Newspaper Document Analysis featuring Connected Line Segmentation

Phillip E Mitchell[1] and Hong Yan[2]

[1]School of Electrical and Information Engineering
University of Sydney, NSW 2006

[2]Department of Electronic Engineering
City University of Hong Kong, Kowloon, Hong Kong

{mitchell, yan}@ee.usyd.edu.au

Abstract

This paper presents an algorithm designed to segment and classify newspaper documents. A notable feature of this algorithm is the ability to detect lines in the document – including lines that are connected to other components. A bottom-up approach is used to segment the image into patterns, and then each pattern is classified into one of seven types. Complete regions are then formed from the classified patterns.

1. Introduction

Newspapers provide a great challenge for document analysis because of the huge variety of layouts and quality. Lines may not be physically connected, perpendicular to the page edge, or even straight. The space between columns may be smaller than some inter-word spaces, and possibly worst of all, separate component may actually be connected. This paper presents an algorithm that is designed to operate on these complex documents. A bottom-up approach is employed (Section 2) to segment the components of the document into patterns, and these are classified into one of nine types (Section 3). A key feature of this algorithm is the ability to extract connected lines, such as underlines, column dividers and boxes, from other components in the document (Section 4). Grouping the classified patterns into entire regions (Section 5) completes the algorithm. The results of the algorithm, when applied to actual newspaper documents, are presented and discussed (Section 6), followed by some concluding remarks (Section 7).

2. Pattern Segmentation

This algorithm uses a bottom-up approach to extract patterns from the document image. The first step in pattern extraction is to locate rectangular regions called *rects* [1]. A *rect* may be thought of as a rectangular region of loosely connected black pixels. More specifically, a *rect* has at least one black pixel in every 9-pixel square. The process of locating *rects* involves scanning 9-pixel squares of the image in raster order. The top left corner of a *rect* is defined by the first 9-pixel square found containing a black pixel. The right edge of the *rect* is located by searching right until a white 9-pixel square is found. Similarly, the bottom edge is located by scanning down until a white 9-pixel square is found. A *rect* is defined in this way so black regions may be found without scanning every pixel.

Patterns may then be defined by merging adjacent *rects*, which means there is no need to refer to the actual image. A *rect* is merged with another *rect* if they are connected to each other. For text regions in a document, patterns are generally single characters or words, depending on the text style and size. For non-text regions, patterns may be all kinds of shapes and sizes. The patterns are stored in a linked list. This enables the patterns to be quickly and easily traversed for classification. Other data structures and methods such as dynamic arrays [4], attributed relational graphs (which are often used for recognition) [5] and spatial map grids [6] were considered, but a simple linked list was sufficient for the task.

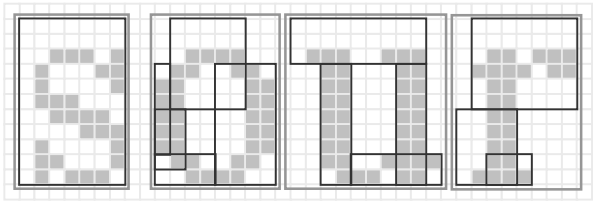

Figure 1. Four patterns (each containing *rects*) are found in the word 'sour', one for each letter.

Many bottom-up image segmentation algorithms use simple connectivity [2,3] to define the basic image components, but patterns offer several advantages. Firstly, all image components, including nested ones, are extracted with only one quick pass through the image. Secondly, a typical document image contains far fewer patterns than connected components, which makes subsequent processing faster. Finally, connected components separated by three or more pixels will not be merged into patterns, which gives good segmentation. Four examples of text patterns are shown in Figure 1. The black rectangles within each pattern represent the *rects*.

3. Pattern Classification

The previous section describes how patterns are created. It is now time to begin to determine what part of the document the patterns belong to. This is known as pattern classification. Ultimately, each region in the document image is to be classified into one of the first seven classes listed in Table 1. The extra classes (8 and 9) correspond to intermediary classes, and are explained in due course. The patterns are classified using a series of rules based on the pattern size, shape, black pixel numbers and run-length characteristics. These rules are explained below and are summarised in Table 2.

