Recursive X–Y Cut using Bounding Boxes of Connected Components

Jaekyu Ha & Robert M. Haralick

Dept. of Electrical Engineering, FT-10
University of Washington
Seattle, WA 98195, U.S.A.

Ihsin T. Phillips

Dept. of Computer Science
Seattle University
Seattle, WA 98122, U.S.A.

Abstract

A top-down page segmentation technique known as the recursive X–Y cut decomposes a document image recursively into a set of rectangular blocks. This paper proposes that the recursive X–Y cut be implemented using bounding boxes of connected components of black pixels instead of using image pixels. The advantage is that great improvement can be achieved in computation. In fact, once bounding boxes of connected components are obtained, the recursive X–Y cut is completed within an order of a second on Sparc–10 workstations for letter-sized document images scanned at 300 dpi resolution.

keywords: page segmentation, recursive X–Y cut, projection profile, connected components

1 Introduction

Document analysis primarily concerns with the layout structure. As an early stage of document analysis, segmentation of document images has to be initiated. This is an essential step for later stages (document classification, graphics coding, etc.). It comprises the separation of multilevel (photographical) and bilevel (textual and graphical) information as well as their mapping into nested layout objects of different levels. The goal is to split the binary image into component-like characters, text lines and text blocks, as well as graphics and image parts.

This paper describes the segmentation technique by projection profile cuts, which is obviously a top-down approach. Nagy and Seth's X–Y tree decomposition technique belongs to this category, though they did not use projection profiles to determine where (horizontal or vertical) cuts need to be placed. Their page decomposition is recursive. At each step, the process must decide whether or not a cut needs to be made. The recursive subdivision of the document page is then converted into a X–Y tree. Though they did not give details to define cuts, it is obvious that the X–Y tree decomposition can be implemented by the recursive X–Y cuts [Witten, 1992].

This paper is organized as follows: Section 2 gives an answer to why we should use the bounding boxes of the connected components of black pixels in the profiling analysis. Section 3 explains details of top-down segmentation by projection profile cuts. Finally, concluding remarks are given in Section 4.

2 Projections of Bounding Boxes

Many kinds of symbols are used in document pages: alphabetic characters, numerals, punctuation marks, mathematical symbols and so on. They are classified into two groups: single-component symbols and multiple-component symbols. Notice that all English alphabets except 'i' and 'j' belong to the former group. Of course, a single-component symbol can be 'broken' into several pieces on a degraded image. In this case, the symbol is also considered as a collection of sub-symbols, that is, a multiple-component symbol.

2.1 Why Bounding Boxes?

Once a paper document is at hand, it has to be been scanned and thresholded, which results in a binarized document image which consists of lots of black and white pixels. Document page decomposition can be accomplished by analyzing the spatial configuration of connected components fo black pixels. At this stage, it is a matter of choice whether we remain in the low level by handling pixels themselves or go to a higher level by handling some important feature of connected components – *bounding boxes*. The bounding box of a connected component (or symbol) is defined to be the

Proceedings of the Third International Conference on Document Analysis and Recognition (ICDAR '95)
0-8186-7128-9/95 $10.00 © 1995 IEEE

smallest rectangle which circumscribes the connected component (or symbol).

It is a good experiment to render the image of a technical document by only drawing the bounding boxes of the connected components. Not only does a human observer get the immediate impression of 'text', but he may also associate categories of meaning, like 'This is the section heading, that's the displayed math!', and so on. It is not necessary to reveal even a single letter.

Traditionally, some efforts were made to initiate document image understanding by projecting image pixels. This pixel projection approach is one of widely used top-down segmentation methods. To decompose document images, local peaks and valleys of pixel projection profiles can be used to determine the location where the division has to take place. The conventional recursive X–Y cut is a typical application of such projection profile cuts.

