Table detection in heterogeneous documents

Conference Paper · January 2010

DOI: 10.1145/1815330.1815339 · Source: DBLP

2 authors, including:

Faisal Shafait
University of Western Australia
178 PUBLICATIONS **3,820** CITATIONS

SEE PROFILE

Table Detection in Heterogeneous Documents

Faisal Shafait*
German Research Center for
Artificial Intelligence (DFKI GmbH)
Kaiserslautern, Germany
faisal.shafait@dfki.de

Ray Smith
Google Inc.
Mountain View, CA, USA
theraysmith@gmail.com

ABSTRACT

Detecting tables in document images is important since not only do tables contain important information, but also most of the layout analysis methods fail in the presence of tables in the document image. Existing approaches for table detection mainly focus on detecting tables in single columns of text and do not work reliably on documents with varying layouts. This paper presents a practical algorithm for table detection that works with a high accuracy on documents with varying layouts (company reports, newspaper articles, magazine pages, . . .). An open source implementation of the algorithm is provided as part of the Tesseract OCR engine. Evaluation of the algorithm on document images from publicly available UNLV dataset shows competitive performance in comparison to the table detection module of a commercial OCR system.

Categories and Subject Descriptors

I.7.5 [**Document and Text Processing**]: Document Capture—*Document Analysis*

Keywords

page segmentation, table detection, document analysis

1. INTRODUCTION

Automatic conversion of paper documents into an editable electronic representation relies on optical character recognition (OCR) technology. A typical OCR system consists of three major steps. First, layout analysis is performed to locate text-lines in the document image and to identify their reading order. Then, a character recognition engine processes the text-line images and generates a text string by recognizing individual characters in the text-line image. Finally, a language modeling module makes corrections in the text string using a dictionary or a language model.

*The author gratefully acknowledges funding from Google Inc. for supporting this work

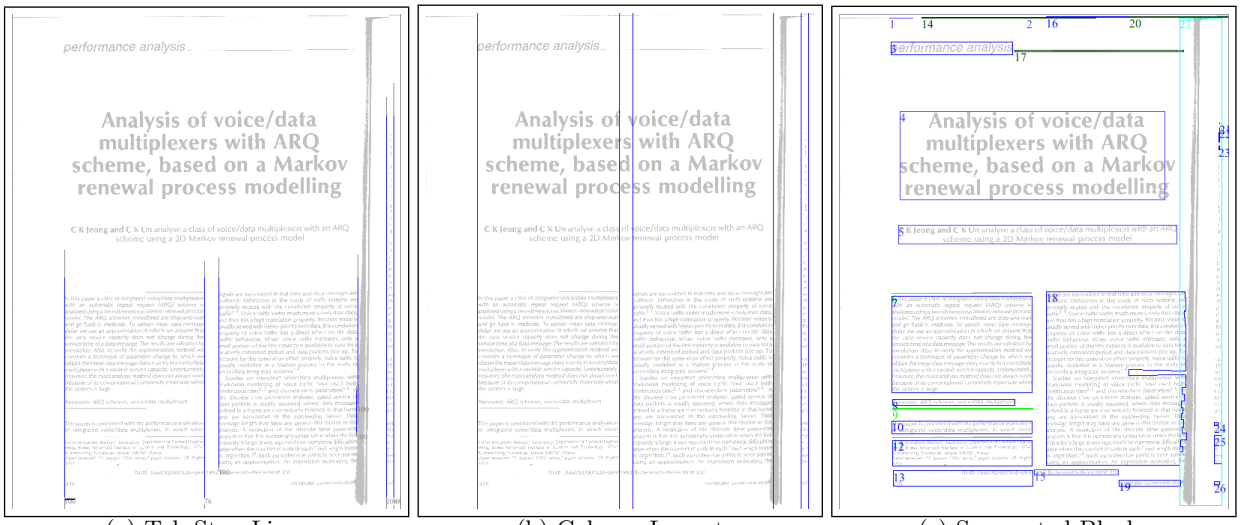
DAS '10, June 9-11, 2010, Boston, MA, USA

Since layout analysis is the first step in such a process, all subsequent stages rely on layout analysis to work correctly. One of the major challenges faced by layout analysis is detecting table regions. Table detection is a hard problem since tables have a large variation in their layouts. Existing open-source OCR systems lack the capability of table detection and their layout analysis modules break down in the presence of table regions. A distinction should be made at this stage between table detection and table recognition [8]. Table detection deals with the problem of finding boundaries of tables in a page image. Table recognition, on the other hand, focuses on analyzing a detected table by finding its rows and columns and tries to extract the structure of the table. Our focus in this paper is on the table detection problem.

One of the pioneering works on table detection and recognition was done by Kieninger et al. [11, 10, 12]. They developed a table spotting and structure extraction system called T-Recs. The system relies on word bounding boxes as input. These word boxes are clustered with a bottom-up approach into regions by building a "segmentation graph". These regions are then designated as candidate table regions if they satisfy certain criterion. The key limitation of the approach is that based only on word boxes, multi-column layouts can not be handled very accurately. Therefore it works well only for single column pages.

