Published in Image Processing On Line on 2026–01–00.
Submitted on 2024–11–17, accepted on 2025–11–21.
ISSN 2105–1232 © 2026 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2026.591

Voronoi Diagrams for Page Segmentation

Marina Gardella[1], Ignacio Ramirez[2]

[1]Université Paris-Saclay, ENS Paris-Saclay, Centre Borelli, Gif-sur-Yvette, France
[2]IIE, Facultad de Ingeniería, Universidad de la República, Uruguay
marina.gardella@ens-paris-saclay.fr, nacho@fing.edu.uy

Communicated by Seginus Mowlavi and Antoine Tadros   *Demo edited by* Marina Gardella

Abstract

Page segmentation is a key task in document processing, enabling effective extraction of structured information from diverse document types. This paper presents an in-depth analysis of the method proposed by Kise et al., a bottom-up approach using area Voronoi diagrams to identify spatial relationships between document parts. Our work provides a detailed description of the method, emphasizing clarity, reproducibility, and transparency, particularly regarding aspects not fully specified in the original paper. We highlight the impact of the parameter settings and preprocessing steps on the method's performance. Through extensive testing, we demonstrate that the method can handle a wide range of layouts but exhibits notable sensitivity to specific document characteristics, especially in handling complex elements like handwritten text, lists, drop-caps, and tables.

Source Code

The reviewed source code and documentation for this algorithm are available from the web page of this article[1]. Usage instructions are included in the README.md file of the archive.

Keywords: document layout analysis; Voronoi diagram

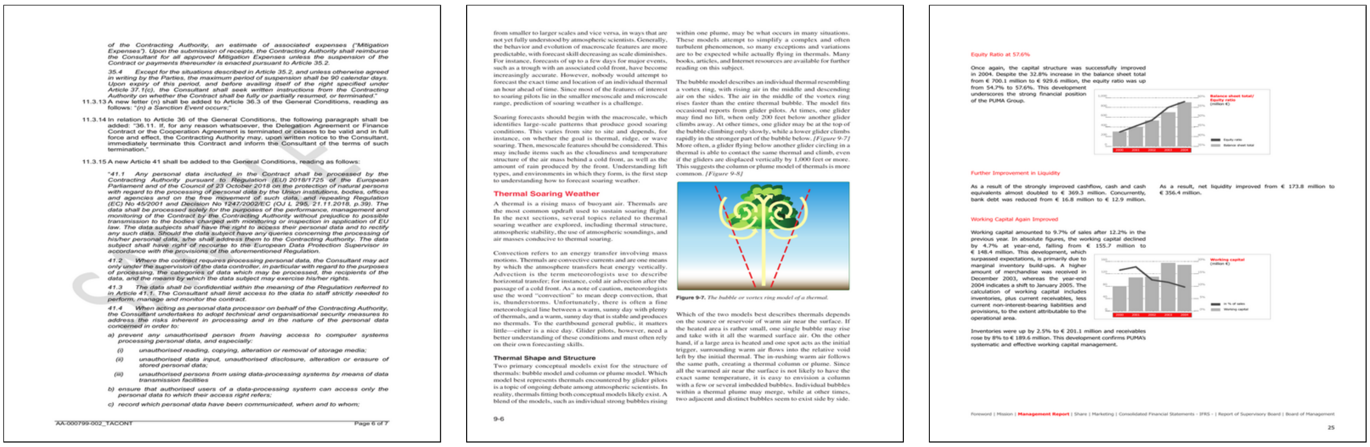
1 Introduction

Document layout analysis is crucial in the field of document processing, as it enables systems to extract meaningful information from diverse types of documents. Page segmentation, in particular, plays a pivotal role in this process by dividing a page into distinct regions, such as text blocks, images, tables, and headers. Accurate segmentation is essential for enabling downstream tasks like optical character recognition (OCR), content extraction, and layout reconstruction, ensuring that different elements are appropriately interpreted.

Document layouts can vary significantly depending on the structure and formatting of the content, making page segmentation a challenging task. Some documents exhibit non-overlapping layouts

[1] https://doi.org/10.5201/ipol.2026.591

Marina Gardella, Ignacio Ramirez , *Voronoi Diagrams for Page Segmentation*, Image Processing On Line, 16 (2026), pp. 1–25.
https://doi.org/10.5201/ipol.2026.591



of the Contracting Authority, an estimate of associated expenses ("Mitigation Expenses"). Upon the submission of receipts, the Contracting Authority shall reimburse the Consultant for all approved Mitigation Expenses unless the suspension of the Contract or payments thereunder is enacted pursuant to Article 35.2.

35.4 Except for the situations described in Article 35.2, and unless otherwise agreed in writing by the Parties, the maximum period of suspension shall be 90 calendar days. Upon expiry of this period, and before availing itself of the right specified under Article 37.1(c), the Consultant shall seek written instructions from the Contracting Authority on whether the Contract shall be fully or partially resumed, or terminated."

11.3.13 A new letter (n) shall be added to Article 36.3 of the General Conditions, reading as follows: "(n) a Sanction Event occurs;"

11.3.14 In relation to Article 36 of the General Conditions, the following paragraph shall be added: "36.11. If, for any reason whatsoever, the Delegation Agreement or Finance Contract or the Cooperation Agreement is terminated or ceases to be valid and in full force and effect, the Contracting Authority may, upon written notice to the Consultant, immediately terminate this Contract and inform the Consultant of the terms of such termination."

11.3.15 A new Article 41 shall be added to the General Conditions, reading as follows:

"41.1 Any personal data included in the Contract shall be processed by the Contracting Authority pursuant to Regulation (EU) 2018/1725 of the European Parliament and of the Council of 23 October 2018 on the protection of natural persons with regard to the processing of personal data by the Union institutions, bodies, offices and agencies and on the free movement of such data, and repealing Regulation (EC) No 45/2001 and Decision No 1247/2002/EC (OJ L 295, 21.11.2018, p.39). The data shall be processed solely for the purposes of the performance, management and monitoring of the Contract by the Contracting Authority without prejudice to possible transmission to the bodies charged with monitoring or inspection in application of EU law. The data subjects shall have the right to access their personal data and to rectify any such data. Should the data subject have any queries concerning the processing of his/her personal data, s/he shall address them to the Contracting Authority. The data subject shall have right of recourse to the European Data Protection Supervisor in accordance with the provisions of the aforementioned Regulation.

41.2 Where the contract requires processing personal data, the Consultant may act only under the supervision of the data controller, in particular with regard to the purposes of processing, the categories of data which may be processed, the recipients of the data, and the means by which the data subject may exercise his/her rights.

41.3 The data shall be confidential within the meaning of the Regulation referred to in Article 41.1. The Consultant shall limit access to the data to staff strictly needed to perform, manage and monitor the contract.

41.4 When acting as personal data processor on behalf of the Contracting Authority, the Consultant undertakes to adopt technical and organisational security measures to address the risks inherent in processing and in the nature of the personal data concerned in order to:

a) prevent any unauthorised person from having access to computer systems processing personal data, and especially:

(i) unauthorised reading, copying, alteration or removal of storage media;

(ii) unauthorised data input, unauthorised disclosure, alteration or erasure of stored personal data;

(iii) unauthorised persons from using data-processing systems by means of data transmission facilities

b) ensure that authorised users of a data-processing system can access only the personal data to which their access right refers;

c) record which personal data have been communicated, when and to whom;

AA-000799-002_TACONT Page 6 of 7

(a) Overlapping Layout

from smaller to larger scales and vice versa, in ways that are not yet fully understood by atmospheric scientists. Generally, the behavior and evolution of macroscale features are more predictable, with forecast skill decreasing as scale diminishes. For instance, forecasts of up to a few days for major events, such as a trough with an associated cold front, have become increasingly accurate. However, nobody would attempt to forecast the exact time and location of an individual thermal an hour ahead of time. Since most of the features of interest to soaring pilots lie in the smaller mesoscale and microscale range, prediction of soaring weather is a challenge.

Soaring forecasts should begin with the macroscale, which identifies large-scale patterns that produce good soaring conditions. This varies from site to site and depends, for instance, on whether the goal is thermal, ridge, or wave soaring. Then, mesoscale features should be considered. This may include items such as the cloudiness and temperature structure of the air mass behind a cold front, as well as the amount of rain produced by the front. Understanding lift types, and environments in which they form, is the first step to understanding how to forecast soaring weather.

Thermal Soaring Weather

A thermal is a rising mass of buoyant air. Thermals are the most common updraft used to sustain soaring flight. In the next sections, several topics related to thermal soaring weather are explored, including thermal structure, atmospheric stability, the use of atmospheric soundings, and air masses conducive to thermal soaring.

Convection refers to an energy transfer involving mass motions. Thermals are convective currents and are one means by which the atmosphere transfers heat energy vertically. Advection is the term meteorologists use to describe horizontal transfer; for instance, cold air advection after the passage of a cold front. As a note of caution, meteorologists use the word "convection" to mean deep convection, that is, thunderstorms. Unfortunately, there is often a fine meteorological line between a warm, sunny day with plenty of thermals, and a warm, sunny day that is stable and produces no thermals. To the earthbound general public, it matters little—either is a nice day. Glider pilots, however, need a better understanding of these conditions and must often rely on their own forecasting skills.

Thermal Shape and Structure

Two primary conceptual models exist for the structure of thermals: bubble model and column or plume model. Which model best represents thermals encountered by glider pilots is a topic of ongoing debate among atmospheric scientists. In reality, thermals fitting both conceptual models likely exist. A blend of the models, such as individual strong bubbles rising within one plume, may be what occurs in many situations. These models attempt to simplify a complex and often turbulent phenomenon, so many exceptions and variations are to be expected while actually flying in thermals. Many books, articles, and Internet resources are available for further reading on this subject.

The bubble model describes an individual thermal resembling a vortex ring, with rising air in the middle and descending air on the sides. The air in the middle of the vortex ring rises faster than the entire thermal bubble. The model fits occasional reports from glider pilots. At times, one glider may find no lift, when only 200 feet below another glider climbs away. At other times, one glider may be at the top of the bubble climbing only slowly, while a lower glider climbs rapidly in the stronger part of the bubble below. *[Figure 9-7]* More often, a glider flying below another glider circling in a thermal is able to contact the same thermal and climb, even if the gliders are displaced vertically by 1,000 feet or more. This suggests the column or plume model of thermals is more common. *[Figure 9-8]*

Figure 9-7. *The bubble or vortex ring model of a thermal.*

Which of the two models best describes thermals depends on the source or reservoir of warm air near the surface. If the heated area is rather small, one single bubble may rise and take with it all the warmed surface air. On the other hand, if a large area is heated and one spot acts as the initial trigger, surrounding warm air flows into the relative void left by the initial thermal. The in-rushing warm air follows the same path, creating a thermal column or plume. Since all the warmed air near the surface is not likely to have the exact same temperature, it is easy to envision a column with a few or several imbedded bubbles. Individual bubbles within a thermal plume may merge, while at other times, two adjacent and distinct bubbles seem to exist side by side.

9-6

(b) Manhattan Layout

Equity Ratio at 57.6%

Once again, the capital structure was successfully improved in 2004. Despite the 32.8% increase in the balance sheet total from € 700.1 million to € 929.6 million, the equity ratio was up from 54.7% to 57.6%. This development underscores the strong financial position of the PUMA Group.

Further Improvement in Liquidity

As a result of the strongly improved cashflow, cash and cash equivalents almost doubled to € 369.3 million. Concurrently, bank debt was reduced from € 16.8 million to € 12.9 million. As a result, net liquidity improved from € 173.8 million to € 356.4 million.

Working Capital Again Improved

Working capital amounted to 9.7% of sales after 12.2% in the previous year. In absolute figures, the working capital declined by 4.7% at year-end, falling from € 155.7 million to € 148.4 million. This development, which surpassed expectations, is primarily due to marginal inventory build-ups. A higher amount of merchandise was received in December 2003, whereas the year-end 2004 indicates a shift to January 2005. The calculation of working capital includes inventories, plus current receivables, less current non-interest-bearing liabilities and provisions, to the extent attributable to the operational area.

Inventories were up by 2.5% to € 201.1 million and receivables rose by 8% to € 189.6 million. This development confirms PUMA's systematic and effective working capital management.

Foreword | Mission | **Management Report** | Share | Marketing | Consolidated Financial Statements - IFRS - | Report of Supervisory Board | Board of Management

25

(c) Non-Manhattan Layout

Figure 1: Examples of documents with different layouts. The document in Figure 1a presents overlapping elements: a stamp overlaps the text. The document in Figure 1b has a Manhattan layout: document parts have rectangular boundaries. In contrast, the document in Figure 1c involves irregular boundaries, following a non-Manhattan layout.

where content blocks, such as text and images, are distinctly separated. More complex documents may include overlapping regions where, for instance, text may flow over images or stamps may appear over the text. Figure 1a shows an example of an overlapping layout. In terms of geometric structure, the so-called *Manhattan layouts* follow rectangular divisions, which are common in standard forms or structured documents, as shown in Figure 1b. Non-Manhattan layouts, often found in more visually dynamic documents like magazines or presentations, involve irregular or slanted boundaries. Such a layout is depicted in Figure 1c.

