Layout Analysis and Content Classification in Digitized Books

Andrea Corbelli, Lorenzo Baraldi(✉), Fabrizio Balducci, Costantino Grana, and Rita Cucchiara

Dipartimento di Ingegneria "Enzo Ferrari",
Università degli Studi di Modena e Reggio Emilia,
Via Vivarelli 10, 41125 Modena, MO, Italy
{andrea.corbelli,lorenzo.baraldi,fabrizio.balducci,
costantino.grana,rita.cucchiara}@unimore.it

Abstract. Automatic layout analysis has proven to be extremely important in the process of digitization of large amounts of documents. In this paper we present a mixed approach to layout analysis, introducing a SVM-aided layout segmentation process and a classification process based on local and geometrical features. The final output of the automatic analysis algorithm is a complete and structured annotation in JSON format, containing the digitalized text as well as all the references to the illustrations of the input page, and which can be used by visualization interfaces as well as annotation interfaces. We evaluate our algorithm on a large dataset built upon the first volume of the "Enciclopedia Treccani".

Keywords: Layout analysis · Content classification · SVM · Annotation interfaces

1 Introduction

Document digitization plays a key role in the preservation and diffusion of historical books: digital archives, indeed, protect fragile and valuable originals from handling, while still presenting their content to a vastly increased audience. Just like multimodal digital libraries need to be properly organized via computer vision and multimedia algorithms [4,5], the simple digital copy of an archive is often not sufficient to present its content in an effective and enjoyable way: beyond the application of Optical Character Recognition (OCR) methods, which are nowadays almost completely reliable, graphical elements, like tables, images and charts, should be automatically segmented and categorized.

Despite the recent advances in this field, layout and content analysis are still unsolved problems due to the high variability of the possible content. In this setting, we propose a novel pipeline for the analysis of structured documents, which includes a page layout analysis algorithm to segment the input document into coherent regions, and a content classification strategy to classify the actual content of each region. Our layout analysis builds upon the Recursive XY-Cut

M. Agosti et al. (Eds.): IRCDL 2016, CCIS 701, pp. 153–165, 2017.
DOI: 10.1007/978-3-319-56300-8_14

algorithm, and extends it with an SVM-aided detector for graphical elements; then, tables are identified by means of the Hough transform, and supervised machine learning techniques are employed in conjuction with local features to classify other graphical elements, like images, charts and scores. The final output is a structured annotation in JSON format, which can be read and modified by an annotation interface, useful for correcting mistakes in the automatic analysis, and by a visualization interface.

The rest of this paper is structured as follows: Section 2 gives a brief discussion of the state of the art in layout analysis, Sect. 3 explains the main components of our pipeline, and Sect. 4 reports the performance evaluation and a comparison against the state of the art.

2 Related Work

Layout analysis has been an active area of research since the seventies. There are two main approaches to layout analysis, *bottom up* and *top down*.

Top-down methods, such as XY cuts [6,13] or methods that exploit white streams [2] or projection profiles [11] are usually fast but tend to fail when dealing with complex layouts. Bottom-up methods are instead more flexible and process the image page from the pixel level and subsequently aggregate into higher level regions but with an higher computational complexity.

These approaches are usually based on mathematical morphology, Connected Components (CCs), Voronoi diagrams [15] or run-length smearing [22]. Many other methods exist which do not fit exactly into either of these categories: the so called mixed or hybrid approaches try to combine the high speed of the top-down approaches with the robustness of the bottom-up ones. Chen *et al.* [7] propose a method based on whitespace rectangles extraction and grouping: initially the foreground CCs are extracted and linked into chains according to their horizontal adjacency relationship; whitespace rectangles are then extracted from the gap between horizontally adjacent CCs; CCs and whitespaces are progressively grouped and filtered to form text lines and afterward text blocks. Lazzara *et al.* [16] provide a chain of steps to first recognize text regions and successively non-text elements. Foreground CCs are extracted, then delimiters (such as lines, whitespaces and tab-stop) are detected with object alignment and morphological algorithms. Since text components are usually well aligned, have a uniform size and are close to each other, the authors propose to regroup CCs by looking for their neighbors. Filters can also be applied on a group of CCs to validate the link between two CCs. Another algorithm based on whitespace analysis has been proposed by Baird *et al.* [3]: the algorithm uses the white space in the page as a layout delimiter and tries to find the biggest background empty rectangles to extract connected regions.