Wang et al. [20] take a statistical learning approach for the table detection problem. Given a set of candidate text-lines, candidate table lines are identified based on gaps between consecutive words. Then, vertically adjacent lines with large gaps and horizontally adjacent words are grouped together to make table entity candidates. Finally, a statistical based learning algorithm is used to refine the table candidates and reduce false alarms. They make the assumption that the maximum number of columns is two and design three templates of page layout (single column, double column, mixed column). They apply a column style classification algorithm to find out the column layout of the page and use this information as *a priori* knowledge for spotting table regions. This approach can handle only those layouts on which it has been trained. Besides, training the algorithm requires a large amount of labeled data.

Hu et al. [6] presented a system for table detection from scanned page images or from plain text documents. Their system assumes a single-column input page that can be eas-



(a) Tab-Stop Lines (b) Column Layout (c) Segmented Blocks

Figure 1: Output of different steps of Tesseract's layout analysis module on a document image.

ily segmented into individual text-lines (for instance by horizontal projection). The table detection problem is then posed as an optimization problem where start and end text-lines belonging to a table are identified by optimizing some quality function. Like previous approaches, this technique can not be applied to multi-column documents.

In [7] Hu et al. evaluated their table detection algorithm on the UW-III dataset [5] by using ground-truth zone information (deciding for each ground-truth zone whether it is a table or not). This evaluation is not practical since segmenting a table as a single zone is actually the hard part of a table detection system. This goes more into the direction of document zone classification [21, 9] where the goal is to assign each of the segmented document zones into a set of pre-defined classes (text, math, table, half-tone, ...).

Cesarini et al. [2] present a system for locating table regions by detecting parallel lines. The table hypothesis formed in this way are then verified by locating perpendicular lines or white spaces in the region included between the parallel lines. However, relying only on horizontal or vertical lines for table detection limits the scope of the system since not all tables have such lines. More recent work in table detection is reported by Gatos et al. [4] and Costa e Silva [3]. Gatos et al. [4] focus on locating tables that have both horizontal and vertical rulings and find their intersection points. Then, table reconstruction is achieved by drawing the corresponding horizontal and vertical lines that connect all line intersection pairs. The system works pretty well for their target documents but can not be used when the tables rows/columns are not separated by ruling lines. The work of Costa e Silva [3] focuses on extracting table regions from PDF documents using Hidden Markov Models (HMMs). They extract text from the PDF using pdftotext Linux utility. The spaces in the extracted text are used for computing the feature vector. Clearly, this approach would not work for document images.

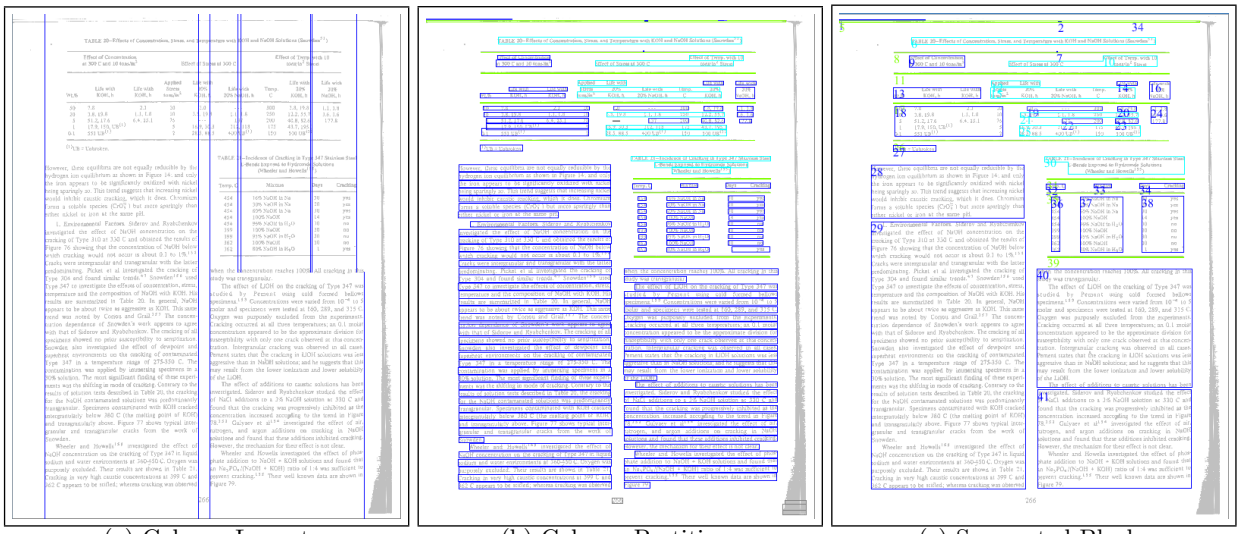
Summarizing the state of the art in table detection, we can see a clear limitation of existing methods. The methods do not work well on multi-column document images. This is probably due to the fact that most of the existing approaches focus on table recognition to extract the structure (rows, columns, cells) of the tables and hence make some simplifying assumptions on the table detection part. This approach works well when one has to deal with some specific classes of document images having simple layouts. However, more robust table detection algorithms are needed when dealing with a heterogeneous collection of documents. In this paper, we try to bridge this gap. Our goal is to accurately spot table regions in complex heterogeneous documents (company reports, journal articles, newspapers, magazines, ...). Once table regions are spotted, one of the existing table recognition techniques (e.g. [10]) could be used to extract the structure of the tables.