Various strategies have been developed to handle page segmentation effectively, each with its own strengths and weaknesses. Bottom-up approaches [8, 9] start with small components, such as connected pixels or characters, and progressively group them into larger structures like words, lines, and paragraphs, based on spatial proximity or similarity. Conversely, top-down approaches [4, 6] begin by dividing the page into large regions, which are then recursively split until the desired level of granularity is achieved (lines, words, characters). Hybrid methods [13, 12] combine elements of both strategies to leverage their respective advantages, offering more flexibility in handling diverse layouts. Finally, there are holistic approaches that attempt to infer the whole structure of the document in one shot. Recent methods based on deep learning [3, 2, 5, 14, 15] belong to this category. These methods rely on learning their task from large amounts of labeled data, which has become increasingly available [11, 16].

This work describes and provides an implementation of the method developed by Kise et al. [8] that is as faithful as possible to the original. This method is a bottom-up approach which uses the so called *Area Voronoi Diagram* of the connected components of the image as an auxiliary data structure for determining the spatial relationship between such components. In describing the method, we focus on clarity of presentation, unambiguous definitions, and total transparency regarding the aspects that are either not fully described in the original paper, or which may have a relevant influence on the performance of the method, such as thresholds, parameters, etc. As for the implementation, we provide a clear, concise, and thoroughly documented implementation with a rich command-line interface. The code can be run as a Python-only implementation, which is readily portable.

2 Voronoi Diagrams

Computational geometry is a branch of computer science that deals with the study and development of algorithms for solving geometric problems. This field often emerges in response to challenges found in image processing and pattern recognition, where the inherent geometry of visual data needs to be analyzed, understood, and manipulated. Among the many tools of computational geometry, Voronoi diagrams are particularly significant, as they arise as a natural partitioning of the space [1]. In this section, we shall review the main concepts of Voronoi diagrams.

2.1 Point Voronoi Diagram

Let $P = \{p_1, \dots, p_n\} \subset \mathbb{R}^2$ be a set of points, which we shall refer to as *generators*. The *Voronoi region* associated with p_i is the set of points that are closer to p_i than to any other point in P. Formally, the Voronoi region associated with p_i is given by

$$\mathscr{v}(p_i) = \{p : d(p, p_i) \leq d(p, p_j),\ \text{for all } j \neq i\} = \bigcap_{j \neq i} \{p : d(p, p_i) \leq d(p, p_j)\}, \tag{1}$$

where d denotes the Euclidean or l_2 distance. According to Equation (1), each Voronoi region is an intersection of half-planes, that is, a convex polygon. The boundaries of such regions are line segments, half-line segments, or infinite lines, which we shall refer to as *Voronoi ridges*[2] and we shall refer to their endpoints as *Voronoi vertices*. Finally, the *Point Voronoi Diagram* associated with the set of generators P is the set of Voronoi regions associated with each point in P

$$\mathscr{V}(P) = \{\mathscr{v}(p_1), \dots, \mathscr{v}(p_n)\}. \tag{2}$$

2.2 Area Voronoi Diagram

Point Voronoi diagrams define regions according to the distance to a set of isolated points. This idea can be generalized to distances to a set of regions of connected points, which we call *components*. Consider $C = \{c_1, \dots, c_n\}$ a set of non-overlapping components in $\mathbb{R}^2$, and $d(p, c_r)$ the distance between a point $p \in \mathbb{R}^2$ and a component c_r, defined as the shortest Euclidean distance from p to any point in c_r. Analogous to the previous case, the Voronoi region associated with c_r is the set of points that are closer to c_r than to any other component in C

$$\mathscr{v}(c_r) = \{p \text{ such that } d(p, c_r) \leq d(p, c_l),\ \text{for all } l \neq r\}. \tag{3}$$

Analogous to the Point Voronoi Diagram, the *Area Voronoi Diagram* associated with C is the set of Voronoi regions associated with each component in C

$$\mathscr{V}(C) = \{\mathscr{v}(c_1), \dots, \mathscr{v}(c_n)\}. \tag{4}$$

2.3 Approximate Area Voronoi Diagram

Computing exact Area Voronoi diagrams can be computationally expensive. However, they can be approximated using the Point Voronoi Diagram. Such a construction involves three steps:

1. For each component c_i, sample points along its boundary $P_{c_i} = \{p_1^{c_i}, \dots, p_{n_{c_i}}^{c_i}\}$.

[2] The original paper refers to these as *Voronoi edges*. We choose to keep the terminology used in the documentation of `SciPy`, the library on which we rely for the computation of the Voronoi diagrams. This choice is intended to facilitate the understanding of the code.

2. Build the Point Voronoi Diagram using $\bigcup_{c_i \in C} P_{c_i}$ as generators.

3. Delete the Voronoi ridges generated from points that belong to the same component.

Note that, while the Voronoi regions in the Area Voronoi Diagram associated with points in $\mathbb{R}^2$ need not be polygons, those arising from this approximation shall be. Indeed, in this approximated version, each ridge is a *polygonal chain* made up of Voronoi ridges that separate the same two connected components.

In the following sections, we will refer to *Voronoi diagrams* as the collection of Voronoi ridges that define the boundaries of the Voronoi regions, rather than as the sets of Voronoi regions themselves as defined in Equation (2) or Equation (4). Note that these two are equivalent.

3 Method

The method proposed by Kise et al. [8] builds on the approximated Area Voronoi Diagram, described in Section 2.3, using the connected components of the document image as generators. The authors state that the ridges obtained using such a construction represent the candidates for the boundaries of the document *parts*[3]. In this setting, page segmentation translates to selecting those Voronoi ridges that are actually associated with those boundaries.

Equivalently, this problem can be stated as the deletion of superfluous ridges that lie between elements in the same document part. To do so, the method relies on two characteristic features of these ridges: i) the connected components separated by them are *close* and ii) their areas are *similar*. The result of the above deletion process or *pruning* usually destroys the structure of the Area Voronoi Diagram, leaving many isolated ridges and ridges that do not define closed boundaries. Therefore, a final pruning step is applied in order to keep only those meaningful ridges that actually define regions.

The following sections describe the algorithm as proposed by the authors. We highlight the details that are left undefined or that are ambiguous in the original article and explicitly describe how we handle these points in our implementation. A concise summary of the method pipeline is given in Figure 2.

3.1 Binarization

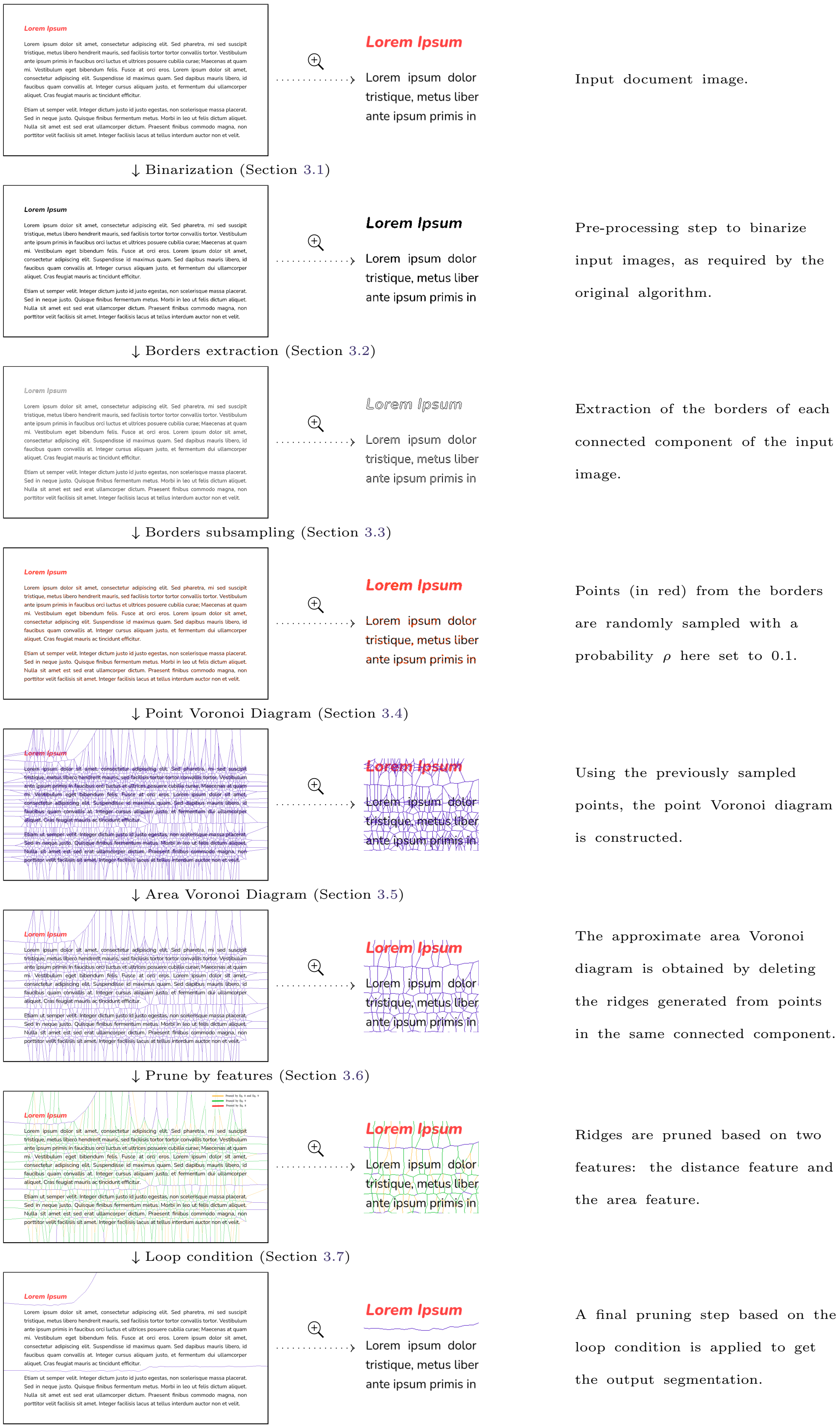
The method [8] takes as input a binary image I whose connected components are assumed to be the relevant objects for document segmentation, such as characters, symbols and graphics. It is not specified how to deal with non-binary inputs, such as grayscale or RGB images.

Our implementation provides a preprocessing step to binarize such images. If the input is a color image, we obtain a grayscale image I_G by averaging the color channels. Then, I_G is thresholded to obtain a binary image. In our implementation, the threshold used at this step can be set manually to an arbitrary value or it can be computed automatically using Otsu's thresholding method [10].

We describe this method here for completeness. Let g be the empirical distribution of the pixels in the grayscale input I_G which range between 0 and $M-1$; g_i being the value for the i-th grayscale level. Otsu's threshold τ is the value that minimizes the intra-class variance of the two classes of pixel intensities

$$\tau = \arg\min_t \left\{ \sum_{i \leq \tau} g_i \left(i - \mu_0\right)^2 + \sum_{j > \tau} g_i \left(j - \mu_1\right)^2 \right\}, \text{ where } \mu_0 = \sum_{i \leq \tau} g_i i \text{ and } \mu_1 = \sum_{j > \tau} g_j j. \quad (5)$$

[3] The original paper refers to these as document *components*, but this is quite confusing as components are also referred to as the connected components of the image. Instead, we shall refer to them as document *parts* hereafter.



Lorem Ipsum

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed pharetra, mi sed suscipit tristique, metus libero hendrerit mauris, sed facilisis tortor tortor convallis tortor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia curae; Maecenas at quam mi. Vestibulum eget bibendum felis. Fusce at orci eros. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse id maximus quam. Sed dapibus mauris libero, id faucibus quam convallis at. Integer cursus aliquam justo, et fermentum dui ullamcorper aliquet. Cras feugiat mauris ac tincidunt efficitur.

Etiam ut semper velit. Integer dictum justo id justo egestas, non scelerisque massa placerat. Sed in neque justo. Quisque finibus fermentum metus. Morbi in leo ut felis dictum aliquet. Nulla sit amet est sed erat ullamcorper dictum. Praesent finibus commodo magna, non porttitor velit facilisis sit amet. Integer facilisis lacus at tellus interdum auctor non et velit.

Input document image.

↓ Binarization (Section 3.1)

Pre-processing step to binarize input images, as required by the original algorithm.

↓ Borders extraction (Section 3.2)

Extraction of the borders of each connected component of the input image.

↓ Borders subsampling (Section 3.3)

Points (in red) from the borders are randomly sampled with a probability ρ here set to 0.1.

↓ Point Voronoi Diagram (Section 3.4)

Using the previously sampled points, the point Voronoi diagram is constructed.

↓ Area Voronoi Diagram (Section 3.5)

The approximate area Voronoi diagram is obtained by deleting the ridges generated from points in the same connected component.

↓ Prune by features (Section 3.6)

Ridges are pruned based on two features: the distance feature and the area feature.

↓ Loop condition (Section 3.7)

A final pruning step based on the loop condition is applied to get the output segmentation.

Figure 2: Summary of the main steps of the method's pipeline together with their intermediate outputs.

Assigning a different value to each of these two classes gives the desired binary output.

3.2 Labeling of Connected Components and Extraction of Border Points

For digital images, which are defined over a finite, discrete grid of coordinates $\Omega \subset \mathbb{N}^2$, the set of components C described in Section 2.2 can be fully specified by a *labeling map*: this is a pseudo-image where a positive value $1 \leq r = C(i,j) \leq n$ indicates that the point at $(i,j) \in \Omega$ belongs to the component $c_r \in C$, and $C(i,j) = 0$ indicates that the corresponding point (i,j) does not belong to a connected component. Using this map, a connected component c_r can be recovered from the label map as $c_r = \{(i,j) \in \Omega : C(i,j) = r\}$. As both definitions of C equivalent, we will use the *label map* formalism hereafter.