Kaur *et al.* [14] and Zanibbi *et al.* [24] present surveys about the approaches applied to table recognition. Mandal *et al.* dealt with the detection and segmentation of tables and formulas [18,19], and proposed a detector with the heuristic that each table has distinct columns which implies that gaps between the fields

are substantially larger than the gaps between the words in text lines; a similar approach was presented by Ferilli *et al.* in [9]. Liu *et al.* [17] deal with table boundary detection and content extraction considering the sparse-line property of table rows, while Bertrand and Lemaitre [8] focus on the recognition of tables and forms.

3 Our Proposal

3.1 Page Layout Segmentation

The first stage of our pipeline is the segmentation of the input page into coherent regions. Our layout analysis step builds upon the well known XY-Cut algorithm [13], and extends it to go beyond its limitations. XY-Cut is an iterative top-down page segmentation algorithm, which takes as input a Region of Interest (ROI) and segments it into rectangular regions which are separated by white spaces. In particular, the algorithm projects the pixels' values on the vertical and horizontal axes of the ROI, and then finds low density regions in the projection histograms, which corresponds to white spaces. If a low density point is found on at least one of the projections, the ROI is split into two subregions, which are then further analyzed (and possibly split) in the next iteration. The resulting segmentation is therefore composed by a set of rectangular regions, each of which is separated from the others by white space on all sides.

The XY-Cut algorithm works well with simple layouts, where elements are actually rectangular and separated by vertical or horizontal white spaces which span on all their sides. However, it often happens that illustrations have complex shapes, or are surrounded by text on more than one side, thus making the application of such a simple technique insufficient (see Fig. 1a for an example).

(a) XY-Cut (b) Detected illustrations (c) Final result

Fig. 1. The page layout segmentation pipeline. First the Recursive XY-Cut algorithm is applied to detect candidate regions inside the page; then, illustrations are detected using local autocorrelation features. A second application of the XY-Cut algorithm gives the final segmentation.

To complete the layout segmentation phase, we therefore apply an additional step whose aim is to detect illustrations (and their corresponding boundaries) inside the page, and which is inspired by the algorithm proposed in [12].

The core assumption of this step is that local autocorrelation statistics are sufficient to distinguish between text and illustration. The autocorrelation matrix of a region, indeed, is an effective feature for finding repeating patterns and is particularly suited in this case since textual textures have a pronounced orientation that heavily differs from that of illustrations. The original image is subdivided into square blocks of size $n \times n$, and for each block the autocorrelation matrix $C(k, l)$, with $k, l \in [-n/2, n/2]$, is computed. Then, the autocorrelation matrix is encoded into a directional histogram $w(\cdot)$, in which each bin contains the sum of the pixels along that direction. Formally,

$$w(\theta) = \sum_{r \in (0, n/2]} C(r \cos\theta, r \sin\theta) \tag{1}$$

We compute the directional histogram in the range $\theta \in [0, 180°]$, and quantize θ with a step of 1°, and r with a step of 1 pixel. The histogram is then concatenated with the vertical and horizontal projections of the autocorrelation matrix, to enhance the repeating pattern of the text lines. The resulting descriptor is fed to a two-class SVM classifier with RBF kernel, trained to distinguish between blocks of text and blocks of illustrations.

Given a region segmented by XY-Cut, we classify each block inside the region as text or illustration, and identify the boundaries of illustrations by finding the connected components created by illustration blocks. Once illustrations have been detected, XY-Cut is again applied on a temporary image where illustrations are removed. The final result of the layout segmentation process is the union of the illustration regions and of the regions found by the second execution of XY-Cut. Figure 1 gives an example of the overall process.

