Table structure understanding and its performance evaluation

Yalin Wang[a],*, Ihsin T. Phillips[b] , Robert M. Haralick[c]

[a] *Department of Electrical Engineering, University of Washington, Seattle, WA 98195, USA*
[b] *Department of Computer Science, Queens College, CUNY, Flushing, NY 11367, USA*
[c] *The Graduate School, CUNY, New York, NY 10016, USA*

Abstract

This paper presents a table structure understanding algorithm designed using optimization methods. The algorithm is probability based, where the probabilities are estimated from geometric measurements made on the various entities in a large training set. The methodology includes a global parameter optimization scheme, a novel automatic table ground truth generation system and a table structure understanding performance evaluation protocol. With a document data set having 518 table and 10,934 cell entities, it performed at the 96.76% accuracy rate on the cell level and 98.32% accuracy rate on the table level.
© 2004 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.

Keywords: Pattern recognition; Document image analysis; Document layout analysis; Table structure understanding; Performance evaluation; Non-parametric statistical modeling; Optimization

1. Introduction

For the last three decades, document image analysis researchers have successfully developed many outstanding methods for character recognition, page segmentation and understanding of text-based documents. Most of these methods were not designed to handle documents containing complex objects, such as tables. Tables are compact and efficient for presenting relational information and most of the documents produced today contain various types of tables. Thus, table structure understanding is an important problem in the document layout analysis field. Its application can be found in image-XML conversion, information retrieval, and document classification, etc.

In this paper, we formulate the table structure understanding problem in the whole document hierarchy. We use a statistical based table structure understanding algorithm.

To systematically evaluate and optimize the algorithms, we use a performance evaluation protocol employing an area overlapping measure. Using this protocol, our global parameter optimization scheme is able to adaptively determine optimum tuning parameter values.

Ground truthing is tedious and time-consuming. We developed an automatic table ground truthing system that can analyze any given ground truth tables and generate documents having similar table elements while adding more variety to both table and non-table parts. Using our content matching ground truthing idea, the table ground truth data for the generated table elements become available with little manual work. We make this software package publicly available.

The remainder of the paper is organized as follows. We give a literature review in Section 2. We present a document structure model and a formal table structure understanding problem statement in Section 3. In Section 4, we present a framework to solve the table structure understanding problem, in which each step is discussed in detail. Large data sets with ground truth are essential in assessing the performance of computer vision algorithms. However, as of the writing of this paper, there are no publicly available table ground truth data sets. We developed a software package that can

---

* Corresponding author. Mathematics Department, Box 951555, UCLA, Los Angeles, CA 90095, USA. Tel.: +1-310-825-8525; fax: +1-310-206-6673.
E-mail addresses: ylwang@math.ucla.edu (Y. Wang), yun@image.cs.qc.edu (I.T. Phillips), haralick@gc.cuny.edu (R.M. Haralick).

simulate any given table ground truth with additional controlled variety. Section 5 describes how we generated the random tables. In Section 6, we describe a table structure understanding performance evaluation protocol based on rectangular overlap. In Section 7, our detailed experimental results are given and compared with other competing algorithms. Conclusions and future work are given in Section 8.

2. Literature review

Tables are an important means for communicating information in documents, and understanding such tables is a challenging problem in document layout analysis. Although there are different document formats, e.g. document images, ASCII text documents, web documents, etc., here we focus on table structure understanding in document images.

Some tables are similar to ruled forms. Many papers were published for form processing [1–3]. In form data extraction, templates are constructed from empty forms and correlated with filled-in forms. Form processing emphasizes extracting data from some given forms. Table structure understanding mainly deals with those tables embedded in the documents. The tables usually have an unknown format. Table structure understanding algorithm segments tables from the document pages and extracts the information from these tables.

The table structure understanding problem includes two subproblems: the table detection problem and the table decomposition problem. Since table detection is a crucial step for table structure understanding, it is also a topic which is of interest to many researchers. By the strategies or methods in which algorithms are used in the table detection, the algorithms can be classified into three categories: (1) pre-defined table layout based; (2) heuristics based; and (3) statistical or optimization based. The pre-defined table layout based algorithms use a set of strict, pre-defined table layout information to detect tables. For a given type of image, it is usually able to have a satisfactory detection performance. However, its extension ability is very limited. The heuristics based algorithms use a set of rules or pre-defined syntax rules of table grammar to derive decisions. The complex heuristics are usually based on local analysis. It sometimes has a even more complicated post-processing part. As for statistical or optimization based algorithms, they either do not need parameters or the needed free parameters which are used in the process are obtained by an off-line training process. We now summarize some selected algorithms within the above defined categories.

Chandran and Kasturi [4] gave a clear algorithm to extract the structure of the table, regardless of the presence or the absence of lines. For the class of tables satisfying their assumption, their method worked well.

Green and Krishnamoorthy [5] developed a strategy for extracting the underlying relational information from the images of printed tables. Given a table model, the visual clues that exist in the images were used for extracting first the physical, and then the logical structure of the tables. A table model was used to extract logical table information from table images.

Shamilian et al. [6] described the architecture of a system for reading machine-printed documents in known pre-defined tabular-data layout styles. In these tables, textual data are presented in record lines made up of fixed-width fields. The row-and-column structure of horizontal tables suggests an analogy with relational databases. The system was applied to more than 400 distinct tabular layouts.

Zuyev [7] introduced a concept of a table grid that can serve for advanced methods of table structure analysis. Table hypothesis generation was guided by visual clues. Classification of a generated hypothesis required some threshold values that were obtained by an analysis of the connected components projection profile.

Kieninger et al. [8,9] presented a bottom-up approach to the table detection algorithm. First an arbitrary word was selected as the seed. Then the local adjacency connection relation was used to expand to a whole text block. Later, some post-processing to the initial block segments was done to get refined results.

A dynamic programming table detection algorithm was given in Hu et al. [10]. By defining table quality measures, it detected tables based on computing an optimal partitioning of a document into some number of tables. Its high-level framework is independent of any particular table quality measure and independent of the document medium. The algorithm works for both ASCII and image documents. According to their global evaluation protocol, the algorithm yields a recall rate of 83% for 25 ASCII documents and 81% for 25 scanned images, and a precision rate of 91% and at 93%, respectively.

Klein et al. [11] introduced three approaches for an industrial document analysis system. These approaches include: searching for a set of known table headers, searching for layout structures which resemble parts of columns, and searching for groupings of similar lines. Approach 1 was not tolerant enough toward some kinds of even minor aberrations and was able to spot about 80% of all table headers. Approach 2 yielded 90% correctness in a test on 1200 real documents. No experimental results were reported for approach 3.

Two papers [12,13] reported their research on the table decomposition problem. Handley [13] presented a table analysis system which reconstructed table formatting information from table images no matter whether or not the cells were explicitly delimited. Inputs to the system were word bounding boxes and any horizontal and vertical lines that delimit cells. Using a sequence of carefully crafted rules, multi-line cells and their inter-relationships are found even though no explicit delimiters are visible. Hu et al. [12] presented algorithms that recognize tables in ASCII text. First hierarchical clustering was used to identify columns and then spatial and lexical criteria were used to classify headers. The algorithm was tested on 26 Wall Street Journal articles in text format (WSJ database) and 16 email messages. The overall

agreement was 82% for the WSJ documents and 73% for the email documents.

Hu et al. [14] introduced intuitive, easy-to-implement evaluation schemes for the related problem of table detection and table structure recognition. They first considered different errors of table detection problem. Then they addressed the problem of evaluating table structure recognition by using a directed acyclic attribute graph, or table DAG. The so-called "graph probing" is a general concept that could be applied to other document recognition tasks as well.

The principle that for every document analysis task there exists a mechanism for creating well-defined ground truth is a widely held tenet. Past experience in the research community with standard data sets providing ground truth for character recognition and page segmentation tasks supports this belief. However, Hu et al. [15] reported a number of serious hurdles connected with the groundtruthing of tables from UW Document Image Database (UWI) [16]. From their experience, there may exist more than one acceptable "truth" and/or incomplete or partial "truth" for table entities.

Although there is continuing interest in the table understanding problem, there are no publicly available table ground truth data sets. In the UW document image database III (UW CDROM III) [17], there are 215 marked table zones but no structure data for them. Detailed table structure information is required for a table detection system evaluation [18]. Clearly, UW CDROM III cannot be directly used to evaluate table detection system.

Nonetheless, large data sets with ground truth are essential in assessing the performance of computer vision algorithms. Manually generating document ground truth proved to be very costly. According to Hu et al.'s research [15], there may exist more than one acceptable "truth" and/or incomplete or partial "truth". Such problems occur because of inherent ambiguity or ground truth bias or inconsistency. They are not easily eliminated. However, using synthetic data [19] at some research phase is a common practice in computer vision field. It has the advantage of extremely low cost, automatic creation of ground truth information, less bias aberrations and more variety than the real images. So far, the problem of generation of synthetic table data sets for research has not received enough attention from the researchers in the document layout analysis field.

3. Table structure understanding problem statement

3.1. Document structure model

We formally define a *Rectangle Layout Structure* (*RLS*) as a triple (C, R, Q), where

- R is a rectangular area;
- Q is a physical label type (e.g. page, textblock, table, etc.);

- C is either empty or a set of rectangle layout structures $\{C_n, R_n, \theta_n\}_{n=1}^N$, satisfying that $\{R_1, \ldots, R_n\}$ is a cover of R.

We denote by $\mathscr{D}$ the set of RLSs in a given document image. Some functions are associated with $\mathscr{D}$.

