2009 10th International Conference on Document Analysis and Recognition
2009年第十届国际文档分析与识别会议

Hybrid Page Layout Analysis via Tab-Stop Detection

通过Tab-Stop检测进行混合页面布局分析

Ray Smith
雷·史密斯
Google Inc. 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA.
Google Inc. 1600 Amphitheatre Parkway，Mountain View，CA 94043，USA。
theraysmith@gmail.com
theraysmith@gmail.com

Abstract

摘要

A new hybrid page layout analysis algorithm is proposed, which uses bottom-up methods to form an initial data-type hypothesis and locate the tab-stops that were used when the page was formatted. The detected tab-stops, are used to deduce the column layout of the page. The column layout is then applied in a top-down manner to impose structure and reading-order on the detected regions.

一种新的混合页面布局分析算法是提议的方法，该方法使用自下而上的方法来形成初始数据类型假设，并找到格式化页面时使用的制表位。检测到的制表位用于推断页面的列布局。然后以自上而下的方式应用列布局，以将结构和读取顺序强加到检测到的区域上。

The complete C++ source code implementation is available as part of the Tesseract open source OCR engine at http://code.google.com/p/tesseract-ocr.

完整的C++源代码实现是可作为Tesseract开源OCR engine的一部分获得，网址为http://code.google.com/p/tesseract-ocr.

1. Introduction

1.简介

Physical Page layout analysis, one of the first steps of OCR, divides an image into areas of text and non-text, as well as splitting multi-column text into columns. This paper does not address *logical* layout analysis, which detects headers, footers, body text, numbered lists, and segmentation into articles.

物理页面布局分析，第一步之一 OCR的功能是将图像分为文本和非文本区域，以及将多列文本分成多个列。本文不涉及逻辑布局分析，而是检测页眉，页脚，正文，编号列表和文章分段。

Physical Layout Analysis is essential to enable an OCR engine to process images of arbitrary pages, such as from books, magazines, journals, newspapers, letters and reports. Methods of physical layout analysis fall roughly into two categories:

物理布局分析对于实现 OCR引擎可处理任意页面的图像，例如书籍，杂志，期刊，报纸，信件和报告。物理布局分析的方法大致分为两类：

Bottom-up methods are both the oldest [1] and more recently published [2,3] methods. They classify small parts of the image (pixels, groups of pixels, or connected components), and gather together like types to form regions. The key advantage of bottom-up methods is that they can handle arbitrarily shaped regions with ease. The key disadvantage is that they struggle to take account of higher-level structures in the image, such as columns. This often leads to over-fragmented regions.

自下而上的方法都是最早的方法[1]甚至更多 最近发表的[2,3]方法。他们将图像的一小部分（像素，像素组或连接的组件）分类，并像类型一样聚集在一起以形成区域。自下而上的方法的主要优点是它们可以轻松处理任意形状的区域。关键缺点是它们努力考虑图像中的高级结构，例如列，这通常导致过度碎片化的区域。

Top-down methods [4] cut the image recursively in vertical and horizontal directions along whitespace that are expected to be column boundaries or paragraph boundaries. Although top-down methods have the advantage that they look at the largest structures on the page first, they are unable to handle the variety of formats that occur in many magazine pages, such as non-rectangular regions and cross-column headings that blend seamlessly into the columns below.

自上而下的方法[4]以递归方式切割图像 沿着空白的垂直和水平方向，这些空白应该是列边界或段落边界。尽管自上而下的方法具有优点是，它们从查看页面上最大的结构开始，因此无法处理许多杂志页面上出现的各种格式，例如非矩形区域和交叉列标题无缝地融合到下面的列中。

A third type of method [5-7] is based on analysis of the whitespace in the image. These methods fix the flaws in the recursive top-down methods, by finding gaps between columns by a bottom-up analysis of the whitespace, explicitly looking for white rectangles. These algorithms mostly still suffer from the problem of being unable to handle non-rectangular regions.