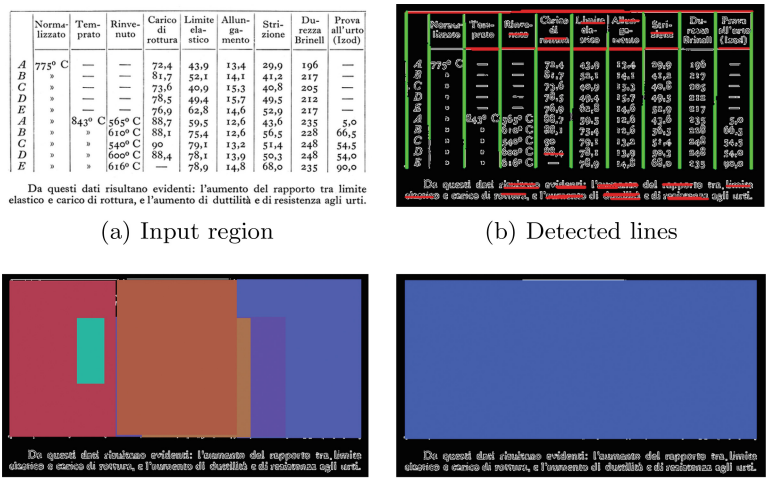
3.2 Table Detection

We developed a simple yet effective method for table detection which is based on the Hough transform [10] and heuristic rules. The Hough transform allows to automatically detect lines in an image, and it is therefore appropriate to detect structures like tables which contain vertical and horizontal lines.

The Hough transform finds objects within a certain class of shapes using a voting procedure, which is carried out in a parameter space from which object candidates are obtained as local maxima, in a so-called accumulator space that is explicitly constructed by the algorithm (Fig. 2).

A straight line in the image space can be expressed as:

- $y = mx + b$ in the Cartesian coordinate system, with the associated point (m, b) in the parameter space
- $r = x \cos\theta + y \sin\theta$ in the Polar coordinate system with the associated point (r, θ), where r is the distance from the origin to the closest point on the



(a) Input region (b) Detected lines

(c) Candidate tables (d) Final result

Fig. 2. The table detection pipeline.

straight line, and θ is the angle between the x axis and the line connecting the origin with that closest point.

The Polar coordinate system permits to overcome the problem of vertical lines, which would rise unbounded values of the slope parameter m in the Cartesian one. Given a single point, the set of all straight lines passing through that point corresponds to a sinusoidal curve in the (r, θ) plane, which is unique to that point. A set of two or more points that form a straight line will produce sinusoids which cross at (r, θ) for that line. In general, a line can be detected by finding the number of intersections between curves: the more curves intersect, the more likely is that a line may be found with those parameters. A threshold can be defined on the minimum number of intersections needed to detect a line.

The proposed algorithm starts detecting all the lines of the image region, using the implementation proposed in [20], and selects the most promising ones using some heuristics; finally, recursively, it uses the detected lines to build a set of rectangles merging those that have an intersection. In this way, starting from small adjacent table pieces (often separated due to inaccuracies of the Hough Transform), the table area is detected.

The pseudo-code for the table detector is provided in Algorithm 1.

It should be noted that, even if the goal of the system is the attribution of meaning to a region which comes from the Recursive XY-Cut, the proposed algorithm is able to detect multiple table structures in a whole page, ensuring that each structure found is the largest (piece of) table which contains all other detected pieces of (the same) table.

Even though the results of the Hough Transform depend on its parameters values and on the properties of the original image, the implementation of the

Algorithm 1. Table detection

```
Lines ← HoughLines(parameters);
/* each line identified by two points (x_i, y_i) and (x_j, y_j)        */
Horizontal_lines ← FindHorizontals(lines);
/* x_i ≠ x_j ∧ y_i = y_j                                               */
Vertical_lines ← FindVerticals(lines);
/* x_i = x_j ∧ y_i ≠ y_j                                               */

tables = { };
forall the hl ∈ Horizontal_lines do
    if ∃ vl ∈ Vertical_lines : vl ∩ hl ≠ ∅ ∧ length(vl)>lenght(l) ∀ l ∈ {l : l ∈
    Vertical_lines, l ∩ hl ≠ ∅} then
        rect ← BoundingRectangle(hl,vl)
    end
    while ∃ t ∈ tables : t ∩ rect ≠ ∅ do
        tables = tables-{ t };
        rect ← merge(rect,t);
    end
    tables ← rect;
end
```