1. *Attributes*
   - We denote by $\mathscr{F}$ the set of format attributes (column number, row number, column justification, etc.).
   - $S: \mathscr{D} \to F$ specifies the format attributes for each rectangle layout structure.
2. *Measurements*
   - We denote by Λ the measurement space.
   - $V : \wp(\mathscr{D}) \to \Lambda$ specifies measurement made on subset of $\mathscr{D}$.

With this model, a *table* is a RLS (C, R, Q), where

- Q is the label table.
- The RLS in C must have labels from a set {table body, row header, column header, horizontal/vertical blank block}.
- One RLS of C must have the label table body.
- At least one RLS of C must have the label horizontal blank block.
- At least one RLS of C must have the label vertical blank block,

where horizontal/vertical blank block are a continuous area of background pixels. Their formal definitions can be founded at [20].

Similarly, we defined *page*, *textblock*, *table row header*, *table column header and table body*, etc. [20]. We give an example of a document hierarchy model using this idea in Fig. 1.

3.2. Problem statement

The table structure understanding problem has two sub-problems: the table detection problem and the table decomposition problem. With the defined document structure model, we can define *tablezone, textzone, documentpage* [20]. Fig. 2(a) shows a document hierarchy including documentpage, tablezone and textzone entities. The problem of *table detection* can be formulated as: *Given a page ϕ having a rectangle area R and a set of words, W, table detection constructs a* RLS (C, R, Q), *where*

- *Q is the label documentpage.*
- *$P((C, R, Q)|\phi)$ is maximized.*
- *Each word of W participates in exactly one RLS in C or in its descendants.*

Fig. 2(b) shows a table structure. Under tablezone level, we have row header, column header and cell entities. Given a detected table entity, the *table decomposition* problem is
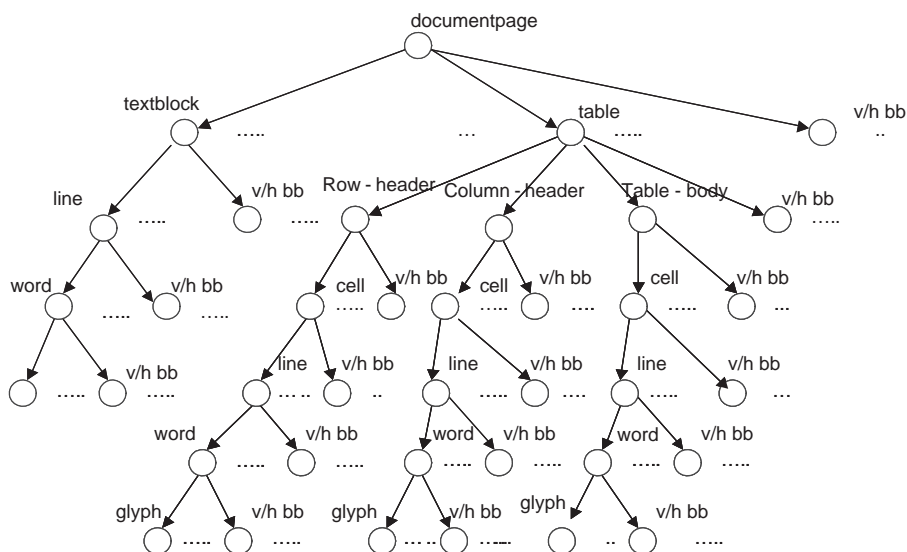
Fig. 1. Illustrates a document hierarchy model, where "*v/hbb*" stands for vertical/horizontal blank block.
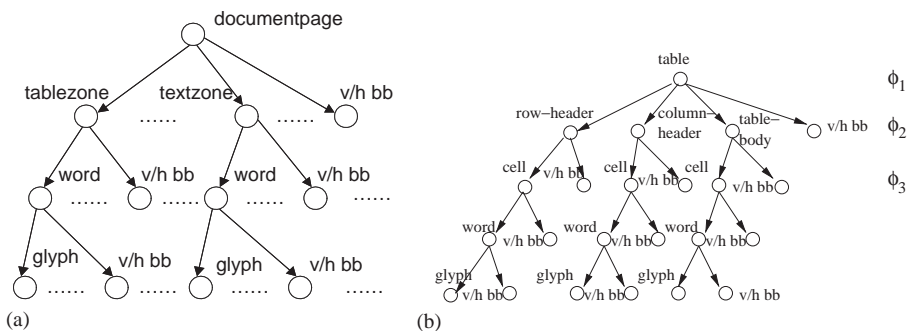
Fig. 2. (a) Illustrates a table hierarchy model for table detection problem, where "*v/hbb*" stands for vertical/horizontal blank block; (b) illustrates a table hierarchy model for table decomposition problem.

to determine its structure and identify its elements such as row/column headers, cells, etc.

4. Table structure understanding algorithm

In this section, we discuss a framework to solve the table structure understanding problem. We present the algorithm details for each step. Among these steps, column style labeling, statistical refinement, iterative updating optimization algorithms are probability based, where the probabilities are estimated from geometric measurements made on the various entities with which the algorithm works in a large training set. The off-line probabilities estimated in the training then drive all decisions in the on-line segmentation module. Thus our algorithms have the advantages such as domain independence, and easy extension.

Fig. 3 gives an overview of the table structure understanding algorithm. Input data to our table structure understanding system are the segmented line and word entities with roughly separated regions [21]. Fig. 4 shows an example of an input image. Assuming the maximum number of columns is two in our data set, we designed a column style labeling algorithm which can label a given page by one of the three column styles: double column, single column with marginal note and single or mixed column style. Using background analysis, we find table candidates by looking for large horizontal blank block equivalence subsets. For the identified table candidates, a statistical refinement module was used to refine the table detection results. Then we consider the table detection problem as an optimization problem. The optimization not only considers the probability of table entities, but also the probability of text block segmentation results. It attempts a probability optimization on the table and its
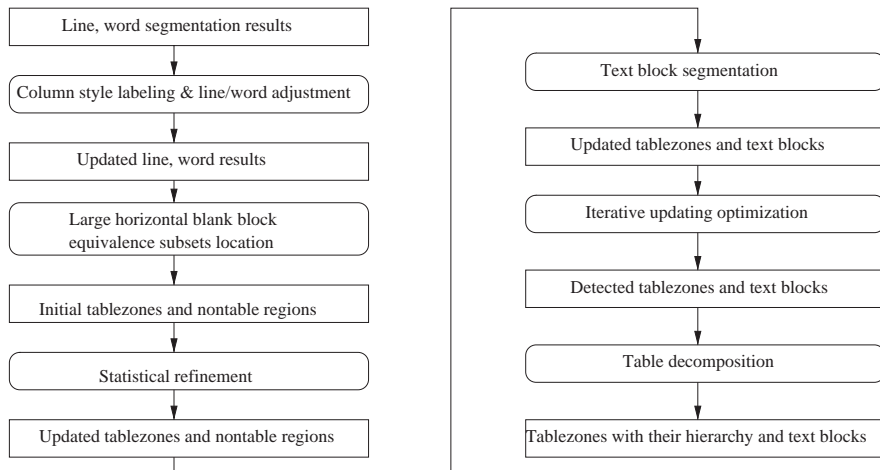
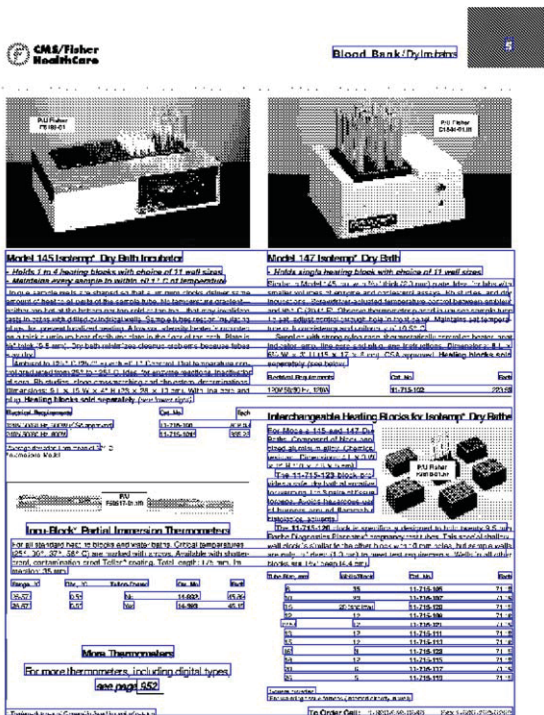Fig. 3. Overview of the table structure understanding algorithm.

Fig. 4. An example of input data to table structure understanding algorithm. The graphic parts in the image have been filtered. Segmented line entities are shown.

adjacent text block segmentation results by an iterative updating method. Our table decomposition algorithm is similar to the X–Y cut technique [22]. We obtain the cell entities by an analysis of word vertical projection results. In each step, some tuning parameters are used. To unify the whole approach and improve the algorithm performance, we employ a global parameter optimization scheme which can optimize the tuning parameter values by using our performance evaluation protocol and some maximum of function methods.

The different stages are shown in Fig. 3. We used a text block segmentation method [23] to get block segmentation results. Other steps, column style labeling, large horizontal blank block equivalence subsets location, statistical refinement, iterative updating optimization, table decomposition algorithm, are described in Sections 4.1–4.5, respectively. The global parameter optimization scheme is introduced in Section 4.6 (Figs. 5 and 6).

4.1. Column style labeling

The motivation of column style labeling is to determine the column type based on the background features. Then by the labels, we update the line detection results. This is a pre-processing for our table structure understanding system.

