Page Segmentation Techniques in Document Analysis

5

Koichi Kise

Contents

K. Kise
Department of Computer Science and Intelligent Systems, Graduate School of Engineering, Osaka Prefecture University, Sakai, Osaka, Japan
e-mail: kise@cs.osakafu-u.ac.jp

D. Doermann, K. Tombre (eds.), *Handbook of Document Image Processing and Recognition*, DOI 10.1007/978-0-85729-859-1_5,

Abstract

In this chapter, we describe various notions and methods of page segmentation, which is to segment page images into homogeneous components such as text blocks, figures, and tables. It constitutes the whole process called layout analysis along with the classification of segmented components described in ▸Chap. 7 (Page Similarity and Classification). This chapter starts with classification of page layout structures from various viewpoints including different levels of components and printing colors. Then we classify methods to handle each class of layout. This is done based on three viewpoints: (1) objects to be analyzed, foreground or background; (2) primitives of analysis, pixels, connected components, maximal empty rectangles, etc.; (3) strategy of analysis, top-down and bottom-up. The details of classified methods are described and compared with one another to know pros and cons of these methods.

Keywords

Background analysis • Bottom-up analysis • Connected component • Docstrum • Foreground analysis • Gabor filters • Layout analysis • Manhattan layout • Mathematical morphology • Maximal empty rectangle • Non-Manhattan layout • Page segmentation • Projection profile • Rectangular layout • Smearing • Top-down analysis • Voronoi diagram • Wavelet analysis • White tile

Introduction

Page segmentation is a task of extracting homogeneous components from page images. As components, we often consider text blocks or zones, text-lines, graphics, tables, and pictures. The task of page segmentation does not include the classification of components, which is described in ▸Chap. 7 (Page Similarity and Classification). It is important to understand that these tasks may not be separated; they are sometimes thought of as different sides of the same coin. Actually the task of both page segmentation and classification is often called "(physical) layout analysis." However, some methods are designed to work without classification. In this chapter, we put the main focus on such methods and touch to methods with classification if needed.

The main target of page segmentation described in this chapter is machine-printed pages with a certain layout, since the majority of efforts have been devoted to their analysis. Other targets of page segmentation are handwritten pages, which are also described shortly at the last part of this chapter.

History and Importance

The history of page segmentation is relatively long in the field of document analysis. After exploring the problem of character recognition, researchers started to address

the problem of character segmentation from a single text-line. This was around late 1970s and the beginning of 1980s. The need of page segmentation was recognized at about this time, since text-lines to be recognized are often laid out in pages, and thus it is necessary to extract text-lines from page images. In other words, in order to let character segmentation and recognition work, it is mandatory to apply layout analysis including page segmentation.

Evolution of the Problem

At very early stages of page segmentation, it was described as a part of a whole system such as "document analysis systems" [1]. In such systems, layout of pages to be processed is not so complicated. However, as soon as researchers attempted to process a wider variety of pages, they realized that the problem of page segmentation was not easily solved. It was recognized that there were pages ranging from easy to hard. This led researchers to classify page layout. As described later, the most fundamental layout is "rectangular" which means that all components can be represented as nonoverlapping rectangles. The problem of page segmentation has been evolved to expand the class of layout to be analyzed. An important subclass of layout is called "Manhattan" which means that each component has boundaries consisting of line segments parallel or perpendicular with one another. In other words, regions of components are not necessarily convex. After much effort was made for addressing the problem of Manhattan layout, the focus of research was moved to process non-Manhattan layout and beyond.

Another problem evolution is about the primitive for page segmentation. The simplest one is connected components (CCs), under the assumption that there is no difficulty to separate the foreground from the background. However, this assumption does not hold for pages in which components are overlapped. In such a case, we need to take pixels as the primitive.

The last evolution is about the strategy of processing. Since layout of pages is hierarchical in such a way that text-lines consist of characters, one can have a strategy to extract larger components first and then divide each extracted component into smaller ones. To be precise, text blocks are first extracted, and then from each text block, text-lines are extracted and so on. This strategy is called splitting or top-down. Readers can easily imagine that we also have a reverse strategy, from the smallest to the largest, i.e., starting from the primitives, they are combined to form larger components. This is called merging or bottom-up.

Applications

The most important application is to feed data to text-line extraction and character segmentation, since the task of page segmentation is designed to solve this problem. However, we also have several other applications, which are to use document images without recognition.

One important application is "reflow" of page contents. This is the process of generating a new image of a different size from its original. The purpose is to read the original document image on a smaller display available on a mobile phone. One idea to implement this functionality is to recognize the input document image to obtain the contents (symbol information) and display the contents on a smaller display by rendering them. However, this process is computationally expensive and not easy to be recovered from recognition errors. Another idea is not to recognize it but to apply "cut and paste" of the input document image to generate an image that fits into the new display. For this purpose, it is necessary to know the layout structure of the document image to apply correctly the cut and paste process.

Another important application is document image retrieval and clustering based on the layout of documents. In companies it is typical to store a large number of documents with the same layout but different contents, such as filled forms. It is also typical that in companies there are many documents with different layouts. Thus in order to deal with such documents efficiently, it is necessary to have a means to retrieve or to apply clustering based on the layout. The process of layout analysis is crucial to realize them.

Main Difficulties

The main difficulty of the page segmentation lies in achieving the generality of processing without losing accuracy and efficiency. It is not so difficult to implement a method of page segmentation for documents with a specific layout. For example, if the layout is strictly fixed, perfect page segmentation is achieved by just cutting the image at the fixed positions. On the contrary, if the layout is totally free, it is necessary to understand the contents of the input document to obtain correct components. The task of page segmentation lies between these two extremes. Layouts with more constraints are easy to handle than those without them. Segmentation of pages with less constraints is harder and thus often more difficult to achieve high accuracy and efficiency.

Therefore it is reasonable to apply page segmentation methods that are capable of dealing with layouts to be analyzed. It may be wasteful to apply too general methods if the layout is with more constraints. For example, it may deteriorate the accuracy and efficiency of segmentation if one applies a page segmentation method for non-Manhattan layout to documents with rectangular layout. On the other hand, if the layout is beyond the one that can be handled by a method, the results of segmentation can be catastrophic. Thus another difficulty is to know the class of layout of documents to be processed.

Summary of the State of the Art

Thanks to the effort that has been made, we can segment documents with constrained layouts such as rectangular and Manhattan. State-of-the-art technologies

of page segmentation are addressing the problem of segmentation of documents beyond Manhattan. As described later, the analysis of pages with layouts such as overlapping is still an open problem. Another important point is that the presence of noise makes the page segmentation difficult. This often occurs for analyzing old documents. Documents with less regularity such as handwriting are also hard to analyze.

To sum up, segmentation of pages with constrained layouts (up to Manhattan) and clean images has almost been solved. The analysis of pages beyond this, such as more general layout, noisy images, and/or presence of handwriting, is still under study.

In this chapter, we first define components of a document in the next section. Then we classify layouts and methods in sections "Classification of Pages and Their Layouts" and "Classification of Methods." Next, we describe details of methods of each class. Finally, most difficult cases of overlapping layout and handwritten documents are described.

Components of a Document

Let us start with the discussion on components of documents. They have a hierarchical structure as shown in Fig. 5.1. First, a document consists of pages each of which may include several components of a page. Typical components are text blocks, figures, tables, pictures, and ruled lines. A text block consists of text-lines, and a text-line is composed of characters. In this chapter, these components of a page are called "page components."

The task of page segmentation is to extract these page components from input document images. In this chapter we consider mainly the extraction of page components larger than text-lines.

Classification of Pages and Their Layouts

Next, let us consider the layout shown in Fig. 5.2 where filled squares indicate characters. For the squares in Fig. 5.2a, which of Fig. 5.2b or c do you think is the better interpretation of text-lines? As shown in Fig. 5.2b, c and d, e are both possible interpretations. This simple example indicates that the correct interpretation cannot be defined without the contents. Although this is an example of text-lines, the same holds for text blocks.

The above example suggests that, if we do not have an access to the contents, layout analysis, and thus page segmentation as its part, requires some assumptions on layout to obtain results of analysis. The purpose of this section is to describe such assumptions used especially in page segmentation.

Assumptions on layout can be classified into two levels: the text-block level and the text-line level. Each of them is described below.

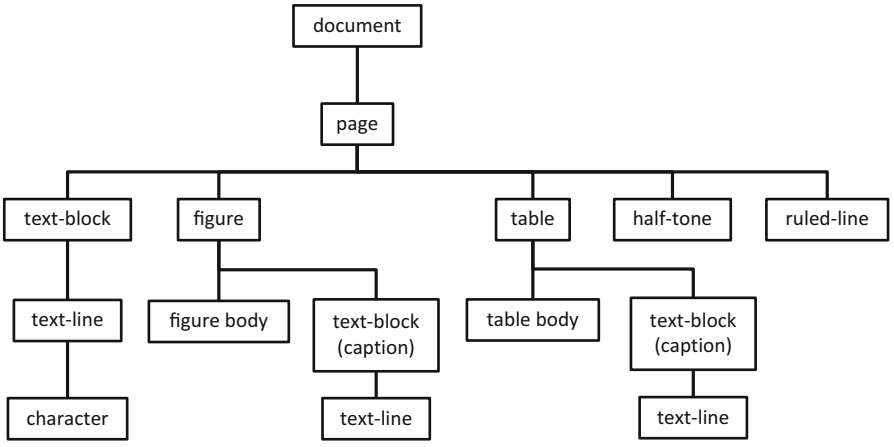

Fig. 5.1 Components of a document

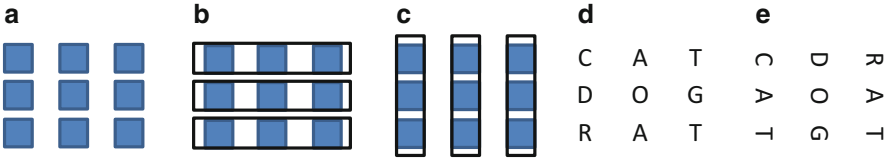

Fig. 5.2 Interpretation of layout

Text-Block Level

Rectangular Layout

The most fundamental layout is called "rectangular" which means all components are circumscribed by nonoverlapping rectangles whose sides are parallel or perpendicular to the border of a page. When pages are without skew, edges of rectangles are vertical or horizontal. Figure 5.3a shows examples of the rectangular layout. It is important to note that most of important publications such as business documents, scientific papers, and most of books are with this layout.

Manhattan Layout

Some components cannot be represented by nonoverlapping rectangles because of their concave shape as shown in Fig. 5.3b. If regions of components are represented by sides parallel or perpendicular with one another as shown in this figure, the layout is called Manhattan. Note that by its definition Manhattan layout includes rectangular layout as its subclass. Documents with multiple columns such as newspapers and magazines are often with this class of layout.

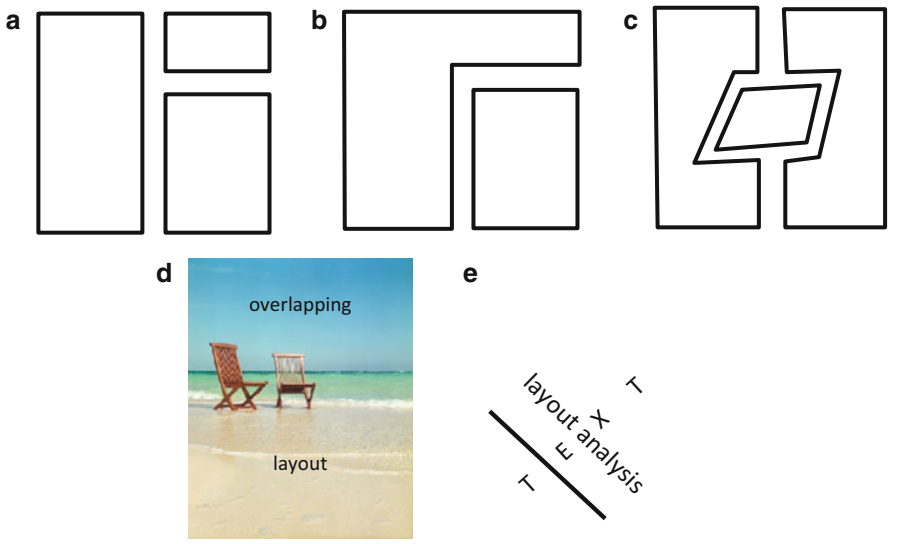

Fig. 5.3 Classes of layout. (**a**) Rectangular, (**b**) Manhattan, (**c**) non-Manhattan, (**d**) overlapping layout 1, and (**e**) overlapping layout 2

Non-Manhattan Layout

There is a class of layout beyond Manhattan. For example, components with slant sides as shown in Fig. 5.3c are not circumscribed by parallel or perpendicular sides. As shown in this example, a class of layout beyond Manhattan but all components are with no overlapping regions is called non-Manhattan. Readers may think that slant borders as shown in Fig. 5.3c can also be represented by using many small parallel or perpendicular sides as in Fig. 5.4. In order to avoid such cases, it is necessary for Manhattan layout to have enough length of sides. As is the relation between rectangular and Manhattan, non-Manhattan layout includes Manhattan as its special case. It is typical to have this class of layout for magazines with larger figures and pictures.

