Layout Analysis of Book Pages

Chad Oliver, Richard Green

***Abstract*—A method is proposed for analysing the geometric and logical structure of pages in a typical book. A Gaussian blur combined with thresholding is used to form connected components which nominally represent words. A bottom-up nearest-neighbour approach is used to find textual lines, and a manually-defined line length parameter is used to remove marginal noise and find the page frame. A state machine is used to group lines and label them according to function. The proposed method is able to correctly segment and label 99.83% of all targeted features in a set of 54 sample pages. Of the four errors encountered in the sample pages, two are instances where paragraphs have been split in half, and the remaining two errors were caused by words on consecutive lines being merged into a single connected component.**

***Index Terms*—Structure Detection, Geometric Layout, Logical Layout, Skew Detection, OCR Pre-processing.**

I. Introduction

There are many old books which are out of print and hard to find. If they could be converted to a digital format, many more people would be able to enjoy them. However, Optical Character Recognition programs available today produce text which suffers from a set of common errors. Headers and page numbers are not recognised as such; words which are split across two lines in the physical text retain their hypens in the digital text; spurious characters are found in images; section titles and captions lose their formatting and meaning.

In order to detect and correct these errors, there needs to be a method which can analyse a page image and label segments of text according to their function. In this paper we present a candidate algorithm for this purpose. The algorithm is capable of detecting lines, paragraphs, section titles, chapter titles, figures (subdivided into images and captions), and headers/footers.

II. Background

A. Geometric Layout Analysis

Geometric Layout Analysis is concerned with segmenting an image into areas of text and non-text, as well as splitting text into columns [1].

Top-down approaches work by recursively splitting an image into smaller sections until some criterion is met. In contrast, bottom-up algorithms combine fundamental units into words, lines, and segment blocks. Hybrid algorithms combine both approaches [2].

1) Recursive X-Y Cuts (RXYC): One of the oldest and most influential algorithms is the Recursive X-Y Cuts (RXYC) algorithm [3]. The page is represented by an X-Y tree of segments, and at each step of the process a segment is tested and potentially split into two or more smaller segments. Segments are split where the horizontal or vertical projection has a peak whiteness above a specified threshold.

The RXYC algorithm is only suited to pages where the layout is Manhattan, that is, all columns can be isolated by a set of horizontal and vertical line-segments drawn through white space. Most books follow this layout, but the RXYC algorithm will give incorrect results if, for example, an image is embedded in the text. Furthermore, the RXYC algorithm is also sensitive to page skew.

2) Shape-Directed Covers: The Shape-Directed Covers (SDC) algorithm [4] segments Manhattan pages by first finding the set of maximal empty rectangles, *i.e.* the largest rectangles which contain only white pixels. The minimum rectangle size is set to the smallest line height, to prevent paragraphs from being segmented into individual lines.

As with the RXYC algorithm, the SDC algorithm is handicapped by its inability to accurately segment non-Manhattan layouts.

3) Docstrum: The Docstrum algorithm [5] uses a unique bottom-up approach based on k-nearest-neighbor clustering of connected components. The Docstrum algorithm has many similarities to the one presented in this page, along with significant differences.

It is assumed that each connected component corresponds to a single letter. For each component, k nearest neighbors are found by measuring the distance between each component's centroid. For $k = 5$, a character might make two or three pairings within a word and across word boundaries within the same line, as well as pairings with characters on upper and lower lines.

A docstrum plot (Figure 1) shows the distance and angle of all k-nearest-neighbour components. The page skew is determined by finding the most common angle after smoothing the angle histogram.

One significant advantage of this technique is that inter-line pairings are clearly distinguishable from within-line pairings. This allows a simple algorithm for grouping components into lines: two components are considered to be on the same line if they are transitively connected by within-line nearest-neighbor pairings.

Lines are determined to be in the same structural block if they "are approximately parallel, close in perpendicular distance, and they either overlap to some specified degree or are separated by only a small distance in parallel distance" [5].