The rest of this paper is organized as follows. First, we describe in Section 2 the layout analysis module of Tesseract [18, 19] that would be used as a basis of our table detection algorithm. Then, our table detection algorithm is illustrated in Section 3. Different performance measures used to evaluate our system are presented in Section 4. Experimental results and discussion is given in Section 5 followed by a conclusion in Section 6.

2. LAYOUT ANALYSIS VIA TAB-STOP DETECTION

The layout analysis of Tesseract is a recent addition to the open source OCR system [19]. It is based on the idea of detecting tab-stops in a document image. When type-setting a document, tab-stops are the locations where text aligns (left, right, center, decimal, ...). Therefore, tab-stops can be used as a reliable indication of where a text block starts or ends. Finding the layout of the page via tab-stop detection proceeds as follows (see Figure 1 for illustration):

- First, a document image pre-processing step is performed to identify horizontal and vertical ruling lines or separators and to locate half-tone or image regions



(a) Column Layout (b) Column Partitions (c) Segmented Blocks

Figure 2: The result of different steps of Tesseract's layout analysis in the presence of table regions. Note that the Column layout is not consistent in the two tables. Similarly, column partitions also sometimes merge text across different table columns and sometimes keep them separate. This results in heavy over-segmentation of the page image in table regions.

in the document. Then, a connected component analysis is performed to identify candidate text components based on their size and stroke width.

- The filtered text components are evaluated as candidates for lying on a tab-stop position. These candidates are grouped into vertical lines to find tab-stop positions that are vertically aligned. As a final step, pairs of connected tab lines are adjusted such that they end at the same y-coordinate (see Figure 1(a)). At this stage, vertical tab lines marks the start and end of text regions.

- Based on the tab-lines, the column layout of the page is inferred and connected components are grouped into *Column Partitions*. A column partition is a sequence of connected components that do not cross any tab line and are of the same type (text, image, . . .). Text column partitions can be regarded as initial candidates for text-lines(see Figure 1(b)).

- The last step creates flows of column partitions such that neighboring column partitions of the same type are grouped into the same block (Figure 1(c)). Text column partitions having different font size and line spacing are grouped into different blocks. Then, the reading order of these blocks is identified. The boundary of the blocks is represented as an isothetic polygon (a polygon that has all edges parallel to the axes).

3. TABLE SPOTTING

Our table detection algorithm is built upon two components of the layout analysis module:

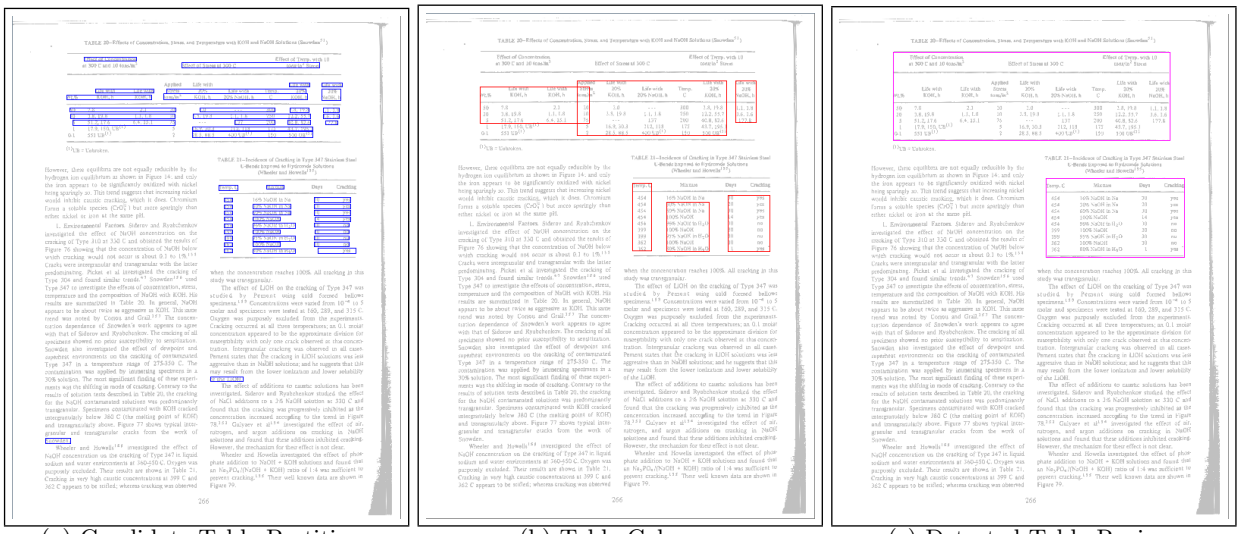
1. Column partitions
2. Column layout

Column partitions give us connected components grouped by their type into partitions that do not cross tab-stop lines. Therefore, text column partitions approximate text-lines in the document. Half-tone regions and horizontal black lines (rulings) are reported as column partitions of "image" and "horizontal line" type. Besides column partitions, column layout gives us the information whether a particular column partition lies completely within one column or is spanning across multiple columns. As shown in Figure 2, both column partition and column layout may give erroneous results in the presence of table regions.

A further analysis of layout analysis results in the presence of table regions shows two major scenarios. In the first case, table columns are reported as page columns thereby destroying the columnar structure of the page. This happens particularly when table cells are very well aligned. The alignment causes a large number of tab-stops to be detected and hence the tab-lines are strong enough to report the presence of a column. Each cell in the table is thereby reported as a single column partition. In the second case, table columns are ignored by the system due to cells that are not well aligned. Hence, the columnar structure of the page is correctly identified. Column partitions in this case span across different columns of the table. Both these cases are illustrated in the example image in Figure 2. Based on this analysis, our table detection algorithm is designed as follows.

3.1 Identifying Table Partitions

The first step in our algorithm identifies text column partitions that could belong to a table region, referred to as *table partitions*. Based on the observations mentioned in the previous paragraph, three types of partitions are marked as table partitions: (1) partitions that have at lease one large gap between their connected components, (2) partitions that consist of only one word (no significant gap between com-



(a) Candidate Table Partitions (b) Table Columns (c) Detected Table Regions

Figure 3: The result of different steps of our table detection algorithm on a sample image.

ponents), (3) partitions that overlap along the y-axis with other partitions within the same column. The first case identifies table partitions that result from merging cells from different columns of a table into one partition. The second case detects table partitions that consists of a single data cell. The third case identifies table partitions that lie in one column but were not joined together due to the presence of a strong tab-line.

This stage tries to find table partition candidates quite aggressively. This has the advantage that even small evidence of the presence of a table is not missed, since any tables that are missed at this stage will not be recoverable at later stages. The disadvantage of the aggressive approach is that several false alarms may originate, for instance from single word section headings, page headers and footers, numbered equations, small parts of text words in the marginal noise, and line drawing regions. A smoothing filter is applied that detects isolated table partitions that have no other table partition neighbor above or below them. These partitions are removed from the candidate table partition list. The candidate table partitions for our example image are shown in Figure 3(a).

3.2 Detecting Page Column Split

The next step is to detect split in the column layout of the page due to the presence of a table. Such a split occurs when the cells of the table are very well aligned. To detect this case, we divide the page into columns and find the ratio of table partitions in each column. Table columns that were erroneously reported as page columns are easily detected since they have a high ratio of table partition as compared to normal text partitions. However, extra care needs to be taken at this stage to undo a column split (i.e. to merge two columns) since a wrong decision would result in merging two text columns leading to a large numbers of errors in page layout analysis itself.

Therefore, we undo a page column split only if sufficient number of text partitions spanning the two columns are present and the split in the columns starts with table partitions. This extra care prevents merging table columns in full-page tables when there is no flowing text in the page. Since the cost of a wrong decision here is very high in terms of layout analysis errors we chose to perform this step defensively.

