the concluding Section 6 we indicate possible future extensions of this research.

2. XY-Tree and Tree Transformations

An XY-tree is formed by successive horizontal and vertical cuts through white space in the page image identified through the use of horizontal and vertical profiles [3,10]. A complete cut of a page image would have the non-cuttable *gray blocks*, typically letter forms, as leaf nodes. Nodes formed by horizontal cuts are (horizontal) *slice* nodes; those formed by vertical cuts are (vertical) *column* nodes. Our convention always has a slice node at the root and slice nodes as leaf nodes.
The fundamental operations performed on the tree are *cut*, and *glue*. Cut and glue are inverse operations as shown in Figure 1. Note that our algorithm examines all possible cuts available in the page image and it performs gluing of horizontally adjacent leaf nodes governed by the application of the *gap width-to-height threshold* described later. Once glued together leaf nodes are not then cut by subsequent processing.

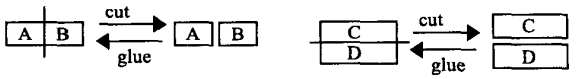

Figure 1: *Cut* and *glue* operations

The cut operations can be used to segment any block of a given XY-tree. Similarly, the glue operations can join two adjacent blocks to create a larger block. Theorem 1 establishes the fundamental character of the cut and glue operations for the XY-trees of a page image.

Theorem 1: Any XY-tree whose leaf node(s) cover the pixels in a page image can be transformed to any other XY-tree on the same set of pixels by a sequence of cut and glue operations.

Proof: It is always possible to reduce the original XY-tree to a degenerate tree with just the single root node by a sequence of glue operations; from there any other XY-tree can be obtained by a sequence of cut operations.

For the work described here it has been sufficient to implement only horizontal gluing since individual text lines rarely exhibit internal, horizontal cuts. Subsequent work will focus on vertical gluing as a means to reduce errors and extend the range of applicability of the algorithm. In our current approach we deliberately over-segment the page image and then only glue horizontally to build the desired XY-tree. The two blocks to be glued must be separated by a vertical cut in the original XY-tree and gluing should remove this cut hence only glue operations are required to build the final tree. There are three cases that can arise during gluing, as shown in Figure 2.

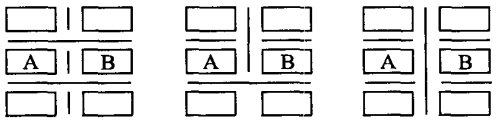

Figure 2: Three cases of horizontally gluing two blocks

In the first case, the two blocks can simply be joined to create a larger leaf node. The second case requires performing a flip operation as shown in Figure 3. The third case requires performing two flip operations.

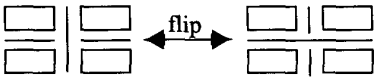

Figure 3: *flip* operation

The flip operation requires identifying the common ancestor of the two blocks to be glued and *reorienting* it, that is, changing its direction of cut (e.g. from vertical to horizontal). Reorientation of a single block can be carried out on the XY-tree by an efficient algorithm [4]. It should be noted, however, that an attempted reorientation of a block might not always succeed because of the absence of a transverse cut. Even when a transverse cut is found, it may require a sequence of flip operations, which could similarly fail. The flip can be regarded as another fundamental local operation on the XY-tree.

Theorem 2: Consider the set of all XY-trees defined on the same set of leaf nodes. Any tree in the set can be transformed to any other tree in the set by a series of flip operations.

Proof: Consider all the junctions of horizontal and vertical cuts in a given XY-tree in the set. Each junction can be in one of two orientations: its higher-level cut is either horizontal or vertical. Therefore, the corresponding junctions in two XY-trees in the set can either be the same or opposite. Therefore, one can be obtained from the other by flipping the junctions that are not the same. Note that in this case the existence of the second XY-tree guarantees that the necessary flip operations are all feasible.

3. Error Measure

To evaluate the performance of our algorithm we compare the set of detected bounding boxes, D, the leaves of the XY-tree, with a set of ground-truth bounding boxes, G, following a similar approach to that of Liang, etal [7], limiting our reporting to the percentage of the area of the set G that is covered by the set D (Table 1). Note that *mis-detection* cannot occur here since our algorithm currently accounts for all pixels, and that since our G only includes text lines we ignore false alarms.