Table 1. Each region is to be classified into one of the first seven types listed here. Types 8 and 9 are intermediary classes.

ID	Description
1	text
2	title
3	inverse text
4	photo
5	graphic/drawing
6	vertical line
7	horizontal line
8	small patterns
9	box

The classification rules are performed in the order they are listed in Table 2. Generally, once a rule is found to be true, that pattern is set to the specified class and is not subjected to any other tests. The only exceptions are rules 'B' and 'I' that modify patterns with an id of '4' and '1' respectively. All that remains now is to explain the terms used in Table 2. Most of the terms are self-explanatory: *w* and *h* refer to the pattern width and height respectively. The pattern *area* is the number of pixels contained within the edges of the pattern (*area* = *w*×*h*). Rule B introduces *rarea*, which is the area covered by the *rects* in the pattern. A pattern, which covers a large area with a relatively small area of *rects*, is most likely part of a graphic. Rule C defines small patterns (id = 8) as those with a very small number of black pixels (*n_black*) or very small *area*. There are often many small patterns in an image and since they are so often noise or punctuation marks, they have simply been ignored in this implementation.

The next two rules (D and E) locate lines by comparing the pattern width and height. Rule E, however, has an extra condition denoted as *line*(). This is a function that determines if the pattern is a line as opposed to a long thin text pattern. It basically checks that most of the pixels through the centre of the pattern are black. Rule F locates graphics patterns by searching for large patterns with a large number of black runs (*blkruns*). Rule G identifies reverse text patterns. It requires a large black to white pixel ration (*bwratio*) and a maximum black run length of at least 3/4 of the pattern width.

All unclassified patterns are then set to text (id = 1). Finally (rule I) locates titles from the text patterns, by comparing the height to the average text pattern height (*avh*). Every class has now been covered with the exception of boxes (id = 9). These are defined in the following section.

Table 2. This table specifies the classification rules in the order in which they are implemented. The lowercase characters represent pattern properties and the uppercase characters are constants.

Rule	ID	Rules ($\cap$ = and, $\cup$ = or)
A	4	$area > A_1 \cap w > D \cap h > D$
B	5	$id = 4 \cap rarea / area < RA$
C	8	$n\_black < BLK \cup area < A_2$
D	6	$(h > L_1 \cap h > w \cdot R_1 \cap w < W_L) \cup (h > w \cdot R_2 \cap w > W_L)$
E	7	$(w > L_1 \cap w > h \cdot R_1 \cap h < W_L \cap line()) \cup (w > h \cdot R_2 \cap h > W_L)$
F	5	$area > A_2 \cap blkruns > 4 \cdot w$
G	3	$w > W_1 \cap bwratio > R_{BW} \cap mbrl > 0.75 \cdot w$
H	1	otherwise
I	2	$id = 1 \cap h > F \cdot avh$

4. Line Extraction

Segmentation becomes very difficult when two or more different components are connected in the image. Some common examples of this are where lines are connected to text characters, other lines or noise. This section presents a solution to this problem, which works if any part of a line has been identified. The basic idea is to locate lines connected to other components by extending known lines through other patterns. An outline of the algorithm is presented below, and a brief explanation follows.

Step 1 Covert box-shaped patterns to lines (see Figure 2).
Step 2 For each line (referred to as cline):
- 2(a) Create a list of segments: patterns that align with cline (see Figure 3).
- 2(b) Create a list of partials: patterns that contain a *rect* that aligns with cline (see Figure 3).
- 2(c) Sort segments and partials from low end to high end (eg: left to right if horizontal line).
- 2(d) Define complete lines from sorted lists, allowing a a maximum gap between elements of twice the width of cline. Each line must contain a pattern with the same id as cline (ie. 6 or 7).
- 2(e) Segment partials that have been used in a line.