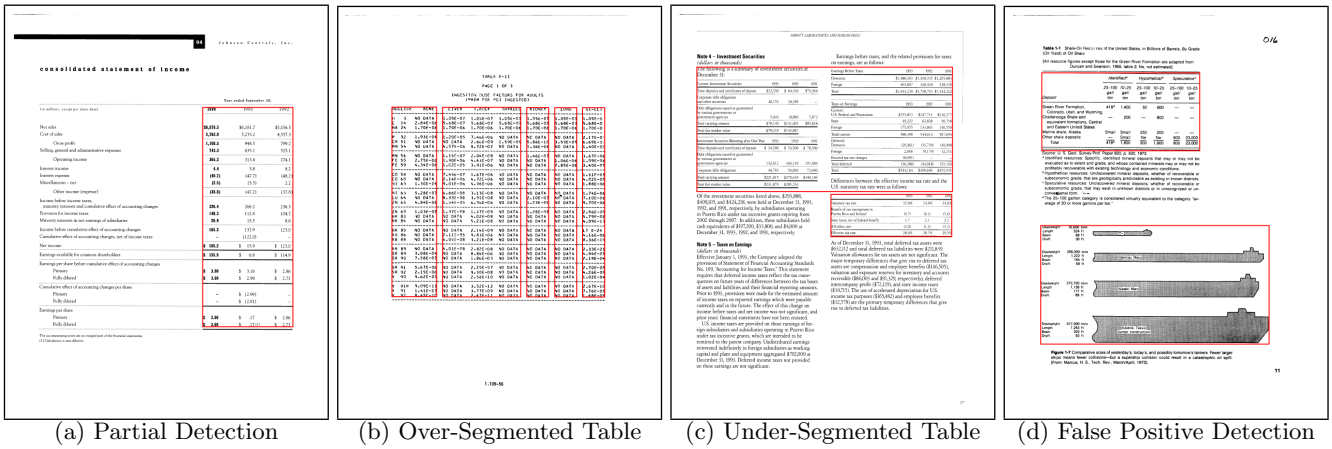
3.3 Locating Table Columns

The goal of this step is to group table partitions into table columns. For this purpose, runs of vertically neighboring table partitions are assigned to a single table column. If a column partition of type “horizontal ruling” is encountered, the run continues. When a partition of any other type is found, the table column obtained so far is finalized. If a table column consists of only one table partition, it is removed as a false alarm. The identified table columns for the example image are shown in Figure 3(b).

3.4 Marking Table Regions

Table columns obtained in the previous steps give a strong hint about the presence of a table in that region. We make a simple assumption here: within a single page column, flowing text does not share space with a table along the y-axis. This assumption holds true for most of the layouts that we encounter in practice since if a table shares space vertically with flowing text, it is hard to see whether the text belongs to the table or not. Based on this assumption, we horizontally expand the boundaries of table columns to the page columns that contain them. Hence we obtain with-in column table regions for each page column.

At this stage, tables that are laid out within one column are correctly identified. However, tables spanning multiple page columns are over-segmented. Although two table regions in neighboring page columns could be merged if their start and end positions align, this might wrongly merge different tables in the two columns. Therefore a merge is carried out only if at least one column partition of any type (text, table,



(a) Partial Detection (b) Over-Segmented Table (c) Under-Segmented Table (d) False Positive Detection

Figure 4: An illustration of different performance measures used in this work. Each figure shows one type of segmentation error that is quantified by the corresponding measure.

horizontal ruling) is found that overlaps with both tables. Table partitions and horizontal ruling partitions that are not included in any table and are directly above or below a table region with a large overlap along the x-axis are also included in the neighboring table. The table regions thus obtained for the example image are shown in Figure 3(c).

3.5 Removing False Alarms

Although most of the false alarms originating from normal text regions are removed in previous stages, other sources of false alarms like marginal noise [17] and figures still remain. Therefore the identified table regions are passed through a simple validity test: a valid table should have at least two columns. False alarms consisting of a single column are removed by analyzing their projection on the x-axis. Projection of a valid table on the x-axis should have at least one zero-valley larger than the global median x-height of the page. Therefore, table candidates that do not have a zero-valley in their vertical projection are removed.

4. PERFORMANCE MEASURES

Different performance measures have been reported in the literature for evaluating table detection algorithms. These range from simple precision and recall based measures [6, 13] to more sophisticated measures for benchmarking complete table structure extraction algorithms [8]. In this paper, since we are only focusing on table spotting, we use standard measures for document image segmentation focusing on the table regions. Hence in accordance with [13, 14, 16, 20] we use several measures for quantitatively evaluating different aspects of our table spotting algorithm.

Both ground-truth tables and tables detected by our algorithm are represented by their bounding boxes. Let G_i represent the bounding box of ith ground-truth table and D_j represent the bounding box of the jth detected table in a document image. The amount of overlap between the two is defined as:

$$A(G_i, D_j) = \frac{2|G_i \cap D_j|}{|G_i| + |D_j|} \quad (1)$$

where $|G_i \cap D_j|$ represents the area of intersection of the two zones, and $|G_i|, |D_j|$ represent the individual areas of the ground-truth and the detected tables. The amount of area overlap A will vary between zero and one depending on the overlap between ground-truth table G_i and detected table D_j. If the two tables do not overlap at all $A = 0$, and if the two tables match perfectly i.e. $|G_i \cap D_j| = |G_i| = |D_j|$, then $A = 1$.

- **Correct Detections:** These are the number of ground-truth tables that have a large overlap ($A \geq 0.9$) with one of the detected tables.
- **Partial Detections:** These are the number of ground-truth tables that have a one-to-one correspondence with a detected table, however the amount of overlap is not large enough ($0.1 < A < 0.9$) to be classified as a correct detection (see Figure 4(a)).
- **Over-Segmented Tables:** These are the number of ground-truth tables that have a major overlap ($0.1 < A < 0.9$) with more than one detected tables. This indicates that different parts of the ground-truth table were detected as separate tables (see Figure 4(b)).
- **Under-Segmented Tables:** These are the number of ground-truth tables that have a major overlap ($0.1 < A < 0.9$) with one detected table, but the corresponding detected table has major overlaps with other ground-truth tables as well. This indicates that more than one table (possibly adjacent) were merged by the detection algorithm and were reported as a single table (see Figure 4(c)).
- **Missed Tables:** These are the number of ground-truth tables that do not have a major overlap with any of the detected tables ($A \leq 0.1$). These tables are regarded as missed by the detection algorithm.
- **False Positive Detections:** These are the number of detected tables that do not have a major overlap with any of the ground-truth tables ($A \leq 0.1$). These tables are regarded as false positive detections since the system mistook some non-table region as a table (see Figure 4(d)).