Overlapping Layout

All the layout classes introduced above are with components whose regions have no overlap. Although these classes cover most document publications, some documents such as graphical magazines are beyond these classes. For example, text can be laid out on a picture as shown in Fig. 5.3d. Another example is illustrated in Fig. 5.3e where text blocks intersect with each other. The former case is different from the latter. In the former case, there is no clear distinction between the foreground and the background; pixels of one component may be adjacent to those of others.

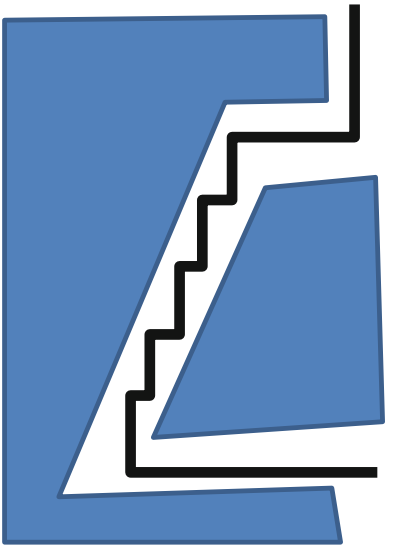

Fig. 5.4 Separation of components of non-Manhattan layout with short edges

On the other hand, in the latter case, the background is clearly separated from the foreground, and pixels of components are adjacent only with those of the background.

Although the former is quite common in modern publications such as advertisements, the latter is rare. An example can be found in maps, but they are out of the scope of this chapter. In the following, in contrast to the overlapping layout, the layouts, rectangular, Manhattan and non-Manhattan described above.

Text-Line Level

Page layout can also be classified with respect to the layout of text-lines. Let us again begin with the most basic one and go to less restricted layouts. Figure 5.5 shows the layouts described below.

Horizontal or Vertical Text-Lines

The most fundamental and thus common layout at the text-line level is that all text-lines are parallel to the horizontal side of the page as shown in Fig. 5.5a, where text-lines are represented as thick black lines. If the page image is without skew, the text-line is horizontal and thus we call this layout horizontal text-line layout. In some Asian languages, text-lines may be laid out vertically, which is also classified into this class. Most documents have this class of layout except for designed pages such as advertisements.

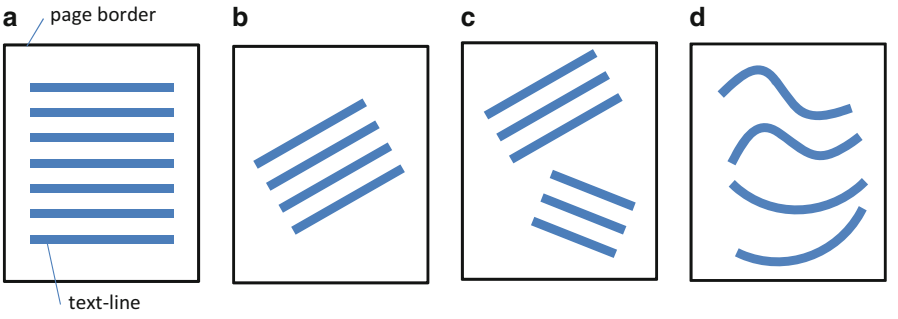

Fig. 5.5 Layout of text-lines

Parallel Text-Lines

Text-lines can be laid out parallel with one another but, as shown in Fig. 5.5b, may not be laid out horizontally. Another layout with more freedom is the layout with partially parallel text-lines as in Fig. 5.5c. These layouts are, however, not common and thus only available in advertisement documents.

Arbitrary Text-Lines

With the highest freedom, text-line may have arbitrary shape such as illustrated in Fig. 5.5d. However, pages only with this layout are uncommon. It may be used with the horizontal text-line layout for their main part.

Colors and Backgrounds

It is still major to have main text in black or dark color. In such cases, the background is often with white or light color. We call such documents black-and-white documents. The main text is also printed in black even in color-printed documents. Some parts are printed in different colors for emphasizing them. For advertisement documents, text may be printed totally in different colors with colored and/or textured background. A typical case would be text on a picture as shown in Fig. 5.3d. As a special case, a single text-line can be partly on a light background and the rest on a dark background. In this case the color of the text-line can be changed partly so as to make it distinguishable from the background.

Other Factors

Printed or Handwritten

Nowadays, printed documents are dominant in our daily life due to the extensive use of computers for their preparation. However, some documents are still in the mixture

of printed and handwritten or purely handwritten. The former case includes printed forms filled out with handwritten characters. The latter can be personal notes or historical documents. In this chapter, we mainly deal with printed documents, and at the end of the chapter, we touch the layout analysis of (partially) handwritten documents.

Clean or Degraded Documents

Thus far, we implicitly assume that all documents are clean enough for the processing. However, this does not hold for certain documents such as historical documents. Pages are degraded and a lot of noise such as stain can be found. In addition to historical documents, modern documents can also suffer from a similar problem if they are captured in an uncontrolled environment.

We have discussed various types of pages with respect to layout. In the history of layout analysis, researchers started their research from the most basic one, i.e., clean documents with rectangular layout and horizontal text-lines, and then gradually remove some assumptions to make their algorithms more general.

Classification of Methods

Before going into the details of various methods of page segmentation, it would be helpful to have a bird's-eye view of these methods. There are several criteria for classifying methods, i.e., (1) an object to be analyzed, foreground or background; (2) a primitive of analysis, such as pixels and CCs; and (3) a strategy of analysis, i.e., top-down or bottom-up. In the following, we describe the details of these criteria focusing mainly on the analysis of black-and-white documents.

Object to Be Analyzed: Foreground or Background

Suppose we analyze documents printed in black and white. It is typical that black parts represent the foreground such as characters and figures, while white parts correspond to the background. In this case, since there is a clear distinction between foreground and background, the analysis can be applied only to one of them. In this chapter the analysis of foreground and background is called foreground analysis and background analysis, respectively.

In both cases foreground and background are represented in terms of "primitives," which will be described in the next subsection. Note that the primitive is an element of images that belongs exclusively to one of the page components or background. Thus there is no need to divide further the primitives in the process of analysis. As shown in Fig. 5.6a, the task of foreground analysis is, therefore, to group primitives to form page components. On the other hand background analysis is to group primitives to obtain borders of page components as shown in Fig. 5.6b.

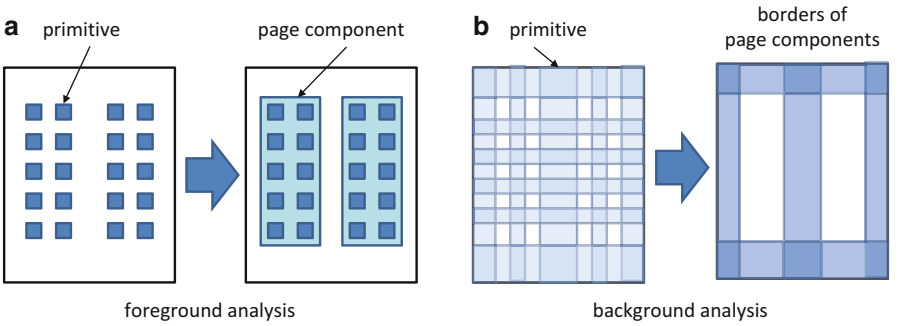

Fig. 5.6 (**a**) Foreground and (**b**) background analysis

If a document to be analyzed is not printed in black and white, the distinction of foreground and background is not trivial; the background of a page component can be the foreground of another page component, when documents have the overlapping layout. Even if a document is with a simple color print, foreground/background separation is not a trivial task; the same color can be used as both foreground and background. Thus the process of separation itself is a part of the goal of page segmentation.

Primitives of Analysis

The next criterion is about the primitive of analysis. Needless to say but the most fundamental and universal primitive is "pixel," which can be used in both foreground and background analysis. However, documents with some classes of layout allow us to use larger primitives that reduce the burden of analysis thanks to their smaller number. In the field of computer vision, such primitives are generally called "superpixels." For the task of page segmentation, superpixels can be of a variety of shapes depending on a class of layout.

For the case of black-and-white pages, an important primitive that can be used for nonoverlapping layout is "connected components" (CCs). As the connectivity for definition of CCs, it is common to employ 8-connectivity for black pixels. In this case, in order to avoid the contradiction of crossing connections, 4-connectivity is used for white pixels.

The burden of the processing can further be reduced by merging CCs. A natural way is to merge CCs if the distance between them is small enough. The question here is how to define the distance between CCs. There have been several methods for this purpose. The simplest is to measure the horizontal and vertical distance assuming that the skew is corrected [1]. A more general way is to use the morphological operation of "closing" that fills the gaps between black pixels. The distance can be defined as the Euclidean distance between centroids of CCs as well as the closest Euclidean distance between pixels of CCs.

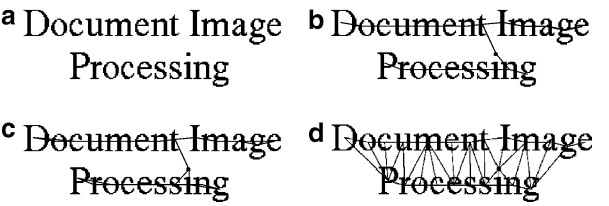

Fig. 5.7 Representations of adjacent relation among CCs. (**a**) Original image, (**b**) MST, (**c**) *k-NN* ($k = 2$), and (**d**) Delaunay triangulation

The above methods of merging CCs are done by just looking locally at the distance between CCs. However, it may depend on other CCs whether or not the CCs of interest should be merged. In order to evaluate such a "context" defined in relation to other CCs, we need a representation of relationship among CCs or a structure among CCs. For this purpose, there have been several methods proposed so far.

In general, a structure can well be represented by using a graph where nodes are CCs and edges are adjacent relations between them. The problem here is how to define the edges. Several ways have been proposed as shown in Fig. 5.7. Figure 5.7b is the minimum spanning tree (MST) that connects all CCs by minimizing the sum of distances of edges [2]. Figure 5.7c is k-nearest neighbors (*k-NN*) from each CC [3]. The last one, Fig. 5.7d is the Delaunay triangulation obtained from the CCs [4]. The details of these representation including pros and cons will be described later. One thing that needs to understand now is that once the page is represented as a graph of CCs, the task of page segmentation is to select necessary edges from the graph or equivalently to delete unnecessary edges from the graph.

The situation is different if we consider background analysis. It is typical that the background is a big single CC so that we need to utilize smaller primitives to form borders of page components. We have several ways to realize this as shown in Fig. 5.8. The most fundamental primitive is "maximal empty rectangles" [5] shown in Fig. 5.8b. A rectangle is empty if it only contains white pixels. An empty rectangle is maximal if it is unable to enlarge its area while keeping it empty. In other words, all sides of an empty rectangle touch to either the border of a page or black pixels. Note that the maximal empty rectangles have sides parallel or perpendicular to the side of a page, assuming that pages have been deskewed. Figure 5.8c shows a way to deal with skewed pages. This representation is called white tiles [6]. The white tiles are also empty rectangles but not maximal. They have variable heights with the maximal width. If the height is small enough, it can represent white space even for skewed pages as shown in Fig. 5.8c.