Let $\mathscr{A}$ be a set of the document pages in our data set. Let $\mathscr{L}$ be a label set of page column style, {double column, single column with marginal note, single or mixed column}. The function $f\colon \mathscr{A} \to \mathscr{L}$ associates each page of $\mathscr{A}$ with a page column style. The function $V\colon \mathscr{A} \to \varLambda$ specifies measurements made on each element of $\mathscr{A}$, where $\varLambda$ is the measurement space.

The page column style classification problem can be formulated as follows: *Given a page set $\mathscr{A}$ and a page column style set $\mathscr{L}$, find a classification function $f\colon \mathscr{A} \to \mathscr{L}$, that maximizes $P(f(\mathscr{A})|V(\mathscr{A}))$.*

We assume conditional independence between the page column style classifications, so the probability in the above equation may be decomposed as $P(f(\mathscr{A})|V(\mathscr{A})) = \prod_{\tau \in \mathscr{A}} P(f(\tau)|V(\tau))$. The problem can be solved by
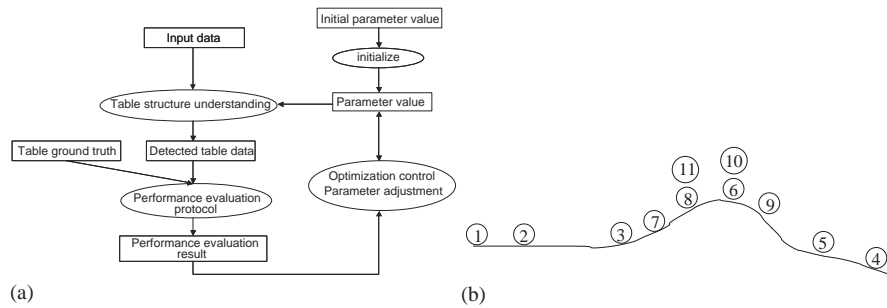
Fig. 5. (a) Diagram of global parameter optimization scheme. (b) Example of a line search method. The starting position is 1, the search direction is right first. The search direction is flipped at 4. It keeps searching until 11.
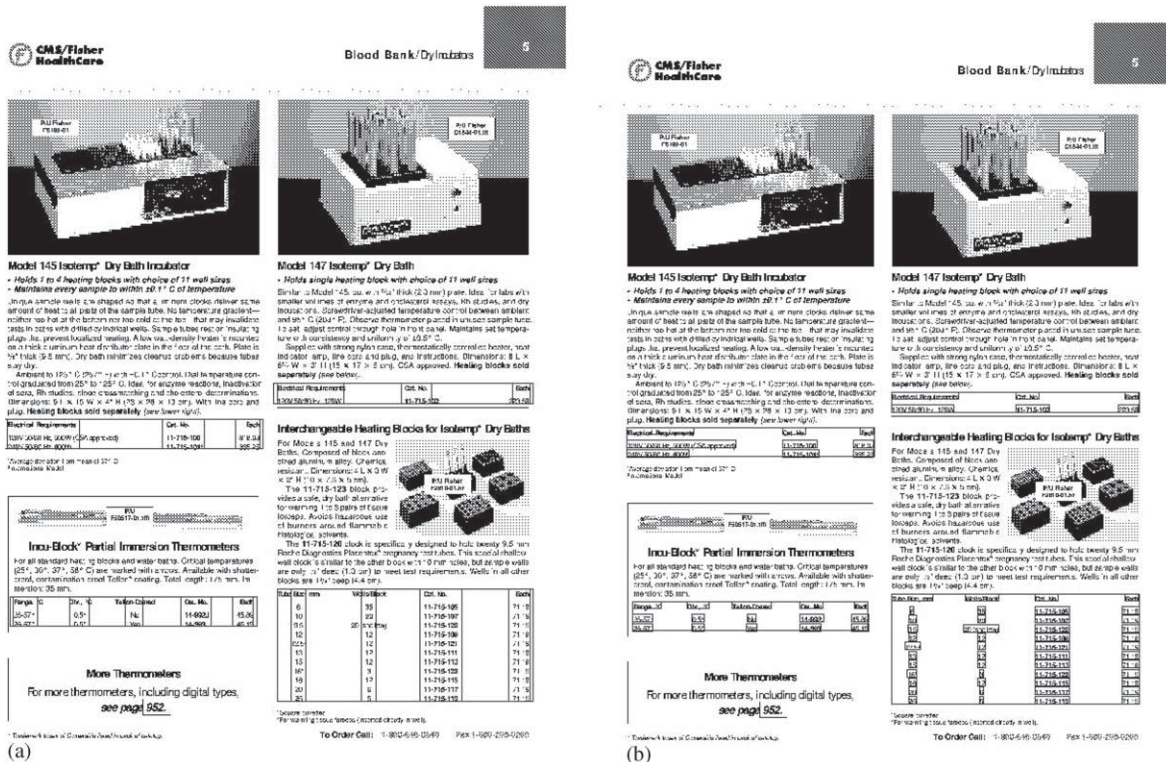

Fig. 6. (a) Located table column entities within the table entities in image in Fig. 4. Table column and table entities are shown. (b) The result of table understanding algorithm. Table cell and table entities are shown.

independently maximizing each individual probability $P(f(\tau)|V(\tau))$ in the above equation, where $\tau \in \mathcal{A}$. The probabilities are estimated from the training set.

The features of $V(\tau)$, are functions of the geometry of the vertical blank block separator on the page τ. There is one vertical blank block separator on each page. A vertical blank block $\mathcal{VR}$ (see Ref. [24] for definition) with b_r rows and b_c columns and with lefttop vertex coordinate (x_{b1}, y_{b1}), is a *vertical blank block separator*, $\mathcal{BK} = (b_r, b_c, x_{b1}, y_{b1})$,

if and only if it satisfies the following conditions:

- Its page normalized number of column is large enough. Specifically, $b_c/mw > 3$, where mw is the median width of text glyphs in the whole page.
- It has the largest number of rows among all the blank blocks. If there is more than one blank block having this largest number of rows, the one with largest number of column is selected. If there is more than one with the same

largest number of column, then one of them is randomly selected.

We use two features for the page column style classifier. They are the length ratio and position ratio of the page's blank block separator, $\mathscr{BK} = (b_r, b_c, x_{b1}, y_{b1})$:

1. The length ratio, $l_{bk} = b_r/lv_r$, where lv_r is the row number of the page live-matter part.
2. Position ratio $p_{bk} = (x_{b1} + b_c/2)/(x_{lv} + lv_c/2)$, where x_{lv} is the column coordinate of the lefttop vertex of the page live-matter part and lv_c is the number of columns of the page live-matter part.

After we identify the column style, we make adjustments to line and word segmentation results according to the labeled column style.

4.2. Large horizontal blank block equivalence subsets location

The next step of the algorithm locates all the horizontal blank blocks (as defined in Ref. [24]) of the page and groups them into a collection of equivalence subsets. The foreground entities adjacent to the equivalence subsets are the table candidates.

Let $\mathscr{B}$ be the set of horizontal blank blocks in a page column. $\mathscr{R}$ is the set of real numbers. Function $top: \mathscr{B} \to \mathscr{R}$ associates a horizontal blank block with its top y coordinate.

For a pair of horizontal blank block a and b, where $a \in \mathscr{B}$ and $b \in \mathscr{B}$, the horizontal distance $d_h(a, b)$ and vertical distance $d_v(a, b)$ between them are defined as

$$d_h(a,b) = \begin{cases} x_b - x_a - w_a & \text{if } x_b > x_a + w_a, \\ x_a - x_b - w_b & \text{if } x_a > x_b + w_b, \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

$$d_v(a,b) = \begin{cases} y_b - y_a - h_a & \text{if } y_b > y_a + h_a, \\ y_a - y_b - h_b & \text{if } y_a > y_b + h_b, \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

The horizontal overlap $o_h(a, b)$ and vertical overlap $o_v(a, b)$ between a and b are defined as

$$o_h(a,b) = \begin{cases} x_a + w_a - x_b & \text{if } x_b > x_a, x_b < x_a + w_a, \\ x_b + w_b - x_a & \text{if } x_a > x_b, x_a < x_b + w_b, \\ 0 & \text{otherwise,} \end{cases} \quad (3)$$

$$o_v(a,b) = \begin{cases} y_a + h_a - y_b & \text{if } y_b > y_a, y_b < y_a + h_a, \\ y_b + h_b - y_a & \text{if } y_a > y_b, y_a < y_b + h_b, \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

Figs. 7(a) and (b) show examples of defined d_h, d_o, o_h and o_v.

We define a as a *low neighbor* of b if $top(a) \geqslant top(b)$ and $O_h(a, b) > 0$. We define a as the *immediate low neighbor* of b if and only if

1. a is a low neighbor of b;
2. $top(a) \leqslant top(a')$, where a' is a low neighbor of b.

We define a relation $\mathscr{G}$ that pairs together those horizontal blank blocks that are vertical adjacent. Define $\mathscr{G} \subset \mathscr{B} \times \mathscr{B}$, $\mathscr{G} = \{(a, b) \in \mathscr{B} \times \mathscr{B} | a \text{ and } b \text{ are vertically adjacent}\}$. Define relation $\mathscr{G} \cdot \mathscr{H} = \{(a, c) | \exists b, (a, b) \in \mathscr{G}, (b, c) \in \mathscr{H}\}$, where $\mathscr{G} \subset \mathscr{B} \times \mathscr{B}$ and $\mathscr{H} \subset \mathscr{B} \times \mathscr{B}$.

The transitive closure of $\mathscr{G}^{\mathscr{T}}$ is defined as $\mathscr{G}^{\mathscr{T}} = \bigcup_{i=1}^{\infty} \underbrace{\mathscr{G} \cdots \mathscr{G}}_{i \text{ times}}$.