The Docstrum method is a capable and accurate page segmentation algorithm which is largely invariant under skew and which generates, as a side effect, the set of all textual lines in the document. In the context of simple page layouts

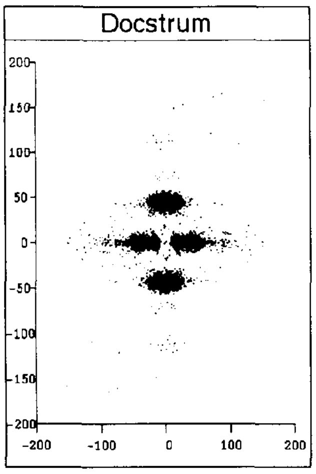


Fig. 1. Docstrum plot for a page of text. Each point represents the distance and angle between a nearest-neighbor pair as the distance and angle from the origin at the center of the plot.

as commonly used in books, there does not seem to be any reason to criticise this algorithm.

4) Voronoi Diagrams: The Voronoi Diagrams algorithm [6] is a bottom-up algorithm which segments a page by examining the relative areas of Voronoi regions and the minimum distance between connected components.

Let $p_1, \ldots, p_n$ be a set of points or *generators* on a plane, and let $d(q, r)$ be the Euclidean distance between points q and r. A *Voronoi region* of a point p_i is the region given by

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

and a *Voronoi edge* is an edge between two regions.

The *point Voronoi diagram* is the set of all *Voronoi regions* generated by the points $p_1, \ldots, p_n$. This can be generalised to the *area Voronoi diagram*, where all points of a connected component are considered to be part of the same generator.

The Voronoi Diagrams algorithm assumes that the majority of letters on the page are represented by a single connected component. The problem of page segmentation therefore becomes the problem of selecting the Voronoi edges which represent segment boundaries. An edge is considered to be significant if the closest points on the two generator components are separated by a minimum distance, or if the ratio of the areas of the two generator components is above a certain value.

A modified version of the algorithm [7] can be used to generate rectangular segments.

In contrast to most other algorithms, the Voronoi Diagrams algorithm is very good at segmenting non-Manhattan layouts which have an arbitary skew. However, in order to determine some parameters it assumes that body text regions are dominant on the page. This is not a valid assumption for many non-fiction books, which may have full-page images. Furthermore, the Voronoi Diagrams algorithm is a pure segmentation algorithm; it does not provide skew estimation or line segments.

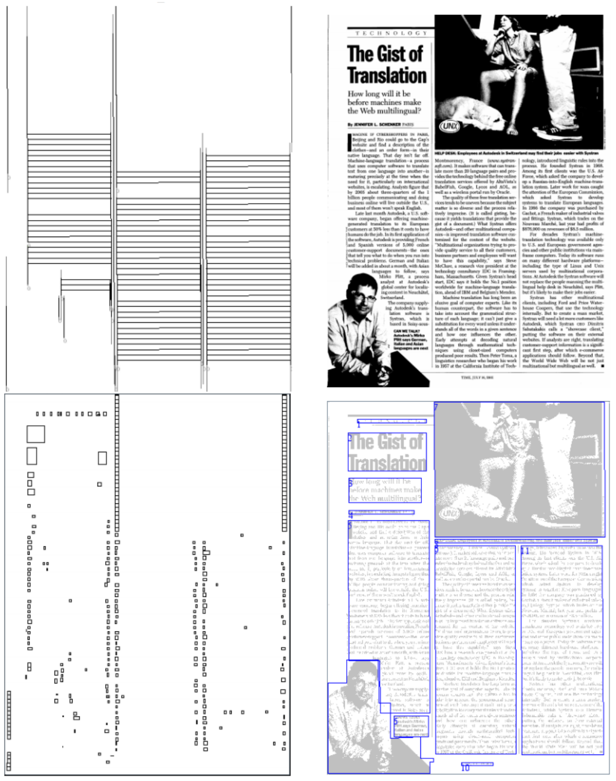


Fig. 2. The Tesseract Algorithm. Clockwise from top left: (a) Original image (b) Candidate tab-stop components (c) Fitted tab lines and traces connections (d) Final segment blocks.

In this sense it is optimised for segmenting layouts which are more complex than typical books.

5) Tesseract: The popular Tesseract OCR engine uses a hybrid tab-stop detection algorithm [8]. A tab-stop is a horizontal position which is used to align the beginning or end of a line of text. For example, a regular column of text will begin at one tab-stop on the left and, if it is right-justified, run to another tab-stop on the right. A secondary tab-stop may be used to align the first line in each paragraph, but this algorithm ignores those tab-stops.

The central process of the tab-stop algorithm involves finding connected components and then grouping candidate tab-stop components which begin or end at the same point. Groups with more than the minimum number of candidates are kept. Line tracking is used to associate each tab-stop component with the component at the other end of the line of text.