Unfortunately, the pixel projection approach is computationally inefficient. It is because each symbol is not treated as a computational unit in this approach. Why not projecting bounding boxes instead of projecting image pixels? The bounding box projection approach has many advantages over the pixel projection approach. It is less computationally involved. It is possible to infer from projection profiles how bounding boxes (and, therefore, primitive symbols) are aligned and/or where significant horizontal and vertical gaps are present. Hence, bounding box projection profiles can be used to determine the reading direction (vertical or horizontal), existence of kernings, text lines, words, and paragraphs.

2.2 Projections and Profiles

The horizontal and vertical projections and profiles can be formulated as follows. Suppose that we are given a set of bounding boxes $B = \{b_1, b_2, \cdots, b_N\}$ where each b_i encloses the region

$$R_{b_i} = \{(x, y) \mid x_{b_i,min} \leq x \leq x_{b_i,max} \text{ and } y_{b_i,min} \leq y \leq y_{b_i,max}\} \quad (1)$$

for $i = 1, \cdots, N$. The horizontal projection of bounding box b_i is the association of b_i with a scalar function H_{b_i} which is called the height function of b_i and is defined by

$$H_{b_i}(y) = \begin{cases} 1 & \text{if } y_{b_i,min} \leq y \leq y_{b_i,max} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where $y \in Z$. Likewise, the vertical projection of a bounding box b_i is the association of b_i with a function V_{b_i} which is called the width function of b_i and is defined by

$$V_{b_i}(x) = \begin{cases} 1 & \text{if } x_{b_i,min} \leq x \leq x_{b_i,max} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where $x \in Z$.

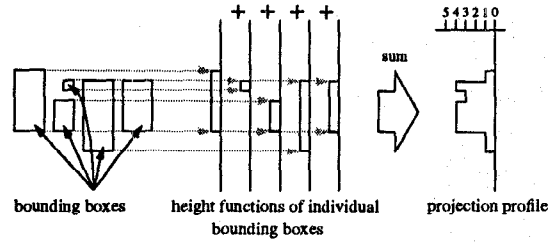

Figure 1: Horizontal Projection Profile, which is obtained by accumulating the bounding boxes onto the vertical line. The projection profile is a histogram which shows frequencies of projected bounding boxes.

The projection profiles of a set of bounding boxes can be defined using the above two projection profiles. The horizontal projection profile of B is a function from Z to Z and is defined by the sum of the horizontal projection profiles of individual bounding boxes:

$$H_B = \sum_{i=1}^{N} H_{b_i} \quad (4)$$

Likewise, the vertical projection profile of B is a function from Z to Z and is defined by the sum of the vertical projection profiles of individual bounding boxes:

$$V_B = \sum_{i=1}^{N} V_{b_i} \quad (5)$$

Figure 1 illustrates how a horizontal projection profile is calculated. A vertical projection profile is calculated in a similar way. As can be seen in Figure 1, a projection profile is a function whose values represent how many bounding boxes are projected onto the projection line.

3 Segmentation by Recursive X–Y Cut

A top-down segmentation technique known as the *recursive X–Y cut* decomposes a document image recursively into a set of rectangular blocks. At each step, the pixel-projection profiles are calculated in both horizontal and vertical directions. Then a zone division is performed at the most prominent valley in either projection profile, and the process is repeated recursively until no sufficiently wide valleys are left in both profiles.

Proceedings of the Third International Conference on Document Analysis and Recognition (ICDAR '95)
0-8186-7128-9/95 $10.00 © 1995 IEEE