For digital images, there is no unique definition of connectivity: the most common alternatives are 4-connectivity and 8-connectivity. In the former case, two pixels are considered connected only if they are either one above the other, or one next to the other. In the latter case, they are also considered connected if they are adjacent in any diagonal direction.

The original work [8] uses a *border following* method [7] which produces an 8-connectivity label map and gives the borders of the connected regions as a byproduct. While the implementation of such a method is necessary to guarantee full reproducibility of the original method, it is outside the scope of this paper, as it would imply the complete reproduction of another algorithm. Furthermore, it is not a core part of the algorithm. Therefore, we opt to rely on off-the-shelf solutions: connected components are extracted from the binary image using the `morphology` module in `scikit-image`, and borders are obtained by eroding the binary image and subtracting the result from the original. More details on the implementation are given in Section 5.

A simple denoising strategy is also applied at this point: small connected components having borders of length smaller than N pixels are discarded. According to Kise et al. [8], the value of N depends on the resolution of the document image. In their experiments, they vary this value between 4 and 13, without giving specific criteria for setting this parameter. The result of this step is the set of all border points, which we shall refer to as B.

3.3 Borders Subsampling

As stated in Section 2.3, the Point Voronoi Diagram is constructed from a subset P of the border points in B. As mentioned in Section 3.2, the original paper [8] uses a border tracking method to extract border points. Based on this, they use this border parametrization to keep every Rth pixel, with R being a parameter of the method.

Our choice is an independent random sampling where a border point from B is kept in P with probability ρ and discarded otherwise. In principle, all the points can be used: this approach would yield the *exact* Area Voronoi Diagram and, according to our experiments, often better results. However, the computational cost of doing so might be too high. This parameter is actually important, as it defines a quality/computational cost trade-off. The value of this parameter has a non-negligible effect on the final result, which we shall explore in Section 6.

3.4 Point Voronoi Diagram

In this step, we construct the Point Voronoi Diagram $\mathcal{V}(P)$ using the set of sampled border points P as input, as described in Section 2.1. Two sampled border points (the generators) are associated with each ridge. These are the two input points that are closest to the ridge and equidistant from it by definition. Depending on whether the ridge is finite, half-finite or infinite it will also have two, one or zero vertices associated with it. These vertices are important for later stages of the algorithm.

3.5 Approximate Area Voronoi Diagram

The next step is to obtain the approximate Area Voronoi Diagram $\mathcal{V}(C)$ from the Point Voronoi Diagram, as described in Section 2.3. This is done by removing the ridges in $\mathcal{V}(P)$ generated by points from the same connected component. In the resulting diagram, the elements are no longer lines or segments but polygonal chains. This is an important difference compared with the original Point Voronoi Diagram. Note that, given the discrete nature of images, this would still be true even if we compute the exact Area Voronoi Diagram. However, this is no longer the case for the Area Voronoi Diagram of a set of connected components in $\mathbb{R}^2$, where ridges may include smooth curves.

3.6 Pruning by Features

All ridges in the resulting Area Voronoi Diagram lie between two neighboring connected components. As document parts are sets of connected components, the set of Voronoi ridges contains the actual boundaries of the different parts. Here we shall introduce two features used by the authors to decide if a ridge is superfluous or not: the distance between the connected components separated by it, and the ratio between their areas.

Given a ridge $r = \{l_1, \dots l_n\}$, there exist two connected components c_i and c_j separated by it. By construction, each line segment l_k in r is generated by a pair of points p_i^k and p_j^k lying on the boundary of c_i and c_j respectively. The *distance feature* associated with the ridge r is defined as [8]

$$D(r) = \min_{k=1,\dots,n} d(p_i^k, p_j^k). \tag{6}$$

Gaps between characters, words and text lines are narrower than those separating paragraphs or columns. However, relying solely on the distance feature in order to discard superfluous ridges may induce errors. Indeed, there are some cases where ridges separating different document parts are not associated with a large value of D. This is where the area of the connected components comes into play, as different document parts that lie close to each other usually have different areas.

This observation is captured by the area feature. Consider again a ridge r and the two connected components c_i and c_j separated by it. The *area feature* associated with the ridge r is defined as [8]

$$A(r) = \frac{\max\{a(c_i), a(c_j)\}}{\min\{a(c_i), a(c_j)\}}, \tag{7}$$

where $a(c)$ denotes the area of the connected component c, defined as its pixel count.

This pruning step aims at discarding ridges based on the above described features. A Voronoi ridge r is deleted if it satisfies one of the following conditions

$$D(r) < T_1, \quad \text{or} \tag{8}$$

$$\frac{D(r)}{T_2} + \frac{A(r)}{T_A} < 1, \tag{9}$$

where T_1 and T_2 are distance thresholds satisfying $T_1 < T_2$ and T_A is an area threshold. The intuition behind these criteria is that very close connected components, such as characters in the same word, are part of the same document part (Equation (8)). Additionally, not-so-close connected components that have similar areas, such as characters in different words or lines, are also part of the same document part (Equation (9)).

These criteria require setting values for the thresholds. The area threshold T_A is fixed by the authors to 40 as they argue that this value approximately corresponds to the largest area ratio between characters of the same font and size. On the other hand, the distance thresholds T_1 and T_2

are estimated from the image itself. To do so, the method relies on the histogram h of ridge distance features, discretized with step 1 and smoothed using an average filter over a symmetric window of size $2 \times w + 1$, where w is an integer parameter. An example of such a histogram is given in Figure 3, together with its corresponding smoothed version when setting $w = 2$. According to the paper [8], the value of w depends on the resolution of the document image. In their experiments, they vary this value between 0 and 2, without giving a specific criterion on how to set this parameter.

This histogram typically has three peaks indicating the most common spacing types: one for distances between letters, another one for the distance between words and another for distances between lines. According to the authors, the two largest ones, h_{v_1} and h_{v_2}, with $v_1 < v_2$ correspond to the inter-character distance and to the inter-line distance, respectively. Indeed, the peak for word spacing is generally smaller because fewer characters are located at the edges of words compared to the number between lines or within words.

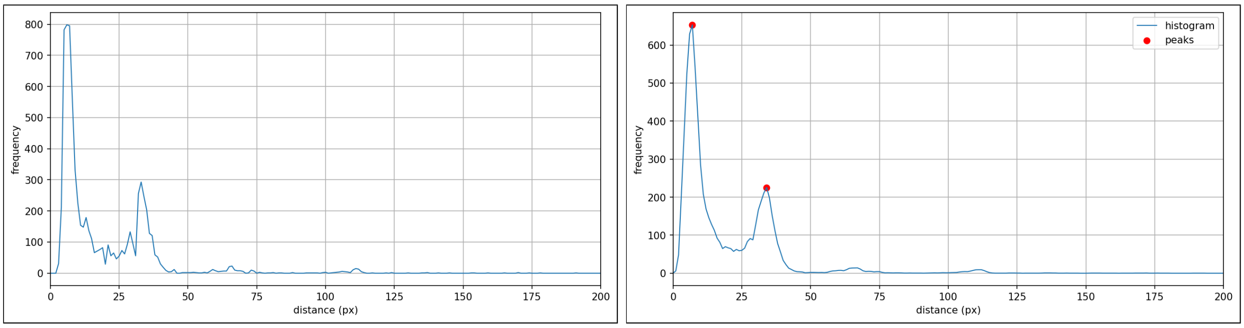


Figure 3: Raw distance features histogram (left) and its smoothed version (right) corresponding to the input document in Figure 2. The smoothed version is obtained using an average filter over a symmetric window of size 5. In red, the largest peaks found in the smoothed histogram.

As the distance between document parts is usually bigger than the inter-character distance, the authors set T_1 to v_1. However, v_2 is not an appropriate value for T_2 as it only captures the maxima and not the whole mode. Indeed, inter-line distances will concentrate around this value but might exhibit bigger values. To tackle this issue, the authors add a margin to v_2 and define T_2 as

$$T_2 = \min T \quad \text{such that} \quad \begin{cases} T & > & v_2 \\ h_T & = & th_{v_2} \end{cases} \tag{10}$$

where t is the *margin parameter*, set to 0.34 by the authors. As the histogram takes on discrete values, the optimum value of T_2 satisfying Equation (10) is found using linear interpolation.

3.7 Loop Condition

The diagram obtained once ridges are deleted based on their distance and area features needs further processing. Indeed, some Voronoi ridges that survive the previously described pruning procedure may not be part of the boundaries of a closed region. In order to discard those ridges, the authors define the *loop condition*: a ridge is said to satisfy the said condition if each of its vertices is either shared by another ridge or lies on the edge of the page. Ridges not satisfying the loop condition are irrelevant for delimiting document parts. Therefore, the final step of the method consists of discarding them.

4 Demo

In this section, we provide a brief explanation of how to use the online demo.

Input. The demo expects a single image as input, which may be binary, grayscale, or color. The most common image formats are accepted, but as soon as the image is uploaded, the system will convert it to an 8-bit-per-channel image. The maximum size allowed by the system is 3000 × 3000 pixels. If the input is larger than this, the demo will still work, but it will resize the image to meet the size requirement. In this case, the demo displays a warning message:

`Input #0 has been preprocessed {resized X%},`

where `X` stands for the resizing factor needed to fulfill the size requirement.

It is important to mention that these restrictions are needed by the system to guarantee a reasonable runtime but are not part of the source code. To avoid those, we advise the interested users to download and run the provided code locally.

Parameters. By default, the demo shows five parameters:

- Binarization mode: Defines the binarization method used for non-binary input images, as described in Section 3.1. The options are *Otsu's thresholding* and *Custom thresholding*. If the user selects the custom option, a threshold parameter is displayed for the user to choose a value in the range 1-255.
- N: Defines the denoising strategy to be applied, according to Section 3.2. If set to zero, no denoising is applied. If set to a value larger than zero, this will remove all blobs whose sizes are smaller than the specified value. The original method [8] does not specify a value for this parameter but, in their experiments, they vary it between 4 and 13. The user may choose a value between 0 (no denoising) and 13, with 4 being the default one.
- ρ: Defines the borders subsampling parameter, as described in Section 3.3. By setting this parameter, the user establishes the probability that a border point is kept. This parameter is actually important, as it defines a quality/computational cost trade-off. This parameter ranges from 0.01 to 1, with 0.1 as the default value.
- w: Defines a window of size $2 \times w + 1$ over which the histogram of distances between connected components is smoothed by averaging during the pruning step (see Section 3.6). The paper [8] does not clearly state a criterion for whether to smooth the distance features histogram. Furthermore, the size of the smoothing window is not clearly defined. We allow the user to set this parameter between 0 (no smoothing) and 4, with 2 as the default value.
- T_A: Defines the area threshold in the pruning criteria defined by Equation (9) during the pruning by features step (Section 3.6). The original method sets it to 40, arguing that this value approximately corresponds to the largest area ratio between characters of the same font and size. The demo allows the user to set this parameter to any value between 10 and 70, allowing users to test other values that may be better suited to their use case.

Outputs. The outputs are displayed as a gallery of PNG images. The images shown in this gallery follow the method's pipeline, as in Figure 2. These are:

- Input: Input image, as submitted by the user.
- Binary input: Binary image obtained after the binarization step (Section 3.1).
- Blobs removed: Resulting image after small blobs (defined by parameter N) are removed. If N is set to 0, this image is the same as the previous one.

- Extracted borders: Image displaying the borders of the connected components extracted by the method according to Section 3.2.
- Borders samples: Image displaying in red, on top of the borders, the points that have been sampled to construct the Voronoi Diagram (see Section 3.3). For visualization purposes, sampled points are slightly thickened.
- Point Voronoi Diagram: Point Voronoi Diagram computed using the sampled border points as generators (see Section 3.4).
- Area Voronoi Diagram: Area Voronoi Diagram obtained once redundant ridges are pruned (see Section 3.5).
- Distances histogram: Histogram of the ridge distance features (Equation (6)), discretized with step 1, as described in Section 3.6.
- Smoothed distances histogram: Histogram after smoothing using an average filter over a symmetric window of size $2 \times w + 1$. Note that here w takes the value chosen by the user. If it is 0, the original histogram will be displayed. The peaks found by the method are painted in red.
- Pruned by features: Diagram obtained once the Area Voronoi Diagram is pruned by the distance and area features, according to Equation (8) and Equation (9). The pruned ridges are painted in different colors according to the criterion that removed them: red for Equation (8), green for Equation (9) and orange for both.
- Final segmentation: Final diagram once the loop condition is imposed, as described in Section 3.7. This is the actual output of the method.

Log file. Besides the visual results displayed in the gallery, the method also outputs a log file which provides the user with information about the execution. For a standard execution, these messages include the number of connected components found in the image, the border points and how many are sampled, the number of ridges and vertices of both the Point Voronoi Diagram and the Area Voronoi Diagram, the numerical value of the peaks found in the histogram and the distance thresholds computed from them, the number of pruned ridges and the criterion used to prune them, finally, the number of ridges in the final diagram. Warning messages can also be displayed in the log file. These indicate potential issues or risks that occurred during the execution but did not stop the program from running. In those cases, we advise the user to check for unintended behavior.

However, if there were errors during the execution, not all of this information shall be provided. In that case, the log file will show a message explaining the type of error encountered. Some common errors include having only a single connected component in the image, missing border points, insufficient points to construct the Point Voronoi Diagram, or the absence of peaks in the distances histogram.