It is easy to prove that the *vertically adjacent* relation satisfies symmetric and reflexive properties. Thus $\mathscr{G}^{\mathscr{T}}$ defines a set of equivalence subsets. Two equivalence subsets are shown in Fig. 7(c).

After we identify the large horizontal blank block equivalence subsets, we can do a horizontal projection of the horizontal blank block equivalence subsets. We can get different regions of the projection. The union of all the words which horizontally overlap ($oh > 0$, oh as defined in Eq. (3)) with a region becomes a table candidate.

4.3. Statistical refinement

For the identified table candidates, we do a vertical projection on the bounding boxes of the words. Because of the table structure, we can expect the vertical projection to have peaks and valleys. Each table column is determined by the valley at its left and the valley at its right. The intersection of the table columns with the rows are the table cells. Clearly, the table candidates have many false alarms among them. A statistical table refinement algorithm is used to validate each table candidate.

For each table candidate, three features are computed.

- Vertical blank block [24] area ratio, ra, taken over each table's area. Let t be an identified table and $\mathscr{B}$ be the set of large vertical blank blocks in it, $ra = \frac{\sum_{\beta \in \mathscr{B}} Area(\beta)}{Area(t)}$;
- Maximum baseline difference mc. Denote the set of the cells in a row i as $\mathscr{RC}_i$, $\mathscr{RC}_i = \{c_{i,1}, c_{i,2}, \ldots, c_{i,i_m}\}$. Denote the set of $\mathscr{RC}_i$ as $\mathscr{RC}$, $\mathscr{RC} = \{\mathscr{RC}_i, i = 1, \ldots, m\}$, where m is the number of rows in the table. Let $baseline(c)$ be the y coordinate of the cell entity bottom line:

$$mc = \max_{\mathscr{RC}_i \in \mathscr{RC}} (\max_{c_{i,j} \in \mathscr{RC}_i} (baseline(c_{i,j})) \\ - \min_{c_{i,j} \in \mathscr{RC}_i} (baseline(c_{i,j})))$$

- Column justification difference, J_i. Denote the set of cells in a column, i, in the table body region $\mathscr{CC}_i = \{c_{i,1}, c_{i,2}, \ldots, c_{i,i_n}\}$. Denote the set of $\mathscr{CC}_i$ as $\mathscr{CC}$, $\mathscr{CC} = \{\mathscr{CC}_i, i = 1, \ldots, n\}$, where n is the column number in the table. Let $B = (x_{i,j}, y_{i,j}, w_{i,j}, h_{i,j})$ be the bounding
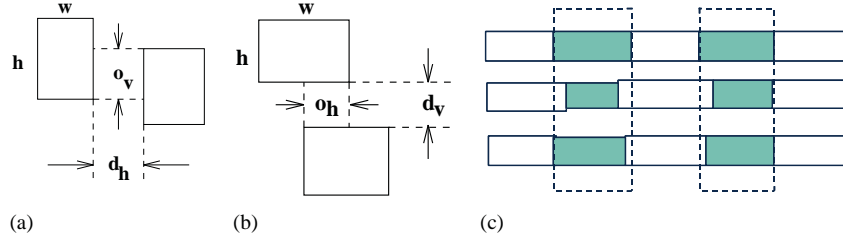
Fig. 7. (a) and (b) illustrate the spatial relations between two bounding boxes that are (a) horizontally adjacent (b) vertically adjacent. (c) Illustrates two examples of equivalence subsets of large horizontal blank blocks. Each gray area is a horizontal blank block between the cells. The equivalence subsets are shown by the dash rectangles.

box of the cell $c_{i,j} \in \mathcal{CC}_i$, where $(x_{i,j}, y_{i,j})$ is the left most point coordinate, $w_{i,j}$ is the width of the bounding box, and $h_{i,j}$ is the height of the bounding box. We estimate the justification of a column, $i, i = 1, \ldots, n$, by computing the vertical projection of the left, center, and right edge of $c_{i,j}, j = 1, \ldots, i_n$:

$$C_{left}[i] = max_{c_{i,j} \in \mathcal{CC}_i}(x_{i,j}) - min_{c_{i,j} \in \mathcal{CC}_i}(x_{i,j}),$$

$$C_{center}[i] = max_{c_{i,j} \in \mathcal{CC}_i}(x_{i,j} + w_{i,j}/2) - min_{c_{i,j} \in \mathcal{CC}_i}$$
$$(x_{i,j} + w_{i,j}/2),$$

$$C_{right}[i] = max_{c_{i,j} \in \mathcal{CC}_i}(x_{i,j} + w_{i,j}) - min_{c_{i,j} \in \mathcal{CC}_i}(x_{i,j} + w_{i,j}),$$

$$J_i = min\{C_{left}[i], C_{center}[i], C_{right}[i]\}.$$

The maximum difference of the justification in a column, mj, is computed as: $mj = max(J_i), i = 1, \ldots, n$.

Then we can compute the table consistent probability for table t as

$$P(consistent(t)|ra(t), mc(t), mj(t)).$$

If $P(consistent(t)|ra(t), mc(t), mj(t)) > 0.5$, we label the table candidate as a table entity.

4.4. Iterative updating optimization

The large horizontal blank block equivalence subsets location and statistical refinement detect tables by considering the local and intrinsic features of table entities. It serves as the means by which an initial starting point may be obtained for the iterative updating optimization that maximizes the probability of the whole page segmentation. That is, not only do we need to get maximum probability on table entities, but also have to maximize the whole page segmentation probability. The whole page segmentation probability includes the probability of table detection results, the probability of text block segmentation results and the probability of the adjacent relationships between table entities and text block entities. Based on this idea, we formulate the table detection problem as a labeling and partitioning problem. The

goal of the problem is to find an optimal solution to label and partition table and text block entities in a given page.

The remainder of the section is organized as follows. We give the problem statement in Section 4.4.1. The probability estimation is stated in Section 4.4.2. We present our algorithm details in Section 4.4.3.

4.4.1. Problem statement

Let $\mathcal{A}$ be a set of segmented zone entities. Let $\mathcal{L}$ be a set of content labels, {table, text-block}. Function $f: \mathcal{A} \to \mathcal{L}$ assigns each element of $\mathcal{A}$ with a label. Function $V: \wp(\mathcal{A}) \to \Lambda$ computes measurements made on subset of $\mathcal{A}$, where Λ is the measurement space.

We define a probability of labeling and measurement function as

$$P(V(\tau): \tau \in \mathcal{A}, f|\mathcal{A}) = P(V(\tau): \tau \in \mathcal{A}|f, \mathcal{A})P(f|\mathcal{A}). \quad (5)$$

By making the assumption of conditional independence that when the label $f_\tau, \tau \in \mathcal{A}$ is known, no knowledge of other labels will alter the probability of $V(\tau)$, we can decompose the probability in Eq. (5) into

$$P(V(\tau): \tau \in \mathcal{A}|f, \mathcal{A}) = \underbrace{\prod_{\tau \in \mathcal{A}} P(V(\tau)|f, \mathcal{A})}_{(a)} \underbrace{P(f|\mathcal{A})}_{(b)}. \quad (6)$$

Expression (a) in Eq. (6) can be computed by applying different measurement functions V_{TAB} and V_{TXT} according to f function values, table or text-block, where V_{TAB} is used for tables and V_{TXT} is used for text-blocks:

$$P(V(\tau): \tau \in \mathcal{A}|f, \mathcal{A}) = \prod_{f_\tau = table}^{\tau \in \mathcal{A}} P(V_{TAB}(\tau)|f, \mathcal{A})$$
$$\times \prod_{f_\tau = text-block}^{\tau \in \mathcal{A}} P(V_{TXT}(\tau)|f, \mathcal{A})P(f|\mathcal{A}). \quad (7)$$

To compute expression (b) in Eq. (6), we consider the discontinuity property between neighbors to two zone entities with different labels. Let $\mathcal{A} = \{A_1, A_2, \ldots, A_M\}$ be the set of document elements extracted from a document page. Each element $A_i \in \mathcal{A}$ is represented by a bounding box (x, y, w, h), where (x, y) is the coordinate of the top-left corner, and w

and h are the width and height of the bounding box, respectively. The spatial relations between two adjacent boxes are shown in Figs. 7(a) and (b).