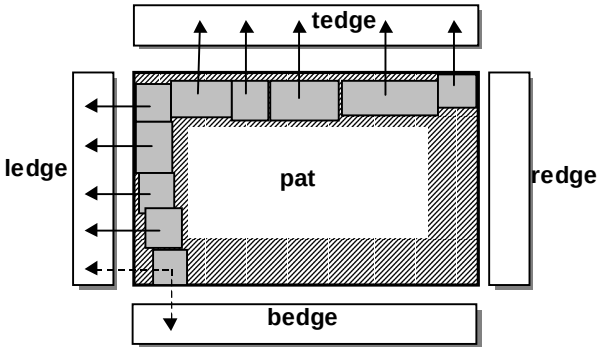

Figure 2. This diagram illustrates the process of detecting box-shaped patterns: the requirement is that all the *rects* (represented as shaded boxes) lie within a small distance of the pattern edge. The diagram also shows how up to four edge lines are extracted from the box pattern.

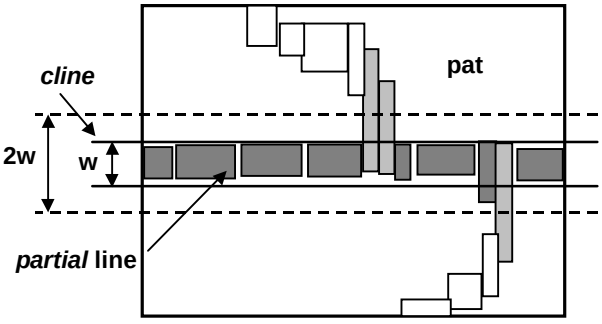

Figure 3. This diagram illustrates how part of a line is detected within a pattern by searching for *rects* that line-up with the current line (*cline*). To account for a slight skew or curve in the line, any *rects* that lies within a region covered by a line twice as thick as *cline* is considered as part of the line.

The first step of this algorithm locates patterns that are boxes or part boxes. These are identified as patterns that only contain rects that are close to the pattern edge. A minimum size is also required to avoid defining text characters as boxes. Lines are created from the boxes by sorting the rects into four groups: one for each edge. Each group of rects forms a new line pattern corresponding to that edge of the box. Some edges may be missing so it is important to group corner rects only with edge lines that exist. Figure 2 illustrates this process.

Complete lines are built in step 2 by taking each identified line as a starting point. Let us call the current line: *cline*. In step 2(a) all the patterns, including the lines, are searched, and a list of segments is created from those that lie along the same line as *cline*. More specifically, these patterns must lie within a region covered by a line twice as thick as *cline* (see Figure 3). There may also be some patterns that only partially lie within this region defined by *cline*: these patterns are added to the partials list if and on if they contain rects which lie wholly within the region. Such rects indicate the pattern may contain a line. After sorting the segments and partials lists (step 2(c)), the complete lines are constructed (step 2(d)). Gaps up to twice the width are allowed within single lines. If larger gaps are present, then separate lines are constructed. The widths of the lines are expanded as necessary to accommodate each new segment. To avoid the creation of false lines, all complete lines must contain at least one segment that is a line of the same type as *cline*. In the final step, 2(e), the partials that have been used in a line are segmented along the edges of the line.

5. Region Formation

At this stage we have a list of patterns, which have all been individually classified. However, the goal is to define entire regions, such as paragraphs, from the document image, so the final step is to group patterns of the same type into complete regions. An issue that arises here is how to decide if separate patterns, of the same type, are part of the same region or different regions. The approach taken in this algorithm is to define a horizontal and vertical gap, which define the minimum distance between separate regions. These spacings are defined in terms of the average text pattern height (*avh*). The values were chosen to allow most text paragraphs to be merged into one block, without merging separate regions. These spacings are defines as follows:

$$hgap = 1.1 \times avh; \quad vgap = 0.8 \times avh$$

This means that a pattern is grouped into a region if (1) the pattern has the same type (id) as all the patterns in the region, and (2) the pattern lies within a horizontal distance of *hgap* and a vertical distance of *vgap* of a pattern in the region. This produces good results for most documents, but fails under certain circumstances. One of the problems is that text and title patterns may be incorrectly classified. It may happen that most of a title is classified as title while a few of the shorter characters are classified as text. Another issue is that separate paragraphs or lines of text, that lie directly above and below each other, should be grouped together. These problems are addressed by modifying the regions as defined above. Firstly, regions containing text or title patterns that overlap should be merged into one region. The resulting region (which should be either text or title) may contain a mixture of text and title patterns, so all the patterns are set to the most common pattern class. Lastly, regions that are horizontally or vertically close may be merged. This allows segmented titles and separate paragraphs to be merged.