Table 1. Definition of error measures, from [LPH97]

Correct detection	One box in G is covered by one box in D
Mis-detection	A box in G is not covered by any box in D
Splitting detection	One box in G is covered by more than one box in D
Merging detection	More than one box in G is covered by one box in D
False alarm	A box in D does not overlap any box in G
Spurious	All other cases

A *merging detection* includes both a merging of text lines in the same column (vertical) as well as merging between text lines in adjacent columns (horizontal). For our algorithm a *vertical merge* is caused by a lack of a cut in the profile between the two text lines, generally due to skew or ascender/descender overlap. A *horizontal merge* corresponds to bridging between columns. This is an *under-segmentation* error since a vertical cut at the location of the column gap was not retained in the XY-tree. A *splitting detection* corresponds to *over-segmentation* or *fragmentation* of a single text line since additional vertical cuts have been made that do not correspond with column gaps.

Column bridging errors are difficult to repair since they bring together portions of two columns which subsequent processing would have to re-cut. Fragmentation errors are relatively easy to correct since their location in the XY-tree continues to reflect their location in the text line.

4. Base Algorithm

Our goal is to capture the columnar structure of a page image in an XY-tree in which the leaf nodes are the text lines found on the page. Our interest in an adaptive algorithm led to the selection of an approach driven by a single primary parameter, the *gap-width-to-height-threshold.* This resolution independent parameter is the ratio of the width of the horizontal gap between two blocks to the height of the gap, measured as the minimum height of the two neighboring blocks. This parameter reflects a nearly universal convention in physical page layout, that the widths of gaps between text-lines in columns are generally larger than the gaps between words and between characters. If the measure of a gap is below this threshold we attempt to glue these two blocks into a single new block, otherwise we consider it to be a column gap. Note that only gaps between exactly two horizontally adjacent blocks that share a mutual vertical overlap are considered for gluing. See Figure 4.

The RXYC algorithm constructs an XY-tree by recursively cutting the initial page with successive horizontal and vertical cuts through white space in the pixel image, down to the non-cuttable *gray block* level, and then performs only flip and horizontal glue operations as we build toward the final tree. As various parts of the tree are constructed we *aggressively glue* horizontally adjacent leaf nodes guided by our threshold parameter. If two leaf nodes should be glued together and are direct siblings in the tree then we glue them immediately. If they are not direct siblings, we attempt a series of tree transformations, flips, to find a tree where they are siblings, if possible, and then glue them in that tree.

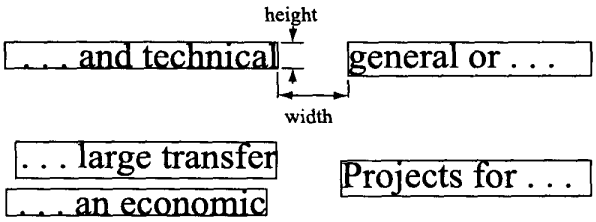

Figure 4. Glueable (upper) and non-glueable (lower) gaps.

Note that if two leaf nodes are glued together, the height of the new leaf node may increase over its two predecessors, possibly affecting the measure of the gaps on either end. Our aggressive strategy is to reexamine gaps for potential gluing whenever they are encountered in the process and to reexamine the final tree exhaustively, applying sequences of flips as necessary to perform all possible glue operations. There is a potential for order dependence in this strategy, although it does not appear to be common.

RXYC algorithm: *Phase (1)* Cut the initial pixel block horizontally into slices. Cut each slice in turn vertically into sub-blocks. Recursively process each sub-block with RXYC to form an XY-tree. Merge the XY-trees from each sub-block horizontally using flips and glues, to form an XY-tree for the entire slice. Merge the XY-trees for each slice vertically to produce the result XY-tree. *Phase (2)* Once the XY-tree has been produced for the initial block, representing the entire page image, exhaustively reexamine the tree to perform all possible remaining glue operations.

As the value of the gap-width-to-height-threshold increases the fragmentation errors decrease and at some point column bridging errors begin to occur. In an ideal document there would be a range of threshold values for which there are no fragment or bridging errors: the segmentation is correct. In real documents we do tolerate occasional fragmentation errors in order to avoid bridging errors. Typically, these occur in left and right justified lines containing word gaps that exceed a column gap.

Table 2 reports the performance of RXYC on images from four document classes, using threshold values

selected to be optimal for small training sets selected from each class. The 2COL & 3COL (UW) classes contain documents from several sources while the IBM and PAMI classes each follow consistent layout styles. While the results are reasonably good for a single parameter process it is clear that simple training is not sufficient, especially when we do not have a good characterization of the layout domain. Most of the errors in the image sets reported are due to one or two exceptional examples.