- **Area Precision:** While the measures defined above help in understanding which types of errors were made by the table detection algorithm, the goal of this measure is to summarize the performance of the algorithm by measuring what percentage of the detected table regions actually belong to a table region in the ground-truth image. A high precision is achieved when the decision about the presence of a table region is made very conservatively.

- **Area Recall:** This measure evaluates the percentage of the ground-truth table regions that was marked as belonging to a table by the algorithm. The concept of precision and recall measures are similar to their use in the information retrieval community [13].

5. EXPERIMENTS AND RESULTS

To evaluate the performance of our table detection algorithm, we chose the UNLV dataset [1]. The UNLV dataset contains a large variety of documents ranging from technical reports and business letters to newspapers and magazines. The dataset was specifically created to analyze the performance of leading commercial OCR systems in the UNLV annual tests of OCR accuracy [15]. It contains more than 10,000 scanned pages at different resolutions and 1000 fax documents. The scanned pages are categorized into bi-tonal and greyscale documents. The bi-tonal documents are again grouped into different scan resolutions (200, 300, and 400 dpi). For each page, manually-keyed ground-truth text is provided, along with manually-determined zone information. The zones are further labeled according to their contents (text, table, half-tone, ...). We picked bi-tonal documents in the 300 dpi class for our experiments since this represents the most common settings for scanning documents. Among these images, 427 pages containing table zones were selected. These page images were further split into a training set of 213 images and a test set of 214 images. The training images were used in the development of the algorithm and different steps of the algorithm were extensively evaluated on these images. The test images were used in the end to evaluate the complete system.

Results of our table detection algorithm on some sample images from the UNLV dataset are shown in Figure 5. Detailed evaluation of the algorithm and its comparison with a state-of-the-art commercial OCR system is given in Table 1 and Figure 6.It should be noted that the ground-truth table zones provided with the UNLV dataset also include the table caption inside the zone. Since table caption is not a tabular structure, it is left out of the table by all OCR systems. Therefore, we edited the ground-truth information by manually marking the table caption regions in all documents. Then this region was excluded from the ground-truth table zones provided with the dataset. This was achieved by shrinking the ground-truth table zones to tightly enclose all foreground pixels that were not part of the table caption. The experimental results show that our system was able to spot table regions with a precision of 86% on the test data. The recall was also quite high (79%) showing a good compromise between precision and recall. The commercial OCR system, on the other hand, had a lower recall (37%) but higher precision (96%).

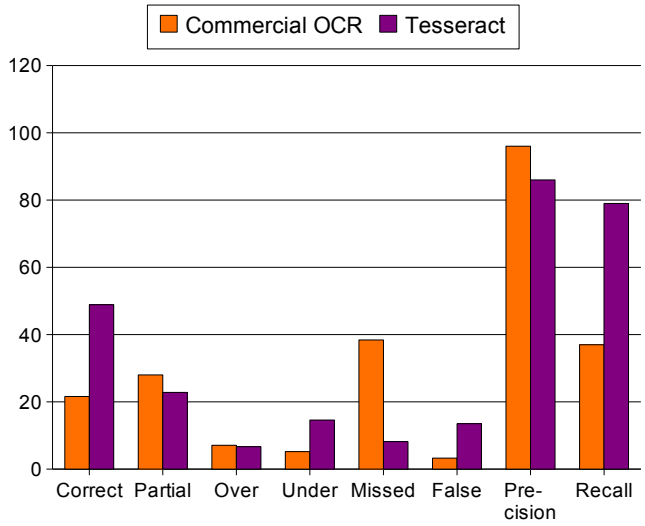


Figure 6: A bar chart of the accuracy of the proposed table detection system with that of a commercial OCR on UNLV test set (214 page containing 268 tables).

Some of the errors made by our algorithm are shown in Figure 4. An analysis of the results shows that the major source of errors are full-page tables. In these cases, the column finding algorithm reports several columns of text. Since newspapers also have several text columns, without using *a priori* knowledge about the type of documents (report, newspaper, ...) it is hard to detect that the large number of columns are due to a full-page table. One typical example is a page containing "table of contents". Such pages are marked as table regions in the ground-truth information provided with the UNLV dataset. However, our algorithm regards them as regular text pages hence either missing these "tables" completely or partially detecting them.