6. Results

This algorithm is designed to segment newspaper sheets, but may be applied to any black and white document. In this section we demonstrate the segmentation algorithm with two sample images, which are extracts from the front page of the newspaper, “To Vima” (“TO BHMA”),

from 6th February 1968. Figure 4, which shall be referred to as sample 1, contains part of the title, several lines and some undesirable marks. These marks, possibly caused by a printing error or a stray pen, are easily identified with the human eye, but are a just another group of black pixels in the computers input sensors. For an algorithm to conclude that these marks are to be ignored is extremely difficult (at least without a document model or template), but it is possible to segment these marks from the genuine patterns they overlap. An algorithm for segmenting lines was described in section 4, and that is just what is needed to segment the bulk of these marks.

Figure 4. Sample 1.

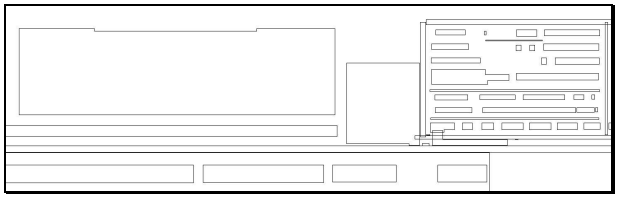

Figure 5. Illustration of regions extracted from Sample 1.

The output of our algorithm is shown in Figure 5. Each line in sample 1 has been successfully extracted, including the four edges of the text box, and the full horizontal line, which passes through the erroneous marks. The larger part of the unwanted mark (above the horizontal line) has been segmented, labelled as a line drawing and causes no more problems. The marks below the line, however, are badly smeared over text and do cause problems: three characters are lost. The rest of sample 1 has been well segmented and classified. The only issue is that not all of the text has been fully grouped into lines and paragraphs. This issue is addressed in the conclusion.

Sample 2, shown in Figure 6, is a larger section from the same newspaper sheet as sample 1. This image contains three columns, including a graphic and reverse text. Figure 7 illustrates the patterns that have been extracted from sample 2 and Figure 8 illustrates the identified regions. Although not specified in Figure 8, the photo, reverse text, line and title regions have all been correctly segmented and classified. The same may be said for the text regions, with a few small exceptions. One of the exceptions is the incomplete grouping of some regions. This was also seen in the results of sample 1. The only other exception is that a small number of text patterns are classified as lines. These patterns may be seen in the first and third columns. This error is due to thin characters such as ‘l’, ‘I’ and ‘i', which may look like lines. These errors can be avoided by setting a lower limit to the length of lines, but this may cause genuine lines to be missed. It is a difficult problem, but the vast majority of patterns are classified correctly.

Figure 6. Sample 2.

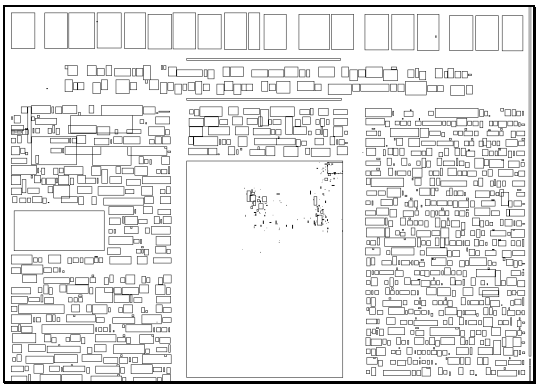

Figure 7. Illustration of the patterns located in Sample 2.