第三类方法[5-7]是基于 图像中的空白。通过对间隙进行自底向上的分析，在列之间查找间隙，从而明确查找白色矩形，从而解决了递归自顶向下方法中的某些缺陷。这些算法大多仍然存在无法处理非矩形区域的问题。

2. Page layout via tab-stop detection

2.通过制表符停止检测进行页面布局

When a page is laid out, either by a professional publishing system or a common word processor, the regions of a page are bounded by *tab-stops*. The margins, column edges, indentation, and columns of a table are all placed at fixed x-positions, at which edges or centers of text lines are aligned vertically. Tab-stops distinguish tables from body text, and they also bound rectangular non-column elements, such as inset images and pull-out quotes.

布置页面时，由专业人员 发布系统或文字处理程序，页面区域由制表符限制，边距，列边缘，缩进和table列都放置在固定的x位置，在这些位置上文本行的边缘或中心垂直对齐。制表位停止将表格与正文分开，它们也限制了矩形的非列 元素，例如插图 图片和引号。

Fig.1. Input image.

图1.输入图像

The tab-stops in the example of Fig. 1 are the column boundaries with an additional tab-stop for the indentation of the first line that is not required for finding the page layout. The non-rectangular inset image, typically, strays outside of the column boundaries.

制表符在 图1的示例是带有用于页面缩进的附加制表符的列边界，这对于查找页面布局不是必需的。非矩形插图通常会偏离列边界。

In some sense, white rectangles match tab-stops, but white rectangles can be disrupted by background noise or background images. Also the ends of white rectangles do not match the ends of regions bounded by the tab-stops, because the white rectangles run on into the perpendicular whitespace.

从某种意义上说，白色矩形与制表符匹配，但是背景矩形可能会干扰白色矩形。也是白色的两端 矩形与制表位限制的区域的末端不匹配，因为白色矩形一直延伸到垂直空白中。

978-0-7695-3725-2/09 $25.00 © 2009 IEEE
978-0-7695-3725-2 / 09 $25.00美元©2009 IEEE 241
DOI 10.1109/ICDAR.2009.257
DOI 10.1109 / ICDAR.2009.257

The proposed algorithm is similar to the whitespace rectangle method [5], in that it uses a top-down approach to find a top-down structure, but instead of finding the space between columns, it looks for the tab-stops that mark the edges of the columns, and uses a combination of bottom-up and top-down methods, copes easily with non-rectangular regions.

There are four main phases: preprocessing, in which bottom-up morphological and connected component analysis form initial hypotheses over the local data types, bottom-up tab-stop detections, finding the column layout, and finally, using the column layout to create an ordered set of typed regions. These phases will be detailed in sections 3-6.

3. Preprocessing

The aim of the preprocessing step is to identify line separators, image regions, and to separate the remaining connected components into likely text components and a smaller number of uncertain type.

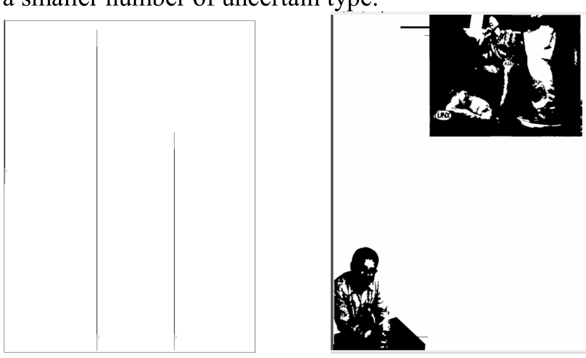

Fig.2. (a) Vertical lines, (b) Image elements.

Starting with the image of Fig. 1, the morphological processing of [8] detects the vertical lines shown in Fig. 2(a) and the image mask shown in Fig. 2(b). These detected elements are subtracted from the input image before passing the cleaned image to connected component analysis.

The connected components (CCs) are filtered by width, w, and height, h, into small, medium and large sizes as follows: CCs with $h < 7$ (at 300ppi) are small. The 75th percentile of the heights of the remainder, h_{75}, is computed, and CCs with $h < h_{75}/2$ are small, $h > 2h_{75}$ or $w > 8h_{75}$ are large, and the rest are medium.