The false positive detection made by our algorithm were also analyzed. We noticed an interesting side-effect of our algorithm. Since many graphics regions have text inside them that is spaced apart, such regions were also spotted as tables. Although such cases were reported as false alarms, in some cases it might be beneficial to additionally spot graphics regions as well. Other cases of false alarms originated from tabulated equations. False alarms in pure text regions were quite rare.

6. CONCLUSION

This paper presented a table detection algorithm as part of the Tesseract open source OCR system. The presented algorithm uses components of the layout analysis module of Tesseract to locate tables in documents having a large variety of layouts. Experimental results on different classes of documents (company reports, journal articles, newspaper articles, magazine pages) from the UNLV dataset showed that our table detection algorithm competes well with that of a commercial OCR system with a much higher recall and slightly lower precision. We plan to extend this work in the direction of table structure extraction in future.

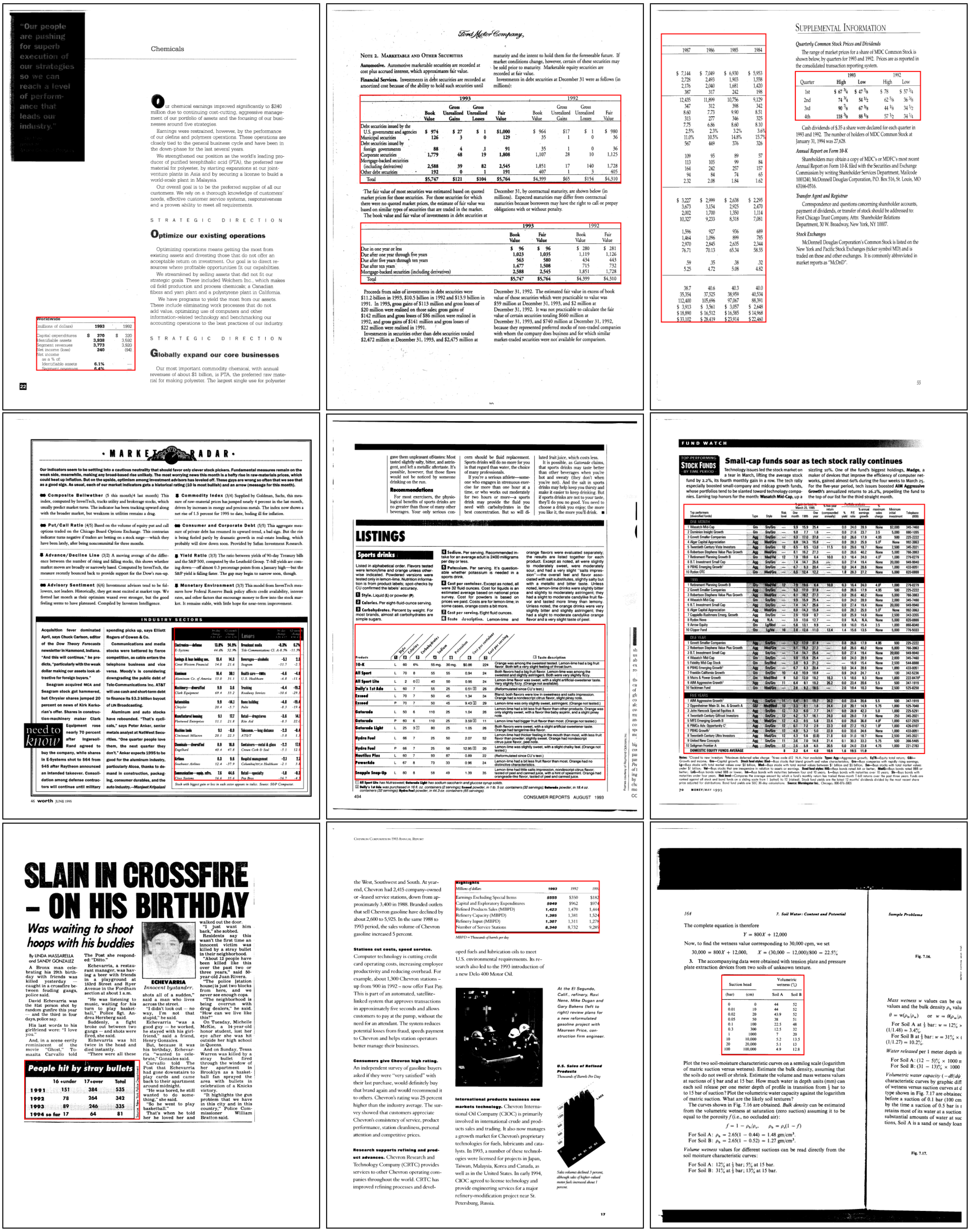


Figure 5: Some sample images from the UNLV dataset showing the table spotting results of our algorithm.

Table 1: Results of evaluating a commercial OCR system and the proposed table detection algorithm on the 427 binary 300-dpi scanned UNLV dataset pages containing table zones.