Similarly to the primitives of foreground, we can also think about the structure among background primitives. An important property is that the representation should keep the topology of background. The area Voronoi diagram [7] shown in Fig. 5.9b realizes a representation with this property. Edges between connected components represent the equidistant positions between borders of connected components. Because the distance does not change by rotation, another important property of the area Voronoi diagram is that it is rotation invariant. This means that

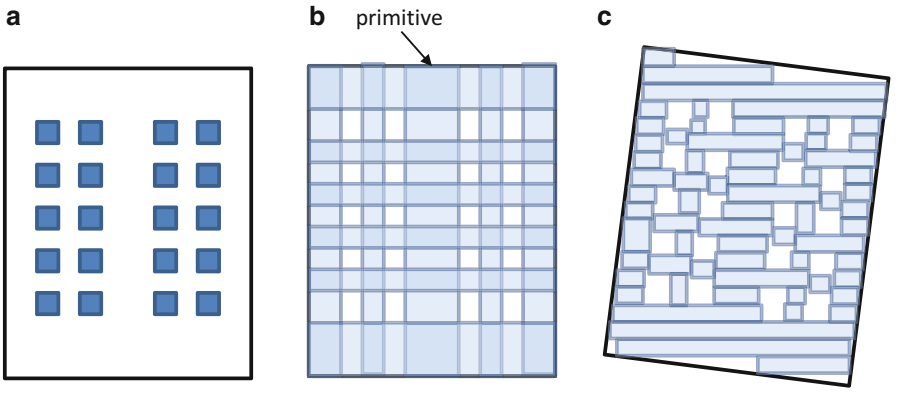

Fig. 5.8 Primitives for representing the background. (**a**) Foreground, (**b**) maximal empty rectangles, and (**c**) white tiles

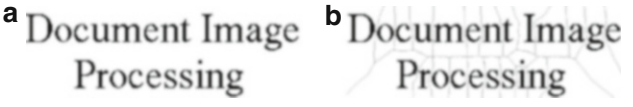

Fig. 5.9 Structural representations of background. (**a**) Original image and (**b**) area Voronoi diagram

even if an input image is rotated, the same representation can be obtained. Once the structural representation of background is constructed, the task here is to select from the graph appropriate edges that represent borders of page components.

For gray-level and color document images, separation of the foreground from the background is not trivial. Thus the primitives for the processing is first to solve this problem. Although the approach of superpixels may also be applied for this purpose, only few attempts have already been made. The most common way is to convert the analysis of gray-level and color documents into that of black-and-white documents. It can be done by applying color clustering; after the clustering, the image of each cluster can be viewed as a black-and-white image. If it is not possible to obtain reasonable results by the clustering, the appropriate primitive is "pixel." For the analysis of overlapping layout, for example, it is normal to analyze images using this primitive.

Strategy of Analysis

The final criterion is about the strategy of analysis. As shown in Fig. 5.1, page layout has a hierarchical structure. The strategy can be divided into two, with respect to from which side (from either the root or the leaves) the processing starts.

The strategy from the root, i.e., the whole page, is called "top-down," while that from the leaves is called "bottom-up." The top-down analysis is to split the whole page into the second level (text blocks, figures, tables) and continues this splitting until having the leaves such as characters or text-lines. On the other hand, the bottom-up analysis is to merge the primitives into the leaves and then continue to merge until obtaining the page components at the highest level. In the literature some methods take an intermediate strategy, starting from the middle (e.g., words); the analysis is to obtain larger components such as text blocks and smaller components such as characters.

The use of the terms top-down and bottom-up is similar to that of parsing in the field of language analysis. As is needed in the language parsing, those strategies generally require "backtracking" if a tentative result of analysis is found incorrect. For example, in the top-down analysis, the analysis may start by assuming that the page layout is two-column. Then after obtaining columns, it may be found that the result is incorrect due, for example, to different column widths. In order to pursue other possibilities of analysis such as three-column layout, the result of two-column should be cancelled by backtracking. In practice, however, the backtracking is rarely applied, simply because it is too time-consuming. Algorithms of page segmentation tend to be designed to work without backtracking. This is similar to the situation of parsing algorithms for programming languages. Note that although the parsing algorithms have a guarantee that it works properly without backtracking, there is no guarantee for the case of page segmentation.

Analysis of Pages with Nonoverlapping Layout

In this section, we describe the details of page segmentation algorithms based upon the criteria introduced in the previous section. We first focus on methods for analyzing pages with nonoverlapping layout. We start from foreground analysis methods, followed by background analysis methods.

Table 5.1 lists representative methods for nonoverlapping layout. As we have already seen, methods are classified into two categories: foreground analysis and background analysis. The former is further divided into three subcategories: projection-based, smearing-based, and connected components-based methods. We describe each method in detail in this section.

Projection-Based Methods

We first introduce projection-based methods which utilize projection profiles of a page image. Figure 5.10 shows the horizontal projection profile of a page. If there is no skew of a page and all text-lines are parallel to the horizontal sides of a page, we can observe the periodical gaps as shown in this figure. The length of gaps is a clue to find a border between page components. Projection profile methods employ this clue to analyze pages. To be precise, gaps are identified as the ranges below the threshold τ_h of the projection profile as shown in Fig. 5.10. We can split blocks vertically

Table 5.1 Representative methods for nonoverlapping layout

Method	Analysis	Primitive/representation	Strategy	Assumption
1. Projection-based methods				
Recursive XY cut (RXYC) [8]	Foreground	Projection profile	Top-down	No skew
Syntactic segmentation [9]	Foreground	Projection profile	Top-down	No skew
White streams [10]	Foreground	Projection profile	Top-down	
Hough transform [11, 12]	Foreground	Hough domain representation	Bottom-up	
2. Smearing-based methods				
Run-length smearing algorithm (RLSA) [1]	Foreground	Smeared pattern	Top-down	No skew
Morphology [13, 14]	Foreground	Pixel	Bottom-up	
3. Connected components-based methods				
Minimum spanning tree [2, 15, 16]	Foreground	Minimum spanning tree	Bottom-up	
Docstrum [3]	Foreground	*k-NN*	Bottom-up	
Delaunay triangulation [4]	Foreground	Delaunay triangulation	Bottom-up	
4. Background analysis methods				
Shape-directed covers [5, 17]	Background	Maximal empty rectangles	Top-down	No skew
White tiles [6]	Background	White tiles	Top-down	
Voronoi diagram [7, 18]	Background/ foreground	Voronoi edges	Top-down	

processor simulator and a detailed memory simulator for the Dash prototype. Tango allows a parallel application to run on a uniprocessor and generates a parallel memory-reference stream. The detailed memory simulator is tightly coupled with Tango and provides feedback on the latency of individual memory operations.

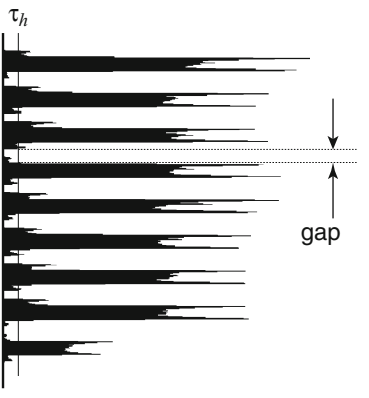

Fig. 5.10 Projection profiles

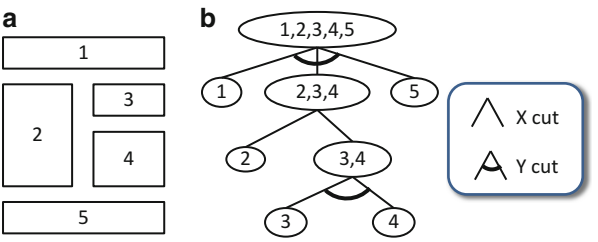

Fig. 5.11 Recursive XY cut. (**a**) Blocks and (**b**) XY tree

by finding wider gaps of horizontal projection profiles. This is called vertical cuts. Likewise, horizontal cuts can be found in vertical projection profiles.

A representative method of this category is called Recursive XY Cut (RXYC) [8]. In order to understand the algorithm, we first need to know the layout structure to be analyzed by this method. Figure 5.11 shows such a layout structure. Nodes of the tree in Fig. 5.11b represent regions. For example, (2,3,4) indicates the region consisting of blocks 2, 3, and 4. Each edge represents either X cuts or Y cuts. Edges of the tree also indicate a part of relation between nodes: regions of children of a node are included in the region of the node. X and Y cuts are applied alternately from top to bottom of the tree. Thus the processing is top-down. The leaves of the tree correspond to the smallest page components. The algorithm of RXYC works to construct the tree by analyzing projection profiles.

The above method employs a general assumption on layouts such as "inter-text-line spacing is narrower than inter-text-block spacing." However, this is not the only way of top-down page segmentation. We can apply the top-down page segmentation which assumes a specific layout structure if we have already known it. The knowledge on a specific layout can be represented as a set of production rules [9]. This method analyzes a projection profile by rewriting it to extract page components. Such methods that employ document-specific knowledge of layout are sometimes called "model-based methods." Note that in general we do not use a model-based method just only for page segmentation; it is used in combination with or in the process of logical layout analysis described in ▸Chap. 6 (Analysis of the Logical Layout of Documents).

Let us go back to methods without knowledge about specific pages. Many methods based on projection profiles assume that input page images are deskewed. In the case that the skew correction is incomplete, they generally fail to segment pages. This problem can be solved by applying projection locally [10]. In this method a page image is split into horizontal belts (blocks of scan lines) with a predetermined height. Then for each belt the projection profile is calculated for finding column gaps. Note that if the height of the belts is small enough, the influence of skew is limited so that column gaps can be found.

Another possible method for coping with skew is to calculate projection profiles by rotating the axis onto which black pixels are projected. It is known that this process is equivalent to the Hough transform [11]. For example, the Hough transform is applied to the centroids of CCs to extract text-lines from line

drawings [12]. However, most of the methods using the Hough transform are not for page segmentation but for skew correction, which are described in ▸Chap. 4 (Imaging Techniques in Document Analysis Processes).

Smearing-Based Methods

Projection-based methods try to find white space (gaps) between page components. The totally reverse idea is to fill the space within each component to extract it. Smearing-based methods are methods based on this idea.

Run-Length Smearing Algorithm (RLSA)

The earliest proposal was called run-length smearing algorithm (RLSA), which is sometimes called run-length smoothing algorithm. Suppose we deal with binary documents where 0s and 1s represent white and black pixels, respectively. The run-length means the number of continuous 0s or 1s along with the scan line. For example, 0000 and 111 are converted into 4 and 3, respectively. The run-length encoding of a binary image is to convert each scan line of the input image to its run-length representation. For example, a bit sequence shown in Fig. 5.12a is transformed into the one in Fig. 5.12b with the minimum length $T = 2$ of white runs (white runs less than or equal to 2 are filled). Note that such a bit operation is extremely efficient, which is an advantage of this method.

In RLSA, the above run-length smearing is applied both horizontally and vertically. Figure 5.13 shows an example. The original image shown in Fig. 5.13a is transformed into horizontally smeared (b) and vertically smeared (c) images. Note that the appropriate threshold value T is different for horizontal and vertical smearing. In Fig. 5.13c, columns are correctly extracted. In Fig. 5.13b, text-lines and figures in different columns are merged. RLSA takes the logical AND of these horizontally and vertically merged images to generate Fig. 5.13d, where most text-lines and figures in each column have been successfully extracted. Similar to the case of the RXYC, RLSA also assumes that the skew has been corrected.