Table 2. Performance results for RXYC for four document sets: Percentage of area of ground truth nodes covered by output leaf nodes. (In each case the top line is the test result and the bottom line is the performance during the training process.) N is the number of images in the set.

Image Set	N	Correct Detection	Split Detection	Merge Detection	
				Vert	Horiz
IBM	17	99.5%	0.5%	0.0%	0.0%
	5	98.8%	0.1%	0.0%	0.0%
PAMI	17	91.5%	8.1%	0.1%	0.2%
	6	91.8%	8.0%	0.0%	0.2%
2COL	20	85.2%	10.0%	0.9%	3.8%
	9	98.8%	1.0%	0.2%	0.0%
3COL	17	96.6%	2.5%	1.0%	0.0%
	5	99.7%	0.1%	0.1%	0.0%

The current algorithm requires cuts to be visible in the profiles and is therefore sensitive to skew and ascender/descender overlap, for example, as well as ruled boxes surrounding text areas. We believe that an adaptation of the algorithm, including implementation of vertical gluing, would allow cutting at arbitrary locations, but the algorithm will continue to require deskewed images or at least knowledge of a (uniform) skew angle.

Also in Phase (2), small artifacts, notably punctuation marks such as periods, commas and dashes (and some noise), are recognized by their small dimensions relative to nearby gray blocks and are glued to them if the separation is small compared to the size of that block.

5. Adaptation

The optimal value in our base algorithm for the gap-width-to-height threshold for a specific page image is just below the onset of column bridging. An adaptive algorithm must recognize the moment at which the initial column bridging occurs for that image. We have identified several measures that can be used to classify gluing operations as either local (leaf nodes within text lines are being glued) or global (leaf nodes are being glued between columns). A simple adaptive extension of the base algorithm might increase the value of the parameter until the change in the tree would be global, and then stop.

When two leaf nodes are glued together a (possibly null) sequence of flip operations is required to make the two leaf nodes direct siblings, enabling the glue operation. The flip operations required in this sequence are confined to the two adjacent subtrees containing the two leaf nodes and rooted at the nearest common ancestor. We use the area associated with these two subtrees, normalized to the size of the median gray block, as a measure of the size of the change. If the two leaf nodes are direct siblings of each other, then the affected area of the tree is the area of the bounding box that encloses the two nodes, a relatively small area. If the two leaf nodes are in adjacent columns, then a flip will occur in the orientation of a vertical cut beneath the common ancestor. The area of this node includes the relatively large area of the two columns

A separate measure of the size of the change is the change in the maximum width of any leaf node, normalized to the width of the median gray block, contained within the affected area as the gluing is done. The initial bridging error typically doubles the maximum width of a leaf node within the affected area.

Our current experiments couple these two measures to determine when a global change has occurred and the threshold has been increased too far. If the minimum area and width of the two subtrees each exceed a threshold, 400 and 15 in the results reported here, then a global change in the tree is likely and gluing is not done.

Since our algorithm is driven by a single parameter a simple search can be performed to locate the optimal threshold value. We have developed an efficient approach to this search. If we produce the complete XY-tree corresponding to a particular value of the threshold, say 2.3, we can then produce the tree for an incremental value, say 2.4, by simply performing an exhaustive glue at this new threshold value. This operation occurs at the tree node level rather than at the pixel level, and can be done quite efficiently. Our search procedure begins by generating the XY-tree for a low threshold value, typically 1.0, and then successively re-glues it at incrementally higher threshold values until a column bridge occurs, as evidenced by these measures of local and global effects.

This extension allows the base algorithm to adapt its selection of the appropriate threshold to a specific page image, but the lack of a strict hierarchy in the widths of character, word and column gaps means there can still be some fragmentation. Since we can, in fact, identify the specific gaps that would cause column bridging if they were glued it is possible to increase the value of the parameter past the onset of column bridging and avoid those glue operations, but continue to do glue operations that correspond to fragmentation. It is this approach that forms our current adaptive algorithm, ARXYC.

ARXYC Algorithm: Given an image-independent series of threshold values, apply Phase (1) of RXYC with the initial (lowest) value of the threshold. Apply Phase (2) of RXYC repeatedly for succeeding values of the threshold.

We have applied this algorithm to 97 page images, with results shown in Table 3. For this set of images, selected from a variety of journals, it is sufficient to first cut the page at a low value of the parameter, in this case 1.0, and then to successively glue the XY-tree at increments of 1.0 up to 5.0. Performance is comparable to algorithms with multiple primary parameters and which require training.