The average line length and the positions of tab-stops are combined to find an estimate for the column layout (which may be occluded by images or headings), and from this lines are grouped into segments.

This algorithm is similar to the one that will be presented in this paper. However, it is a pure segmentation algorithm and does not determine the skew and or group connected components into lines of text.

B. Page Skew Detection and Line Recovery

There are three common approaches to finding the location and orientation of text lines in a page. Projection profiles at various angles may be compared; the page skew corresponds to the projection angle which produces the most variance in the projection profile [9]. This approach is limited to small page skews of less than $\pm 10°$. The Hough Transform may be used to find the dominant geometric lines in the image; when tested this approach was found to give inaccurate or spurious angle estimates. Finally, nearest-neighbour clustering such as demonstrated in [5] may be used to find textual lines in a bottom-up manner without first determining page skew.

Recent work has focused on variants of nearest-neighbour clustering, as this general approach appears to offer greater flexibility and accuracy. In [10] Konya et. al. find the Euclidean minimum spanning tree of the centres of all connected components. Branches of the minimum spanning tree will tend to run along each line, and thus the skew angle can be determined from the histogram of minimum spanning tree edges. This method also examines the ratio of ascenders to descenders in order to determine skew angle over the full range of $\pm 180°$.

In [11] Meng et. al. propose a method for skew estimation by examining many different visual clues rather than just lines of text. This method is an accurate and general-purpose method, but its performace comes at the expense of greater computational requirements.

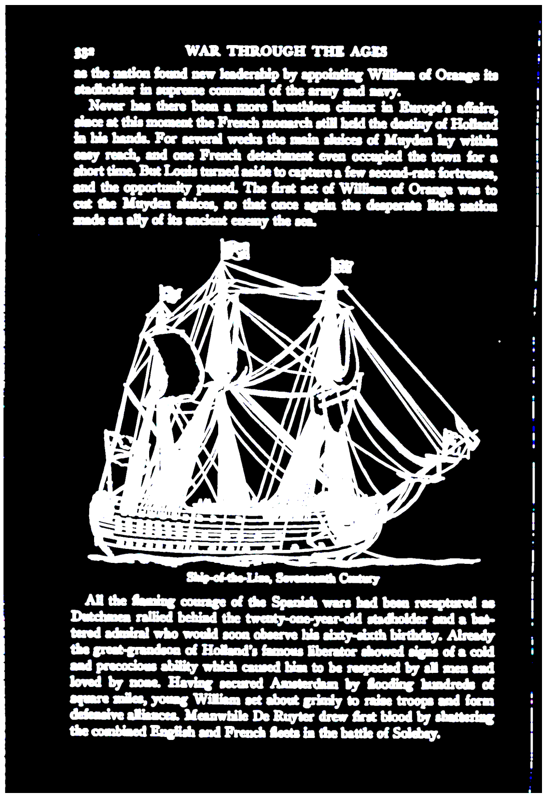
C. Logial Layout Analysis

Logical Layout Analysis is concerned with detecting the structure of the text, as perceived by the reader. This includes detecting headers, footers, captions, and chapters.

Derrien-Peden describes a method for determining the logical layout of a page using macro-typographical analysis [12]. This is based on the idea that in certain classes of documents, structure is indicated by changes in typeface size and styling, indentation, paragraph spacing etc. Thus the logical layout of the document may be determined without the need to interpret the semantic content. This approach is essentially similar to the one presented here, except that we characterise lines rather individual letters.

Meunier describes a method for determining logical structure by solely assessing different combinations of horizontal and vertical cuts across the document [13]. Vertical cuts are preferred to horizontal cuts, in order to correctly handle text columns. This method is restricted to Manhattan layouts and can only determine reading order. It is not capable of labeling segments according to function.

Krishnamoorthy et. al. propose that technical documents may be simultaneously segmented (geometric layout) and classified (logical layout) using the application of context-free grammars [14]. They demonstrate this using the compiler utilities *Yacc* and *Lex*. This is a novel and flexible approach which appears to produce accurate results, but it suffers from two problems. Firstly, it is only capable of processing Manhattan layouts; secondly, the subject of parser design is sufficiently different to document layout analysis as to make it unlikely that one person may be capable of implementing a system combining both fields.


332 WAR THROUGH THE AGES

as the nation found new leadership by appointing William of Orange its stadholder in supreme command of the army and navy.