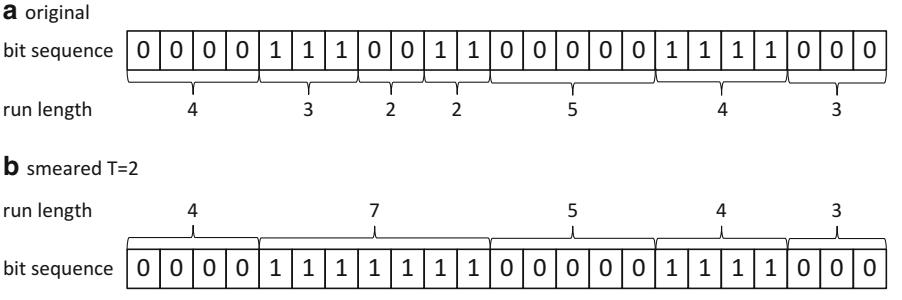

Fig. 5.12 Run-length smearing

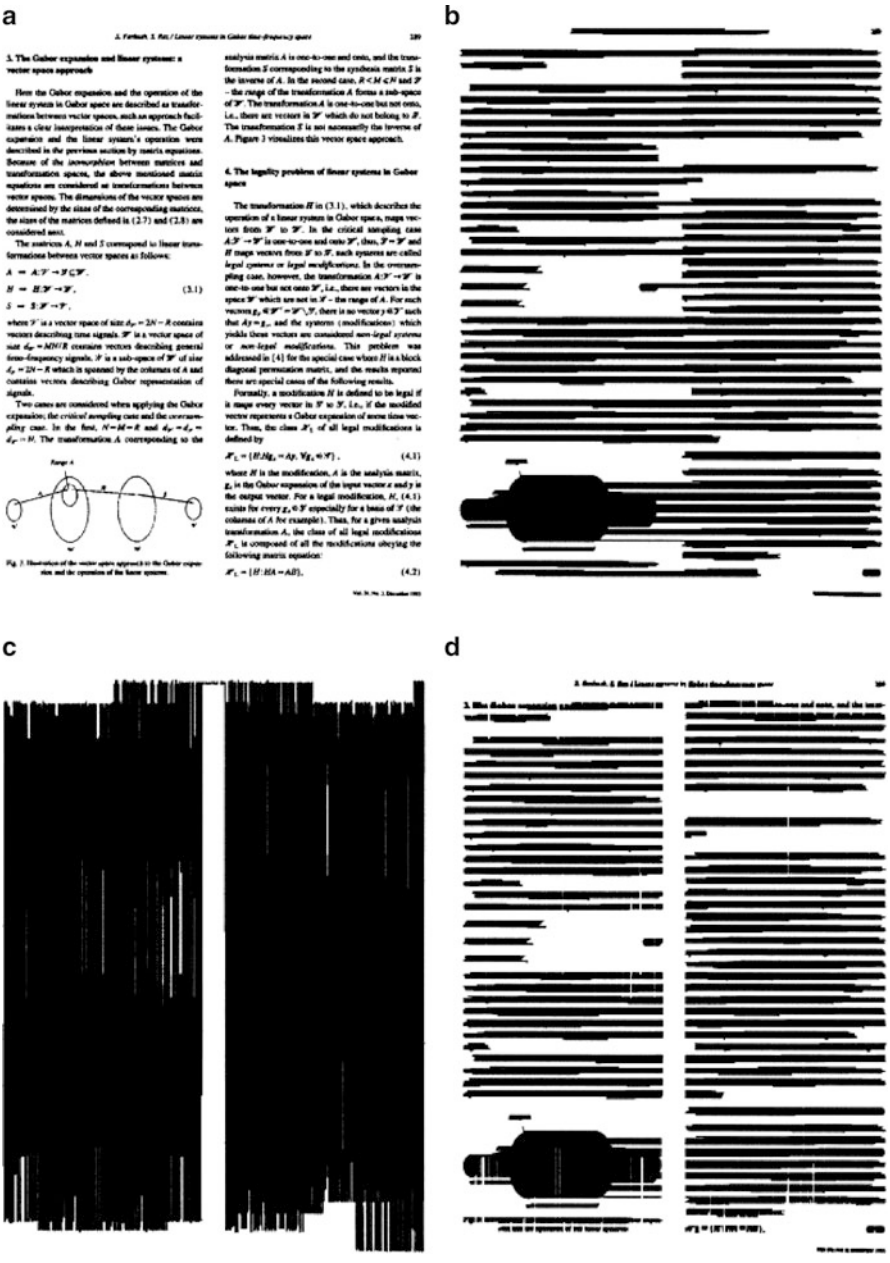

Fig. 5.13 Run-length smearing algorithm (RLSA) [1] (Reprinted from Fig. 5.4 in Yin [19])

Morphology-Based Method

The RLSA algorithm described above can be thought of as a special case of more general processing which we call here "morphology-based methods." The fundamental operations of morphology-based methods are dilation and erosion from mathematical morphology:

$$\text{Dilation: } A \oplus B = \bigcup_{b \in B} A_b$$

$$\text{Erosion: } A \ominus B = \bigcap_{b \in B} A_{-b}$$

where A is an input image, B is a structural element, and A_b is the translation of A along the pixel vector b. As the names indicate, dilation and erosion are to dilate and erode the foreground of original image A by the amount the structural element B defines, respectively. Based on these two operations, opening and closing are defined as follows:

$$\text{Opening: } A \circ B = (A \ominus B) \oplus A$$
$$\text{Closing: } A \cdot B = (A \oplus B) \ominus A$$

Opening allows us to remove objects which are small with respect to the structural element, while closing fills gaps which are small with respect to it. If the structural element is isotropic, images with skew can also be handled. With the knowledge about layout structure such as the direction of text-lines, an anisotropic structural element allows us to find text-lines and blocks more effectively. The closing operation with the structural element $1 \times T$ represents the horizontal smearing shown in Fig. 5.12.

In order to make the segmentation process efficient and reliable, a series of morphological operations is applied. Bloomberg has proposed multiresolution morphology to separate picture regions from text [13]. Its fundamental processing is as follows. Since the distance between pixels in halftones is smaller than that between characters, one can apply closing for filling the gaps in halftone pixels. This results in large blobs that represent parts of halftones. Then the opening is applied to erase characters. The remaining is the large blobs that become "seeds" to find pictures. This process requires the application of relatively large structural elements, which lowers the efficiency of processing. Actually, one of the major drawbacks of morphology is its computational burden. This problem has been solved by using multiresolution morphology called "threshold reduction." In this method, 2×2 pixels are transformed into one pixel by using a threshold for the sum of 2×2 pixel values. This allows us to mimic opening and closing depending on the threshold. Applying the threshold reduction several times with different thresholds, it is possible to achieve a similar effect with a much lower processing cost.

Connected Component-Based Methods

In this section, we introduce the analysis based on CCs. There are many methods of connected component analysis which can be characterized by the representation of structure among the CCs as we have already seen in section "Primitives of Analysis."

Minimum Spanning Tree (MST)

The first method we describe here is based on the minimum spanning tree (MST). We define the distance between CCs as the Euclidean distance between centroids of CCs. Consider here a graph whose nodes are CCs and whose edges are between centroids and thus associated with the distance. Then the minimum spanning tree is defined as the tree structure that connects all nodes with the minimum sum of distance of edges [2].

The construction of MST is quite easy; greedy methods allow us to obtain the MST. Here we describe Kruskal's. First, find the edge whose distance is the shortest among all possible edges between CCs and delete it from the set of all possible edges. Then select the next edge whose distance is the shortest among the remaining edges, on condition that the inclusion of the selected edge still keeps the tree structure. Otherwise, the edge is thrown and the next one is selected. When there is no more remaining edge, the resultant tree represents the MST.

Once the MST is constructed, it can be used to extract page components as subtrees of the MST [15]. This is based on the following assumption about the distance between connected components. In general, the distance between CCs in the same page component is smaller than that between CCs in different page components. If this assumption holds, the process of page segmentation is to select edges that lie between different components and delete them. Dias employs the distance associated with edges as well as information on CCs for the selection [16]. The threshold is calculated based on the distribution of distance.

Docstrum

The problem of MST-based page segmentation is that the assumption does not always hold. In other words, some edges lying between different page components have smaller distance than those within the same page component. A way to solve this problem is to select edges for deletion not only based on the distance but also other features. Another possible way is to change the representation.

The document spectrum or *docstrum* proposed by O'Gorman [3] is a method that overcomes the above difficulty. The idea is simple; the problem can be rephrased as the lack of edges within page components. In order to have more chances not to lose such "within" component edges, edges for the representation are increased. For this purpose, k-nearest neighbor (k-*NN*) graph is employed in the docstrum, where nodes again represent CCs, and edges indicate k-*NN* from each CC. An example is shown in Fig. 5.14.

Fig. 5.14 5-NN of connected components

Taking *k-NN* of CCs is simple but powerful to obtain statistical information of the page. For each *k-NN* pair (*ij*) of CCs, we can observe (d, ϕ) where d is the distance and ϕ is the angle of each edge. The 2D plot of (d, ϕ) is called docstrum. The value of k is typically set to 5. From upright page images, it is typical to have clear peaks at 0 and $\pm 90°$ which correspond to angles of edges in text-line and between text-lines, respectively. This shows that based on these statistics, the skew angle, within-line and between-line spacing, is estimated as follows: First, the highest peak of angle distribution indicates the skew angle. The within-line and between-line spacing are estimated from the distance distribution of edges close and perpendicular to the estimated skew angle, respectively.

Then the above estimated parameters are employed for page segmentation. The processing is fully bottom-up. First, in order to extract text-lines, connected components are merged by using the estimated skew angle and within-line spacing. Next, text blocks are extracted by merging text-lines again by taking into consideration parameters such as angles of text-lines and distance between text-lines. Note also that there are several clear peaks in the distance distribution. The peak with the smallest distance shows the inter-character distance, and the peak with the largest distance indicates the inter-text-line distance. By using these values, it is possible to automatically adjust the parameters for merging CCs to obtain page components.

The above estimation of parameters is based upon the assumption that the values of these parameters are common in the page. In other words, the parameters are globally estimated. In the case that there are several page components with different values of parameters, the additional processing is necessary. The simplest one is to apply clustering in the parameter space so that each cluster represents possible values.

Delaunay Triangulation

Although the docstrum is powerful and effective, a problem still remains in setting the parameter k since the appropriate value of k depends on the layout. If it is larger than the appropriate value, the *k-NN* graph includes edges spurious to estimate within-line and between-line spacing. For example, a larger k produces edges that connect not only neighboring CCs but also further CCs. For example, in the word "example" an edge from "*x*" to "*m*" can also be set in addition to "*a*." On the other hand, the value of k smaller than the appropriate value prevents us from having enough edges between text-lines, which makes the estimation of between-line spacing unreliable. In order to solve this problem, we need a representation

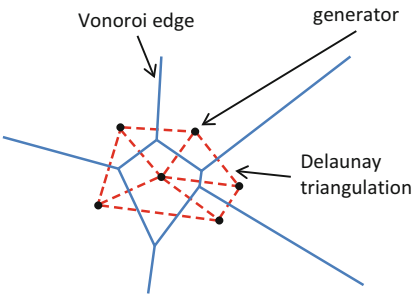

Fig. 5.15 Point Voronoi diagram and its corresponding Delaunay triangulation

that limits edges to the ones between the "neighboring" connected components. Note that the MST cannot give us a solution, since it tends to miss edges on the neighboring CCs.

This problem can be solved by the diagram called Delaunay triangulation. First, let us introduce some definitions by using Fig. 5.15 as an example. Let $P = \{p_1, \ldots, p_n\}$ be a set of points or *generators* and $d(p, q)$ be the distance between points p and q. A Voronoi region $V(p_i)$ of a point p_i is defined as

$$V(p_i) = \{p | d(p, p_i) \leq d(p, p_j), \forall j \neq i\}.$$

The Voronoi diagram $V(P)$ generated from the point set P is defined as a set of Voronoi regions:

$$V(P) = \{V(p_1), \ldots, V(p_n)\}.$$

The boundaries of Voronoi regions are called Voronoi edges.

The dual graph of the Voronoi diagram is called the Delaunay triangulation. An example is also shown in Fig. 5.15 using dashed lines. Nodes of the Delaunay triangulation are generators, and edges exist between pairs of generators whose Voronoi regions share a Voronoi edge.

We can construct the Voronoi diagram and the Delaunay triangulation by taking centroids of CCs as generators. However, if the shape of CCs is far from the point and the size of CCs are not uniform, it is better to take their shape into account when we construct the Voronoi diagram and the Delaunay triangulation.

First, let us consider the case of the Voronoi diagram. The Voronoi diagram generated from figures of arbitrary shape is called the area Voronoi diagram. In order to distinguish it from the previously defined Voronoi diagram, we call the previous one the point Voronoi diagram. The definition of the area Voronoi diagram is as follows. Let $G = \{g_1, \ldots, g_n\}$ be a set of nonoverlapping figures (in our case CCs) and let $d(p, g_i)$ be the distance between a point and a figure defined as

$$d(p, g_i) = \min_{q \in g_i} d(p, q).$$

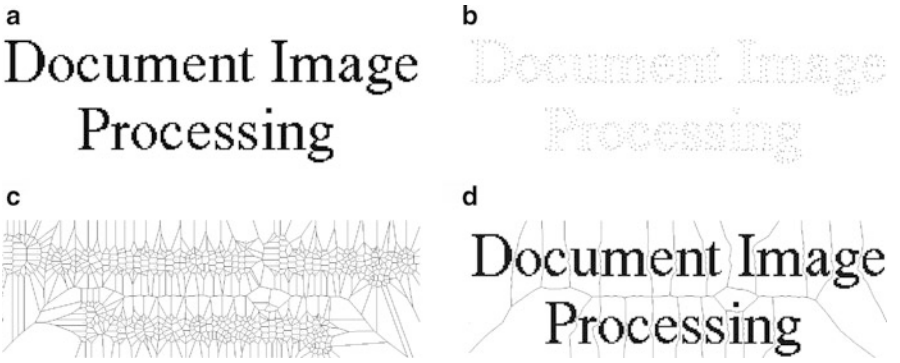

Fig. 5.16 Approximate construction of the area Voronoi diagram [7]. (**a**) Original image, (**b**) sampled points, (**c**) point Voronoi diagram, and (**d**) area Voronoi diagram

The Voronoi region $V(g_i)$ and the Voronoi diagram $V(G)$ are defined as

$$V(g_i) = \{p | d(p, g_i) \leq d(p, g_j), \forall j \neq i\},$$
$$V(G) = \{V(g_1), \ldots, V(g_n)\}.$$

The area Voronoi diagram can easily be constructed in an approximate manner by sampling points on borders of CCs as generators and delete Voronoi edges lying on the same CC. Figure 5.16 shows an example. From the points (Fig. 5.16b) of CCs (Fig. 5.16a), the point Voronoi diagram (Fig. 5.16c) is generated. By deleting edges lying between the same connected components, the area Voronoi diagram (Fig. 5.16d) is obtained.