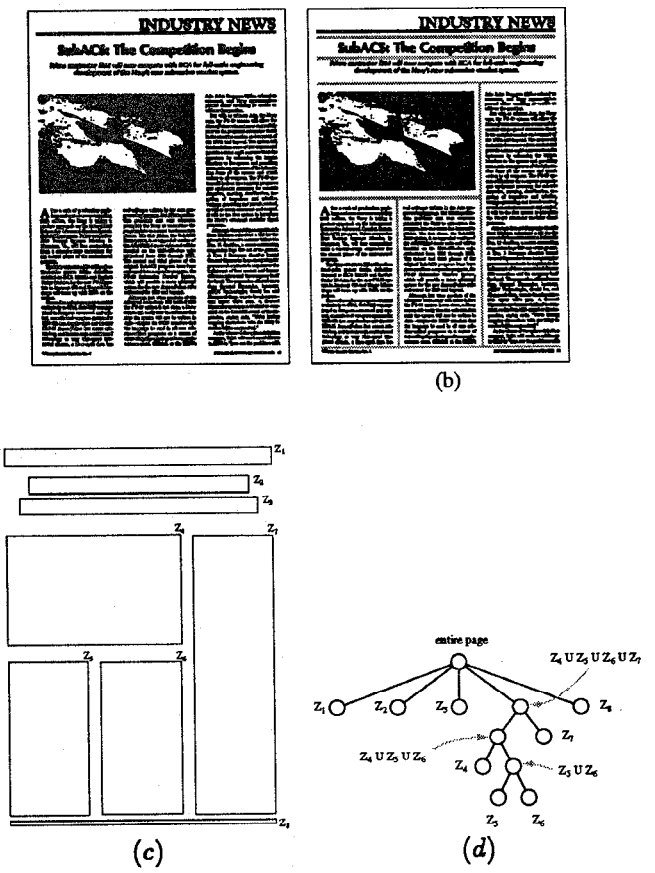

Figure 2: Segmentation by the recursive X–Y cut algorithm: (a) an original document image, (b) placement of cuts, (c) zones subdivided by the recursive X–Y cut, (d) X–Y tree of the page layout structure of the document image shown in (a).

Of course, the conventioanl recursive X–Y cut algorithm can be implemented using bounding boxes instead of using pixels. The effect is to greatly improve the computation. Figure 2 shows an example image and the segmentation result by the recursive X–Y cut. The figure shows the cuts as horizontal and vertical thick lines. The algorithm being implemented using bounding boxes, it takes nearly an order of a second on Sparc–10 workstations to complete the segmentation.

The structured subdivision of a document image by the recursive X–Y cut makes the document to be represented in the form of a tree with nested rectangular blocks — called an X–Y tree. The X–Y tree is a spatial data structure which shows the hierarchical subdivision of a document page by recursive X–Y (X–horizontal and Y–vertical) cuts. The root node of an X–Y tree is the bounding rectangle of the full page. Each node in the tree represents a rectangle in the page. The children of a node are obtained by subdividing the rectangle of the parent node either horizontally or vertically, with horizontal and vertical cuts being alternately employed in successive levels in the tree. The first subdivision may be arbitrarily set to either horizontal or vertical. The leaves constitute a tiling of the page. Horizontal and vertical subdivisions alternate strictly, level by level.

Nagy and Seth (1984) proposed an X–Y tree as the representation of a page layout. The X–Y tree representation has the following properties:

1. The page is gauranteed to be completely tiled, leaving no portion unaccounted for. Nested subdivisions appear well suited to the hierarchical structure of technical and business records.

2. Only rectangles are generated, allowing identical processing steps at every level.

3. At each level, only a linear (i.e., either horizontal or vertical) subdivision must be considered, which allows a well-ordered sequential examination of the blocks.

4. X–Y trees can be readily extended to three or more dimensions; this may be advantageous in dealing with multi-page documents, and possibly in other applications.

The original X–Y tree decomposition was intended to acomplish a complete tiling of a document page. However, all rectangular background regions, for example, between text columns being forced to be tree nodes, the tree representation is made to be complicated.

We will use the term X–Y tree in a strict sense. The tree structure does not include background regions. Two consecutive tree siblings should be well separated. That is, if gaps in a region are not prominent compared to the *em* size of the dominant font, cuts will not be made. Based on the above discussions, the algorithm for the recursive X–Y cut is described as follws: Given a document image,

1. Do preprocessing (eg. noise removal, skew correction, etc.).

2. Apply a connected component labeling algorithm.

3. Obtain the bounding boxes of connected components.

Proceedings of the Third International Conference on Document Analysis and Recognition (ICDAR '95)
0-8186-7128-9/95 $10.00 © 1995 IEEE

4. Create a root node.

5. Calculate the horizontal and vertical projection profiles within the region of interest.

6. Do divisions at large gaps in the projection profiles whose widths exceed a certain threshold value V_{thr}. Whenever divisions are made, create a new child node. At each recursion level, horizontal and vertical divisions alternate.