This filtration is important, since small CCs (noise or vertical punctuation) and large non-text (line art, logos, or frames) are likely to confuse the text-line algorithms, but large text headings are important to reading order detection. Large CCs are considered text if their left or right neighbor has a similar stroke width. On "stressed" fonts, the stroke width is greater on vertical lines than on horizontal lines, so stroke width is calculated separately in both directions. Stroke width is calculated from horizontal and vertical local maxima of the distance function of the binary image of the CC. Fig. 3 shows the CCs are filtered as medium or large text.

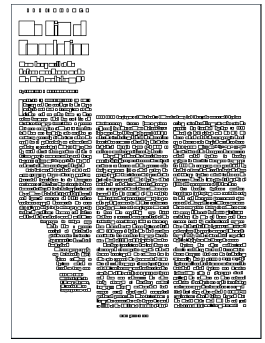

Fig.3. Filtered CCs

4. Finding tab positions as line segments

The process of finding tab-stop line segments has several major sub-steps: CCs that look like they may be at the edge of a text region are found and then grouped into tab-stop lines, then connections between tab-stop lines are traced, enabling removal of false positives.

4.1. Finding candidate tab-stop components

The initial candidate tab-stop CCs are found by a radial search from each filtered CC from preprocessing. Assuming that the CC is at a tab-stop, the search looks for aligned neighbors and neighbors in the gutter where there should be a space. Each CC is independently marked as a candidate based on whether it is a candidate left tab, right tab or neither. Fig. 4(a) illustrates the candidate tab-stop CCs.

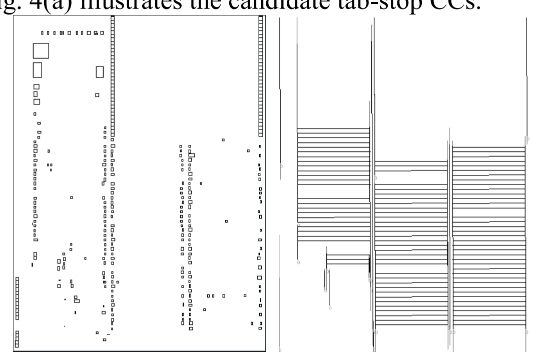

Fig.4. (a) Candidate tab-stop components (b) Fitted tab lines and traces connections.

4.2. Grouping candidate tab components

Candidate tab CCs are grouped into lines, and, if there are enough of them in a group, they are kept. A least median of squares algorithm is used to fit a line to the appropriate (left or right) edge of each CC in a group. After finding all tab-stop line segments, all the lines are refitted to the page-mean direction,

978-0-7695-3725-2/09 $25.00 © 2009 IEEE

DOI 10.1109/ICDAR.2009.257

all text-line members fall on the same side of the line segment.

4.3. Tracking text lines to connect tab stops

The next step connects tab-stops by tracking text lines from one tab-stop to another. Close neighbor, vertically overlapping CCs qualify, but large gaps cannot be jumped. Tab-stops that have a text-line connecting them are associated with each other, as being like opposite sides of a text column. Fig. 4(b) shows the tab-stop lines and connecting text lines. Tab-stop lines that have no connection are discarded.

The most frequently occurring widths of the text lines connecting tab stops are recorded for use in finding the column layout.

4.4. Cleaning up tab stop ends

The final step attempts to extend the ends of the tab line to the last member CC at the same y-coordinate, by allowing the ends to move between the last member CC whose edge is used to define the tab line, and the first non-member CC that the line intersects. Fig. 5 shows the final tab line segments.

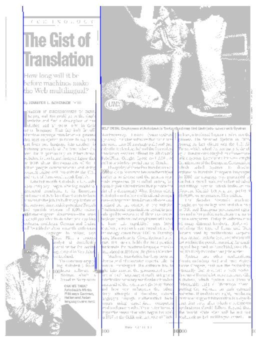

Fig.5. Cleaned tab-stops.