As the dual graph of the area Voronoi diagram, we can also define the Delaunay triangulation that connects figures of arbitrary shape sharing the Voronoi edge as shown in Fig. 5.7d. In this Delaunay triangulation, the distance associated with edges is defined as

$$d(g_i, g_j) = \min_{p_i \in g_i, p_j \in g_j,} d(p_i, p_j).$$

In the approximate construction, p_i and p_j are sampling points of g_i and g_j, respectively.

Based on the Delaunay triangulation of a page, the task of page segmentation is to delete edges that connect different page components. In [4], a bottom-up method for the selection has been proposed. In this method, the area ratio of CCs connected by an edge as well as the angle of an edge is employed for the selection. Starting from the Delaunay triangulation shown in Fig. 5.17b, the method first selects seeds, which are reliable parts of text-lines, and then extends them to form full text-lines as in Fig. 5.17c. By merging text-lines taking into account their orientation, length, and distances, we can obtain text blocks.

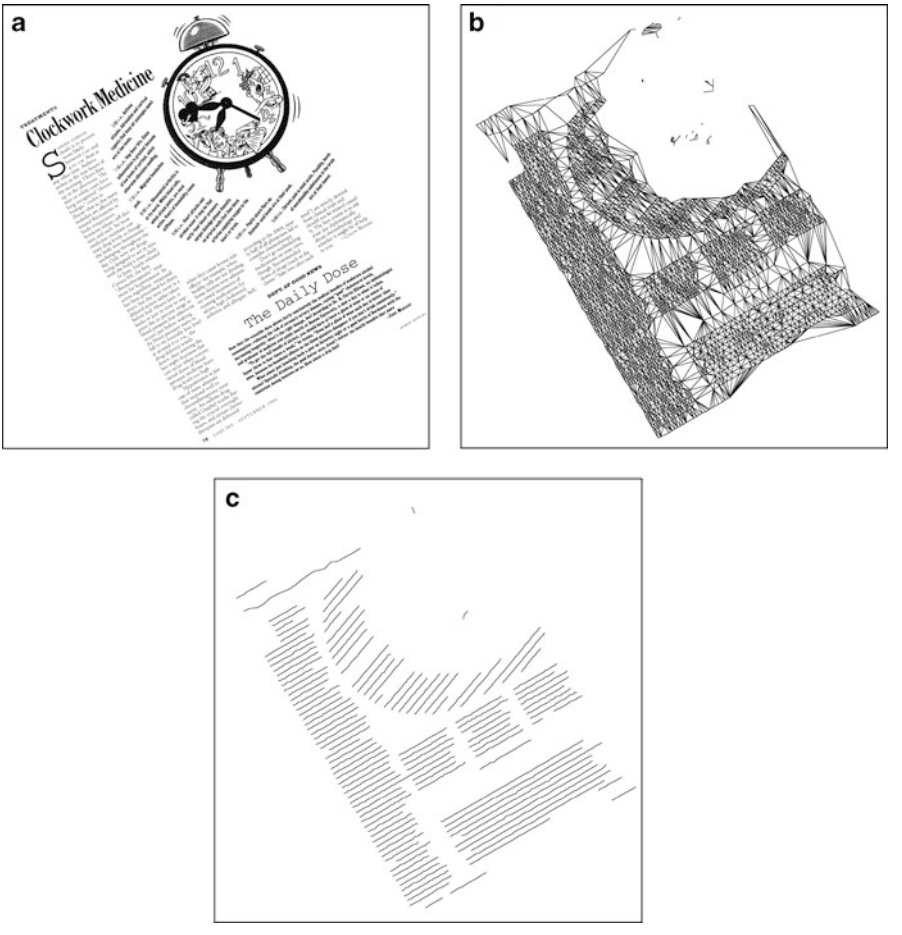

Fig. 5.17 Delaunay triangulation and text-lines as its subgraphs. (**a**) Original image, (**b**) Delaunay triangulation, and (**c**) text-lines

Background Analysis Methods

We have seen methods that work directly on the foreground to extract page components by merging primitives. As described earlier, there exist different methods that deal with the background to segment the whole page into page components. In this section, some representative methods of this category are described.

Shape-Directed Covers

The method "shape-directed covers" proposed by Baird [5] is the first well-known method of this category. Figure 5.18b shows blank regions around page components. As you can see, these regions can be represented by white rectangles each of which

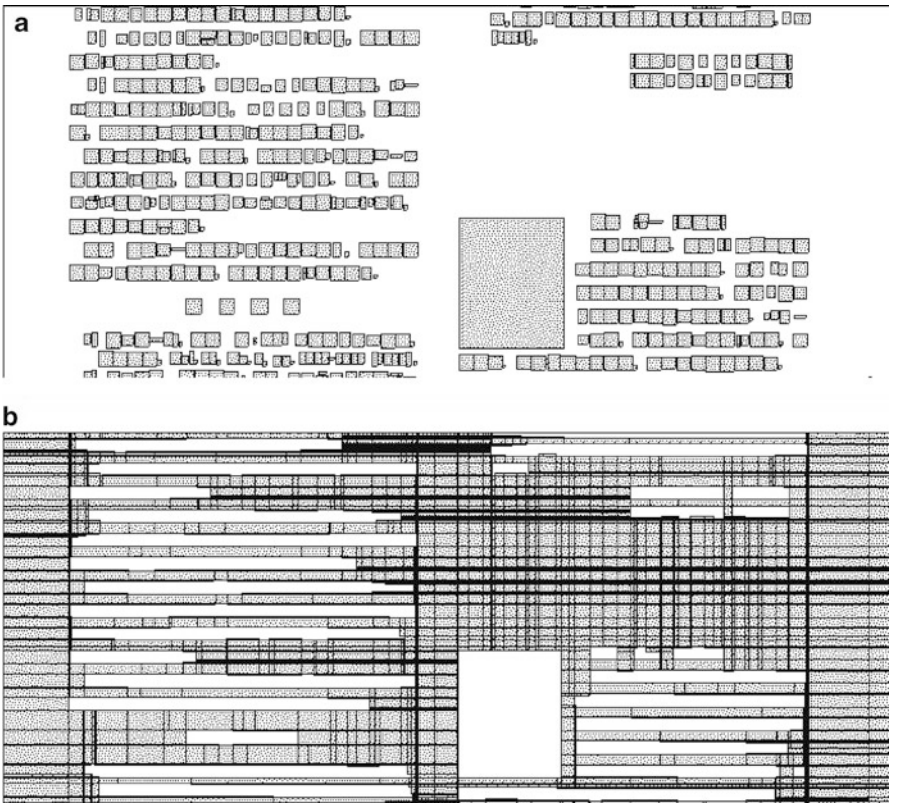

Fig. 5.18 Examples of (**a**) black and (**b**) white rectangles (Reprinted from Fig. 5.4 in "Page Segmentation Based on Effective Area of Background," Trans. IEE of Japan, Vol. 116, No. 9, p. 1038, 1996)

maximally covers a part of the blank regions. The first question here is how to extract such white rectangles in an efficient way.

Although there are several methods proposed for this purpose, it has been pointed out that they are not always easy to implement. In this section we describe an algorithm inspired by Breuel [17] which is to extract "maximum"(largest) white rectangle based on the branch and bound.

The input to the algorithm is a set of black rectangles $B = \{b_1, \ldots, b_n\}$ in the page bounding box w_p. As black rectangles b_i, we often utilize bounding boxes of CCs. Examples are shown in Fig. 5.18a. The task here is to find a set of empty (white) rectangles $W = \{w_1, \ldots, w_m\}$, under the conditions that (1) $w_i \subset w_p$ where $\subset$ means the inclusion of rectangles, (2) $\forall i, j\, w_i \cap b_j = \varnothing$ where $\cap$ indicates their intersection, and (3) $\forall i\, w_i$ is maximal, i.e., all sides of the white rectangle touch either on the side of w_p or that of a black rectangle b_i.

The key idea of the algorithm is shown in Fig. 5.19 where the outer rectangle in Fig. 5.19a is a bound and smaller rectangles in the bound are black rectangles.

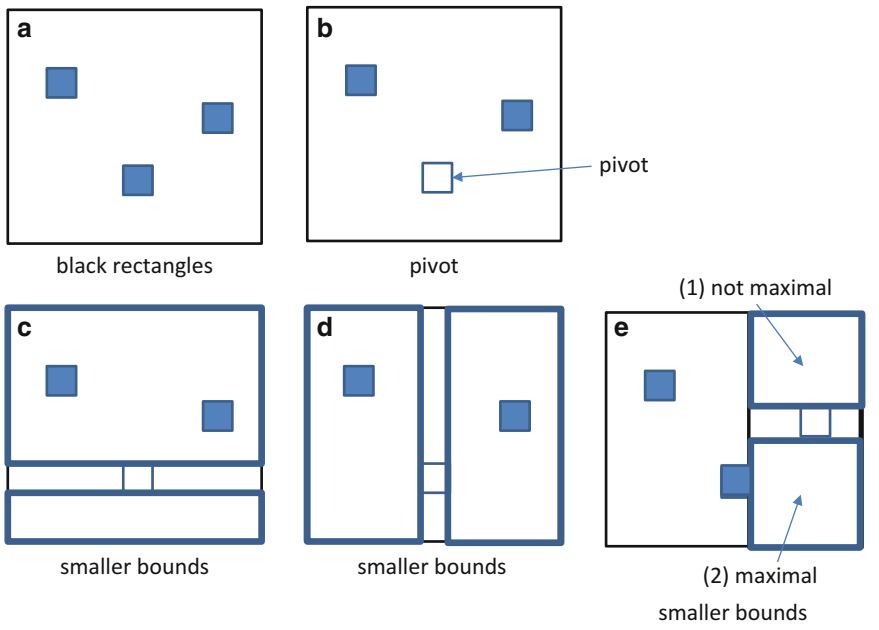

Fig. 5.19 A branch-and-bound algorithm for maximal empty rectangles. (**a**) Black rectangles, (**b**) pivot, (**c**) smaller bounds, (**d**) smaller bounds, and (**e**) smaller bounds

At the beginning, the bound is set to the page bounding box. The algorithm works as follows. If the bound is not empty, i.e., including black rectangles, the bound is divided. For each black rectangle, take it as a pivot as shown in Fig. 5.19b and generate four smaller bounds as in Fig. 5.18c, d. Then for each bound, select black rectangles that overlap with the bound and call recursively the procedure. On the other hand, if the bound is empty, it may be a maximal white rectangle. The maximality can be confirmed by checking whether all sides of the bound touch to either a black rectangle or a side of the page bounding box. The bound (1) in Fig. 5.19e is not maximal since the left side touches nothing. The bound (2), on the other hand, is a maximal white rectangle.

Once all maximal white rectangles are extracted, page segmentation can be done by selecting white rectangles appropriate as borders of page components. Baird et al. have proposed to use the shape score $s = a \times r$ where a is the area of a white rectangle and r is a truncated aspect ratio (the ratio of the longer to the shorter side length, but if it is larger than a threshold, say 16, it is set to 0). White rectangles are sorted by the score and select as borders from top to a certain limit.

White Tiles

The white rectangles are effective under the following assumptions: (1) the page is precisely deskewed and (2) the layout is Manhattan. These assumptions are removed by the method called the white tiles proposed by Antonacopoulos [6]. A white tile

is a vertically concatenated white runs with the following condition. Let (x_{si}, x_{ei}) be the x coordinates of the start and the end of the ith white run. The white tile is a concatenated white runs from jth to kth ($j \leq k$) on condition that

$$\sum_{i=j}^{k-1}(|x_{si+1} - x_{si} + 1| + |x_{ei+1} - x_{ei} + 1|) < T,$$

where T is a threshold. An example of white tiles is shown in Fig. 5.20c, which is constructed from the smeared image (Fig. 5.20b) of the original page image (Fig. 5.20a). Based on the extracted white tiles, the task of page segmentation is to connect them to form the borders of page components. In [6] contours of white tiles are traced to circumscribe a region that is recognized as a page component. An example of traced contours is shown in Fig. 5.20d.

Voronoi Diagram

A disadvantage of the white tiles is that the representation is not rotation invariant. Thus the parameter for merging the while tiles should be determined based upon a possible range of skew angles. There is no problem if the skew can be limited within a narrow range such as in the case that documents are scanned by a flatbed scanner. If it does not hold, on the other hand, it is better to use a method with a rotation invariant representation.