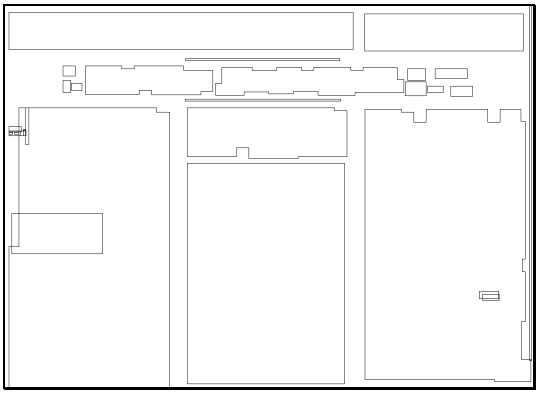

Figure 8. Illustration of regions extracted from Sample 2.

7. Conclusion

This paper has presented a complete algorithm for segmenting and classifying newspaper documents. The algorithm may equally be applied to any black and white document, because no assumptions are made about the document layout. The document is first scanned for rectangular regions called *rects*, and these are then grouped into patterns. As Figure 7 illustrates, patterns are small enough to keep separate regions separate, but larger then connected components, which allows for faster processing. These patterns are classified individually, using both general size and shape properties as well as run-length characteristics. Overlapping regions, such as the noise in the image shown in Figure 4, are a major hurdle for successful document analysis. An algorithm that addresses this problem for lines was presented in section 4. If any part of a line is recognised, then this algorithm enables the entire line to be extracted. The last step is region formation. This is where all the patterns are grouped into regions of the same class. Finally, section 6 presented some results, which illustrate the algorithms capabilities. One issue that arose was that of segmented text regions. It might seem that the solution is simple: increase the allowable gap between text patterns. This would indeed work, but would cause close text columns to merge. A solution may be to identify column boundaries by searching for long white spaces, and to merge all text lying between the columns. This can be a difficult task with the complicated layouts of many newspaper sheets, but it appears to be a way forward.

References

[1] P. Mitchell and H.Yan, "Document Page Segmentation based on Pattern Spread Analysis", Optical Engineering, Vol. 39, No. 3, pp724-734, March, 2000. J. Ha and R.M.

[2] Haralick, "Document Page Decomposition by the Bounding-Box Projection Technique", Proc. 3rd Int. Conf. Doc. Anal. Recogn. (ICDAR), Vol. II, pp. 1119-1122, 1995.

[3] D. Drivas and A. Amin, "Page Segmentation and Classification Utilising Bottom-Up Approach", Proc. 3rd Int. Conf. Doc. Anal. Recogn. (ICDAR), Vol. II, pp. 610 - 614, 1995.

[4] Antonacopoulos, "Page Segmentation Using the Description of the Background", Computer Vision and Image Understanding, Vol. 70, No. 3, June, pp. 350 -369, 1998.

[5] K.P. Chan and Y.S. Cheung, "Fuzzy-Attribute Graph with Application to Chinese Recognition", IEEE Trans. on Sys. Man and Cybernetics, Vol. 22, No. 1, pp153-160, Jan/Feb.1992.

[6] J. Lii and S.N. Srihari, "Location of Name and Address on Fax Cover Pages", Proc. 3rd Int. Conf. Doc. Anal. Recogn. (ICDAR), Vol. II, pp. 756-759, 1995.

[7] Y.Y. Tang, S. Lee and C.Y. Suen, "Automated Document Processing: A Survey", Patt. Recog, Vol. 29, No. 12, pp. 1931-1952, 1996.

[8] T. Watanabe and T. Sobue, "Layout Analysis of Complex Documents", Proc. 15th Int. Conf on Patt. Recog. (ICPR), Vol. 4, pp. 447-450, 2000.

[9] D. Ryu, S. Kang and S. Lee, "Parameter-independent Geometric Document Layout Analysis", Proc. 15th Int. Conf on Patt. Recog. (ICPR), Vol. 4, pp. 397-400, 2000.