5 Detailed Implementation

The whole algorithm is implemented in a single executable Python file `voronoi.py`; a few utilities are provided separately in `util.py`. The program provides a command-line where the user can optionally modify the default parameters of all stages and choose alternatives for those parts that are not completely specified.

Algorithm 1: Voronoi document layout analysis algorithm.

```
Input $I$: binary image.
Output $\mathcal{V}^*$: resulting segmentation diagram.
$C \leftarrow$ get_connected_components($I$, $N$)
$B \leftarrow$ get_borders($C$)
$P \leftarrow$ sample_border_points($B, \rho$)
$\mathcal{V}(P) \leftarrow$ get_point_voronoi($P$)
$R \leftarrow$ eval_redundancy_criterion($\mathcal{V}(P), C$)            # bool. list, one entry per ridge
$\mathcal{V}(C) \leftarrow$ ridges $r_i$ in $\mathcal{V}(P)$ such that $R_{r_i} = 1$
$D, A \leftarrow$ compute_ridge_features($\mathcal{V}(C), C$)
$T_1, T_2 \leftarrow$ compute_thresholds($D$, $t$, $w$)
$\mathtt{P} \leftarrow$ eval_pruning_criteria($\mathcal{V}(C), D, A, T_1, T_2, T_A$)      # bool. list, one entry per ridge
$\overline{\mathcal{V}} \leftarrow$ ridges $r_i$ in $\mathcal{V}(C)$ such that $\mathtt{P}_{r_i} = 1$
$\mathcal{V}^* \leftarrow$ prune_by_loop_condition($\overline{\mathcal{V}}$)
return $\mathcal{V}^*$
```

The complete method, which is described in Algorithm 1, is divided into functions that correspond to each step defined in Section 3. We will now describe each of these using the actual function names for easy cross-reference with the code.

$I =$ **get_binary_image(I_{input})** Converts the submitted image I_{input} into a binary image I. This includes removing transparency (alpha channel) if present, converting color images to grayscale, and thresholding the grayscale image in order to obtain a binary representation. Details on the latter are provided in Section 3.1.

$C =$ **get_connected_components(I, N)** Produces a label map C where $C(i, j) = 0$ indicates that the pixel at position (i, j) in image I is background and $C(i, j) = c > 0$ means that the corresponding pixel in I belongs to a connected component identified by the label c. Connected components having borders of length smaller than N pixels are discarded.

$B =$ **get_borders(C)** Returns the bordering pixels of the connected components as a binary image. This is obtained as the difference between the binarized input image and a binary erosion of it using a 4-neighbors kernel.

$P =$ **sample_border_points(B, ρ)** Samples border points randomly using a $Z \sim \text{Bernoulli}(\rho)$ process where $Z = 1$ indicates to keep the point and 0 to discard it.

$\mathcal{V} =$ **get_point_voronoi(P)** Takes the sampled border points as inputs and produces the initial Point Voronoi Diagram. This is computed using `SciPy`.

$R =$ **eval_redundancy_criterion($\mathcal{V}(P), C$)** Returns a list of boolean values with a 1 in the i-th position for those Point Voronoi ridges that should be kept, and 0 otherwise. This is described in Algorithm 2. The approximate Area Voronoi Diagram $\mathcal{V}(C)$ is obtained once redundant ridges are discarded.

Note that, at this step we **do not** merge the Point Voronoi ridges into Area Voronoi ridges. Ridges are structurally kept as the original line segments. This detail is important in Algorithm 3.

$D, A =$ **compute_ridge_features**$(\mathcal{V}(C), C)$ Computes the distance $D(r)$ and area $A(r)$ ratio features of each ridge r in $\mathcal{V}(C)$. This is explained in Section 3.6. Algorithm 3 provides the pseudocode for this function.

Note that, when computing D_r, the distances are not the actual distances between the components but the distances between sampled points from their borders.

$T_1, T_2 =$ **compute_thresholds**(D, t, w) Computes the thresholds defined in Section 3.6. The pseudocode is provided in Algorithm 4.

$\mathtt{P} =$ **eval_pruning_criteria**$(\mathcal{V}(C), C, D, A, T_1, T_2, T_A)$ Returns a list of boolean values $\mathtt{P}$ where $\mathtt{P}_i = 0$ indicates that the i-th ridge, r_i, does not satisfy Equation (8) and Equation (9) and thus should be removed. The pseudocode for this function is given in Algorithm 5.

$L =$ **eval_loop_condition**$(\overline{\mathcal{V}})$ Returns a boolean list L where a 0 in the i-th element indicates that the corresponding ridge in $\overline{\mathcal{V}}$ does not satisfy the loop condition. See Algorithm 6.

$\mathcal{V}^* =$**prune_by_loop_condition**$(\overline{\mathcal{V}})$ This function takes $\overline{\mathcal{V}}$ as input and prunes those elements that do not satisfy the loop condition iteratively until no further pruning occurs, producing $\mathcal{V}^*$. This is described in Algorithm 7.

Algorithm 2: eval_redundancy_criterion.

```
Input $\mathcal{V}(P)$: Point Voronoi Diagram
Input $C$: label map
Output $R$: pruning indicator vector.
for each ridge $r_k$ in $\mathcal{V}(P)$ do
    $p_1, p_2 \leftarrow$ generator points of ridge $r_k$
    $R_k \leftarrow \mathtt{boolean}(C(p_1) \neq C(p_2))$
return $R$
```

Algorithm 3: compute_ridge_features.

```
Input $\mathcal{V}(C)$: Voronoi ridges (line segments)
Input $C$: label map
Output $D$: distance features
Output $A$: area ratio features
$n \leftarrow \max C$                                  # number of connected components
$D_{i,j} \leftarrow +\infty,\ i,j = 1,\dots,n$
$A_{i,j} \leftarrow 0,\ i,j = 1,\dots,n$                # initial value does not matter
for each line segment $l$ in $\mathcal{V}(C)$ do
    $p_1, p_2 \leftarrow$ generator points of line segment $l$
    $i, j \leftarrow C(p_1), C(p_2)$          # labels of the connected components of $p_1, p_2$ resp.
    $c_i, c_j \leftarrow$ connected components for labels $i$ and $j$ in $C$
    $A_{i,j} \leftarrow \frac{\max(|c_i|,|c_j|)}{\min(|c_i|,|c_j|)}$                # computes Equation (7)
    $D_{i,j} \leftarrow \min\{D_{i,j},\ \|p_1 - p_2\|_2\}$         # iteratively computes Equation (6)
return $D, A$
```

Algorithm 4: compute_thresholds.

```
Input $D$: distance features
Param $w$: smoothing window size parameter
Param $t$: margin parameter (set to 0.34)
Output $T_1, T_2$: thresholds
$\tilde{h} \leftarrow$ histogram of $D$
$h \leftarrow$ smoothed histogram, $h_i = \frac{1}{2w+1} \sum_{k=-w}^{w} \tilde{h}_{i+k}$    # out-of-range values are repeated
# below, if only one peak is found, we set $v_1 = v_2$
$v_1, v_2 \leftarrow$ index of the two largest peaks of $h$ in ascending order
$T_1 \leftarrow v_1$
$T \leftarrow v_2$
while $h_T > th_{v_2}$ do
    $T \leftarrow T + 1$
$T_2 \leftarrow T - 1 + \frac{th_{v_2} - h_{T-1}}{h_T - h_{T-1}}$    # computes Equation (10) using linear interpolation
return $T_1, T_2$
```

Algorithm 5: eval_pruning_criteria.

```
Input $\mathcal{V}(C)$: Voronoi ridges (line segments)
Input $C$: label map
Input $D$: distance features
Input $A$: area ratio features
Param $T_1$: Threshold 1
Param $T_2$: Threshold 2
Param $T_A$: area threshold
Output P: pruning indicator vector (boolean)
for each line segment $l_k$ in $\mathcal{V}(C)$ do
    $p_1, p_2 \leftarrow$ generator points of line segment $l_k$
    $i, j \leftarrow C(p_1), C(p_2)$    # labels of the connected components of $p_1, p_2$ resp.
    $\mathtt{P}_k \leftarrow D_{i,j} \geq T_1$ and $D_{i,j}/T_2 + A_{i,j}/T_A \geq 1$ # $\mathtt{P}_k = 1$ means keep
return P
```

Algorithm 6: eval_loop_condition.

```
Input $\mathcal{V}(C)$: Voronoi ridges
Output $L$: Pruning indicator vector
for each line segment $l_k$ in $\mathcal{V}(C)$ do
    $v_1, v_2 \leftarrow$ vertices of $l_k$
    # the vertex value $\infty$ below means border vertex; $L_k = 1$ means keep
    $L_k \leftarrow v_1$ not only in $l_i$ or $v_1 = \infty$ and $v_2$ not only in $l_i$ or $v_2 = \infty$
return $L$
```

Algorithm 7: prune_by_loop_condition.

```
Input $\overline{\mathcal{V}}$: Voronoi ridges
Output $\mathcal{V}^*$: Pruned ridges
$\mathcal{V}^* \leftarrow \overline{\mathcal{V}}$
repeat
    $L \leftarrow$ eval_loop_condition($\mathcal{V}^*$)
    $\mathcal{V}^* = \{r_k \in \mathcal{V}^* : L_k = 1, k = 1, \ldots, |\mathcal{V}^*|\}$
until $L$ is all ones
return $\mathcal{V}^*$
```

6 Experiments

6.1 Impact of the Parameters

Binarization mode. While Otsu's thresholding method enables us to automate the binarization procedure by minimizing intra-class intensity variance, custom thresholding might work better in some cases, as shown in Figure 4. In this example, the background is completely white and the foreground exhibits several grayscale intensities. Otsu's threshold takes parts of the figures as background, causing an over-segmentation of the document that does not respect the figures' borders. On the other hand, a custom threshold set to a value close to 255 (250 in this example) correctly identifies the pictures in the document as foreground. The final segmentation with this thresholding approach correctly respects figures' boundaries.

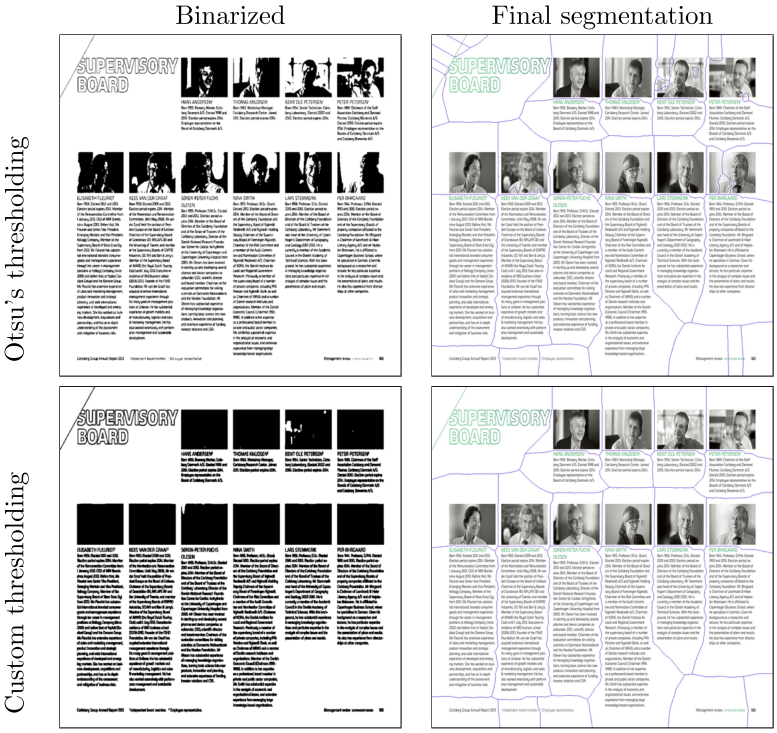


Figure 4: Comparison between Otsu's thresholding and custom thresholding. When binarization is performed using Otsu's threshold, parts of the pictures in the document are taken as background, causing an incorrect segmentation of those parts. A custom threshold set to 250 prevents such behavior and outputs correct region borders. The results shown here are obtained by setting $N = 4$, $w = 2$, $\rho = 0.4$ and $T_A = 40$.

Blobs removal parameter N. The parameter N sets the size of blobs to be discarded as noise. Its optimal value depends, of course, on the noise present in the document as well as on its resolution. Kise et al. [8] vary this parameter across their experiments without setting a clear criterion for its selection. While this parameter can be irrelevant for noiseless documents such as most modern digital documents, it might be crucial to achieve a satisfactory result in noisy ones. Figure 5 shows such an example. If no denoising is applied, the final segmentation presents several document parts that are semantically meaningless to the document layout. This is because each blob is considered a generator for the Area Voronoi Diagram construction. When setting $N = 2$, this phenomenon is mitigated, but $N = 2$ is not enough to successfully remove all the noise. Setting $N = 3$ indeed produces the desired output: blobs due to noise are correctly removed and the final segmentation actually reflects the document layout. However, there is an undesirable side effect: characters (or part of them) may also be removed. This becomes more evident when setting $N = 13$. In this case, entire parts of the text are deleted and this causes an incorrect segmentation of the document parts.