Such a representation is realized by the area Voronoi diagram shown in Fig. 5.9. As shown in Fig. 5.21b, it represents the background structure of a page. Based on this representation, the task of page segmentation is to select Voronoi edges that form the borders of page components. This can be paraphrased as to remove unnecessary Voronoi edges. As the feature of deletion, Kise et al. employ simple features such as the minimum distance between CCs sharing an edge as their border, as well as the area ratio of the CCs [7]. The threshold for the distance is calculated based on statistics on distance distribution in the page. It is often the case that there is clear peaks that represent the inter-character and the inter-text-line gaps of the body text. This idea comes from the NN distance of docstrum. Note that in the case of the area Voronoi diagram, we do not have the parameter k of *k-NN* that affects the distribution.

Using these values as thresholds, pages are segmented adaptively to the layout of pages. Figure 5.21c is the result of removing unnecessary edges. By further removing edges with open terminals, i.e., a terminal of an edge without connection to others, we can obtain the segmentation as shown in Fig. 5.21d.

Comparison of Methods

We have described representative methods of page segmentation for binary images with nonoverlapping layout. For this category of layout, many more methods have

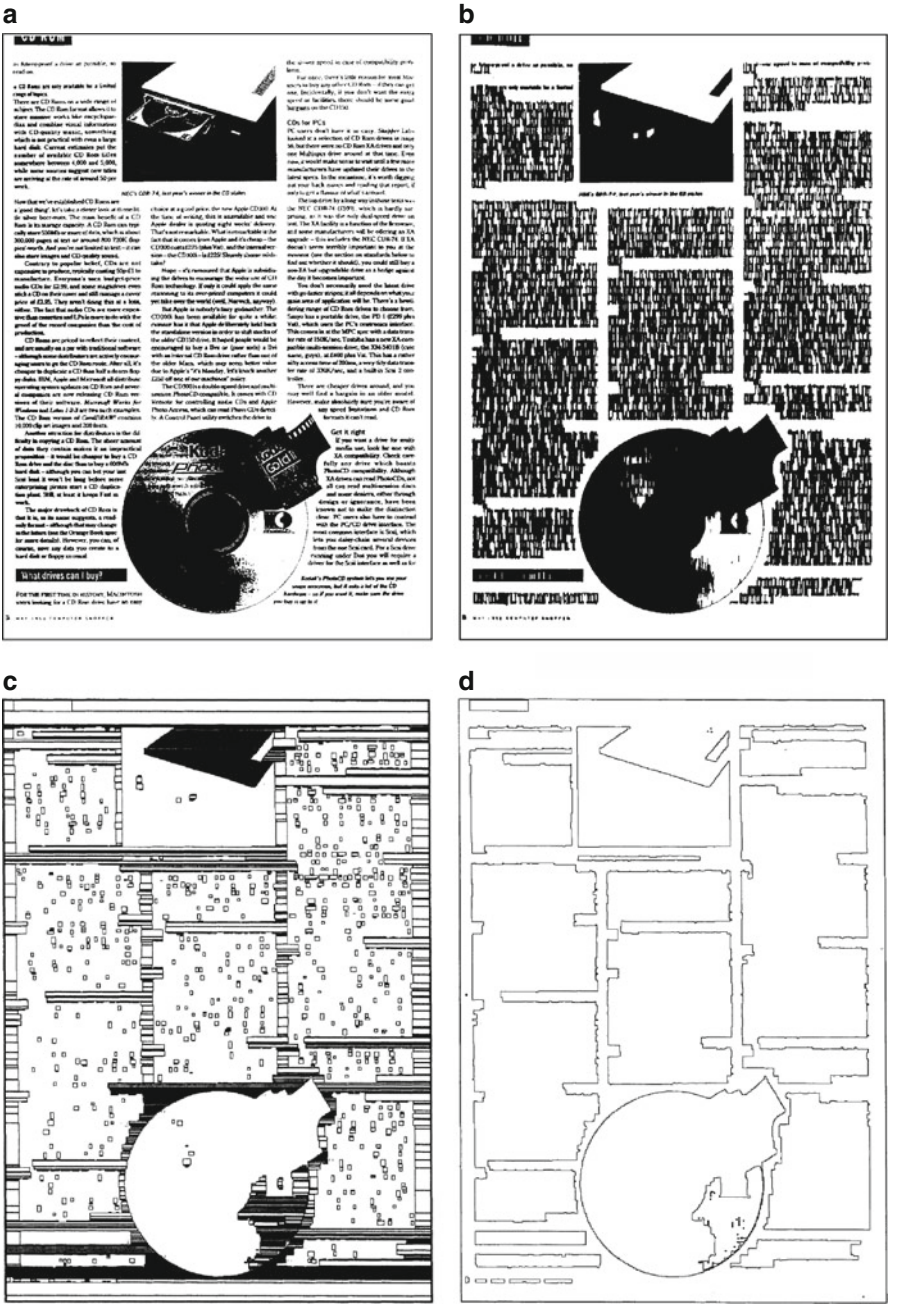

Fig. 5.20 Page segmentation based on white tiles (**a**) Original image, (**b**) smeared image, (**c**) white tiles (**d**) segmentation result (Reprinted from Figs. 1, 7, 9, 13 of [6])

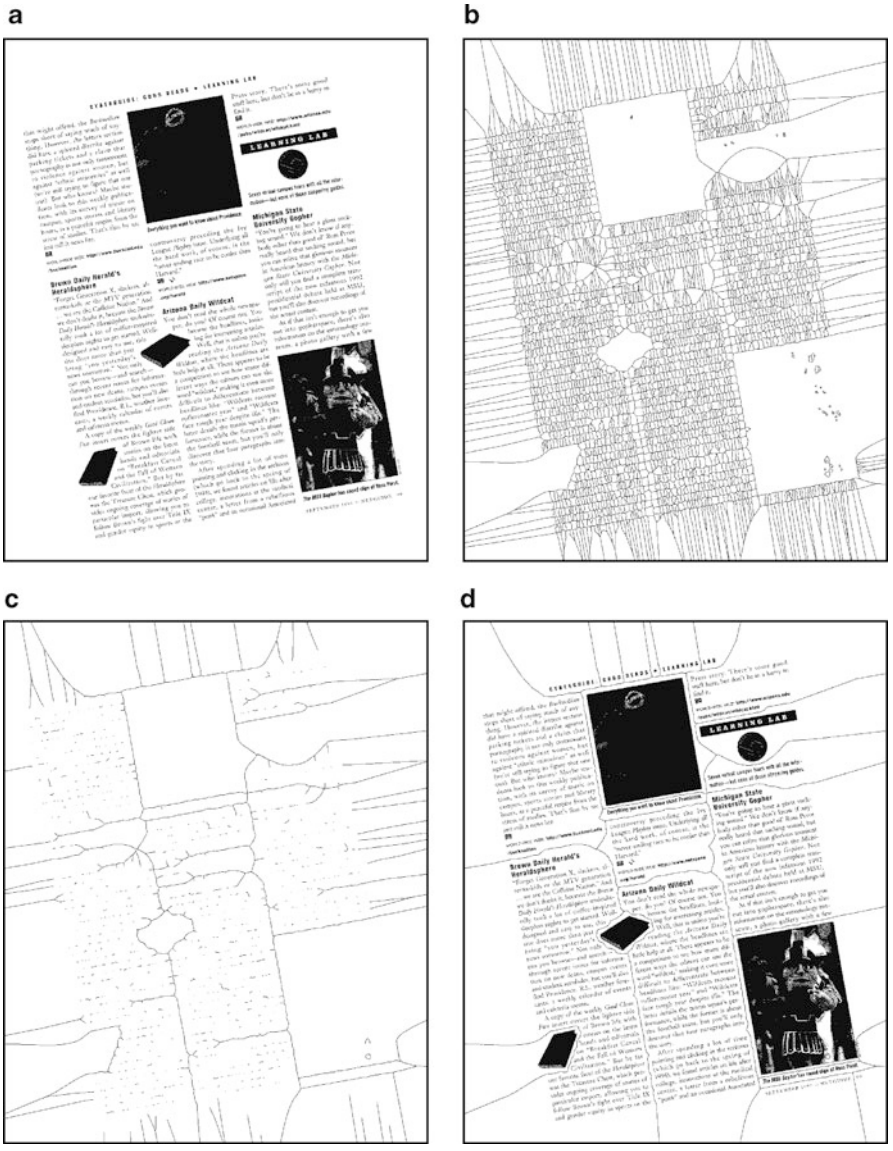

Fig. 5.21 Area Voronoi diagram and its use for page segmentation (Reprinted from Fig. 4 of [7])

been proposed. Thus readers may be interested in finding which the best algorithm is among them.

In order to answer to this question, several research activities have been performed. A representative is the "page segmentation contest" starting as the newspaper page segmentation contest in 2001 and continued until 2009 as a

competition in biannual ICDAR conferences (e.g., [20]). The details can be found in ▸Chap. 29 (Datasets and Annotations for Document Analysis and Recognition). Another activity is the publication of benchmarking papers [21, 40]. An important conclusion of these papers is that the best algorithm must be determined based on the constraints on pages to be analyzed. For example, if the layout is rectangular and there is no skew, it is not always a good choice to use more general methods such as docstrum and the Voronoi in terms of the accuracy. Another important note is that among representative methods including docstrum and the Voronoi, there may be no statistically significant difference even when their accuracies differ [22]. Pages hard to segment by one method are also hard by other methods. In other words, the variation caused by pages is much larger than that by algorithms.

As for computation time and memory requirement, we should be careful to the following points. In methods that work directly on the image such as projection-based and smearing-based methods, the size of the image determines both the computation time and memory. Thus it is better to work not on the original image with high resolution but on its downsampled version. For methods based on larger primitives and representations such as connected component-based and background analysis methods, once the primitives and representations are extracted, computation time and memory consumption are independent from the resolution. Especially, for methods based on computational geometry such as MST and Voronoi, a lot of sophisticated algorithms are available for computing representations. Thus these methods have an advantage with respect to computation time and memory.

Analysis of Gray-Level and Color Pages

It is sometimes inappropriate to capture documents as black-and-white images. There are two cases. One is the case that documents are inherently printed in gray-level or color. The other case is that even for documents printed in black and white, it is sometimes more appropriate to capture them as gray-level or color images due to the degradation. The latter is typical for historical documents.

Strategies for the analysis can be broadly divided into two categories: whether or not the foreground is separated from the background by preprocessing. If we can apply foreground separation, the problem of page segmentation can be transformed into that for black-and-white pages [23], and thus methods described above can be applied. Here we briefly describe the preprocessing for the purpose of page segmentation.

For gray-level images, a simple strategy is to apply a thresholding. For color images, the equivalent is clustering to obtain colors that represent foreground. Note that they are not just an application of simple thresholding or color clustering, because gray-level and color printing is based on fine dots or *dither*. If we scan such documents at high resolution, these dots are directly obtained as shown in Fig. 5.22 and make the processing difficult. Lower resolution images hardly have this problem as in Fig. 5.22. However, simple downsampling by averaging in local areas blurs

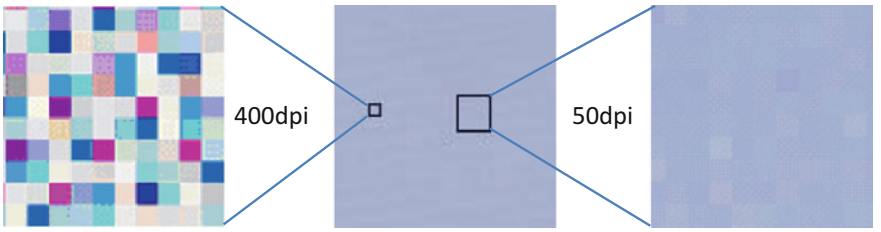

Fig. 5.22 Effect of resolution change (Reprinted from Fig. 9 of [24])

edges of the foreground. To cope with this problem, Hase et al. proposed a method of *selective* averaging images in local areas [24]. The value of each pixel is determined by taking into account its surrounding local area whose center is the pixel. They assume that a local area contains at most two different regions. If the local area contains only a single region, the value of the current pixel at the center is set to the average of pixel values in the local area. Otherwise a simple discriminant analysis is applied to determine two regions, and the average of the region the current pixel belongs to is set to the current pixel.

If readers are interested in thresholding or color clustering, see ▸Chap. 4 (Imaging Techniques in Document Analysis Processes).

For the purpose of page segmentation, it is better not only to consider pixel values, but also features of page components. Perroud et al. have proposed a method that takes into account spatial information in addition to color information [25]. They employ a four-dimensional histogram, i.e., RGB and the Y-coordinate of a pixel, for the clustering. This enables us to take into account the vertical proximity to form a cluster of pixels.

In addition to the thresholding, we have another strategy for page segmentation of gray-level images. In gray-level images, text regions are characterized by strong edges from characters. Yuan and Tan have proposed a method that employs edge detection [26]. In their method text regions are characterized by those with horizontal edges. By grouping neighboring horizontal edges, text regions are extracted.