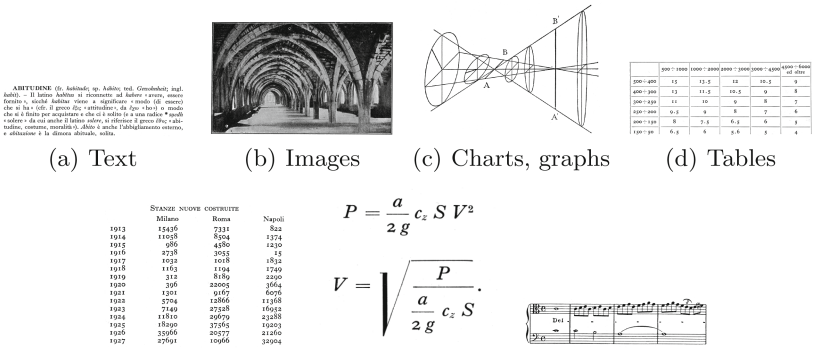
proposed algorithm is scalable and permits great customization: in particular, the method which finds the longest vertical line that intersects an horizontal one (candidate to be a side of the rectangle) permits to specify:

- the minimum number of vertical lines which must intersect an horizontal one for a candidate horizontal line to be considered
- the minimum length of the maximum vertical line which must intersect an horizontal one
- the offset (number of pixels) for considering an horizontal line intersecting a vertical one
- the offset (number of pixels) for considering a vertical line intersecting an horizontal one

3.3 Content Classification

The classification step is necessary to assign each region to a specific class. In particular, in addition to tables, we consider six different classes: text, images, charts and graphs, formulas, scores, and borderless tables. An example of each class is given in Fig. 3.

In order to classify each region, dense SIFT descriptors are computed using the Harris-Laplace detector. This step results in a variable number of 128-dimensional descriptors for each region. To obtain a representation for a region, with fixed size, we summarize SIFT descriptors using the Bag-of-Words technique. To include spatial information into the feature vector we also add the position, the dimensions and the aspect ratio of the image bounding box. A SVM classifier with RBF



(a) Text (b) Images (c) Charts, graphs (d) Tables

(e) Borderless tables (f) Formulas (g) Scores

Fig. 3. We consider seven different content categories: text, images, charts and graphs, tables, borderless tables, formulas, and scores.

kernel is then trained using the feature vectors described earlier. The output of this classifier is the class that will be assigned to the region.

3.4 JSON Description

The output of the overall pipeline is a structured JSON description. For each page, indeed, a JSON file is created with the corresponding OCR results for each text entry and with all the illustrations found inside the page. Moreover, the proposed JSON schema allows textual entries to be linked together. This can be particularly beneficial in the case of encyclopedias, in which each paragraph belongs to a lemma.

An example of the JSON description is reported in Listing 1.1. The `entries` element contains all the textual entries of the page. Each entry contains the body of the paragraph, the column the entry belongs to (in case of multi-column documents), a boolean `is_tabbed` that indicates whether the first line of the paragraph is tabbed or not, and a reference to the lemma. In case the considered text entry is a lemma, the `opening` element is used set a lemma identifier, which can in turn be used to reference other paragraphs, using the `lemma_ref` element. Each text entry also contains the location of the text region inside the page.

Graphical elements are instead grouped into the `graphics` element. Each of them is described by its own caption, position and size inside the page, and contain a reference to the lemma it belongs to. It also contains a `type` field which reports the category of the graphic element, obtained with the content classification pipeline.

3.5 Annotation and Visualization Interfaces

Two more tools have been developed, the first one is an annotation tool which allows a user to visualize the result of the analysis process and, if needed, allows

```
{
   "filename":"0006_00_T_V01.K_scaled.png",
   "layout": {
       "entries": [
            {
              "body":"Sottilissimo nell'ordine ionico greco, era quivi ordinariamente...",
              "centered": false,
              "closing": {
                  "author":"",
                  "selected": false,
                  "sign":"",
                  "valid": false
              },
              "col": 0,
              "is_tabbed": true,
              "lemma_ref": {
                  "major":"ABACO",
                  "minor":"",
                  "pageNumber": 5,
              },
              "opening": {
                  "major":"",
                  "minor":"",
              },
              "par": {
                  "rect": {
                      "height": 137,
                      "width": 562,
                      "x": 125,
                      "y": 835
                  },
              },
          },
       ],
      "graphics": [
           {
              "caption": {
                  "rect": {
                      "height": 24,
                      "width": 347,
                      "x": 524,
                      "y": 788
                  },
                  "text":"CAPITELLI DELL'ATRIO DI S. MARCO A VENEZIA",
              },
              "type":'IMAGE',
              "hascaption": true,
              "image": {
                  "rect": {
                      "height": 639,
                      "width": 1162,
                      "x": 123,
                      "y": 150
                  },
              },
              "lemma_ref": {
                  "major":"",
                  "minor":"",
                  "pageNumber": 5,
                  "x": 0,
                  "y": 0
              },
          },
       ],
  }
}
```