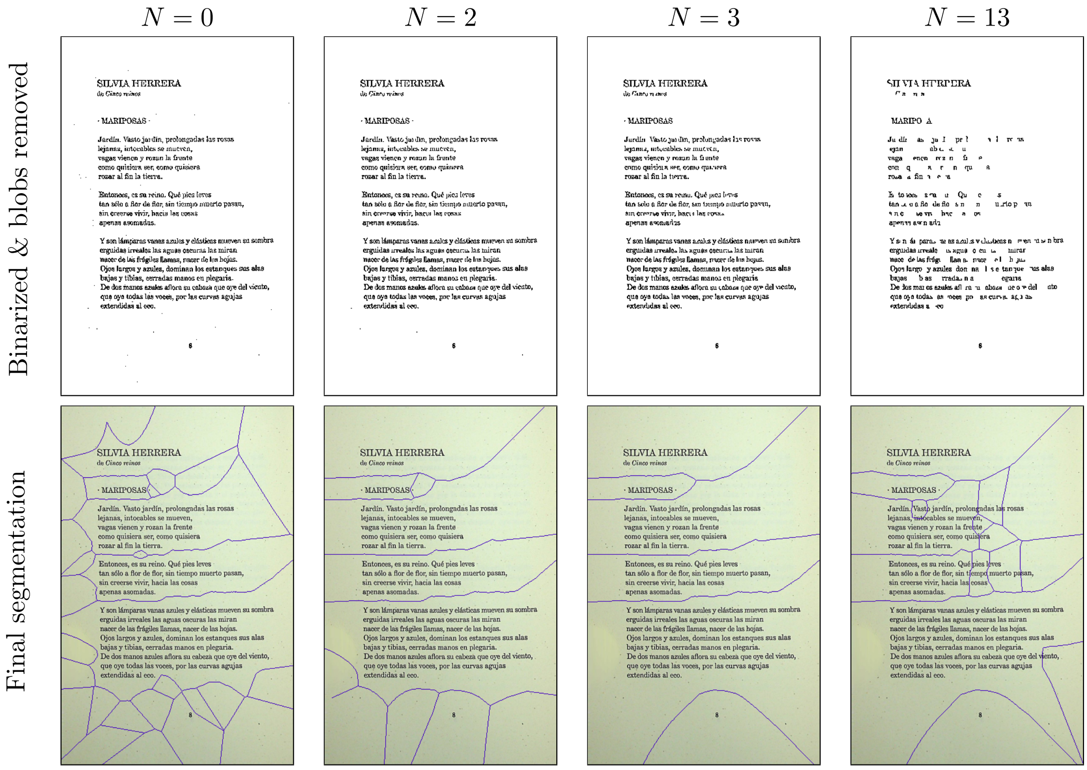


Figure 5: Comparison between different values for the blobs parameter N. When no denoising is applied ($N = 0$) or when it is insufficient, the final segmentation generates document parts that are meaningless for the document layout. Setting $N = 3$ produces the correct output. An excessive denoising ($N = 13$) can produce the loss of parts of the text and artifacts in the final segmentation. The results shown here can be replicated by setting *Otsu's thresholding* as binarization mode, $w = 2$, $\rho = 1$ and $T_A = 40$.

While tuning the parameter N can be effective in many cases, there exist degraded documents for which no value yields a satisfactory result. Figure 6 illustrates such an edge case, where the document suffers from two simultaneous issues: the presence of large blobs of noise, and the fragmentation of characters due to insufficient ink or fading. In this scenario, setting N too low results in the inclusion of noisy regions as valid layout elements, while increasing N inevitably removes legitimate text components that appear small due to degradation. As a consequence, segmentation either fails to discard noisy areas or disrupts the structure of textual elements. This example highlights the limitations of blob-size filtering alone and suggests that more robust noise handling strategies may be necessary for highly degraded documents.

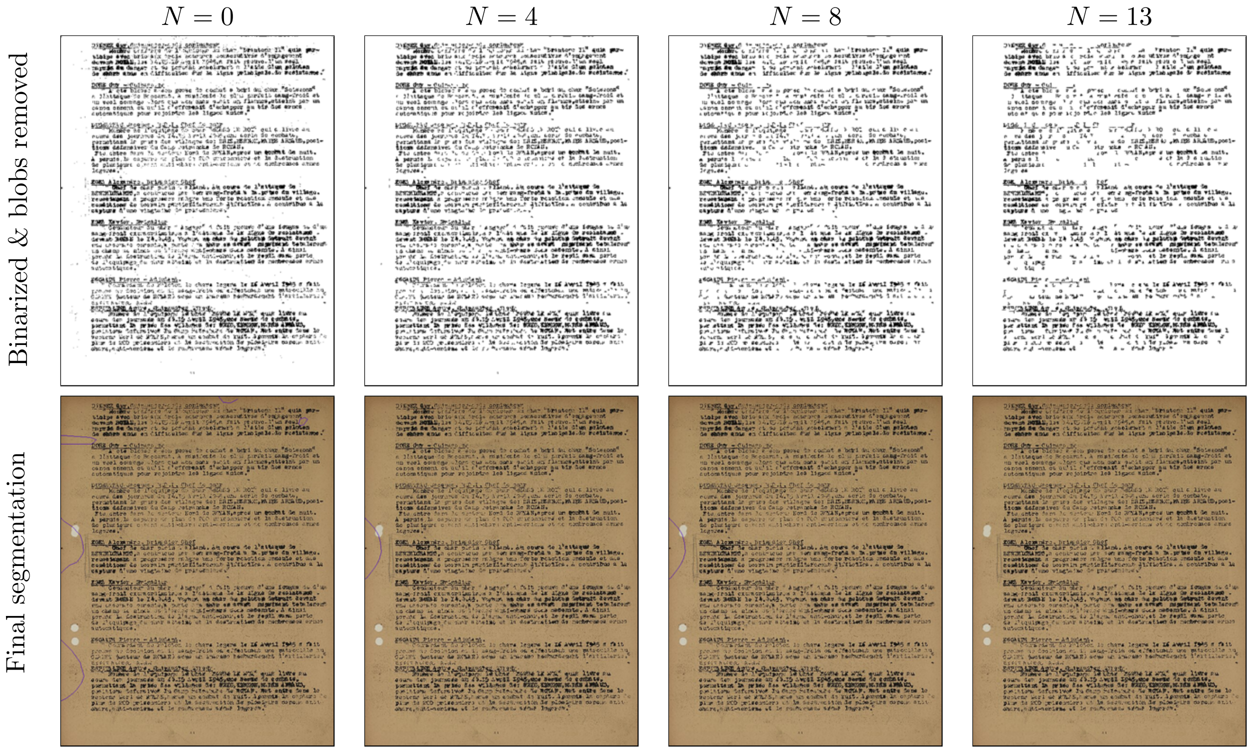


Figure 6: Limitations of blob-size filtering in a severely degraded document. The input document contains both large blobs of noise and fragmented characters due to ink loss. Varying the parameter N demonstrates the trade-off: small values fail to remove noisy regions, while larger values eliminate valid character fragments. No value of N yields a clean segmentation, illustrating the need for more sophisticated denoising strategies in such cases. The results shown here can be replicated by setting *Otsu's thresholding* as binarization mode, $w = 2$, $\rho = 0.1$ and $T_A = 40$.

Smoothing window size w. A key element in the success of the method [8] is the determination of the thresholds T_1 and T_2, which are derived from the two largest peaks in the histogram of distances between the components, as described in Section 3.6. The rationale behind the determination of T_1 and T_2 from the two largest peaks is that these peaks correspond to the centers of the two main modes typically observed in these histograms: one for distances between letters, and another for distances between lines.

However, the distance between two components, be it within a word or between different lines, is actually a noisy estimator of the inter-character or inter-line distances. Indeed, as characters have different shapes, this distance is affected by the particular characters involved in the measurement. Features like tall letters (ascenders), letters that go below the line (descenders), and capital letters especially affect the measurement of line spacing. Additionally, the distance between two components is measured using the generator points rather than the component itself. Therefore, the subsampling parameter ρ also adds noise to the histogram.

To reduce this noise and make the estimates clearer, the original method proposes smoothing the histogram using an average filter over a symmetric window of size $2 \times w + 1$. The size of this window, given by w, is left as a parameter of the method. In Figure 7 we illustrate the effect of this smoothing. When no smoothing is applied ($w = 0$), the two highest peaks are very close and seem to be part of the same noisy mode. Presumably, both represent the inter-character distance as, in the final segmentation, words are separated into different cells. On the other hand, when smoothing is applied ($w = 2$ or $w = 4$), these two peaks are merged into one and the second highest peak in this case seems to accurately correspond to the inter-line distance. The final segmentation seems to corroborate this, as paragraphs are correctly identified as belonging to the same region. It is worth mentioning that the main difference between setting $w = 2$ and $w = 4$ is that while the peaks are

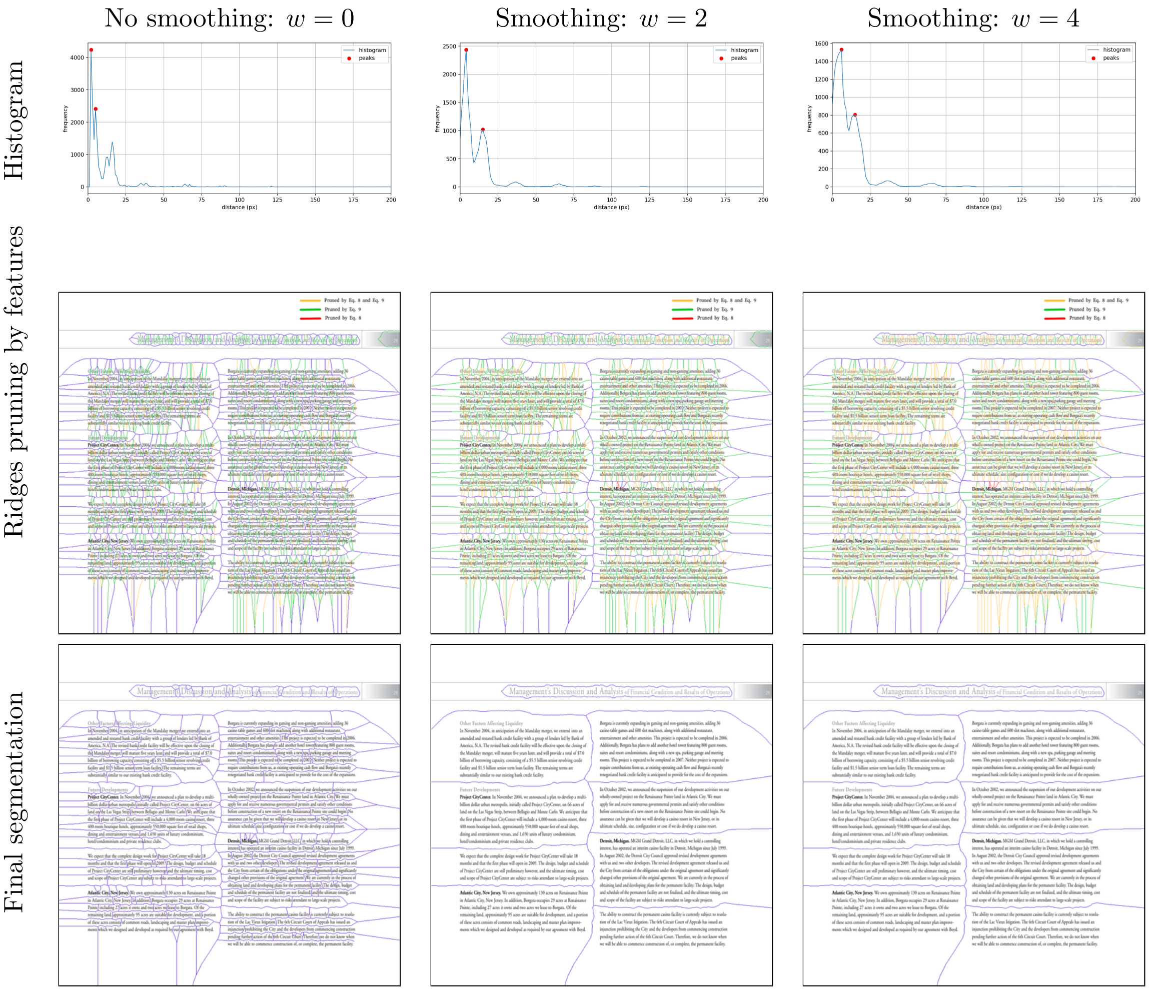


Figure 7: Effect of the smoothing window size w on the distances histogram, the pruning by features step, and the final segmentation. When no smoothing is applied ($w = 0$), the two largest peaks seem to correspond to the same noisy mode, resulting in an erroneous segmentation. On the other hand, when smoothing is applied, these two peaks are merged into one and the second highest peak in this case seems to correspond to the inter-line distance. In this case, the final segmentation accurately represents the document layout. The results shown here can be replicated by setting the binarization threshold to 200, $N = 4$, $\rho = 0.3$ and $T_A = 40$.

still visible with both window sizes, their distance, relative heights, and sharpness change. Larger values of w shall result in the merging of these two modes.

Subsampling parameter ρ. Subsampling was introduced in the original paper [8] in order to reduce the computation time involved in computing the *exact* Area Voronoi Diagram. Our implementation keeps a random fraction ρ of the border pixels, which is set by default to $\rho = 0.1$. The maximum value $\rho = 1$ keeps all border pixels, resulting in the exact Area Voronoi Diagram.