In case that the foreground-background separation is not easy to achieve by the above methods, such as the case of overlapping layout, it is necessary to solve this problem by taking into consideration features of page components at a signal level. Details are described in the next section.

Analysis of Pages with Overlapping Layout

As we discussed in section "Text-Block Level," overlapping layout of type shown in Fig. 5.3d is far different from nonoverlapping layout in the sense that there is no clear distinction between foreground and background; as the case of text on a picture, one

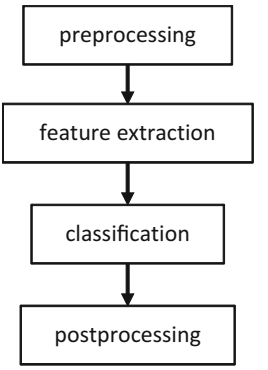

Fig. 5.23 Analysis of pages with overlapping layout

page component can be the background of another page component. Thus the simple separation between foreground and background is meaningless. In other words, page segmentation should be done with classification of page components. Note that some page classification methods assume that page components have already been extracted. Methods of page classification including this type are described in ▸Chap. 7 (Page Similarity and Classification). In this chapter, we focus on methods that segment page components simultaneously with their classification.

Most of the methods for this layout are based on signal properties of page components. To be precise, features and strategies based on texture analysis technologies enable us page segmentation of this class of layout. Figure 5.23 shows a general processing flow. After the preprocessing such as noise reduction, texture features are extracted from an input image, which is mostly given as gray-level or color. As a unit for feature extraction, it is common to consider each pixel of the image. A feature vector is associated with each pixel and employed to classify it. Methods of classification can be divided into two types – unsupervised and supervised. Classes of page components for classification depend on methods. The most basic classes are text and non-text. Some methods employ more detailed classes such as pictures, line drawings, and handwriting. In general classification results tend to be noisy due to limited amount of "local" information used at the classification. The task of postprocessing is, therefore, to make the results more reliable by using global constraints or contexts.

Table 5.2 shows representative methods of this category. Methods can be characterized by the items in the table: classes of page components to be extracted, features used for the classification, methods of classification, and postprocessing. In the following, details of each method are described.

Let us start with one of the pioneering work in this category, i.e., the method proposed by Jain and Bhattacharjee [27]. Their method is to classify each pixel of the input image into either text or non-text. As the feature for each pixel, they employ multichannel Gabor filters. By using filters of different scales and orientations (two scales and four orientations), the method can process images with various layouts and skew angles.

Table 5.2 Representative methods for overlapping layout

Method	Classes	Feature	Classification method	Postprocessing
[27]	Text, non-text	Multichannel Gabor filters	***Unsupervised:*** CLUSTER (3 clusters) ***Supervised:*** k-*NN* ($k = 7$)	Heuristics (deletion of small and thin CCs, etc.)
[28]	Text + graphics Picture Background	Learned masks (16 masks of size 7×7)	***Supervised:*** Neural network	Morphology
[29]	Text Picture Graphics Background	Wavelet packets	***Supervised:*** Local decision by neural network + decision integration (Coarse-to-fine)	Morphology (closing)
[30]	Text Graphics Background	M-band wavelets	***Unsupervised:*** k-means	
[31]	Text Picture Background	Matched wavelets	***Supervised:*** Fisher	MRF
[32]	Text Picture Background	Haar wavelet	***Supervised:*** Multiscale Bayesian	
[33]	Machine-printed Handwriting Picture Background		***Supervised:*** Iterated classification	Implemented as iterated classification

In their method, the feature vector $\boldsymbol{e}$ of a pixel at (x, y) is given by

$$\boldsymbol{e}(x, y) = (e_1, \ldots, e_n)$$

$$e_k(x, y) = \frac{1}{M^2} \sum_{(a,b) \in W_{xy}} |\varphi(r_k(a, b))|, k = 1, \ldots, n$$

where W_{xy} is a window of size $M \times M$ centered at the pixel (x, y), $r_k(x, y)$ is the kth filtered image as described below, and φ corresponds to a function for thresholding:

$$\varphi(t) = \tan h(\alpha t)$$

where α is a parameter and typically set to 0.25. As filters to obtain filtered images, they propose to select appropriate ones from the following bank of Gabor filters with various radial frequencies with four orientations $\theta = 0°, 45°, 90°, 135°$. For the processing of newspaper images of size 256×512, they employed two radial

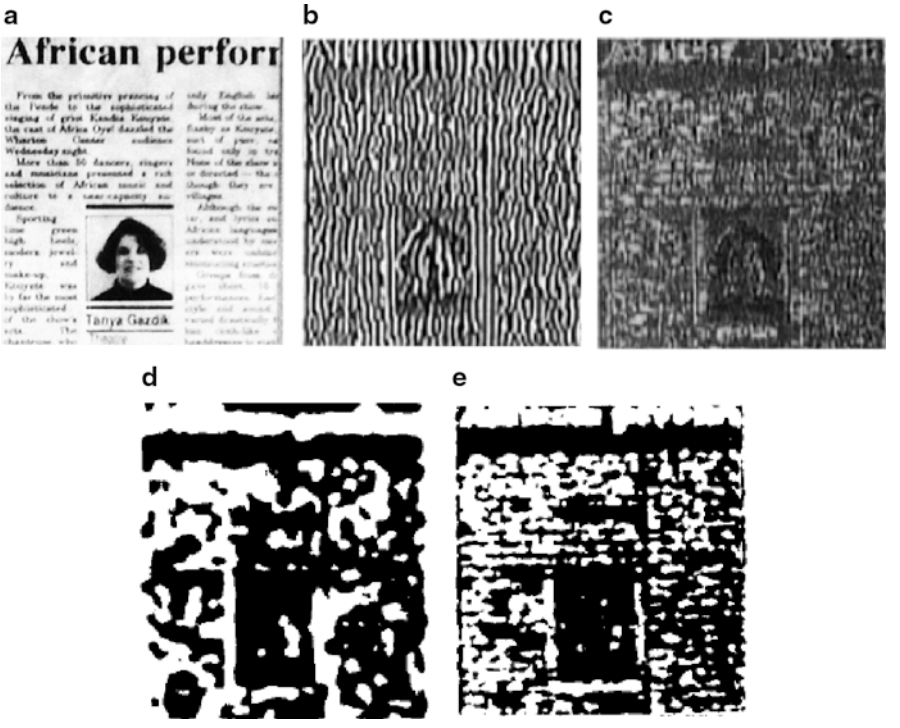

Fig. 5.24 Examples of filtered images and features. (**a**) Original image, (**b**) a filtered image with the radial frequency $32\sqrt{2}$ and orientation $0°$, (**c**) radial frequency $64\sqrt{2}$ and orientation $0°$, (**d**) the feature obtained from (**b**), and (**e**) the feature from (**c**) (Reprinted from Figs. 2–4 of [27])

frequencies $32\sqrt{2}, 64\sqrt{2}$. Examples of filtered images and elements of the feature vector are shown in Fig. 5.24.

Then the feature vector is classified by two methods. One is unsupervised based on the CLUSTER algorithm. Note that the number of clusters is not two but three: text, non-text, and their border. The other method is supervised. To be precise, the *k-NN* with $k = 7$ is employed for two-class classification with 4,000 training samples (pixels). As the postprocessing, they utilize some heuristics for cleaning up the results. Figure 5.25 shows an example of processing results.

The above method is based on general features that require a higher computational load. In other words, more efficient processing is possible if features are tuned for a specific purpose. To achieve this, Jain and Zhong have proposed a method using a learning capability [28]. Features are obtained by a feed-forward artificial neural network learned with the back-propagation algorithm. They first use three classes, text or graphics, picture, and background. Then after the classification, the class text

Fig. 5.25 An example of segmentation results by the method. (**a**) Input image, (**b**) extracted text region shown in *gray color*, and (**c**) text segments after postprocessing (Reprinted from Fig. 7 of [27])

or graphics is further divided based on different features. As the postprocessing, they employ noise removal and morphological operators to enclose text regions.

Etemad et al. have also proposed a method which employs "tuned" texture features [29]. Specifically, they used wavelet packets that can be adaptively designed based on learning data. One of the advantages of their method is that it is much more computationally efficient than Gabor filter-based methods, thanks to the fast algorithms for computing wavelet packets. Another advantage is in their decision process. Severe overlaps of page components prevent us from making hard decision based only on local information. This problem is solved by using "soft" (weighted) decision, followed by decision integration based on context information, i.e., classification results of neighboring pixels. As the postprocessing they also employ morphological operators to eliminate noise.

In addition to the above basic texture features (Gabor filters and wavelet packets), there are several approaches that employ more advanced features. Acharyya and Kundu have proposed to use M-band wavelet features ($M > 2$) for better analysis of high-frequency signals as compared to the normal wavelet with $M = 2$ [30]. Their method employs a simple k-means clustering for classification. Even with such a simple method, it has been experimentally shown that their method works better than previously proposed methods such as [27] and [29]. The method allows us results with less noise. A different way of adapting features is proposed by Kumar et al. [31]. Their method uses matched wavelets to implement a trainable segmentation together with the Fisher classifier for classification. A postprocessing method based on Markov Random Fields (MRF) is employed for improving the quality of segmentation results.

In general, most methods in this category including those mentioned above can be characterized by the following concepts: multiscale, learning, and context representation for postprocessing. Each method has their own way to deal with these concepts, but most of them lack a unified theoretical framework for these concepts. For example, some methods employ MRF for postprocessing, but it is independent of the classification and the multiscale feature representation. Needless to say, it is better to have a unified framework to achieve a better performance obtained by global optimization than a heuristic combination of these concepts.

Cheng and Bouman [32] have proposed a method to achieve this goal. Their method is based on a multiscale Bayesian framework in which a Markov chain in scale is utilized for modeling the texture features and the contextual dependencies for the image. The algorithm called trainable sequential maximum a posteriori (TSMAP), which is used to calculate the MAP estimates sequentially along with the multiscales, is capable of improving the performance with increasing the number of samples for learning as shown in Fig. 5.26.

There is another method that is worth being mentioned. All of the above methods, if they are supervised, require learning process which is in general computationally expensive. When the number of training samples becomes very large, the learning process is prohibitive due to the computational cost. This problem can be solved by using a simpler supervised method with less learning cost. For the classifier based on the nearest neighbor (NN) scheme, for example, the learning is just to record the samples with labels. The cost of matching can also be reduced by using approximate NN search. The method proposed by An et al. [33] embodies this concept. In their method, each pixel is characterized by a feature vector with 77 dimensions including a current result of classification for the pixel of interest and its surrounding pixels with a fixed radius (5–7 pixels). An interesting strategy of their method is that they do not have a single postprocessing step but with many steps of postprocessing: the method consists of cascaded classifiers, and each classifier is trained based on the result of a previous classifier. With such a multistage training and testing, error rates can be reduced even for a more complex task of distinguishing four classes: machine printed, handwriting, picture, and background.