Listing 1.1. JSON Description of a sample page with one text entry and one graphic element

for modifications to the segmentation results. This tool is useful for many reasons: it makes the creation of an annotated dataset possible for all the subsequent learning and evaluation processes and allows users to apply corrections to the processed data. A screenshot of the annotation tool is reported in Fig. 4.

The second tool is a visualization interface used to present and browse the content of the encyclopedia, making all the information easily accessible. This tool lets the users access the content at different levels and from different points of view, it's possible to browse the encyclopedia page by page, lemma by lemma

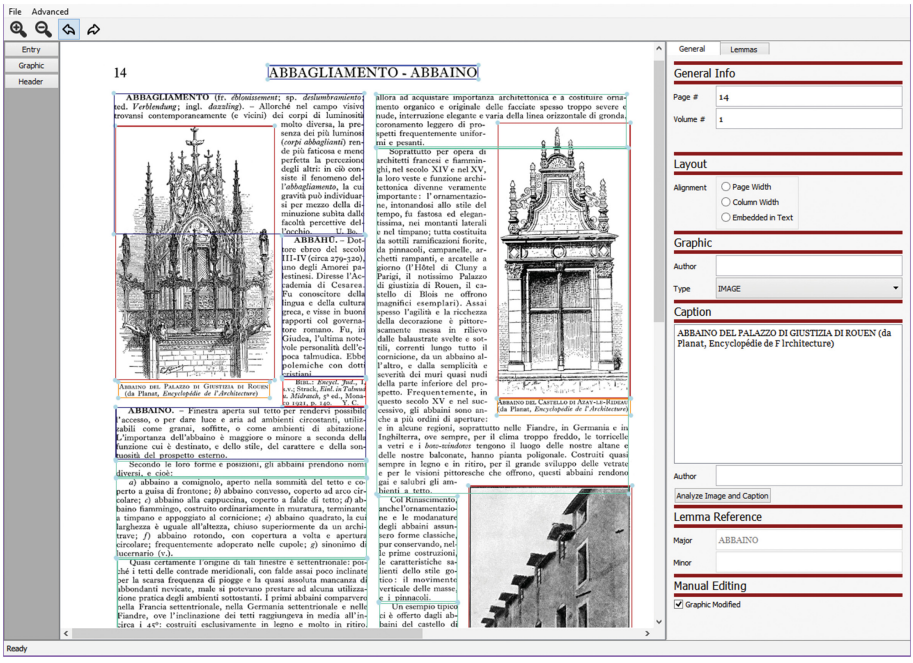


Fig. 4. The main view of the annotation tool. A processed page is visible on the left while on the right all the information relative to a particular page element are displayed

(a) (b)

Fig. 5. Displaying a page in the visualization interface. The page content is highlighted in blue when the cursor hovers on it. (Color figure online)

and image by image in each volume. The full text is also accessible and readable in HTML format. Hovering the cursor over a page shows the underlying extracted content and double clicking on it takes the user to a different view which displays the digitized version of the document. The visualization tool is visible in Fig. 5.

4 Experimental Evaluation

The performance evaluation of layout analysis algorithms can be conducted using two different approaches, namely pixel-level and region-level. The first evaluates how each pixel has been classified in a single page comparing the class assigned to the pixel with the class assigned to the same pixel in the ground truth annotation, the accuracy for a single page is then calculated as the percentage of correctly classified pixels. A region-level approach tries instead to find the best matching between areas of a page that are semantically coherent, called regions, between the analyzed page and the ground truth annotation. Once the matching process is completed, the accuracy value for a single page is calculated with regard to the matching quality. In both cases a cumulative accuracy measure can be calculated as the mean accuracy over multiple pages.