After construction of the tab-stops, the CCs are re-classified, as "Text" or "Image", using the same text line tracking algorithm as was used above to find connections between tab stops. If a group of CCs of significant width form a text line, then they are classified as text. Artificial image CCs of about the same size as the body text CCs are created from the image mask from the morphological preprocessing.

5. Finding the column layout

The next major step is to find the column layout of the page. All the rest of the steps make use of the Column Partition (CP) objects which are created now.

Scanning the CCs from left to right and top to bottom, the runs of similar type (text, image or unknown) CCs are gathered into CPs, subject to the constraint that no CP may cross a tab stop line. Fig. 6 shows the result of this process. A collection of CPs from a single horizontal scan is stored in a Column Partition Set (CPset).

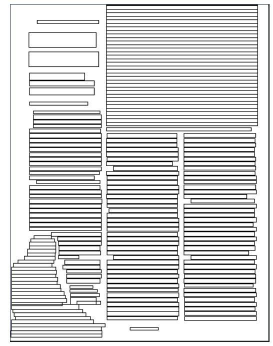

Fig.6. Column Partitions (CPs)

Each CPset is potentially a division of the page into columns at that vertical position. Finding the column layout is therefore a process of finding an optimal set of CPsets that best "explains" (see below) all the CPsets on the page, but first some definitions:

A *good* CP either touches a tab line on both vertical edges of its bounding box, or its width is close to a frequently occurring width. (See 4.3.)

The *coverage* of a CPset is the total width of all the good CPs that it contains.

CPset A *is better than* CPset B if A has greater coverage, or equal coverage, but more good CPs, or equal good CPs, but more total CPs.

CPset A *explains* set B **unless** one or more of the following is true:

1. The edge of one of B's CPs lies outside of all of A's CPs. This is not allowed, as it shows that B has more text than A.
2. The edges of one of B's CPs fall in different CPs of A, and the width of the B CP is a common one. This means that A has split a column of common width.
3. The right edge of a CP of B is in the same CP of A as the left edge of the next B CP, and the B CPs are of roughly the same width. It looks like A has a different number of columns to B. The same width condition allows A to explain B with a pull-out.
4. Both edges of two CPs of B fall in the same CP of A. This means that A has merged two columns of B.

Note that the two edges of one of B's CPs are allowed to fall into two CPs of A, as long as the width is not a common one. This allows headings that merge columns in B to be explained by A.

Fig.7. Columns.

A list of column candidates is made from the set of CPsets on the page, ordered best first, and with elimination by the A explains B rules above. In this process, all image CPs are ignored.

After the initial candidates are made, they are improved by adding new CPsets, and widening existing CPs by using the edge of a CP in a different CPSet while widening doesn't cause overlap of CPs.

978-0-7695-3725-2/09 $25.00 © 2009 IEEE

DOI 10.1109/ICDAR.2009.257

An iterative process then labels the longest segment of each column candidate (allowing very small regions of failure) page y-coordinates that is explained by one of the column candidates. Fig. 7 shows the result of this process.

6. Finding the regions

After the columns are found, CPs are given a type according to how many columns they span. CPs within a single column are *flowing*, partitions that touch more than one column, but do not span to the outer edges of either are *pull-out*, and partitions that completely span more than one column are *heading*.

6.1. Create flows of CPs

Each CP chooses its best matching upper and lower partner, i.e. vertically nearest that overlaps horizontally. Since each CP registers itself with its chosen partner, each CP may have zero or more registered upper and lower partners.

The size of the list of registered partners is forced to zero or one for both upper and lower, using the following rules in order:

1. Type. If there are multiple types, text can only stay with its own (exact) type, whereas image can stay with any other image type.
2. Transitive partner shortcuts are broken. If A has 2 partners B and C, and also B has C as a partner in the same direction, then delete C as a partner of A, leaving a clean chain A-B-C. Also if A has a partner B, and B has a partner A in the same direction, break the cycle.
3. (Text only) If A still has 2 partners B, C, chase B and C's partners to see which has the longest chain. Delete from A the partner that has the shortest chain, and convert the type of the shortest chain to pull-out.
4. (Image) Choose the partner with the largest horizontal overlap.