As shown in Figure 8, and confirmed in many experiments, for moderate to high resolution images (e.g., above 150 DPI) the approximation obtained with the default value does not introduce artifacts or errors in the resulting segmentation and reduces computational time. On the other hand, for low resolution images (say, below 150 DPI), the default value may result in all border points of an entire component being skipped (in the example, the letter “t”), often degrading the resulting segmentation. At the same time, with modern computational resources, small images are quick to process even with the maximum value $\rho = 1$. Therefore, our recommendation is to use $\rho = 0.1$ unless the input image is small, in which case a larger value is recommended.

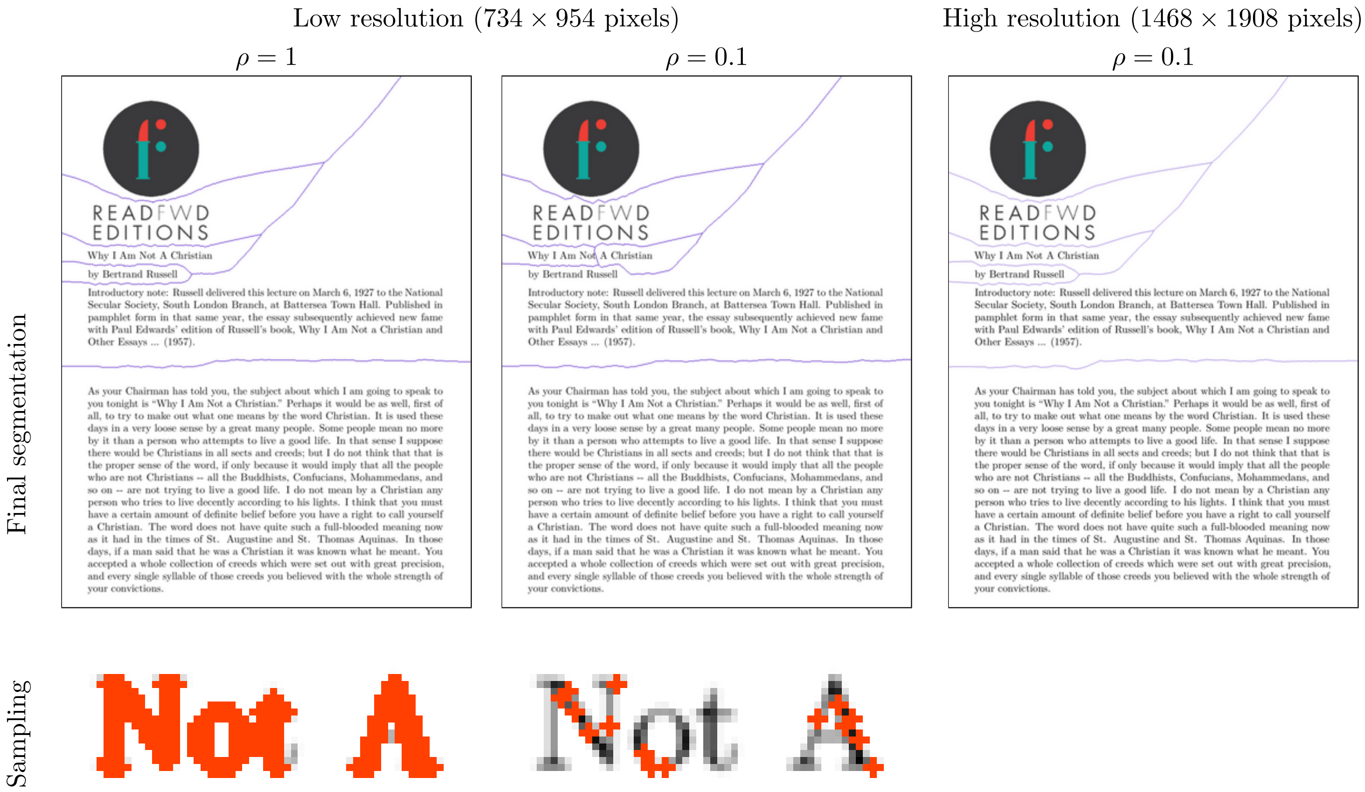


Figure 8: Effect of subsampling parameter ρ. The first row shows the results obtained on the same page for different resolutions and values of ρ. From left to right: low resolution with $\rho = 1$, low resolution with $\rho = 0.1$ and high resolution with $\rho = 0.1$. There is an error in the segmentation in the second case (middle image), with the title of the book "Why I Am Not A Christian" divided between "Not" and "A". In the second row we show a zoom of that case where the source of the error can be clearly attributed to the subsampling process: as the letter "t" was completely skipped, the space between the "o" and the "A" surpasses the threshold and therefore it was considered a significant edge. This does not happen when using the full border, as shown on the bottom right image.

Area threshold T_A. The area threshold has a significant role in Equation (9). Still, setting it is not straightforward: it should be permissive enough not to segment characters from the same font and size, even when their areas differ substantially, while still being discriminative enough to distinguish slight differences in the font size. Kise et al. fixed this threshold to 40 once and for all. In our implementation, we allow users to modify this value according to their particular needs. In Figure 9 we show an example of how modifying this threshold can impact the output of the method. Indeed, when setting T_A to 40, the subtitle "Takeoff and Climb" is not segmented separately from the body text because it lies close to the text and the font size is not significantly larger. Lowering the threshold to 30 results in a better segmentation as the method is better able to discriminate characters having different font sizes.

6.2 Analysis of Sample Cases

In this section, we present a series of examples obtained using the analyzed method to illustrate its performance in various scenarios. In each case, we aim to provide a comprehensive understanding of the method's performance and its strengths and weaknesses. All the examples shown here can be replicated by setting *Otsu's thresholding* as binarization mode, $N = 4$, $w = 2$ and $\rho = 1$.

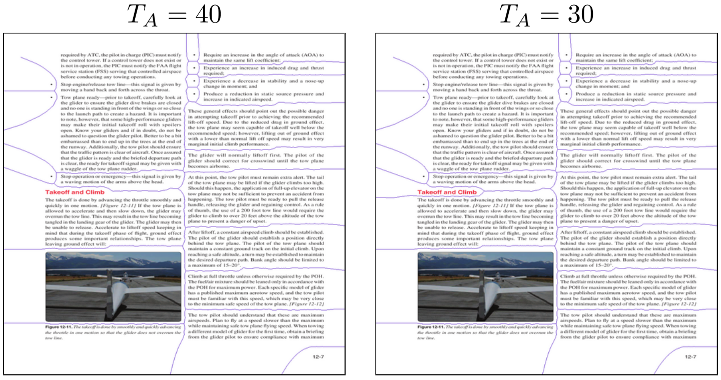


Figure 9: Comparison between different area thresholds (T_A). When setting T_A to 40, as done in the original article, the subtitle "Takeoff and Climb" is not segmented separately as it lies close to the text and the font size is not significantly larger. Setting an area threshold of 30 produces the correct segmentation, as the method becomes more discriminative. The results shown here can be replicated by setting *Otsu's thresholding* as binarization mode, $N = 4$, $w = 2$ and $\rho = 1$.

Simple layouts. Basic layout structures should represent the least challenging scenario for the method. Still, even in these cases, some mistakes can happen. Figure 10 shows the segmentation obtained for three examples. We observe that the method is able to correctly handle multiple columns, as in the first and second document. Still, it is worth mentioning that, when paragraphs are determined by indentation (first document), the method is not able to segment them. On the other hand, when spacing between paragraphs is used (second document), paragraphs are segmented separately. The second document contains a list where inter-line spacing is bigger than in the rest of the document. Note that the method is not consistent in handling such a structure, as some lines are merged together while others are not. This depends on the spacing between ascenders and descenders in each line. Finally, the third example depicts a limitation of the method: text having bigger inter-word spacing than the predominant spacing is likely to be over-segmented.

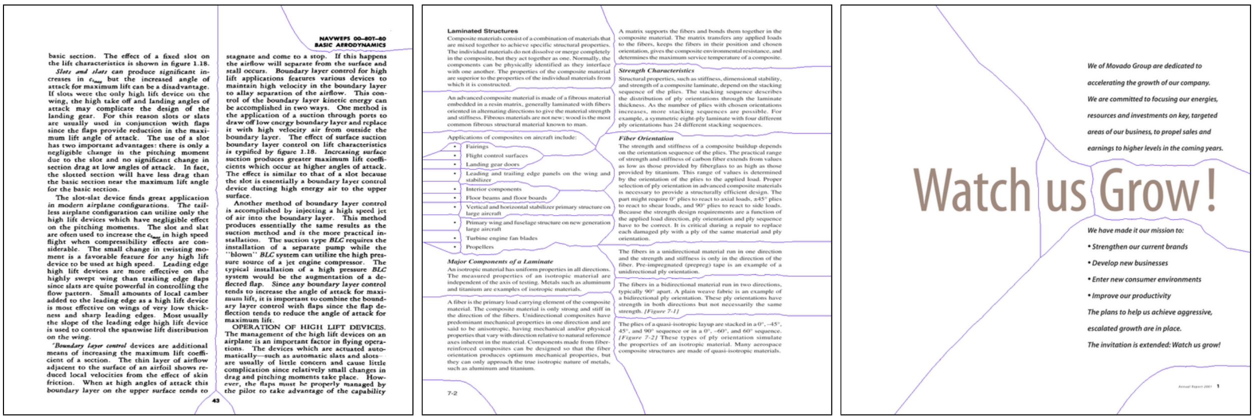


Figure 10: Results on simple layouts. The method is able to correctly handle multi-column text as well as paragraph separation through spacing. However, document parts with different inter-line or inter-word spacing than in the main text can cause errors in the segmentation.

Tables. Tables are versatile layout elements widely used in documents. Naturally, their design can vary depending on the purpose and style of the document: the use of grid-lines, alignment and spacing shall be different in each case. Figure 11 depicts the segmentation obtained in two examples. Simple tables, such as the one in the first document, are correctly handled by the method as multi-column text. However, complex tables, such as the one in the second example, are poorly handled. In this case, grid-lines spanning over multiple columns cause text in different columns to be merged into a single cell. On the other hand, extra spacing in the table causes over-segmentation within a single column.

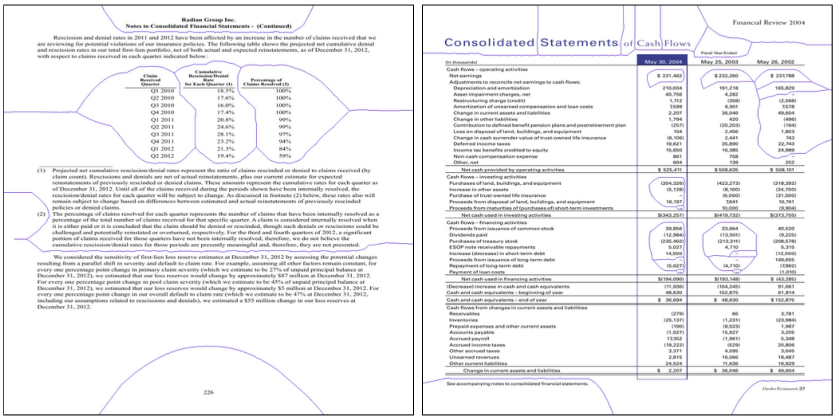

Radian Group Inc.
Notes to Consolidated Financial Statements - (Continued)
Consolidated Statements of Cash Flows
Financial Review 2004

Figure 11: Results on documents containing tables. Simple tables are correctly handled as multi-column text. More complex ones generate an inconsistent segmentation: text in different columns is merged into a single cell while text in the same column is sometimes segmented.

Light characters on dark backgrounds. The binarization step in the current implementation assumes that background pixels are lighter than foreground ones. Therefore, whenever text is written in lighter colors than the background, the segmentation shall be unsatisfactory. This situation is depicted in Figure 12. In the first example, simply inverting the colors would be enough to avoid this problem. However, the other two examples would require a local approach, as in those cases dark and light fonts as well as background colors are intermixed throughout the document. We advise users to handle these situations before submitting the image to the demo.

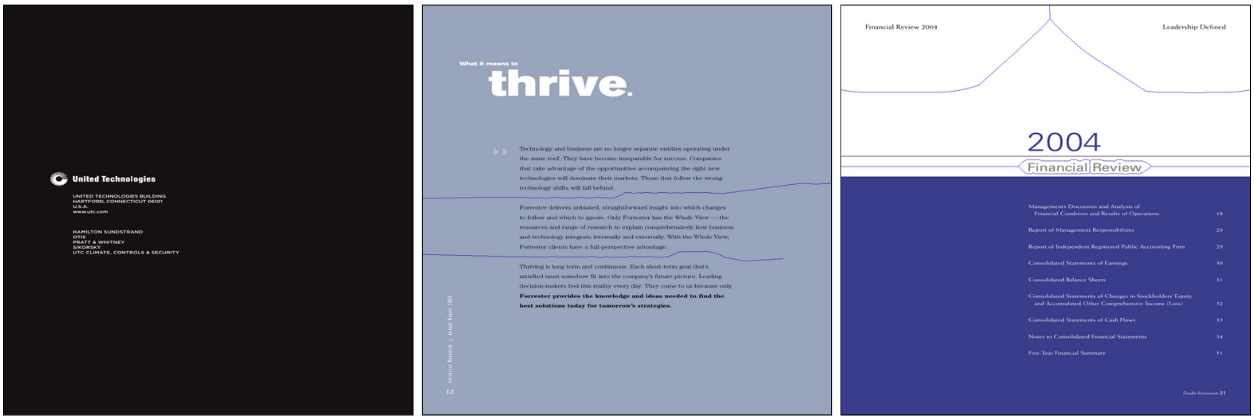
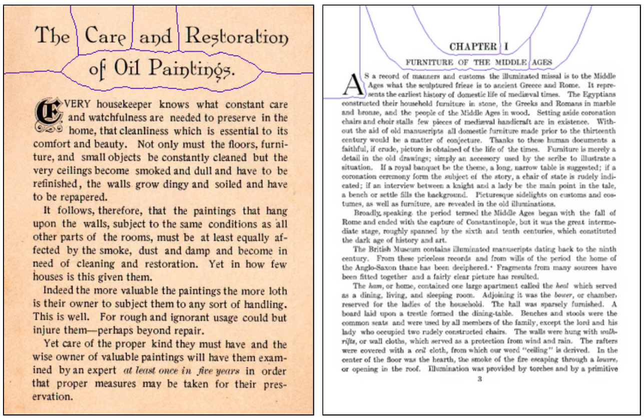


Figure 12: Results on documents with dark backgrounds and light-colored text. The binarization step in the current implementation assumes that background pixels are lighter than foreground ones. Therefore, the segmentation results in these examples are unsatisfactory.