Since we use a top-down page segmentation algorithm in our method and the final output of the page processing pipeline consists of polygons containing pixels of a page classified of the same type, we have chosen a region-level approach to performance evaluation. We used the matching method described by Phillips and Chhabra [21], which has also been used in the ICDAR 2003 page segmentation competition [1]. We used the suggested *acceptance threshold* and *rejection threshold*, respectively 0.85 and 0.05, and we used intersection over union as a similarity function to determine match scores. To evaluate the classification performances we compared the results against the ground truth annotations creating a confusion matrix and calculating accuracy values, one for each class and a cumulative one.

All our tests have been conducted on the first volume of the "Enciclopedia Treccani" (http://www.treccani.it), which was published in 1929 and is composed of 999 pages. The block size n was set to 64, and we used the Tesseract OCR [23]. Considering the Table detector algorithm, all the parameters, both for the Hough Transform and for the heuristic rules, have been set empirically performing several tests on the first 20 tables of the dataset.

Concerning the page segmentation algorithm step we have compared three different algorithms, the standard XY-Cut, the Whitespace Analysis algorithm, proposed by Baird in [3], and, finally our method. Results are shown in Table 1 and it is clear how our method performs largely better than the other two.

Classification results are shown in Tables 2 and 3.

Table 1. Page segmentation experimental results.

	XY-Cut	Whitespace analysis	Our method
Accuracy	61.8%	71.4%	93.8%

Table 2. Classification accuracies

Class	# Elements	Accuracy
Images	1102	0.71
Graphics	145	0.66
Formulas	535	0.77
Tables	91	0.99
Scores	132	0.08
Borderless tables	142	0.10
Text	13628	0.96
TOTAL	**15775**	**0.92**

Table 3. Confusion matrix for classification results of the SVM classifier.

		Truth					
		I	**G**	**F**	**S**	**BT**	**Txt**
Prediction	**I**	74.8%	13.2%	5.2%	0.8%	0.8%	5.3%
	G	6.6%	70.6%	11.0%	1.5%	1.5%	8.8%
	F	3.2%	4.5%	77.4%	0%	2.1%	12.8%
	S	16.0%	6.4%	62.4%	8.8%	1.6%	4.8%
	BT	4.1%	7.4%	37.7%	0%	11.5%	39.3%
	Txt	0.2%	0.3%	1.8%	0%	0.3%	97.3%

5 Conclusion

We presented a complete pipeline for layout analysis and content classification in digitalized documents. The layout analysis algorithm is based on the Recursive XY-Cut and an SVM-aided illustration detection, while the content classification pipeline builds on the Hough transform for table classification and on local features for the classification of images, scores, formulas and charts. The final output is a JSON description, which can in turn be used by two tools, useful for the display and correction of analyzed data. Experimental results showed the effectiveness of our method when tested on the first volume of the "Enciclopedia Treccani".

References

1. Antonacopoulos, A., Gatos, B., Karatzas, D.: ICDAR 2003 page segmentation competition. In: ICDAR, p. 688. IEEE (2003)
2. Appiani, E., Cesarini, F., Colla, A.M., Diligenti, M., Gori, M., Marinai, S., Soda, G.: Automatic document classification and indexing in high-volume applications. Int. J. Doc. Anal. Recogn. **4**(2), 69–83 (2001)
3. Baird, H., Jones, S., Fortune, S.: Image segmentation by shape-directed covers. In: International Conference on Pattern Recognition, vol. 1, pp. 820–825, June 1990