The neighbor set is defined as

$$\mathcal{N} = \{(v_a, v_b)| v_a \text{ and } v_b \text{ horizontally or vertically adjacent,}$$

$$\times v_a \in \mathcal{V}, v_b \in \mathcal{V}\}.$$

Assuming the conditional independence between each neighborhood relationship, expression (b) in Eq. (6) can be computed as

$$P(f|\mathcal{A}) = \prod_{\{p,q\} \in \mathcal{N}} P_{\{p,q\}}(f_p, f_q | p, q), \tag{8}$$

where $P_{\{p,q\}}(f_p, f_q | p, q)$ has the property

$$P_{\{p,q\}}(f_p, f_q | p, q) = \begin{cases} P_{\{p,q\}}(f_p, f_q | p, q), & f_p \neq f_q, \\ 0, & f_p = f_q. \end{cases} \tag{9}$$

Eq. (7) can be written as

$$P(V(\tau): \tau \in \mathcal{A} | f, \mathcal{A}) = \prod_{\substack{\tau \in \mathcal{A} \\ f_\tau = table}} P(V_{TAB}(\tau) | f, \mathcal{A})$$

$$\times \prod_{\substack{\tau \in \mathcal{A} \\ f_\tau = text\text{-}block}} P(V_{TXT}(\tau) | f, \mathcal{A})$$

$$\times \prod_{\{p,q\} \in \mathcal{N}} P(f_p, f_q | p, q). \tag{10}$$

The iterative updating optimization problem can be formulated as follows: *Given initial set $\mathcal{A}^0$, find a new set $\mathcal{A}^s$ and a labeling function $f^s: A^s \rightarrow L$, that maximizes the probability*:

$$P(V(\tau): \tau \in \mathcal{A}^s | f^s, \mathcal{A}^s) = \prod_{\substack{\tau \in \mathcal{A}^s \\ f_\tau^s = table}} P(V_{TAB}(\tau) | f^s, \mathcal{A}^s)$$

$$\times \prod_{\substack{\tau \in \mathcal{A}^s \\ f_\tau^s = text\text{-}block}} P(V_{TXT}(\tau) | f^s, \mathcal{A}^s)$$

$$\times \prod_{\{p,q\} \in \mathcal{N}} P(f_p^s, f_q^s | p, q). \tag{11}$$

Our goal is to maximize the probability in Eq. (10) by iteratively updating $\mathcal{A}^k$ and f^k. Our iterative updating optimization system works as follows: we take the statistical refinement results as the preliminary table detection results and use some existing text block segmentation algorithm to get text blocks [21]. Then we systematically adjust the labeling to maximize the probability until no further improvement can be made.

4.4.2. Probability estimation

1. *Table and text separator probability*. Given a table, t, and its vertically adjacent neighboring text block B, we compute the probability of the separator between them being a table and text separator as

$$P(f_t, f_B | t, B) = P(TableTextSeparator | o_h(t, B), d_v(t, B)),$$

where the definitions of $d_v(t, B)$ and $o_h(t, B)$ can be found at Eqs. (2) and (3).

2. *Table measurement probability*. To facilitate table detection, we applied our table decomposition algorithm (Section 4.5) on each detected table. Based on the table decomposition results, three features are computed. These features are given below:

- Ratio of total large vertical blank block [24] and large horizontal blank block [24] areas over identified table area. Let t be an identified table and $\mathcal{B}$ be the set of large horizontal and vertical blank blocks and in it, $ra = \sum_{\beta \in \mathcal{B}} Area(\beta)/Area(t)$.
- Maximum difference of the cell baselines in a row. Denote the set of the cells in a row i as $\mathcal{RC}_i$, $\mathcal{RC}_i = \{c_{i,1}, c_{i,2}, \ldots, c_{i,i_m}\}$. Denote the set of $\mathcal{RC}_i$ as $\mathcal{RC}$, $\mathcal{RC} = \{\mathcal{RC}_i, i = 1, \ldots, m\}$, where m is the row number in the table. Let $baseline(c)$ be the y coordinate of the cell entity bottom line, $mc = max_{\mathcal{RC}_i \in \mathcal{RC}}(mzx_{c_{i,j} \in \mathcal{RC}_i}(baseline(c_{i,j})) - min_{c_{i,j} \in \mathcal{RC}_i}(baseline(c_{i,j})))$;
- Accumulated difference of the justification in all columns. Denote the set of cells in a column, i, in the table $\mathcal{CC}_i = \{c_{i,1}, c_{i,2}, \ldots, c_{i,i_n}\}$. Denote the set of $\mathcal{CC}_i$ as $\mathcal{CC}$, $\mathcal{CC} = \{\mathcal{CC}_i, i = 1, \ldots, n\}$, where n is the column number in the table. Let $(x_{i,j}, y_{i,j}, w_{i,j}, h_{i,j})$ represent the bounding box of the cell $c_{i,j} \in \mathcal{CC}_i$. We estimate the justification of a column, $i, i = 1, \ldots, n$, by computing the vertical projection of the left, center, and right edge of $c_{i,j}, j = 1, \ldots, i_n$:

$$C_{left}[i] = \max_{c_{i,j} \in \mathcal{CC}_i}(x_{i,j}) - \min_{c_{i,j} \in \mathcal{CC}_i}(x_{i,j}),$$

$$C_{center}[i] = \max_{c_{i,j} \in \mathcal{CC}_i}(x_{i,j} + w_{i,j}/2) - \min_{c_{i,j} \in \mathcal{CC}_i}(x_{i,j} + w_{i,j}/2),$$

$$C_{right}[i] = \max_{c_{i,j} \in \mathcal{CC}_i}(x_{i,j} + w_{i,j}) - \min_{c_{i,j} \in \mathcal{CC}_i}(x_{i,j} + w_{i,j}),$$

$$J_i = min\{C_{left}[i], C_{center}[i], C_{right}[i]\}.$$

The accumulated difference of the justification in all columns, mj, is computed as: $mj = \sum_{i=1}^{n} J_i$.

Finally, we can compute the table consistent probability for table t as

$$P(V_{TAB}(t)) = P(consistent(t) | ra(t), mc(t), mj(t)).$$

3. *Text block measurement probability*. A text block, in general, has a homogeneous inter-line spacing and a alignment type (such as left-justified, etc.) Given a detected text block B, we compute the probability that B has homogeneous inter-line spacing, and a text alignment type. As in Liang et al. [23], we compute text block measurement probability as

$$P(V_{TXT}(B)) = P(V_{TXT}(B) | Leading(B), Alignment(B)).$$

By making the assumption of conditional independence, we can rewrite the above equation as

$$P(V_{TXT}(B)) = P(V_{TXT}(B)|Leading(B)P(V_{TXT}(B)|$$

$$Alignment(B)).$$

Let $B = (l_1, \ldots, l_n)$ be an extracted block. $D_B = (d(1), d(2), \ldots, d(n-1))$ is a sequence of inter-line space distances, where $d(j)$ is the space distance between l_j and l_{j+1}. We compute the median and the maximum value of the elements of D_B. The probability is

$$P(V_{TXT}(B)|Leading(B)) = P(median(D_B), max(D_B)|$$

$$Leading(B)).$$

Given a text block B that consists of a group of text lines $B = (l_1, l_2, \ldots, l_n)$, we determine the text alignment of B by observing the alignment of the text line edges. Let e_{li} be the left edge of the text line l_i and let e_{ci} and e_{ri} be the center and right edges of the line box, respectively. Let E_l be the left edges of text line 2 to n, such that $E_l = \{e_{li}|2 \leqslant i \leqslant n\}$. E_c is the center edges of text line 2 to $n-1$, and E_r is the right edges of text line 1 to $n-1$. We first estimate the median of E_l, then compute the absolute deviation D_l of the elements of E_l from its median:

$$D_l = \{d_i|d_i = |e_{li} - median(E_l)|, 2 \leqslant i \leqslant n\}.$$

Similarly, we estimate the absolute deviation of the center edges and right edges: D_c and D_r. Then, we compute the probability of B being left, center, right, or both justified by observing the mean absolute deviation of the left, center and right edges:

$$P(V_{TXT}(B)|Alignment(B)) = P(mean(D_l), mean(D_c),$$

$$\times mean(D_r)|Alignment(B)). \tag{12}$$

4.4.3. Iterative updating optimization algorithm

Fig. 8 shows a block diagram of our algorithm. Given a labeled page, first we estimate its segmentation probability. For each table, we consider several adjustments, which are to keep it as a table, to grow the table to include its upper and lower neighbors, to merge the table with its upper and lower neighbors and label it as text block. For each adjustment, we compute the new probability. We select the adjustment which produces the biggest improvement upon the initial page segmentation probability. This process is repeated until no improvement can be made. The details of the algorithm are described below.

Algorithm 4.1. *Iterative updating optimization.*

1. The input data to the algorithm are table statistical refinement algorithm result and text block segmentation results [23]. They are a set of block entities, $\mathscr{A}^0$ and function $f^0: \mathscr{A}^0 \to \mathscr{L}$.

2. Set $k = 0$.
3. For each hypothesized table, i, $i = 1, \ldots, N$, where N is the number of tables in $\mathscr{A}^k$. Compute the different probabilities under different adjustments:
   - Keep the table. Compute the probability $P_{(i,1)}$ following Eq. (10).
   - Merge table i with its upper text neighbor and label it as a new table. Compute the new probability $P_{(i,2)}$ following Eq. (10).
   - Merge table i with its upper text neighbor and label it as a new text block. Compute the new probability $P_{(i,3)}$ following Eq. (10).
   - Merge table i with its lower text neighbor and label it as a new table. Compute the new probability $P_{(i,4)}$ following Eq. (10).
   - Merge table i with its lower text neighbor and label it as a new text block. Compute the new probability $P_{(i,5)}$ following Eq. (10).
4. Compute $P_{max} = max(P_{(i,j)}), i = 1, \ldots, N, j = 1, \ldots, 5$ and get its appropriate adjustment *action*.
5. If the *action* is to keep the table, then, return the labeling result $\mathscr{A}^k$ as $\mathscr{A}^s$ and stop the algorithm.
6. If the *action* is not to keep the table, then take the adjustment *action* and we get $\mathscr{A}^{k+1}$ and $f^{k+1}: \mathscr{A}^{k+1} \to \mathscr{L}$.
7. Set $k = k + 1$ and go back to 3.

4.5. Table decomposition algorithm

According to our experience and other researcher's report [12], it is extremely difficult to detect table headers using only geometric information without lexical criteria. The goal of our table decomposition algorithm is to decompose the detected table zones into cell structures using only geometric information.

Similar to the recursive X–Y cut in [22], we do a vertical projection on the word level in each identified table. Because of the table structure, we can expect the projection result to have peaks and valleys after binarization of projection result. We can separate each table column which starts from a valley and ends at the next valley. Fig. 6(a) shows the table columns we get on the image shown in Fig. 4. After we construct the table columns, we can get cell structures and their attributes such as starting/ending row, starting/ending column. In Fig. 6(b), we show an example output from the table structure understanding algorithm.