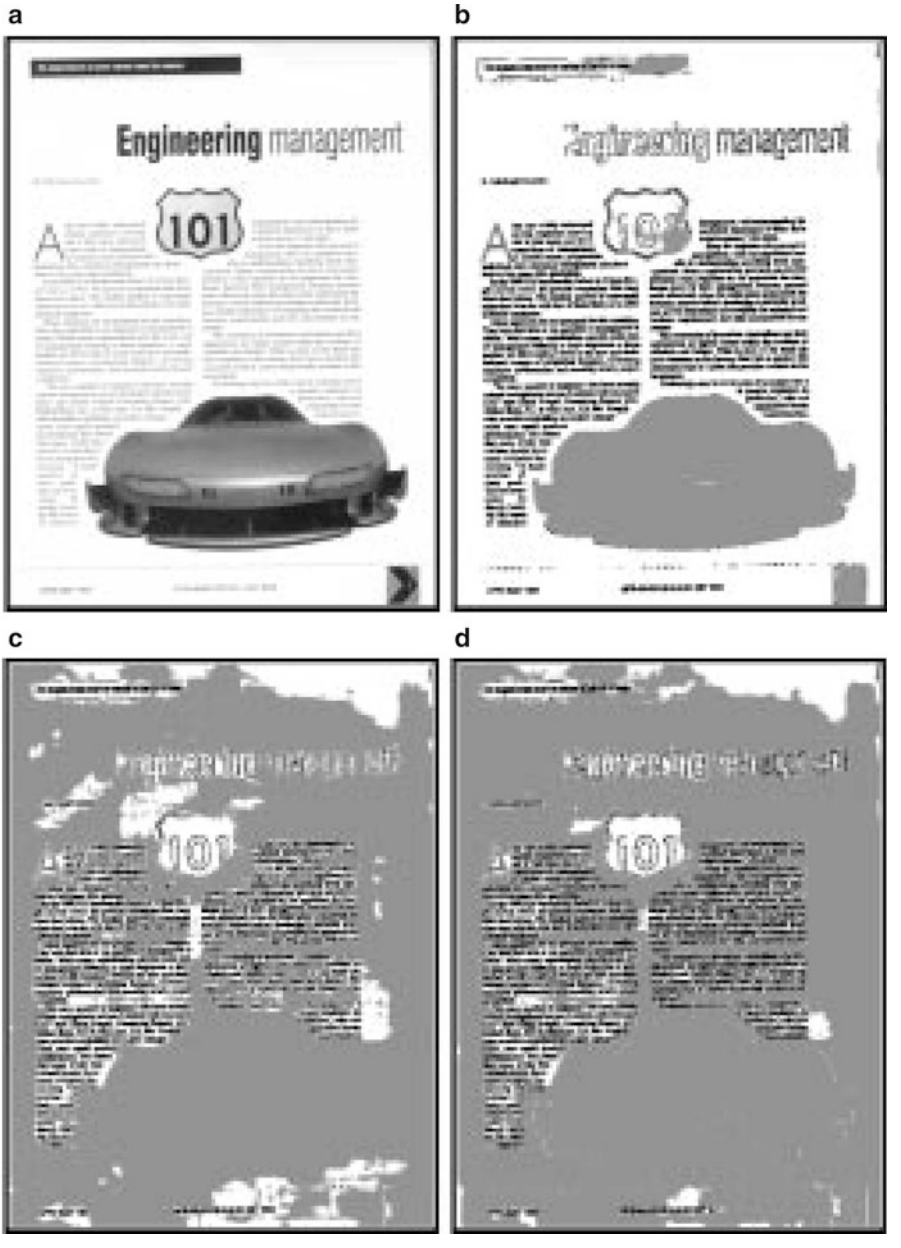

Fig. 5.26 Effect of training by the TSMAP algorithm. (**a**) Input image; (**b**) segmentation results when trained on 20 images where *black*, *gray*, and *white pixels* indicate text, pictures, and background, respectively; (**c**) results with 10 training images; and (**d**) results with 5 training images (Reprinted from Fig. 15 of [32])

Analysis of Handwritten Pages

The final topic in this chapter is the analysis of handwritten pages. The task here is to distinguish handwriting from machine-printed part in a page image. As we have already seen that texture-based methods such as [33] offer a means for the distinction, many methods for this task are quite similar to those in the previous section. In this section, let us briefly describe an example, a method proposed by Zheng et al. [34].

In order to reduce the computational cost, the method employs connected components as units of processing. Thus the goal here is to classify connected components into either machine printed or handwriting. In order to achieve this goal for noisy documents, Zheng et al. define their task as classification into either machine printed, handwriting, or noise. Features used for classification are similar to those in texture-based page segmentation such as Gabor filters and bi-level co-occurrence. After the features are extracted from each connected component, the next step is classification. In the method, the Fisher classifier is applied after feature selection by principal component analysis. Since the classification is error prone due to the limited amount of information from each connected component, the postprocessing is necessary to clean the results by taking into account the contextual information. Zhang et al. employ a method based on Markov random field – another resemblance to methods in the previous section.

Conclusion

In this chapter, we have described various methods of page segmentation focusing mainly on segmenting machine-printed page components such as text blocks, graphics, tables, and pictures. The research activities are in mature for nonoverlapping layout such as rectangular, Manhattan, and non-Manhattan. If the restriction on the layout is stronger, more stable methods with less computational costs are available. For the analysis of overlapping layout, methods are still under development. The difficulty of the analysis is similar to that of camera-based document analysis for unrestricted environments. We have seen that pixel-level classification by using texture features is a promising way to achieve the task.

The history of development of page segmentation methods is a series of challenges to remove the assumptions/limitations on page layout. A possible future trend in the research is to extend this challenge for adaptive and learnable page segmentation. Since most of the printed documents are prepared digitally, fully automated learning methods would be developed if appropriate degradation models of images are available.

Cross-References

▸Analysis of the Logical Layout of Documents
▸Datasets and Annotations for Document Analysis and Recognition

▸Imaging Techniques in Document Analysis Processes
▸Page Similarity and Classification

References

1. Wong KY, Casey RG, Wahl FM (1982) Document analysis system. IBM J Res Dev 26(6): 647–656
2. Ittner DJ, Baird HS (1993) Language-free layout analysis. In: Proceedings of the second ICDAR, Tsukuba, pp 336–340
3. O'Gorman L (1993) The document spectrum for page layout analysis. IEEE Trans PAMI 15(11):1162–1172
4. Kise K, Iwata M, Matsumoto K, Dengel A (1998) A computational geometric approach to text-line extraction from binary document images. In: Proceedings of the 3rd DAS, Nagano, pp 346–355
5. Baird HS (1992) Anatomy of a versatile page reader. Proc IEEE 80(7):1059–1065
6. Antonacopoulos A (1998) Page segmentation using the description of the background. Comput Vis Image Underst 70(3):350–369
7. Kise K, Sato A, Iwata M (1998) Segmentation of page images using the area Voronoi diagram. Comput Vis Image Underst 70(3):370–382
8. Nagy G, Seth S (1984) Hierarchical representation of optically scanned documents. In: Proceedings of the 7th ICPR, Montreal, pp 347–349
9. Krishnamoorthy M, Nagy G, Seth S, Viswanathan M (1993) Syntactic segmentation and labeling of digitized pages from technical journals. IEEE Trans PAMI 15(7):737–747
10. Pavlidis T, Zhou J (1992) Page segmentation and classification. CVGIP: Graph Models Image Process 54(6):484–496
11. Srihari SN, Govindaraju V (1989) Analysis of textual images using the Hough transform. Mach Vis Appl 2:141–153
12. Fletcher LA, Kasturi R (1988) A robust algorithm for text string separation from mixed text/graphics images. IEEE Trans PAMI 10(6):910–918
13. Bloomberg DS (1996) Textured reduction for document image analysis. In: Proceedings of the IS&T/SPIE EI'96, conference 2660: document recognition III, San Jose
14. Bukhari SS, Shafait F, Breuel TM (2011) Improved document image segmentation algorithm using multiresolution morphology. In: SPIE document recognition and retrieval XVIII, DRR'11, San Jose
15. Dias AP (1996) Minimum spanning trees for text segmentation. In: Proceedings of the 5th annual symposium on document analysis and information retrieval, Las Vegas
16. Simon A, Pret J-C, Johnson AP (1997) A fast algorithm for bottom-up document layout analysis. IEEE Trans PAMI 19(3):273–277
17. Breuel TM (2002) Two geometric algorithms for layout analysis. In: Proceedings of the DAS2002, Princeton, pp 188–199
18. Agrawal M, Doermann D (2010) Context-aware and content-based dynamic Voronoi page segmentation. In: Proceedings of the 9th DAS, Boston, pp 73–80
19. Yin P-Y (2001) Skew detection and block classification of printed documents. Image Vis Comput 19(8):567–579
20. Antonacopoulos A, Pletschacher S, Bridson D, Papadopoulos C (2009) ICDAR 2009 page segmentation competition. In: Proceedings of the 10th ICDAR, Barcelona, pp 1370–1374
21. Shafait F, Keysers D, Breuel TM (2008) Performance evaluation and benchmarking of six-page segmentation algorithms. IEEE Trans PAMI 30(6):941–954
22. Mao S, Kanungo T (2001) Empirical performance evaluation methodology and its application to page segmentation algorithms. IEEE Trans PAMI 23(3):242–256
23. Strouthopoulos C, Papamarkos N, Atsalakis AE (2002) Text extraction in complex color documents. Pattern Recognit 35:1743–1758

24. Hase H, Yoneda M, Tokai S, Kato J, Suen CY (2004) Color segmentation for text extraction. IJDAR 6:271–284
25. Perroud T, Sobottka K, Bunke H, Hall L (2001) Text extraction from color documents – clustering approaches in three and four dimensions. In: Proceedings of the 6th ICDAR, Seattle, pp 937–941
26. Yuan Q, Tan CL (2001) Text extraction from gray scale document images using edge information. In: Proceedings of the 6th ICDAR, Seattle, pp 302–306
27. Jain AK, Bhattacharjee S (1992) Text segmentation using Gabor filters for automatic document processing. Mach Vis Appl 5:169–184
28. Jain AK, Zhong Y (1996) Page segmentation using texture analysis. Pattern Recognit 29(5):743–770
29. Etemad K, Doermann D, Chellappa R (1997) Multiscale segmentation of unstructured document pages using soft decision integration. IEEE Trans PAMI 19(1):92–97
30. Acharyya M, Kundu MK (2002) Document image segmentation using wavelet scale-space features. IEEE Trans Circuits Syst Video Technol 12(12):1117–1127
31. Kumar S, Gupta R, Khanna N, Chaudhury S, Joshi SD (2007) Text extraction and document image segmentation using matched wavelets and MRF model. IEEE Trans Image Process 16(8):2117–2128
32. Cheng H, Bouman CA (2001) Multiscale Bayesian segmentation using a trainable context model. IEEE Trans Image Process 10(4):511–525
33. An C, Baird HS, Xiu P (2007) Iterated document content classification. In: Proceedings of the 9th ICDAR, Curitiba, pp 252–256
34. Zheng Y, Li H, Doermann D (2004) Machine printed text and handwriting identification in noisy document images. IEEE Trans PAMI 26(3):337–353
35. O'Gorman L, Kasturi R (1995) Document image analysis. IEEE Computer Society, Los Alamitos
36. Dori D, Doermann D, Shin C, Haralick R, Phillips I, Buchman M, Ross D (1997) The representation of document structure: a generic object-process analysis. In: Bunke H, Wang PSP (eds) Handbook of character recognition and document image analysis. World Scientific, Singapore, pp 421–456
37. Jain AK, Yu B (1998) Document representation and its application to page decomposition. IEEE Trans PAMI 20(3):294–308
38. Okun O, Pietikäinen M (1999) A survery of texture-based methods for document layout analysis. In: Pietikäinen M (ed) Texture analysis in machine vision. Series in machine perception and artificial intelligence, vol 40. World Scientific, Singapore
39. Nagy G (2000) Twenty years of document image analysis in PAMI. IEEE Trans PAMI 22(1):38–62
40. Mao S, Rosenfeld A, Kanungo T (2003) Document structure analysis algorithms: a literature survey. In: Proceedings of the document recognition and retrieval X, Santa Clara, pp 197–207
41. Namboodiri AM, Jain A (2007) Document structure and layout analysis. In: Chaudhuri BB (ed) Digital document processing: major directions and recent advances. Springer, London, pp 29–48. ISBN:978-1-84628-501-1
42. Cattoni R, Coianz T, Messelodi S, Modena CM (1998) Geometric layout analysis techniques for document image understanding: a review. Technical report TR9703-09, ITC-irst
43. Normand N, Viard-Gaudina C (1995) A background based adaptive page segmentation algorithm. In: Proceedings of the 3rd ICDAR, Montreal, pp 138–141
44. Kise K, Yanagida O, Takamatsu S (1996) Page segmentation based on thinning of background. In: Proceedings of the 13th ICPR, Vienna, pp 788–792

Further Reading

In addition to papers in conferences such as ICDAR and DAS, it is recommended to read some chapters in handbooks [35, 36] and survey papers [37–41] if readers are interested in finding more

methods of page segmentation. Although most of them are published before 2000, major activities of page segmentation are covered by them. The most comprehensive list of methods before 2000 can be found in [42].

The following are some additional comments we do not touch in the above.

In section "Smearing Based Methods," we introduced a smearing-based method using the mathematical morphology. A limitation of this method is that it can only deal with separation of text from halftones. A recent progress to solve this problem has been proposed by Bukhari et al. [14].

In section "Background Analysis Methods," we mainly focused on methods based on computational geometry, since they allow us efficient processing. However, there is another type of methods based on binary digital image processing. It is easy to understand that a rotation invariant representation of background can be obtained by using maximal empty "circles" [43]. This representation can be obtained by using the algorithm of skeletonization. A disadvantage of using skeletons is that they do not keep the information of connectivity among regions represented by empty circles. In the field of digital image processing, algorithms of thinning have been proposed to solve this problem. An example of using thinning for page segmentation can be found in [44].

Methods for improving the Voronoi-based segmentation have also been proposed. One of the serious problems of the Voronoi-based method is that the threshold for deleting unnecessary Voronoi edges is globally determined. Thus if a page contains page components with different sizes such as a bigger title with a smaller body text, the global threshold may cause the problem of over-segmentation for a nondominant text such as titles. This problem has been addressed by Agrawal and Doermann [18]. In this method, the deletion of Voronoi edges works adaptively depending on the "contexts," i.e., the surrounding CCs, with the help of new features and decision rules.