4. Baraldi, L., Grana, C., Cucchiara, R.: A deep siamese network for scene detection in broadcast videos. In: ACM International Conference on Multimedia, pp. 1199–1202. ACM (2015)
5. Bertini, M., Del Bimbo, A., Serra, G., Torniai, C., Cucchiara, R., Grana, C., Vezzani, R.: Dynamic pictorial ontologies for video digital libraries annotation. In: IEEE MultiMedia Magazine, pp. 42–51. ACM (2009)
6. Cesarini, F., Lastri, M., Marinai, S., Soda, G.: Encoding of modified XY trees for document classification. In: Proceedings of the Sixth International Conference on Document Analysis and Recognition, pp. 1131–1136. IEEE (2001)
7. Chen, K., Yin, F., Liu, C.L.: Hybrid page segmentation with efficient whitespace rectangles extraction and grouping. In: 12th International Conference on Document Analysis and Recognition (ICDAR), pp. 958–962. IEEE (2013)
8. Coüasnon, B., Lemaitre, A.: Recognition of tables and forms. In: Doermann, D., Tombre, K. (eds.) Handbook of Document Image Processing and Recognition, pp. 647–677. Springer, London (2014)
9. Mauro, N., Ferilli, S., Esposito, F.: Learning to Recognize Critical Cells in Document Tables. In: Agosti, M., Esposito, F., Ferilli, S., Ferro, N. (eds.) IRCDL 2012. CCIS, vol. 354, pp. 105–116. Springer, Heidelberg (2013). doi:10.1007/978-3-642-35834-0_12
10. Duda, R.O., Hart, P.E.: Use of the Hough transformation to detect lines and curves in pictures. Commun. ACM **15**(1), 11–15 (1972)
11. Esposito, F., Malerba, D., Lisi, F.A.: Machine learning for intelligent processing of printed documents. J. Intell. Inf. Syst. **14**(2–3), 175–198 (2000)
12. Grana, C., Serra, G., Manfredi, M., Coppi, D., Cucchiara, R.: Layout analysis and content enrichment of digitized books. Multimed. Tools Appl. **75**(7), 3879–3900 (2016)
13. Ha, J., Haralick, R.M., Phillips, I.T.: Recursive XY cut using bounding boxes of connected components. In: Proceedings of the Third International Conference on Document Analysis and Recognition, vol. 2, pp. 952–955. IEEE (1995)
14. Kaur, S., Sharma, D.V.: Table structure identification from document images: a survey. Int. J. Innov. Adv. Comput. Sci. **4**, 581–585 (2015)
15. Kise, K., Sato, A., Iwata, M.: Segmentation of page images using the area Voronoi diagram. Comput. Vis. Image Underst. **70**(3), 370–382 (1998)
16. Lazzara, G., Levillain, R., Géraud, T., Jacquelet, Y., Marquegnies, J., Crépin-Leblond, A.: The scribo module of the olena platform: a free software framework for document image analysis. In: 2011 International Conference on Document Analysis and Recognition (ICDAR), pp. 252–258. IEEE (2011)
17. Liu, Y., Mitra, P., Giles, C.L.: A fast preprocessing method for table boundary detection: narrowing down the sparse lines using solely coordinate information. In: The Eighth IAPR International Workshop on Document Analysis Systems, pp. 431–438. IEEE (2008)
18. Mandal, S., Chowdhury, S.P., Das, A.K., Chanda, B.: Detection and segmentation of tables and math-zones from document images. In: Proceedings of the 2006 ACM Symposium on Applied Computing. SAC 2006, pp. 841–846. ACM (2006)
19. Mandal, S., Chowdhury, S., Das, A., Chanda, B.: A simple and effective table detection system from document images. Int. J. Doc. Anal. Recogn. (IJDAR) **8**(2–3), 172–182 (2006)
20. Matas, J., Galambos, C., Kittler, J.: Robust detection of lines using the progressive probabilistic Hough transform. Comput. Vis. Image Underst. **78**(1), 119–137 (2000). http://dx.doi.org/10.1006/cviu.1999.0831

21. Phillips, I.T., Chhabra, A.K.: Empirical performance evaluation of graphics recognition systems. IEEE Trans. Pattern Anal. Mach. Intell. **21**(9), 849–870 (1999)
22. Sebastiani, F.: Machine learning in automated text categorization. ACM Comput. Surv. (CSUR) **34**(1), 1–47 (2002)
23. Smith, R.: An overview of the Tesseract OCR engine. In: International Conference on Document Analysis and Recognition, pp. 629–633. IEEE (2007)
24. Zanibbi, R., Blostein, D., Cordy, J.: A survey of table recognition. Doc. Anal. Recogn. **7**(1), 1–16 (2004)