4.6. Global parameter optimization scheme

Fig. 3 shows the steps of the table understanding algorithm. Each step has some tuning parameters. Fig. 9 shows the input, output structures and the tuning parameters used for each step. The parameter values are first estimated locally and then these locally estimated values are used as the initial value in a global parameter optimization for the whole procedure.
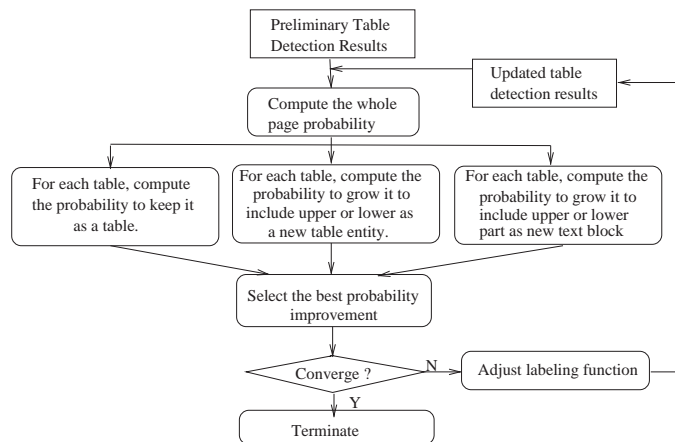
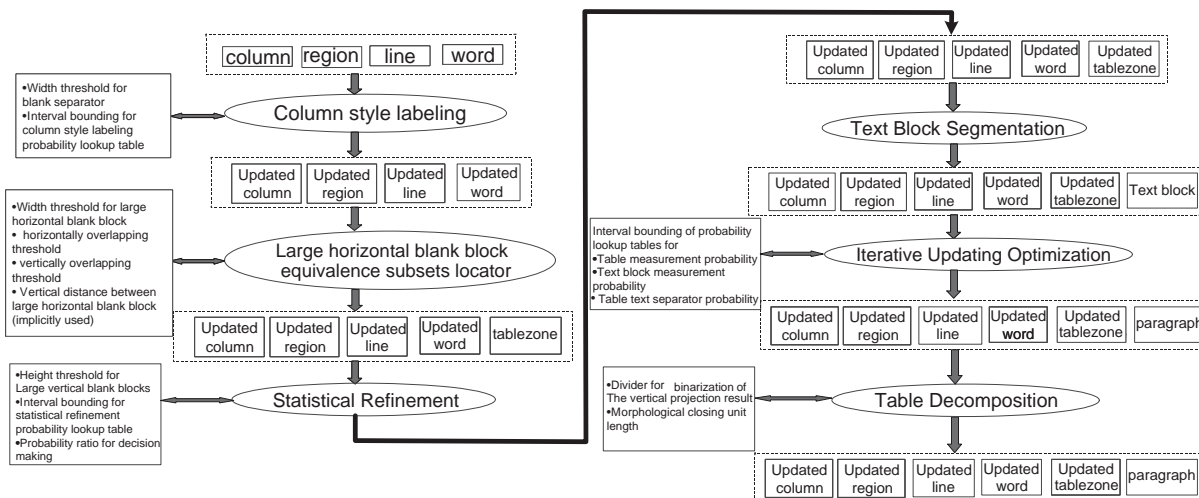Fig. 8. Overview of the iterative updating optimization algorithm.

Fig. 9. Details of table understanding algorithm. For each step, its input/output data structures and the parameters used are shown. The global parameter optimization scheme is designed to optimize these parameters.

Fig. 5(a) shows the diagram of the global parameter optimization scheme. First, the tuning parameter set is initialized by the initial parameter values. We use the performance evaluation protocol presented in Section 6 to evaluate the function value with the given parameter set. An independent optimization control module reads the tuning parameter set and its performance. Then it uses an optimization method to adjust the parameter values to maximize the performance. The performance criteria we used is the fraction of ground truth cells that are correctly identified plus the fraction of the detected cells that are correctly identified, the fraction being with respect to the whole data set. There are various optimization methods that could be used, such as golden section search in one dimension [25], line search in one dimension, or conjugate gradient methods in multidimensions [25], etc.

Below we will give a brief introduction to the line search in one dimension which is what we used.

Given an initial point, a direction in a given dimension, and an initial step interval, the line search method starts searching with the initial step interval. As long as the function increases in value, the step size doubles for each trial. If the function value decreases, the pass terminates, the direction is flipped and the new step size is changed to the initial step size. The interval between the pass starting position and the flipping position of the pass is a search region for the pass. This procedure repeats until the number of trials exceeds some threshold or the search region length is equal to the initial step size. Fig. 5(b) shows a working example of the algorithm. It starts the search at position 1. The initial search direction is right. It flips directions at positions

4, 7, 9, 11. It stops at 11. The maximum function value is taken at position 10. For an N-dimensional optimization, this line search cycle through all dimensions until there are no changes.

5. Automatic table ground truth generation

We developed a software package that can analyze any given set of table ground truth data and estimate the mean and variance of all the parameters required to synthesize similar kinds of tables. Given the parameter values, the software can synthesize pseudo-random tables and their ground truth. Automatic ground truthing of generated tables is facilitated by making table contents unique in each given document image so that the table structure can be determined by content matching. We demonstrated the feasibility of our algorithm on a real image data set and used the synthetic table data to aid our table structure understanding research. The software package is publicly available at Ref. [26].

5.1. Table ground truth specification

We define the table structure in a hierarchical structure, as shown in Fig. 2(b). In the table ground truth, we need specify the hierarchical structure between table, row/column header, table body and cell entities. For each cell, the following attributes have to be recorded: (**1**) starting/ending row, sr and er; (**2**) starting/ending column, sc and ec; (**3**) justification, cj. Its possible values are left, center, right and decimal.

Note that although we do not explicitly describe row and column structures, such information can be readily obtained by examining cell attributes. As explained in the next section, the table hierarchical structure and its cell attributes are automatically generated by our table ground truth generation tool.

5.2. Automatic table ground truth generation

Fig. 10(a) shows the diagram of the system and Fig. 10(b) an example of the automatic table ground truth generation results. The following parts describe the automatic table ground truth generation procedure.

5.2.1. Parameter generator

This software is used to analyze a given table ground truth or non-table ground truth. Two kinds of parameter sets, $\mathcal{T}$ and $\mathcal{N}$, are designed. There are 12 table layout parameters in $\mathcal{T}$, e.g. column justification, spanning cell position, etc. There are three non-table layout parameters in $\mathcal{N}$, e.g. text column number, if there are marginal notes, etc. Clearly, $\mathcal{T}$ is designed to add more variety to table instances and test the mis-detection performance of any table detection algorithm. $\mathcal{N}$ is designed to add more variety to non-table

instances and test the false alarm performance of any table detection algorithm. Currently, the part which automatically estimates non-table parameters has not been implemented, so we enclose them in dashed lines in Fig. 10(a). The table parameter set and non-table parameter set definitions can be found in Ref. [20].
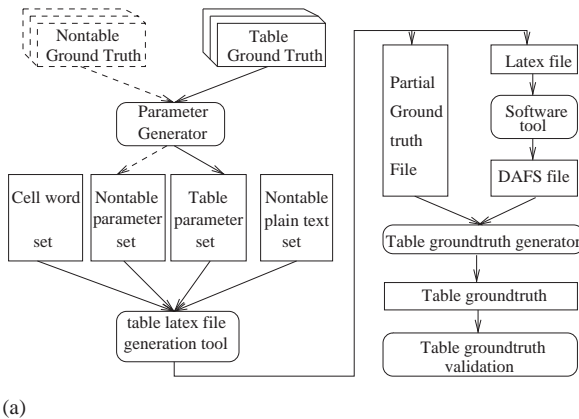
5.2.2. Table latex file generation tool

This software randomly selects two parameter elements from sets $\mathcal{T}$ and $\mathcal{N}$. The resulting parameter for a page is a reasonable element in $\mathcal{T} \times \mathcal{N}$. We pre-computed two content sets $\mathcal{C}$, $\mathcal{P}$. They are cell word set and non-table plain text set. Elements of $\mathcal{C}$ are random, meaningless English character strings. Elements of $\mathcal{P}$ are the text ground truth file from UW CDROM III [17]. Sets $\mathcal{C}$, $\mathcal{P}$ are the contents of table entities and non-table entities in the generated LaTeX [27] files, respectively. We make sure every element in $\mathcal{C}$ is unique in both $\mathcal{C}$ and $\mathcal{P}$ and it can only be used once for a given file. This software writes out two files: a LaTeX file and a partial ground truth file. In the partial ground truth file, there are table, row header, column header and cell entities with their content and attributes such as cell starting/ending column number, etc.

5.2.3. DAFS file generation tools

Several software tools are used and some minimum manual work is required in this step. LaTeX produces DVI files. The DVI2TIFF software [28] converts DVI file to a TIFF file and a so-called character ground truth file which contains the bounding box coordinates, the type and size of the font, and the ASCII code for every individual character in the image. The CHARTRU2DAFS software [26] combines each TIFF file and its character ground truth file and converts it to a DAFS file [29]. The DAFS file has content ground truth for every glyph, which is the basis of content matching in the next step. Then line segmentation and word segmentation software [30,31] segments word entities from DAFS file. Since we cannot guarantee a 100% word segmentation accuracy, a minimum of manual work using Illuminator [32] tool is required to fix any incorrect word segmentation results inside tables.