Never has there been a more breathless climax in Europe's affairs, since at this moment the French monarch still held the destiny of Holland in his hands. For several weeks the main sluices of Muyden lay within easy reach, and one French detachment even occupied the town for a short time. But Louis turned aside to capture a few second-rate fortresses, and the opportunity passed. The first act of William of Orange was to cut the Muyden sluices, so that once again the desperate little nation made an ally of its ancient enemy the sea.

Ship-of-the-Line, Seventeenth Century

All the flaming courage of the Spanish wars had been recaptured as Dutchmen rallied behind the twenty-one-year-old stadholder and a battered admiral who would soon observe his sixty-sixth birthday. Already the great-grandson of Holland's famous liberator showed signs of a cold and precocious ability which caused him to be respected by all men and loved by none. Having secured Amsterdam by flooding hundreds of square miles, young William set about grimly to raise troops and form defensive alliances. Meanwhile De Ruyter drew first blood by shattering the combined English and French fleets in the battle of Solebay.

Fig. 3. The image after preprocessing. Each connected component nominally represents a words, although there are components along the margin which represent noise.

III. Method

A. Preprocessing and Word Extraction

The goal here is to form connected components which nominally represent words.

The process begins with a binary document image which typically contains salt-and-pepper noise along with marginal noise. A gaussian blur is applied with a user-specified kernel, and Otsu's Method [15] is used to return the image to binary form.

The value of the user-specified kernel determines how the dark pixels are joined together into connected components. A small kernel will result in each connected component typically representing a single character, while a large kernel will result in almost all dark pixels becomming merged into a single connected component. The kernel must be selected so that connected components normally correspond to a single word. This generally proves to be a simple task.

At this point, all connected components with an area less than A_{min} are discarded.

B. Grouping Words into Candidate Lines

Connected components (i.e. words) are grouped into lines by forming a graph of nearest-neighbour connections using the weighted Euclidean distance.

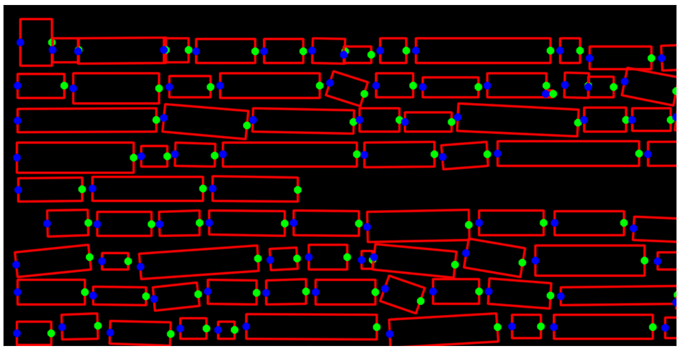


Fig. 4. An illustration of how each word's start- and end-points are found. Red boxes indicate minimum-area bounding boxes around connected components; blue dots represent start-points; green dots represent end-points.

Before finding nearest-neighbour connections, it is important to determine the start-point and end-point of each word. The distance between two words a and b is measured from the start-point of a to the end-point of b, or vice versa. This is done instead of using word centroids because it significantly reduces the number of inter-line connections.

These start-points and end-points should be chosen so that the end-point of one word is close to the start-point of the following word. A naive method is to simply choose the left-most and right-most points of each connected component. However, this can lead to points which are positioned at the end of a descender or ascender, especially if the page is skewed.

Instead, a minimum-area bounding box is fitted around each component, and the start- and end-points are chosen to be half-way up each of the vertical sides of the box. This process is illustrated in Figure 4.

A graph is now formed by connecting each component's end-point to the nearest start-point, as measured using a weighted distance function (see below). Similarly, each component's start-point is connected to the weighted nearest end-point.

The weighted distance function ensures that words on the same line are more likely to be connected than words on different lines. Let p and q be two points, and let p_x be the x-axis position of point p. Then the weighted distance function $d(p, q)$ is defined as:

$$d(p,q) = \sqrt{(p_x - q_x)^2 + 1.5(p_y - q_y)^2},$$

Once the graph has been formed, each disjoint section of the graph becomes a line. That is, lines are found in the process of creating the graph, not by analysing the graph.

C. Page Frame Detection

There are two steps in this process: the right and left tab-stops are found and used to remove marginal noise, then the page frame is found and used to derive metrics such as indentation.