All CPs now have 0 or 1 partners. Even so, (re)run the above rules. This will reduce all chains of text to a single type and splits text chains from image chains. Image chains are purified by setting all CPs in a chain to the most general type in the chain. Fig. 8 shows the final typed CPs, where flowing text is blue, heading text is cyan, heading image is magenta, and pull-out image is orange.

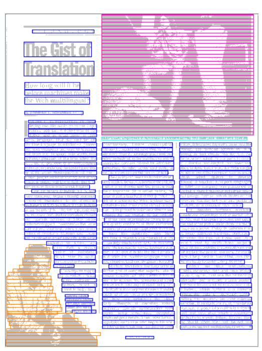

Fig. 8. Typed partition chains.

Chains of text CPs are further divided into groups of uniform line-spacing to make text blocks. Now each chain of CPs represents a candidate region, but the regions have to be ordered.

6.2. Reading order determination

Recall that image and text partitions are typed as one of the possibilities flowing, pull-out, or heading. Also, the page is divided into sections of a consistent column layout. With this information, a reasonable reading order is obtained with a few simple rules:

1. Flowing blocks follow by y position within a column.
2. Pull-out blocks follow by y position in an imaginary column between the real columns that they touch.
3. A heading spans multiple columns and follows anything that is above it in the columns spanned or between them. Anything that lies in the same columns below the heading follows after it.
4. A change in column layout works just like a heading. Anything in any columns that are changed (or between them) goes before anything in the new columns. Unchanged columns are unaffected by a change in column layout.
5. Between headings, the content of columns is ordered from left to right.

6.3. Find the polygon boundary for each region

For simplicity of implementation, the region polygons are restricted to be rectilinear, with edges being horizontal and parallel to the mean tab line (Approximately vertical.) The polygon edges are chosen to minimize the area of the region while satisfying the constraint that all CPs are contained within the polygon, and no CP from another region is contained. Fig. 9 shows the final blocks created for the input image of Fig. 1.

Fig. 9. Final blocks.

7. Testing and results

The algorithm described herein is implemented in C++, and the code is available as part of the Tesseract open source OCR system [9,10]. It runs on a typical 8MPixel image in approximately 1 second on a 3.4 GHz Pentium 4.

978-0-7695-3725-2/09 $25.00 © 2009 IEEE

DOI 10.1109/ICDAR.2009.257

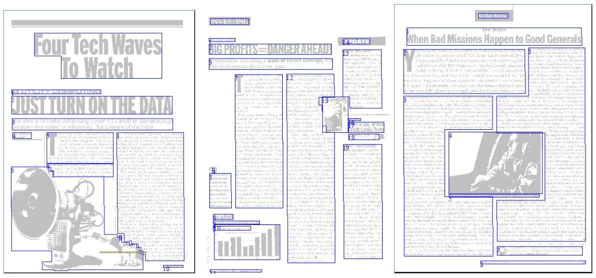

Fig. 10. Results on some of the ICDAR2007 set.

Properly testing page layout analysis is a difficult problem[11], [1] with very little publicly available ground truth for complex magazine pages. The UNLV test set [12], only measures text regions, and counts errors unless figure captions are placed after all the body text.

The ICDAR page layout analysis competitions provide better overall measurement of accuracy, and the results of this algorithm appear in the 2009 competition [13]. Some graphical results are shown in Fig. 10 and numerical comparisons with the entrants in the ICDAR 2007 competition are shown in Table 1. The results in Table 1 are computed on only the 2007 test set, and the author would like to thank Apostolos Antonacopoulos for providing these results. For details on the testing methodology, see references [11] and [13].

Table 1. Results on the ICDAR 2007 set.