Drop-caps. Drop-caps are large, decorative capital letters used at the beginning of a section or paragraph in a document. In terms of semantics, drop-caps are part of the said section or paragraph. However, their style does not match the rest of the body-text. Handling drop-caps is ambiguous and defining whether they should be segmented separately or not depends on downstream applications. Figure 13 presents the segmentation obtained for two examples having drop-caps. We observe that the method is not consistent in its handling of these elements and that the final segmentation depends on each case. If drop-caps lie very close to the text and their area falls within the limits imposed by Equation (9), they are likely to be merged with the text body, as in the first example. On the other hand, when these criteria are not met, either because they are too large or lie too far from the text (or a combination of both), drop-caps are segmented apart, as in the second example.



The Care and Restoration of Oil Paintings.

EVERY housekeeper knows what constant care and watchfulness are needed to preserve in the home, that cleanliness which is essential to its comfort and beauty. Not only must the floors, furniture, and small objects be constantly cleaned but the very ceilings become smoked and dull and have to be refinished, the walls grow dingy and soiled and have to be repapered.

It follows, therefore, that the paintings that hang upon the walls, subject to the same conditions as all other parts of the rooms, must be at least equally affected by the smoke, dust and damp and become in need of cleaning and restoration. Yet in how few houses is this given them.

Indeed the more valuable the paintings the more loth is their owner to subject them to any sort of handling. This is well. For rough and ignorant usage could but injure them—perhaps beyond repair.

Yet care of the proper kind they must have and the wise owner of valuable paintings will have them examined by an expert *at least once in five years* in order that proper measures may be taken for their preservation.

CHAPTER I

FURNITURE OF THE MIDDLE AGES

AS a record of manners and customs the illuminated missal is to the Middle Ages what the sculptured frieze is to ancient Greece and Rome. It represents the earliest history of domestic life of mediæval times. The Egyptians constructed their household furniture in stone, the Greeks and Romans in marble and bronze, and the people of the Middle Ages in wood. Setting aside coronation chairs and choir stalls few pieces of mediæval handicraft are in existence. Without the aid of old manuscripts all domestic furniture made prior to the thirteenth century would be a matter of conjecture. Thanks to these human documents a faithful, if crude, picture is obtained of the life of the times. Furniture is merely a detail in the old drawings; simply an accessory used by the scribe to illustrate a situation. If a royal banquet be the theme, a long, narrow table is suggested; if a coronation ceremony form the subject of the story, a chair of state is rudely indicated; if an interview between a knight and a lady be the main point in the tale, a bench or settle fills the background. Picturesque sidelights on customs and costumes, as well as furniture, are revealed in the old illuminations.

Broadly speaking the period termed the Middle Ages began with the fall of Rome and ended with the capture of Constantinople, but it was the great intermediate stage, roughly spanned by the sixth and tenth centuries, which constituted the dark age of history and art.

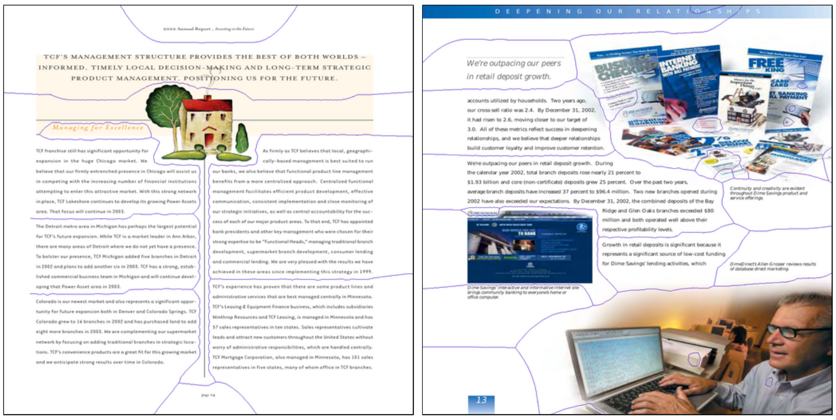
The British Museum contains illuminated manuscripts dating back to the ninth century. From these priceless records and from wills of the period the home of the Anglo-Saxon thane has been deciphered.* Fragments from many sources have been fitted together and a fairly clear picture has resulted.

The *ham*, or home, contained one large apartment called the *heal* which served as a dining, living, and sleeping room. Adjoining it was the *bower*, or chamber, reserved for the ladies of the household. The hall was sparsely furnished. A board laid upon a trestle formed the dining-table. Benches and stools were the common seats and were used by all members of the family, except the lord and his lady who occupied two rudely constructed chairs. The walls were hung with *walh-rifts*, or wall cloths, which served as a protection from wind and rain. The rafters were covered with a *ceil* cloth, from which our word "ceiling" is derived. In the center of the floor was the hearth, the smoke of the fire escaping through a *louvre*, or opening in the roof. Illumination was provided by torches and by a primitive

3

Figure 13: Results on documents with drop-caps. The segmentation depends on each particular case, mostly on the distance between the drop-cap and the main text as well as the relative area with respect to the neighboring characters.

Non-Manhattan layouts. Contrary to most of the document layout analysis techniques in the literature [9, 3, 2, 5, 14, 15], handling non-Manhattan layouts involving irregular boundaries is one of the main strengths of the method. Figure 14 illustrates the method's ability to segment such layouts. In both examples, the irregular boundaries of the text surrounding the figures are correctly managed. Still, note that in the second example, there is an over-segmentation of the figure in the top-right of the document due to binarization artifacts.



2002 Annual Report _ *Investing in the Future*

TCF'S MANAGEMENT STRUCTURE PROVIDES THE BEST OF BOTH WORLDS – INFORMED, TIMELY LOCAL DECISION-MAKING AND LONG-TERM STRATEGIC PRODUCT MANAGEMENT, POSITIONING US FOR THE FUTURE.

Managing for Excellence

TCF franchise still has significant opportunity for expansion in the huge Chicago market. We believe that our firmly entrenched presence in Chicago will assist us in competing with the increasing number of financial institutions attempting to enter this attractive market. With this strong network in place, TCF Lakeshore continues to develop its growing Power Assets area. That focus will continue in 2003.

The Detroit metro area in Michigan has perhaps the largest potential for TCF's future expansion. While TCF is a market leader in Ann Arbor, there are many areas of Detroit where we do not yet have a presence. To bolster our presence, TCF Michigan added five branches in Detroit in 2002 and plans to add another six in 2003. TCF has a strong, established commercial business team in Michigan and will continue developing that Power Asset area in 2003.

Colorado is our newest market and also represents a significant opportunity for future expansion both in Denver and Colorado Springs. TCF Colorado grew to 16 branches in 2002 and has purchased land to add eight more branches in 2003. We are complementing our supermarket network by focusing on adding traditional branches in strategic locations. TCF's convenience products are a great fit for this growing market and we anticipate strong results over time in Colorado.

As firmly as TCF believes that local, geographically-based management is best suited to run our banks, we also believe that functional product line management benefits from a more centralized approach. Centralized functional management facilitates efficient product development, effective communication, consistent implementation and close monitoring of our strategic initiatives, as well as central accountability for the success of each of our major product areas. To that end, TCF has appointed bank presidents and other key management who were chosen for their strong expertise to be "Functional Heads," managing traditional branch development, supermarket branch development, consumer lending and commercial lending. We are very pleased with the results we have achieved in these areas since implementing this strategy in 1999.

TCF's experience has proven that there are some product lines and administrative services that are best managed centrally in Minnesota. TCF's Leasing & Equipment Finance business, which includes subsidiaries Winthrop Resources and TCF Leasing, is managed in Minnesota and has 57 sales representatives in ten states. Sales representatives cultivate leads and attract new customers throughout the United States without worry of administrative responsibilities, which are handled centrally. TCF Mortgage Corporation, also managed in Minnesota, has 101 sales representatives in five states, many of whom office in TCF branches.

page 14

DEEPENING OUR RELATIONSHIPS

We're outpacing our peers in retail deposit growth.

accounts utilized by households. Two years ago, our cross-sell ratio was 2.4. By December 31, 2002, it had risen to 2.6, moving closer to our target of 3.0. All of these metrics reflect success in deepening relationships, and we believe that deeper relationships build customer loyalty and improve customer retention.

We're outpacing our peers in retail deposit growth. During the calendar year 2002, total branch deposits rose nearly 21 percent to $1.93 billion and core (non-certificate) deposits grew 25 percent. Over the past two years, average branch deposits have increased 37 percent to $96.4 million. Two new branches opened during 2002 have also exceeded our expectations. By December 31, 2002, the combined deposits of the Bay Ridge and Glen Oaks branches exceeded $80 million and both operated well above their respective profitability levels.

Growth in retail deposits is significant because it represents a significant source of low-cost funding for Dime Savings' lending activities, which

Continuity and creativity are evident throughout Dime Savings product and service offerings.

Dime Savings' interactive and informative Internet site brings community banking to everyone's home or office computer.

DimeDirect's Allen Grosser reviews results of database direct marketing.

13

Figure 14: Results on non-Manhattan documents. In both examples, the irregular boundaries of the text surrounding the figures are correctly managed.

Overlapping layouts. The method is not designed to work on documents where overlapping elements are present. This is, indeed, the most challenging scenario for document layout analysis. When analyzing such documents, two situations can be encountered. First, the overlapping elements can be taken as part of the background when binarizing the document; consequently, the resulting segmentation is the same as if those elements did not exist. Alternatively, the overlapping elements can be treated as a single element spanning the union of the intersecting parts. Figure 15 depicts these two situations: in the first case, the watermark is considered as part of the background, while in the second, the seal is merged into a single element together with the parts it intersects.

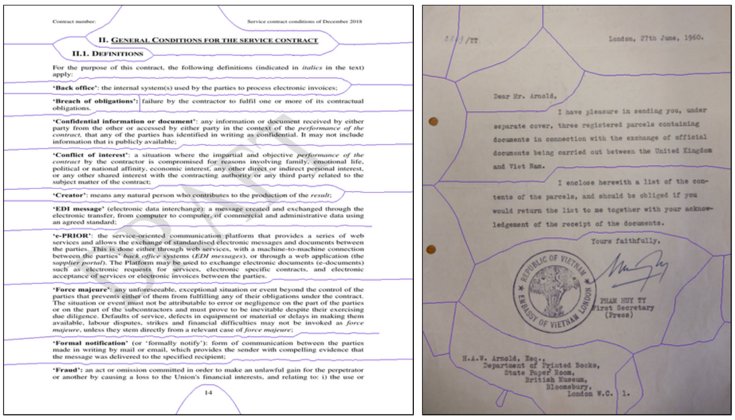


Figure 15: Results on documents with overlapping elements. In the first case, the watermark is considered as part of the background when binarizing the image. Therefore, the segmentation is the same as if it did not exist. In the second case, the seal is considered as a single element together with the other parts it intersects.

Non-Latin scripts. Though the method is not explicitly designed for Latin scripts, there are some implicit assumptions that call into question its ability to work on other scripts. It is assumed that characters are, in most cases, a single connected component and that the most frequent spacings are the inter-character spacing and the inter-line spacing. Furthermore, characters in the same font are assumed to have a similar area. This is not the case in every language.

For instance, Arabic or Persian scripts are cursive, meaning that the characters are often connected within a word, the spacing between words and characters is less uniform due to this cursive nature, and the use of diacritics is more prevalent. Three examples of documents in these languages are given in Figure 16. Here we observe that, even if the overall layout is correct, the method struggles to manage diacritics correctly.

Another language that challenges these hypotheses is Korean. Korean characters typically consist of complex, dense structures that are not necessarily connected. The fourth example in Figure 16 shows the result of the method on a Korean document. All in all, the result on this simple layout is satisfactory despite the language's complexities.