The method described here depends on knowing the maximum textual line length. Based on this, the set of full-length lines is constructed by examining the minimum-area bounding box of each line. The left tab-stop is found by fitting a line through the left-most points of all full-length lines, and the right tab-stop is found in a similar manner. A connected component is considered to be marginal noise if its centroid does not fall between the left and right tab-stops. This method is simplistic and would be expected to miss marginal noise along the top and bottom of the page, but in reality it removes all marginal noise in the 54 test pages which were examined.

Once all marginal noise has been removed, a more accurate page frame is found. In the majority of cases it is sufficient to find the minimum bounding box of all remaining connected components, but this can fail if the content is very short, such as for the last page in a chapter. Therefore the page skew is detected by averaging the skew of all remaining textual lines, and a bounding box with the same skew is fitted around all connected components.

The left-indentation of each line is found by measuring the distance between the start of the line and the left edge of the page frame. The right-indentation is found in an analagous manner.

D. Finding the Logical Structure

The logical structure of a page is found by using a state machine to group lines into higher-level constructs such as paragraphs, figures, and sections titles. The state machine allows for a capable rule-set.

Images are distinguished from textual lines by the height of their bounding box. The threshold is arbitarily set to 300 pixels; any 'line' with a height greater than this is considered to be an image. Caption lines are detected as centered lines which immediately follow an image or another caption line. A figure consists of an image and zero or more caption lines.

Section titles are detected as centered lines which followed a paragraph.

Any line which was not an image, caption, or section title is detected to be part of a paragraph. Left- and right-indents are used to detect the beginning and end of paragraphs; otherwise lines are appended to the current paragraph.

IV. Results

The algorithm presented in this paper is able to accurately detect and label lines, paragraphs, section titles, chapter titles, figures (including images and captions), and headers/footers.

The algorithm was tested on a representative sample of 54 pages. This sample contained thirteen section titles, three chapter titles, four figures, and numerous lines, paragraphs, and headers/footers.

The algorithm's accuracy in detecting each feature type is summarised in Table I.

The two paragraph errors are instances where a paragraph has been split in half; it is not understood why this occured.

The two line errors both occur in a quote, where the text size and line spacing are reduced. These factors caused two words on consecutive lines to merge into a single connected

TABLE I
FEATURE DETECTION ACCURACY

Feature	Num. Instances	Num. Errors	Accuracy
Line	1916	2	99.89%
Paragraph	301	2	99.34%
Header/Footer	54	0	100%
Section Titles	13	0	100%
Chapter Titles	3	0	100%
Figures	4	0	100%
Total	2291	4	99.83%

component, which disrupted the graph creation process. One of the two pages with line errors is shown in Figure 6.

The algorithm requires an average of 3.0s to process each page when using a Core 2 Duo CPU running at 2.20GHz.

V. Discussion

The most significant limitation of this algorithm is that the method for finding words is very sensitive to the ratio of blur kernel to text size. Further work could involve developing a method to automatically adjust the kernel size. Alternatively, aspects of the Docstrum algorithm could be adopted to allow lines to be found without having to first find connected components representing words.

The method for determining the page frame depends on knowing the maximum line length and assumes that the text is fully justified. Future work could involve either developing an algorithm to detect the maximum line length, or developing a page frame detection algorithm which does not depend on this information.

The algorithm presented here is incapable of detecting block quotes.

VI. Conclusion

In this paper we proposed a method for analysing the pages of a typical book. This method uses a bottom-up nearest-neighbour approach for grouping connected components into textual lines, which obviates the need to explicitly determine the page skew. Textual lines are categorised according to logical structure using a state machine.

The proposed method distinguishes itself from other methods because it encompasses the complete process from binary image to labelled features.