Table 3. Performance of ARXYC for four document sets: Percentage of ground truth area covered by output leaf nodes. N is the number of images in each set.

Image Set	N	Correct Detection	Split Detection	Merge Detection	
				Vert	Horiz
IBM	21	100%	0.0%	0.0%	0.0%
PAMI	22	98.9%	0.6%	0.5%	0.0%
2COL	32	98.1%	1.0%	1.0%	0.0%
3COL	22	98.6%	0.9%	0.9%	0.0%

6. Conclusion

Considering that page layouts are not governed by strict rules, we find it remarkable that a segmentation algorithm based on just one primary parameter should perform as well on a variety of documents. Remarkably, also, although the parameter definition was motivated by observing how text lines are constructed, experimental results show it to perform well also in segmenting non-text regions from text regions.

Our experiments on IBM and PAMI with the base RXYC algorithm indicate that a fixed parameter value for each class of documents can segment all pages without column bridging and with minimal line fragmentation. More complex cases of errors do not occur. A single fixed parameter value does not work as well across the varied layouts represented in our UW image classes.

The adaptive algorithm, ARXYC, performs well across all 97 page images in our test sets with the same series of threshold values applied to all images, with no column bridging.

The most interesting extension of this algorithm is to incorporate non-white-space cutting by extending the glue operation to the vertical dimension. Treatment of punctuation marks should also be formalized.

References

[1] H. S. Baird, Background Structure in Document Images, in H, Bunke (Ed.), *Advances in Structural and Syntactic Pattern Recognition*, World Scientific, Singapore, 1992, pp. 253-269.

[2] R. Cattoni, T. Coianiz, S. Messelodi, C. M. Modena, "Geometric Layout Analysis Techniques for Document Image" *ITC-IRST*, Via Sommarive, I-38050 Povo, Trento, Italy January 1998.

[3] J. Kanai, "Knowledge-Based Document Image Analysis System", PhD Dissertation, RPI, 1990.

[4] J. Kanai, M. S. Krishnamoorthy, T. Spencer, "Algorithms for Manipulating Nested Block Represented Images", *Advance Printing of Paper Summaries, 26th Fall SPSE Conf and Exhibition on Electronic Imaging*, pp. 190-193, 1986.

[5] M. Krishnamoorthy, G. Nagy, S. Seth, M. Viswanathan, "Syntactical segmentation and labeling of digitized pages from technical journals", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol 15(7), pp. 737-747, July 1993.

[6] K-H Lee, Y-C Choy, and S-B Cho, "Geometric Structure Analysis of Document Images: A Knowledge-Based Approach", *Transactions on Pattern Analysis & Machine Intelligenc*, Vol. 22(11), November, 2000, pp. 1224-1240.

[7] J. Liang, I.T. Phillips, and R.M. Haralick, "Performance evaluation of document layout analysis on the UW data set", in: L.M. Vincent and J.J. Hull, eds., *Proceedings Document Recognition IV* (San Jose, CA, 1997) 149--160.

[8] Jisheng Liang, Ihsin T Phillips, Robert Haralick, "A Statistically-Based, Highly Accurate Text-line Segmentation Method", *Proc. of the 53th International Conference on Document Analysis and Recognition,* ICDAR-99.

[9] S. Mao and T. Kanungo, "Automatic training of page segmentation algorithms: An optimization approach", *Proceedings of International Conference on Pattern Recognition*, pp. 531--534, 2000.

[10] G. Nagy and S. Seth, "Hierarchical Representation of Optically Scanned Documents" *Proc. 7th Int. Conf. on Pattern Recognition*, pp. 347-349, 1984.

[11] G. Nagy, "Twenty Years of Document Image Analysis in PAMI", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 1: January 2000, pp. 38-62.

[12] T. Pavlidis and J. Zhou "Page segmentation and classification", *CVGIP: Graphical Models and Image Processing*, Vol. 54 (6), pp. 484-496, November, 1992.

[13] L. O'Gorman and R. Kasturi, *Document Image Analysis*, IEEE Computer Society Press, Los Alamitos, CA, 1995.

[14] I. Phillips, *User's Reference Manual*, CD-ROM, UW-III Document Image Database-III.

[15] D. Sylwester, S. Seth, "A Trainable, Single-Pass Algorithm for Column Segmentation", *Proc. 3rd Int. Conf. on Document Analysis and Recognition (ICDAR95)*, pp. 615-618, 1995.