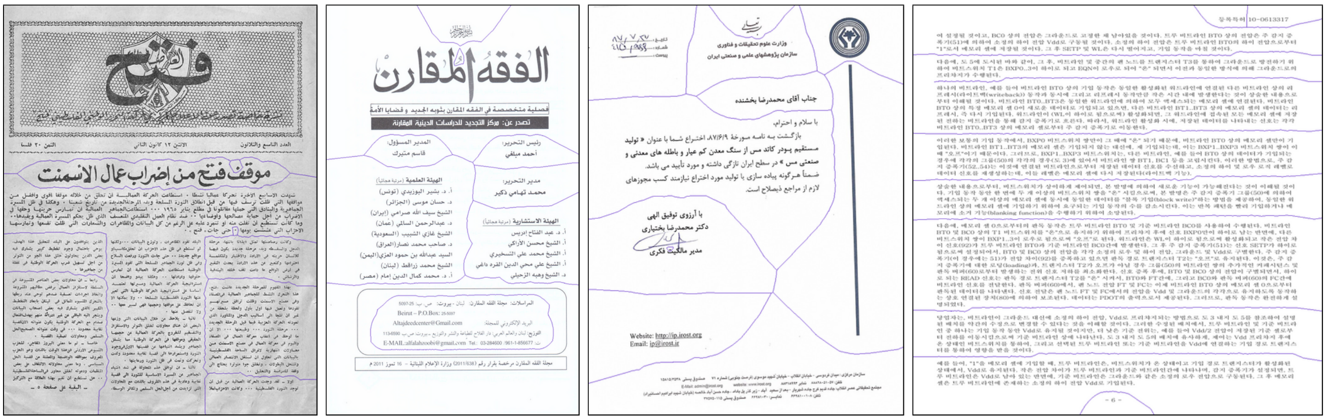
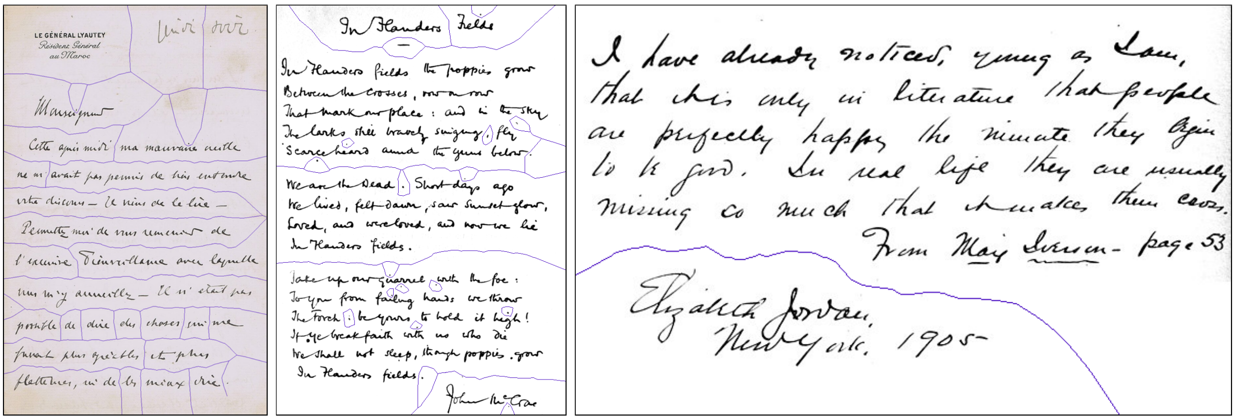


Figure 16: Results on documents written in different languages using other than Latin scripts. Note that, even if the results are overall good, the method struggles with the use of diacritics in Arabic and Persian.

Handwritten text. Handwritten text presents several unique challenges for document layout analysis that make it significantly more complex than printed text. Unlike printed text, where spacing between characters, words and lines is consistent, handwriting often features irregular spacing, making it more difficult to determine semantically meaningful thresholds T_1 and T_2. Furthermore, individual writing styles such as the use of cursive or ligatures between words and the spacing between punctuation marks and characters may affect the result. Figure 17 shows three examples of the results obtained by the method on handwritten text. In the first two examples we observe that some lines of text as well as some punctuation marks are segmented separately due to the mentioned non-uniformity throughout the text. Note that this problem could also arise in fully justified (not necessarily handwritten) text, as spacing is also non-uniform. On the other hand, the third example shows a satisfactory segmentation, reinforcing the idea that the performance of the method on handwritten documents depends on the specific writing style.



LE GÉNÉRAL LYAUTEY
Résident Général
au Maroc

Lundi soir

Monseigneur

Cette après midi ma mauvaise oreille ne m'avait pas permis de bien entendre votre discours — Je viens de le lire — Permettez moi de vous remercier de l'excessive bienveillance avec laquelle vous m'y accueillez — Il n'était pas possible de dire des choses qui me fussent plus agréables et plus flatteuses, ni de les mieux dire.

In Flanders Fields

—

In Flanders fields the poppies grow
Between the crosses, row on row
That mark our place: and in the sky
The larks still bravely singing, fly
Scarce heard amid the guns below.

We are the Dead. Short days ago
We lived, felt dawn, saw sunset glow,
Loved, and were loved, and now we lie
In Flanders fields.

Take up our quarrel with the foe:
To you from failing hands we throw
The Torch; be yours to hold it high!
If ye break faith with us who die
We shall not sleep, though poppies grow
In Flanders fields.

John McCrae

I have already noticed, young as I am, that it is only in literature that people are perfectly happy the minute they begin to be good. In real life they are usually missing so much that it makes them cross.

From May Iverson - page 53

Elizabeth Jordan,
New York, 1905

Figure 17: Results on handwritten documents. The performance depends on the particular writing style: while in the first two cases the non-uniform spacing causes a wrong segmentation, in the third example the method achieves a satisfactory result.

7 Conclusions

In this work, we presented an efficient and faithful implementation of the method proposed by Kise et al. [8], rigorously testing it across a diverse set of samples. We showed that the method has a reasonable performance in diverse scenarios and is able to handle some complex layouts. Still, we found some inconsistencies in how elements such as lists, drop-caps, and tables are handled, suggesting that the particular styling of each of these can affect the final result.

Furthermore, our experiments revealed a significant sensitivity of the method to parameter settings, with even small adjustments leading to notable changes in the output. Additionally, the method's performance is also affected by the binarization process, highlighting the importance of careful preprocessing even if this step is left unaddressed in the original article.

One key finding is that border sampling, which may seem like a minor detail, has a considerable impact on the final results. This indicates that seemingly peripheral choices in the implementation can alter the method's behavior substantially. Overall, our analysis shows that there is no single set of default parameters that consistently works well across all cases. Effective use of the method requires a context-dependent tuning of parameters.

Acknowledgments

Work supported by Bpifrance, CollabNext project.

Image Credits

By the authors.

DocLayNet dataset [11]. License: CDLA-Permissive-1.0.

Title: *Otro '45 : antología poética*. Publisher: Biblioteca Nacional (Montevideo). License: CC BY 2.0[4].

Wikimedia Commons. Author: Leclerc (Philippe François Marie Leclerc de Hauteclocque, dit) (1902-11-22 - 1947-11-28). License: Creative Commons CC0 1.0 Universal Public Domain Dedication[5].

Wikimedia Commons. Author: Political Party Fatah. License: Public Domain[6].

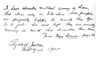
Wikimedia Commons. Author: Shirali110. License: CC-BY-SA-4.0[7].

Wikimedia Commons. Author: Zarbarg. License: CC-BY-SA-4.0[8].

Wikimedia Commons. Author: Hubert Lyautey (letter) - Jean-François Miniac (photograph). License: CC BY-SA 3.0[9].

Wikimedia Commons. Author: Oliver Brothers. License: Public Domain[10].

Wikimedia Commons. Author: Virginia Robie. License: Public Domain[11].

 Wikimedia Commons. Author: Charlotte Harding. License: Public Domain[12].

Wikimedia Commons. Author: John McCrae. License: Public Domain[13].

References

[1] F. Aurenhammer, *Voronoi Diagrams-A Survey of a Fundamental Geometric Data Structure*, ACM Computing Surveys, 23 (1991), p. 345–405, https://doi.org/10.1145/116873.116880.

[2] A. Banerjee, S. Biswas, J. Lladós, and U. Pal, *SwinDocSegmenter: An End-To-End Unified Domain Adaptive Transformer For Document Instance Segmentation*, in International Conference on Document Analysis and Recognition (ICDAR), 2023, pp. 307–325, https://doi.org/10.1007/978-3-031-41676-7_18.

[4] https://www.bibna.gub.uy/?artwork=otro-45

[5] https://commons.wikimedia.org/wiki/File:Ordre_g%C3%A9n%C3%A9ral_n%C2%B0_88_du_21_juin_1945_-_Citations_%C3%A0_l%27Ordre_de_BrigadeR%C3%A9giment,_Division,_Corps_d%27Arm%C3%A9e_(Alsace_-_Allem,_2002.2218(316).jpg

[6] https://commons.wikimedia.org/wiki/File:Fatah_39_January_1969.jpg

[7] https://commons.wikimedia.org/wiki/File:Alfeqhalmuqaranp1.jpg

[8] https://commons.wikimedia.org/wiki/Category:Documents_in_Persian#/media/File:IROST-MSRT.jpg

[9] https://commons.wikimedia.org/wiki/Category:Handwriting_in_French#/media/File:Courrier_d%27Hubert_Lyautey_%C3%A0_Louis_Duchesne,_au_soir_de_la_r%C3%A9ception_de_Lyautey_%C3%A0_l%27Acad%C3%A9mie_fran%C3%A7aise_en_1920,_collection_Marie-Anne_Miniac,_Rennes,_France..jpg

[10] https://commons.wikimedia.org/wiki/File:Old-brochure-page-one.jpg

[11] https://commons.wikimedia.org/wiki/File:Historic_styles_in_furniture_(IA_cu31924063909075).pdf

[12] https://commons.wikimedia.org/wiki/File:Handwritten_author%27s_letter--May_Iverson,_her_story.jpg

[13] https://commons.wikimedia.org/wiki/File:In_Flanders_fields_and_other_poems,_handwritten.png

[3] A. Banerjee, S. Biswas, J. Lladós, and U. Pal, *SemiDocSeg: Harnessing Semi-Supervised Learning for Document Layout Analysis*, International Journal on Document Analysis and Recognition, 27 (2024), p. 317–334, https://doi.org/10.1007/s10032-024-00473-y.

[4] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma, *VIPS: a Vision-Based Page Segmentation Algorithm*, tech. report, Microsoft Research, 01 2003. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-2003-79.pdf.

[5] C. Da, C. Luo, Q. Zheng, and C. Yao, *Vision Grid Transformer for Document Layout Analysis*, in IEEE/CVF International Conference on Computer Vision (ICCV), 2023, pp. 19405–19415, https://doi.org/10.1109/ICCV51070.2023.01783.

[6] J. Ha, R. Haralick, and I. Phillips, *Recursive X-Y Cut Using Bounding Boxes of Connected Components*, in International Conference on Document Analysis and Recognition (ICDAR), vol. 2, 1995, pp. 952–955 vol.2, https://doi.org/10.1109/ICDAR.1995.602059.

[7] Y. Ishiyama, C. Funaoka, F. Kubo, H. Takahashi, and F. Tomita, *Labeling Board Based on Boundary Tracking*, in IAPR International Conference on Pattern Recognition. Vol. IV. Conference D: Architectures for Vision and Pattern Recognition,, 1992, pp. 34–38, https://doi.org/10.1109/ICPR.1992.202125.

[8] K. Kise, A. Sato, and M. Iwata, *Segmentation of Page Images Using the Area Voronoi Diagram*, Computer Vision and Image Understanding, 70 (1998), pp. 370–382, https://doi.org/10.1006/cviu.1998.0684.

[9] L. O'Gorman, *The Document Spectrum for Page Layout Analysis*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 15 (1993), pp. 1162–1173, https://doi.org/10.1109/34.244677.

[10] N. Otsu, *A Threshold Selection Method from Gray-Level Histograms*, IEEE Transactions on Systems, Man, and Cybernetics, 9 (1979), pp. 62–66, https://doi.org/10.1109/TSMC.1979.4310076.

[11] B. Pfitzmann, C. Auer, M. Dolfi, A. S. Nassar, and P. W. J. Staar, *DocLayNet: A Large Human-Annotated Dataset for Document-Layout Segmentation*, ACM SIGKDD Conference on Knowledge Discovery and Data Mining, (2022), p. 3743–3751, https://doi.org/10.1145/3534678.3539043.

[12] J. Y. Ramel, S. Leriche, M. L. Demonet, and S. Busson, *User-Driven Page Layout Analysis of Historical Printed Books*, International Journal on Document Analysis and Recognition, 9 (2007), p. 243–261, https://doi.org/10.1007/s10032-007-0040-6.

[13] T. Shehzadi, D. Stricker, and M. Z. Afzal, *A Hybrid Approach for Document Layout Analysis in Document Images*, ArXiv, (2024), https://doi.org/10.48550/arXiv.2404.17888.

[14] Z. Shen, R. Zhang, M. Dell, B. C. G. Lee, J. Carlson, and W. Li, *LayoutParser: A Unified Toolkit for Deep Learning Based Document Image Analysis*, in International Conference on Document Analysis and Recognition (ICDAR), 2021, pp. 131–146, https://doi.org/10.1007/978-3-030-86549-8_9.

[15] Z. Tang, Z. Yang, G. Wang, Y. Fang, Y. Liu, C. Zhu, M. Zeng, C. Zhang, and M. Bansal, *Unifying Vision, Text, and Layout for Universal Document Processing*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023, pp. 19254–19264, https://doi.org/10.1109/CVPR52729.2023.01845.

[16] X. Zhong, J. Tang, and A. Jimeno Yepes, *PubLayNet: Largest Dataset Ever for Document Layout Analysis*, in International Conference on Document Analysis and Recognition (ICDAR), 2019, pp. 1015–1022, https://doi.org/10.1109/ICDAR.2019.00166.