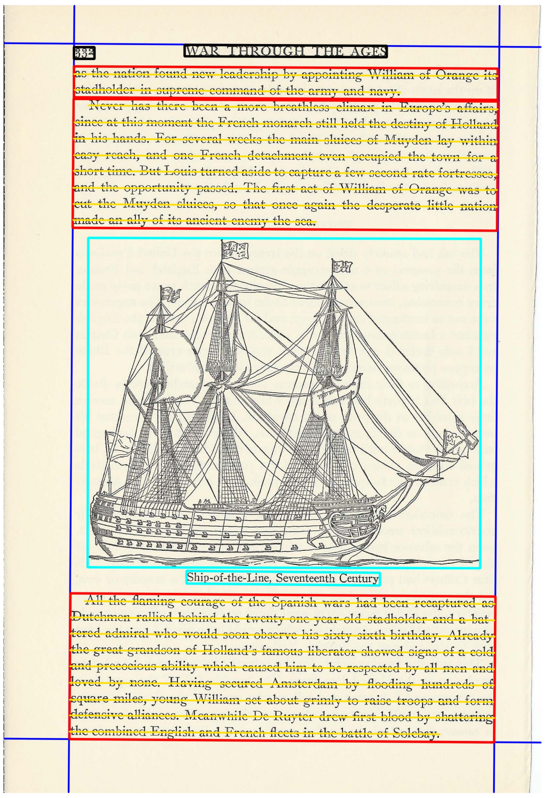
The proposed method is able to correctly segment and label 99.83% of all targeted features in a set of 54 sample pages.

The proposed method is very sensitive to the size of the blur kernel and the size of the maximum line length. Future research should attempt to find methods to automatically derive these parameters.

References

[1] R. Smith, “Hybrid page layout analysis via tab-stop detection,” in *Document Analysis and Recognition, 2009. ICDAR ’09. 10th International Conference on*, 2009, pp. 241–245.

[2] S. Mao, A. Rosenfeld, and T. Kanungo, “Document structure analysis algorithms: a literature survey,” in *Electronic Imaging 2003*. International Society for Optics and Photonics, 2003, pp. 197–207.



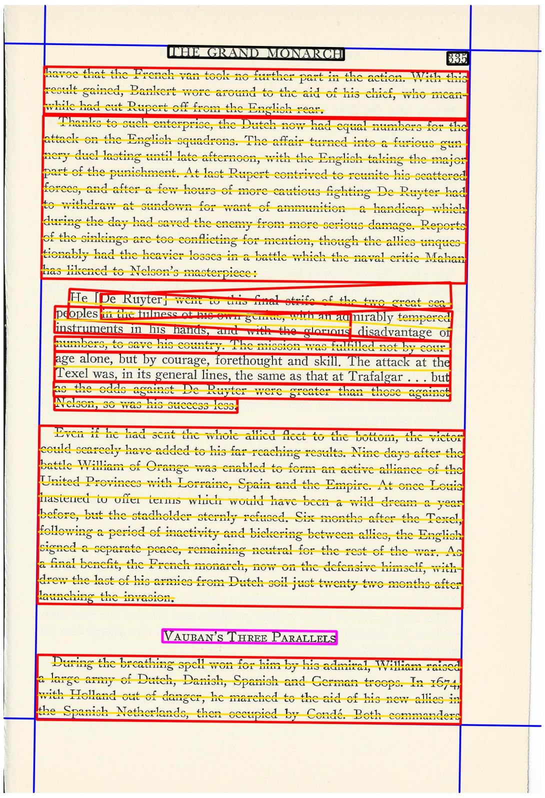
332 WAR THROUGH THE AGES

as the nation found new leadership by appointing William of Orange its stadholder in supreme command of the army and navy.

Never has there been a more breathless climax in Europe's affairs, since at this moment the French monarch still held the destiny of Holland in his hands. For several weeks the main sluices of Muyden lay within easy reach, and one French detachment even occupied the town for a short time. But Louis turned aside to capture a few second rate fortresses, and the opportunity passed. The first act of William of Orange was to cut the Muyden sluices, so that once again the desperate little nation made an ally of its ancient enemy the sea.

All the flaming courage of the Spanish wars had been recaptured as Dutchmen rallied behind the twenty one year-old stadholder and a battered admiral who would soon observe his sixty-sixth birthday. Already the great grandson of Holland's famous liberator showed signs of a cold and precocious ability which caused him to be respected by all men and loved by none. Having secured Amsterdam by flooding hundreds of square miles, young William set about grimly to raise troops and form defensive alliances. Meanwhile De Ruyter drew first blood by shattering the combined English and French fleets in the battle of Solebay.

Fig. 5. A good result. Yellow lines represent textual lines; red boxes represent paragraphs; dark blue lines represent the page frame; cyan boxes represent a figure; black boxes represent headers and footers; magenta boxes represent section titles.



THE GRAND MONARCH 335

havoc that the French van took no further part in the action. With this result gained, Bankert wore around to the aid of his chief, who meanwhile had cut Rupert off from the English rear.