5.2.4. Table ground truth generator

Since we know every word in the tables appears once, we can use a content matching method to locate any table related entity of interest. Our software locates each unique word from the partial ground truth file in the DAFS file. If this cannot be done, an error is reported. Here is the way to make the previous step even simpler. We only need to run the table ground truth generator twice. The only places we need look at are the files with some errors in the first run. After the correction, we run this software again to obtain the final table ground truth data.

Fig. 10. (a) Illustrates automatic table ground truth generation procedure; (b) illustrates an example of generated table page.

5.2.5. Table ground truth validation

For normal ground truthing work, validation is a required step to make sure that we have correct ground truth. Our table ground truth validation is also automatically done. It checks the geometric relations among table, row, column and cell entities. If there is any discrepancy, the page can be either removed or given for further manual checking.

6. Table structure understanding performance evaluation

We can view the table structure understanding evaluation problem as three correspondence problems between the detection results and ground truth in the three different sets, ϕ_1, ϕ_2, ϕ_3, where $\phi_1 = \{$RLS rectangle areas whose Q value is table$\}$, $\phi_2 = \{$RLS rectangle areas whose Q values are row header, column header or table body$\}$, and $\phi_3 = \{$RLS rectangle areas whose Q value is cell$\}$.

Suppose we are given two sets $\mathscr{G} = \{G_1, G_2, \ldots, G_M\}$ for ground truthed foreground table related entities, e.g. cell entities in ϕ_3, and $\mathscr{D} = \{D_1, D_2, \ldots, D_N\}$ for detected table related entities. The algorithm performance evaluation can be done by solving the correspondence problem between the two sets. Performance metrics developed in Ref. [30] can be directly computed in each rectangular layout structure set. Any detection errors such as table/cell misdetections, false alarms, splitting errors and merging errors can be reported

by studying the metrics. In particular, the correspondence problem in set ϕ_1 is the evaluation of the table identification algorithm, and those in sets ϕ_2 and ϕ_3 are the evaluation of table decomposition algorithm. The total cost of the table structure understanding process can be calculated by the linear combination of the costs in three sets:

$$Cost = W_{\phi_1} \, Cost_{\phi_1} + W_{\phi_2} \, Cost_{\phi_2} + W_{\phi_3} \, Cost_{\phi_3},$$

where W_{ϕ_1}, W_{ϕ_2} and W_{ϕ_3} are weights for the three evaluation costs.

For our table structure understanding algorithm, our final goal is to extract tables from document pages and decompose into different cell entities. So our table structure understanding performance evaluation is done in ϕ_3. Most of the published table structure understanding algorithms focus on the table detection problem. To compare our algorithms with these algorithms, we also did performance evaluation on the ϕ_1 level.

7. Experimental results

Our testing data set has 1125 document pages. All of them are machine printed, noise free data. Among them, 565 pages are real data from different business and law books. Another 550 pages are synthetic data generated using the method described in Section 5. From the real document images, we used our software package to estimate the table

Table 1
Cell level performance of the statistical optimization table structure understanding algorithm on real data set and whole data set

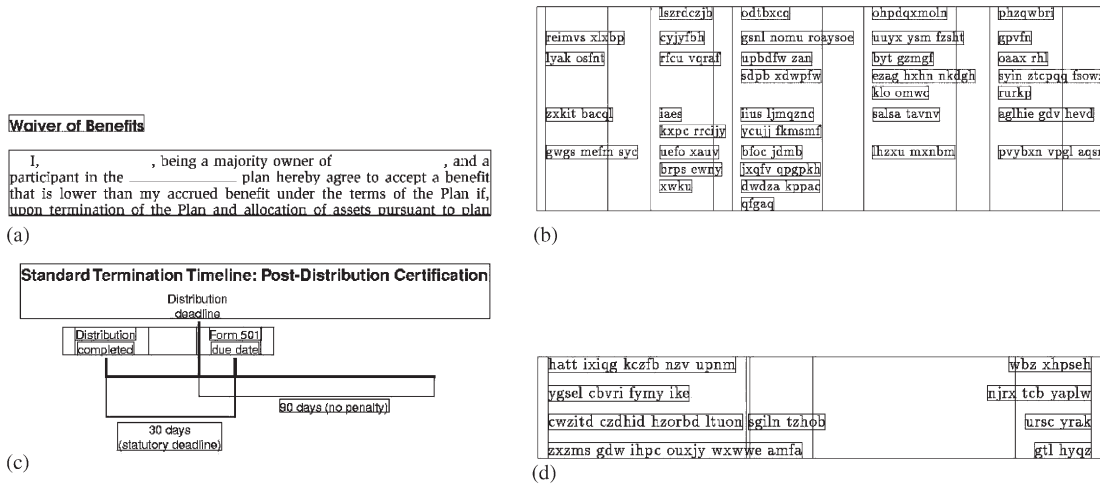|		Total	Correct	Splitting	Merging	Mis-false	Spurious
Real data set	Ground truth	679	657 (96.76%)	3 (0.44%)	15 (2.21%)	4 (0.59%)	0 (0.00%)
	Detected	700	657 (93.86%)	6 (0.86%)	7 (1.00%)	30 (4.29%)	0 (0.00%)
Whole data set	Ground truth	10934	10461 (95.67%)	132 (1.21%)	45 (0.41%)	296 (2.71%)	0 (0.00%)
	Detected	10779	10461 (97.05%)	264 (2.45%)	18 (0.17%)	36 (0.33%)	0 (0.00%)

Fig. 11. Illustrates the table structure understanding results: (a) and (b) correct results; (c) and (d) failed results.

parameters from seven types of tables. The parameter files we used in this experiment can be obtained at Ref. [26]. A hold-out cross validation experiment [33] was conducted on all the data with $N = 9$. Discrete lookup tables were used to represent the estimated joint and conditional probabilities used at each of the algorithm decision steps. The performance evaluation was done using the protocol in Section 6.

7.1. Statistical optimization algorithm experimental results on ϕ_3 level

The numbers and percentages of miss, false, correct, splitting, merging and spurious detections on real data set and on the whole data set are shown in Table 1. Here the tuning parameters take the local maximum values before we apply the global parameter optimization scheme. The performance on the real data set was $0.9676 + 0.9386 = 1.9062$ and the performance on the whole data set was $0.9600 + 0.9695 = 1.9295$.

Fig. 11 shows a few table structure understanding examples. Figs. 11(a) and (b) are two correct results. Fig. 11(a)

shows a correct text block segmentation result. Fig. 11(b) shows a correct table detection and decomposition result. Figs. 11(c) and (d) illustrates some failed examples. Fig. 11(c) shows a false alarm example. Some texts in a figure are detected as a table entity. Fig. 11(d) shows an error example where our table decomposition algorithm failed.

7.2. Statistical optimization algorithm with local maximum line search experimental results on ϕ_3 level

To verify if the tuning parameter value used in the previous section is on the local maximum point, we applied the global parameter optimization scheme (Section 4.6) to do the line search on the tuning parameter values. Using our cross validation method, each time, eight nineth partitions are used as a training data set. Given a training data set, we sequentially used the global parameter optimization scheme to do the line search on each dimension to find a new local maximum point. Then we used this parameter set to test on the last one nineth data set. The whole experimental results

Table 2
Cell level performance of the statistical optimization table structure understanding algorithm with line search on real data set and whole data set

		Total	Correct	Splitting	Merging	Mis-false	Spurious
Real data set	Ground truth	679	657 (96.76%)	3 (0.44%)	15 (2.21%)	4 (0.59%)	0 (0.00%)
	Detected	698	657 (94.13%)	6 (0.86%)	7 (1.00%)	28 (4.01%)	0 (0.00%)
Whole data set	Ground truth	10934	10497 (96.00%)	117 (1.07%)	123 (1.12%)	197 (1.80%)	0 (0.00%)
	Detected	10827	10497 (96.95%)	234 (2.16%)	54 (0.50%)	42 (0.39%)	0 (0.00%)

Table 3
Hu et al.'s dynamic programming based table extraction algorithm [10] performance results on whole image data set performance evaluation result on the table level

		Total	Correct	Splitting	Merging	Mis-false	Spurious
Real data set	Ground truth	36	32 (88.89%)	0 (0.00%)	2 (5.56%)	2 (5.56%)	0 (0.00%)
	Detected	148	32 (21.62%)	0 (0.00%)	1 (0.68%)	115 (77.70%)	0 (0.00%)
Synthetic data set	Ground truth	480	420 (87.50%)	51 (10.62%)	2 (0.42%)	7 (1.46%)	0 (0.00%)
	Detected	545	420 (77.06%)	104 (19.08%)	1 (0.18%)	20 (3.67%)	0 (0.00%)
Whole data set	Ground truth	516	452 (87.60%)	51 (9.9%)	4 (0.78%)	9 (1.74%)	0 (0.00%)
	Detected	693	452 (65.22%)	104 (15%)	2 (0.29%)	135 (19.48%)	0 (0.00%)

are shown in Table 2. Comparing Tables 1 and 2, we will see the line search results are a little better but there is no significant difference. For example, in the whole data set, the line search results have smaller mis-detection rate but a large false alarm rate. The performance on the real data set was $0.9676 + 0.9413 = 1.9089$ and on the whole data set was $0.9567 + 0.9705 = 1.9272$.

The line search of course resulted in an improvement of performance on the training set side. The numbers reported in the results are on the testing set side. So although these was an overall improvement on the training side, there was not an overall improvement on the testing side (1.9295 before line search and 1.9272 after line search). This suggests that the initial parameter set values were most probably set close to the optimal values.