	Training Images (302 tables)		Test Images (268 tables)	
	Commercial System	Tesseract	Commercial System	Tesseract
Correct Detections	79	130	58	131
Partial Detections	66	65	75	61
Over-Segmented Tables	25	30	19	18
Under-Segmented Table	17	55	14	39
Missed Tables	120	31	103	22
False Positive Detections	6	17	7	29
Area Precision	97.4%	90%	96.3%	86%
Area Recall	40.7%	78%	36.7%	79%

7. REFERENCES

[1] http://www.isri.unlv.edu/ISRI/OCRtk.

[2] F. Cesarini, S. Marinai, L. Sarti, and G. Soda. Trainable table location in document images. In *Proc. Int. Conf. on Pattern Recognition*, pages 236–240, Quebec, Canada, Aug. 2002.

[3] A. C. e Silva. Learning rich hidden markov models in document analysis: Table location. In *Proc. Int. Conf. on Document Analysis and Recognition*, pages 843–847, Barcelona, Spain, July 2009.

[4] B. Gatos, D. Danatsas, I. Pratikakis, and S. J. Perantonis. Automatic table detection in document images. In *Proc. Int. Conf. on Advances in Pattern Recognition*, pages 612–621, Path, UK, Aug. 2005.

[5] I. Guyon, R. M. Haralick, J. J. Hull, and I. T. Phillips. Data sets for OCR and document image understanding research. In H. Bunke and P. Wang, editors, *Handbook of character recognition and document image analysis*, pages 779–799. World Scientific, Singapore, 1997.

[6] J. Hu, R. Kashi, D. Lopresti, and G. Wilfong. Medium-independent table detection. In *Proc. SPIE Document Recognition and Retrieval VII*, pages 291–302, San Jose, CA, USA, Jan. 2000.

[7] J. Hu, R. S. Kashi, D. Lopresti, and G. Wilfong. Experiments in table recognition. In *Proc. Int. Workshop on Document Layout Interpretation and Applications*, Seattle, WA, USA, Sep. 2001.

[8] J. Hu, R. S. Kashi, D. Lopresti, and G. Wilfong. Evaluating the performance of table processing algorithms. *Int. Jour. on Document Analysis and Recognition*, 4(3):140–153, 2002.

[9] D. Keysers, F. Shafait, and T. M. Breuel. Document image zone classification - a simple high-performance approach. In *2nd Int. Conf. on Computer Vision Theory and Applications*, pages 44–51, Barcelona, Spain, Mar. 2007.

[10] T. Kieninger and A. Dengel. A paper-to-HTML table converting system. In *Proc. Document Analysis Systems*, pages 356–365, Nagano, Japan, Nov. 1998.

[11] T. Kieninger and A. Dengel. Table recognition and labeling using intrinsic layout features. In *Proc. Int. Conf. on Advances in Pattern Recognition*, Plymouth, UK, Nov. 1998.

[12] T. Kieninger and A. Dengel. Applying the T-RECS table recognition system to the business letter domain. In *Proc. Int. Conf. on Document Analysis and Recognition*, pages 518–522, Seattle, WA, USA, Sep. 2001.

[13] T. Kieninger and A. Dengel. An approach towards benchmarking of table structure recognition results. In *Proc. 8th Int. Conf. on Document Analysis and Recognition*, pages 1232–1236, Seoul, Korea, Aug. 2005.

[14] S. Mandal, S. Chowdhury, A. Das, and B. Chanda. A simple and effective table detection system from document images. *Int. Jour. on Document Analysis and Recognition*, 8(2-3):172–182, 2006.

[15] S. V. Rice, F. R. Jenkins, and T. A. Nartker. The fourth annual test of OCR accuracy. Technical report, Information Science Research Institute, University of Nevada, Las Vegas, 1995.

[16] F. Shafait, D. Keysers, and T. M. Breuel. Performance evaluation and benchmarking of six page segmentation algorithms. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 30(6):941–954, 2008.

[17] F. Shafait, J. van Beusekom, D. Keysers, and T. M. Breuel. Document cleanup using page frame detection. *Int. Jour. on Document Analysis and Recognition*, 11(2):81–96, 2008.

[18] R. Smith. An overview of the Tesseract OCR engine. In *Proc. 9th Int. Conf. on Document Analysis and Recognition*, pages 629–633, Curitiba, Brazil, Sep. 2007.

[19] R. Smith. Hybrid page layout analysis via tab-stop detection. In *Proc. Int. Conf. on Document Analysis and Recognition*, pages 241–245, Barcelona, Spain, July 2009.

[20] Y. Wang, R. Haralick, and I. T. Phillips. Automatic table ground truth generation and a background-analysis-based table structure extraction method. In *Proc. Int. Conf. on Document Analysis and Recognition*, pages 528–532, Seattle, WA, USA, Sep. 2001.

[21] Y. Wang, I. Phillips, and R. Haralick. Document zone content classification and its performance evaluation. *Pattern Recognition*, 39(1):57–73, 2006.