Thanks to such enterprise, the Dutch now had equal numbers for the attack on the English squadrons. The affair turned into a furious gunnery duel lasting until late afternoon, with the English taking the major part of the punishment. At last Rupert contrived to reunite his scattered forces, and after a few hours of more cautious fighting De Ruyter had to withdraw at sundown for want of ammunition—a handicap which during the day had saved the enemy from more serious damage. Reports of the sinkings are too conflicting for mention, though the allies unquestionably had the heavier losses in a battle which the naval critic Mahan has likened to Nelson's masterpiece:

He [De Ruyter] went to this final strife of the two great sea-peoples in the fulness of his own genius, with an admirably tempered instruments in his hands, and with the glorious disadvantage of numbers, to save his country. The mission was fulfilled not by courage alone, but by courage, forethought and skill. The attack at the Texel was, in its general lines, the same as that at Trafalgar . . . but as the odds against De Ruyter were greater than those against Nelson, so was his success less.

Even if he had sent the whole allied fleet to the bottom, the victor could scarcely have added to his far-reaching results. Nine days after the battle William of Orange was enabled to form an active alliance of the United Provinces with Lorraine, Spain and the Empire. At once Louis hastened to offer terms which would have been a wild dream a year before, but the stadholder sternly refused. Six months after the Texel, following a period of inactivity and bickering between allies, the English signed a separate peace, remaining neutral for the rest of the war. As a final benefit, the French monarch, now on the defensive himself, withdrew the last of his armies from Dutch soil just twenty two months after launching the invasion.

VAUBAN'S THREE PARALLELS

During the breathing spell won for him by his admiral, William raised a large army of Dutch, Danish, Spanish and German troops. In 1674, with Holland out of danger, he marched to the aid of his new allies in the Spanish Netherlands, then occupied by Condé. Both commanders

Fig. 6. A bad result. The smaller line spacing in the block quote causes two words on consecutive lines to merge into a single connected component.

[3] G. Nagy and S. Seth, "Hierarchical representation of optically scanned documents," in *Proceedings of International Conference on Pattern Recognition*, vol. 1, 1984, pp. 347–349.

[4] H. S. Baird, S. E. Jones, and S. J. Fortune, "Image segmentation by shape-directed covers," in *Pattern Recognition, 1990. Proceedings., 10th International Conference on*, vol. 1. IEEE, 1990, pp. 820–825.

[5] L. O'Gorman, "The document spectrum for page layout analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 15, no. 11, pp. 1162–1173, 1993.

[6] K. Kise, A. Sato, and M. Iwata, "Segmentation of page images using the area voronoi diagram," *Computer Vision and Image Understanding*, vol. 70, no. 3, pp. 370–382, 1998.

[7] S. Mao and T. Kanungo, "Empirical performance evaluation methodology and its application to page segmentation algorithms," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 3, pp. 242–256, 2001.

[8] R. W. Smith, "Hybrid page layout analysis via tab-stop detection," in *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*. IEEE, 2009, pp. 241–245.

[9] H. S. Baird, "The skew angle of printed documents," in *Document image analysis*. IEEE Computer Society Press, 1995, pp. 204–208.

[10] I. Konya, S. Eickeler, and C. Seibert, "Fast seamless skew and orientation detection in document images," in *Pattern Recognition (ICPR), 2010 20th International Conference on*. IEEE, 2010, pp. 1924–1928.

[11] G. Meng, C. Pan, N. Zheng, and C. Sun, "Skew estimation of document images using bagging," *Image Processing, IEEE Transactions on*, vol. 19, no. 7, pp. 1837–1846, 2010.

[12] D. Derrien-Peden, "Frame-based system for macro-typographical structure analysis in scientific papers," in *Pro ceedings of International Conference on Document Analysis and Recognition*, 1991, pp. 311–319.

[13] J.-L. Meunier, "Method and apparatus for determining logical document structure," Jun. 24 2008, uS Patent 7,392,473.

[14] M. Krishnamoorthy, G. Nagy, S. Seth, and M. Viswanathan, "Syntactic segmentation and labeling of digitized pages from technical journals," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 15, no. 7, pp. 737–747, 1993.

[15] N. Otsu, "A threshold selection method from gray-level histograms," *Automatica*, vol. 11, no. 285-296, pp. 23–27, 1975.