Method	Noise	Sep	Text	Image	Overall
PRImA Metric					
2007-Besus	86.8%	76.9%	37.4%	42.5%	35.9%
2007-TH1	68.0%	79.7%	76.1%	46.2%	67.6%
2007-TH2	67.6%	79.6%	72.9%	48.4%	65.7%
Tesseract	65.6%	74.1%	72.1%	55.3%	68.4%
F-Measure					
2007-Besus	62.9%	76.2%	95.8%	57.2%	90.2%
2007-TH1	79.2%	80.7%	91.9%	72.1%	88.2%
2007-TH2	79.2%	80.6%	92.3%	72.4%	88.6%
Tesseract	79.2%	70.9%	93.3%	82.0%	91.3%
Recall					
2007-Besus	65.7%	71.7%	94.9%	67.0%	88.2%
2007-TH1	65.6%	79.5%	96.9%	66.4%	89.8%
2007-TH2	65.6%	79.5%	97.2%	66.9%	90.2%
Tesseract	65.6%	81.4%	97.9%	76.5%	93.8%
Precision					
2007-Besus	60.4%	81.3%	96.7%	50.0%	92.2%
2007-TH1	100.0%	81.9%	87.4%	79.0%	86.7%
2007-TH2	100.0%	81.7%	87.9%	79.0%	87.0%
Tesseract	100.0%	62.8%	89.0%	88.3%	88.9%

10. Conclusion and further work

Tab-stops make an interesting and useful alternative to white rectangles for finding the column structure of a page. Combining the top-down concept of column structure with bottom-up classification methods enables page layout analysis to easily handle the complex non-rectangular layouts of modern magazine pages without losing sight of the "bigger picture" that often happens when bottom-up methods are used alone.

The algorithm described has no table detection or analysis, but the tab-stops make particularly useful features for both, so table analysis will be added in the future.

11. References

[1] F. Wahl, K. Wong, R. Casey, "Block segmentation and text extraction in mixed text/image documents," *Computer Graphics and Image Processing*, **20**, 1982, pp375-390.

[2] M. Chen, X. Q. Ding, "Unified HMM-based Layout Analysis Framework and Algorithm," SCI CHINA Ser F, **46**(6), Dec. 2003, pp401-408.

[3] S.P. Chowdhury, S. Mandal, A.K. Das, B. Chanda, "Segmentation of Text and Graphics from Document Images," *Proc. of Int. Conf. 9th Int. Conf. on Document Analysis and Recognition*, IEEE, Curitiba, Brazil, Sep 2007, pp619-623.

[4] G. Nagy, S.C. Seth, "Hierarchical Representation of Optically Scanned Documents," *Proc. 7th Int. Conf. on Pattern Recognition*, Montreal, Canada, 1984, pp347-349.

[5] H.S. Baird, S.E. Jones, S.J. Fortune, "Image Segmentation by Shape-directed Covers," *Proc. 10th Int. Conference on Pattern Recognition*, IEEE, Atlantic City, NJ, 1990, pp820-825.

[6] T. Pavlidis, J. Zhou, "Page Segmentation and Classification," *CVGIP: Graphical Models and Image Processing*, **54**(6), November 1992, pp484-496.

[7] T.M. Breuel, "Two Geometric Algorithms for Layout Analysis," *Proc. of the 5th Int. Workshop on Document Analysis Systems V*, Springer-Verlag 2002, pp188-199.

[8] Leptonica image processing and analysis library. http://www.leptonica.com.

[9] R. Smith, "An overview of the Tesseract OCR Engine," *Proc 9th Int. Conf. on Document Analysis and Recognition*, IEEE, Curitiba, Brazil, Sep 2007, pp629-633.

[10] The Tesseract open source OCR engine. http://code.google.com/p/tesseract-ocr.

[11] A. Antonacopoulos, B. Gatos, D. Bridson, "ICDAR2007 Page Segmentation Competition," *Proc 9th Int. Conf. on Document Analysis and Recognition*, IEEE, Curitiba, Brazil, Sep 2007, pp1279-1283.

[12] UNLV ISRI OCR testing toolkit and database http://www.isri.unlv.edu/ISRI/OCRtk.

[13] A. Antonacopoulos et. al. "ICDAR2009 Page Segmentation Competition," *Proc 10th Int. Conf. on Document Analysis and Recognition*, 2009, Barcelona, Spain, Jul 2009.

978-0-7695-3725-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDAR.2009.257