7. Do Step 5–6 recursively until no further divisions are possible.

Figure 2(a)–(d) show a document image, X–Y cut positions, resulting rectangular blocks, and the X–Y tree structure. The threshold value V_{thr} is determined by the dominant font size. Let the *em* size of the dominant font be denoted by S_{dom}. Then, $V_{thr} \approx S_{dom}$.

4 Discussions

This paper discusses an improved version of the recursive X–Y cut. Improvement was possible by choosing, as computational units, the bounding boxes of connected components of black pixels instead of the image pixels. For usual text documents, this leads to a tremendous reduction of items. To give an impression: if a letter sized document page is scanned at 300 dpi resolution, this yields $2,550 \times 3,300 = 8,415,000$ pixels. If the page is typewritten with 35 lines per page, it may contain approximately 2,500 connected components. On Sparc–10 workstations, about two seconds are required in average for segmentation of a letter-sized image of 300 dpi resolution, though the computation time depends on the number of connected components.

Our improved recursive X–Y cut described in this paper can be combined with the page decomposition technique by [Ha *et. al.*, 1994] to produce the complete segmentation of document pages to which isothetic (Manhattan) spatial subdivisions can be applied.

Furthermore, the recursive X–Y cut recursively decompose a document page into blocks until no further decomposition is possible. Thus, the X–Y tree of a X–Y *cuttable* document page will have only simple leaf nodes which contains only one homogeneous region. Whereas, in complex document pages, an X–Y tree will have complex leaf nodes which contain two or more homogeneous regions. To complete the page segmentation for a complex document page, we can first construct a X–Y tree and then do further decomposition for each complex leaf nodes of the X–Y tree using other methods that can handle document pages with complex layout in the literature.

We have applied this method over 150 document images contained in the "UW English Document Image Database I" [Phillips *et. al.*, 1993], which was developed at the University of Washington in 1993 and was intended for researchers in the areas of OCR and document image understanding. The results were perfect under our definition of the X–Y tree given in Section 3.2. Currently, an experimental protocol is being desinged for systematically evaluating the performance of the proposed method. We expect to report the experimental results in a near future.

References

[1] H.S. Baird, *Structured Document Image Analysis*, Springer Verlag, 1992.

[2] R.M. Haralick and L.G. Shapiro, *Computer and Robot Vision*, Volume I, Addison-Wesley, 1992.

[3] W. Horak, "Office Document Architecture and Office Document Interchange Format, Current Status of International Standardization," *IEEE Computer*, October, 1985.

[4] G. Nagy and S. Seth, "Hierarchical Representation of Opically Scanned Documents," *7th ICPR*, Montreal, 1984, p347-349.

[5] G. Nagy, S. Seth and S. Stoddard, "Document Analysis with an Expert System," *Pattern Recognition in Practice II*, Elsevier Science Pbli. B. V., pp. 149-155, 1984.

[6] G. Nagy, S. Seth and M. Viswanathan, "A Prototype Document Image Analysis System for Technical Journals," *IEEE Computer*, vol. 25, no. 7, pp. 10-22, July 1992.

[7] I.H. Witten, A. Moffat, and T.C. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images*, Van Nostrand Reinhold, 1994.

[8] J. Ha, I.T. Phillips and R.M. Haralick, "Document Page Decomposition using Bounding Boxes of Connected Components of Black Pixels," ISL report, Dept. Electrical Eng., University of Washington, 1994.

[9] I.T. Phillips, S. Chen and R.M. Haralick, "English Document Database Standard," *Proc. ICDAR*, Japan, 1993.

Proceedings of the Third International Conference on Document Analysis and Recognition (ICDAR '95)
0-8186-7128-9/95 $10.00 © 1995 IEEE