7.3. Table detection result comparison

Most of table structure understanding research focused on the table detection problem [9–11]. The output of their

algorithms are detected table regions. To compare our table detection algorithm, we also did table detection performance evaluation on table level (ϕ_1 level in Fig. 2(b)). We implemented the algorithm of Hu et al. [10]. Their performance evaluation results on our data set are shown in Table 3. The performance evaluation results of the automaton algorithm [34] are reported in Table 4. Both algorithms take detected column and word structure as the input. We manually generated the column ground truth in our data set. The column and detected word structures are used as the input of their algorithms. The performance evaluation results of our statistical optimization algorithm are reported in Table 5. Since our performance evaluation is very strict, the result is counted as a correct detection only when the detected result is totally matched the ground truth data. If the detected table has one more or less row/column, our performance evaluation algorithm counts it as incorrect.

The best performance results of three algorithms were 88.89% (Hu et al.'s dynamic programming algorithm), 82.71% (Wasserman et al.'s automaton algorithm), and
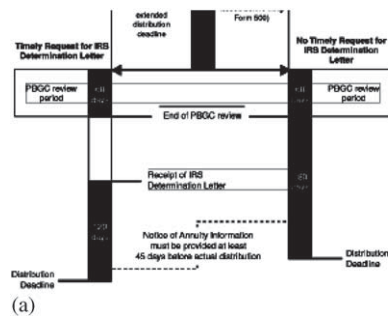
Table 4

Wasserman et al.'s automaton based table extraction algorithm [34] performance results on whole image data set performance evaluation result on the table level

		Total	Correct	Splitting	Merging	Mis-false	Spurious
Real data set	Ground truth	36	21 (58.33%)	5 (13.89%)	2 (5.56%)	8 (22.22%)	0 (0.00%)
	Detected	53	21 (39.62%)	12 (22.64%)	1 (1.89%)	19 (35.85%)	0 (0.00%)
Synthetic data set	Ground truth	480	397 (82.71%)	54 (11.25%)	0 (0.00%)	29 (6.04%)	0 (0.00%)
	Detected	519	397 (76.49%)	109 (21.00%)	0 (0.00%)	13 (2.50%)	0 (0.00%)
Whole data set	Ground truth	516	418 (81.00%)	59 (11.43%)	2 (0.39%)	37 (7.17%)	0 (0.00%)
	Detected	572	418 (73.08%)	121 (21.15%)	1 (0.17%)	32 (5.59%)	0 (0.00%)

Table 5

The optimization based table extraction algorithm performance results on whole image data set performance evaluation result on the table level

		Total	Correct	Splitting	Merging	Mis-false	Spurious
Real data set	Ground truth	36	32 (88.89%)	0 (0.00%)	4 (11.11%)	0 (0.00%)	0 (0.00%)
	Detected	38	32 (84.21%)	0 (0.00%)	2 (5.26%)	4 (10.53%)	0 (0.00%)
Synthetic data set	Ground truth	480	469 (97.71%)	4 (0.83%)	0 (0.00%)	7 (1.46%)	0 (0.00%)
	Detected	477	469 (98.32%)	8 (1.68%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
Whole data set	Ground truth	516	501 (97.09%)	4 (0.78%)	4 (0.78%)	7 (1.36%)	0 (0.00%)
	Detected	515	501 (97.28%)	8 (1.55%)	2 (0.39%)	4 (0.78%)	0 (0.00%)

Fig. 12. (a) and (b) failed table detection results due to the limited table simulation ability of automatic table groundtruth generation.

Table 6
A proposal to generalize this framework to math detection problem

Step description	Math detection	Table detection
Coarse detection	Special symbol detection	Large horizontal blank block equivalence subsets location
Statistical refinement	Math structure statistical study	Table structure statistical study
Iterative updating optimization	Text/math measurement/separator probabilities	Text/table measurement/separator probabilities
Global parameter optimization scheme	Applicable	Applicable

98.32% (our algorithm). On the whole data set, the performance of Hu et al.'s dynamic programming algorithm [10] was $0.8760 + 0.6522 = 1.5282$ and the performance of Wasserman et al.'s automaton algorithm [34] was $0.8100 + 0.7308 = 1.5408$. The performance of our algorithm was $0.9709 + 0.9728 = 1.9437$. From the results, we can see our algorithm results are better than the other two algorithms because our algorithm has more statistical validation modules. Considering the different input data, the other two algorithms took perfect column structure as the input while our algorithm's took previously segmented column structure, which could have many errors, as the input. It also demonstrated our algorithm is more robust and could easily be integrated into any practical document layout analysis system.

Another observation is that the performance on the synthetic data set and on the real data set are different. The reason is that although the synthetic data set has a similar table structure to the real data set, the non-table parts are still a little different. Figs. 12(a) and (b) show two failed cases. In Fig. 12(a), the false alarm is due to the text on the graphics. In Fig. 12(b), the merging error is due to two adjacent table entities. These cases are not simulated in the synthetic data set. This partially explains why there is a small difference between the results from synthetic data set and real data set.

8. Conclusion and future work

In this paper, we formulated the table structure understanding problem in the whole page segmentation framework. We presented a statistical based table structure understanding algorithm using optimization methods. We showed that an algorithm designed to locally maximize table detection is not satisfactory. We improved the table detection result by optimizing the whole page segmentation probability, including table entities and text block entities.

Some turning parameters are used in our algorithm. Instead of fixing these parameters, we determined the tuning parameter values on the fly with our performance evaluation protocol and some maximum function method. A novel

automatic table ground truth generation system which can efficiently generate a large amount of accurate table ground truth suitable for the development of table detection algorithms was discussed. We implemented our algorithm and tested on a data set which included 1125 document pages with 10,934 table cell entities. Among them, 565 pages were real data from different business and law books. Another 560 pages were synthetic data.

As shown in Fig. 11(d), our current table decomposition result can be further refined by a statistically based table decomposition algorithm. Finally, the framework proposed in this paper is very general. It has the potential to be applied to other page segmentation problems, such as text/math and text/figure segmentation problems. Table 6 shows some ideas to generalize this framework to math detection problem.

9. Summary

With the large number of existing documents and the increasing speed in the production of new documents, finding efficient methods to process these documents for their content retrieval and storage becomes critical. Tables are a popular and efficient document element type. Therefore, table structure understanding is an important problem in the document layout analysis field. This paper presents a table structure understanding algorithm using optimization methods. It includes steps of column style labeling, large horizontal blank block equivalence subsets location, statistical refinement, iterative updating optimization and table decomposition. The column style labeling, statistical refinement and iterative updating optimization steps are probability based, where the probabilities are estimated from geometric measurements made on the various entities with which the algorithm works in a large training set.

Each step of our table structure understanding algorithm has some tuning parameters. We initially set the parameters with some conjectural values. Then with a global parameter optimization scheme, we update these values using a line search optimization algorithm. We use a performance evaluation protocol employing an area overlapping measure. With

this scheme, we can obtain statistically satisfactory tuning parameter values on the fly.

Large data sets with ground truth are essential in assessing the performance of a computer vision algorithm. Manually generating document ground truth proved to be very costly and prone to involve subjective errors. We address this problem by using an automatic table ground truth generation system which can efficiently generate a large amount of accurate ground truth suitable for the development of table structure understanding algorithms. This software package is publicly available.

The training and testing data set for the algorithm include 1125 document pages having 518 table entities and a total of 10,934 cell entities. The algorithm performed at the 96.76% accuracy rate on the cell level and 98.32% accuracy rate on the table level. We implemented and tested two other published table structure understanding algorithms. In the same data set, with the perfect column structure as the input, the other two algorithms performed at the 88.89% and 82.71% accuracy rate on the table level. Comparing with them, our algorithm demonstrated a favorable result.

References

[1] T. Watanabe, Q. Luo, N. Sugie, Layout recognition of multi-kinds of table-form documents, IEEE Trans. Pattern Anal. Mach. Intell. 17 (4) (1995) 432–445.

[2] B. Yu, A.K. Jain, A generic system for form dropout, IEEE Trans. Pattern Anal. Mach. Intell. 18 (11) (1996) 1127–1134.

[3] F. Cesarini, M. Gori, S. Marinai, G. Soda, Informys: a flexible invoice-like form-reader system, IEEE Trans. Pattern Anal. Mach. Intell. 20 (7) (1998) 690–706.

[4] S. Chandran, R. Kasturi, Structural recognition of tabulated data, in: Proceedings of International Conference on Advances in Pattern Recognition (ICAPR) 93, Tsukuba Science City, Japan, October 1993, pp. 516–519.

[5] E. Green, M. Krishnamoorthy, Model-based analysis of printed tables, in: Proceedings of the Third ICDAR, Canada, August 1995, pp. 214–217.

[6] J.H. Shamilian, H.S. Baird, T.L. Wood, A retargetable table reader, in: Proceedings of the Fourth ICDAR, Germany, August 1997, pp. 158–163.

[7] K. Zuyev, Table image segmentation, in: Proceedings of the International Conference on Document Analysis and Recognition (ICDAR) '97, Ulm, Germany, August 1997, pp. 705–708.

[8] T.G. Kieninger, Table structure recognition based on robust block segmentation, Document Recognition V, January 1998, pp. 22–32.

[9] T. Kieninger, A. Dengel, Applying the t-rec table recognition system to the business letter domain, in: Sixth International Conference on Document Analysis and Recognition (ICDAR01), Seattle, WA, September 2001, pp. 518–522.

[10] J. Hu, R. Kashi, D. Lopresti, G. Wilfong, Medium-independent table detection, in: SPIE Document Recognition and Retrieval VII, San Jose, CA, January 2000, pp. 291–302.

[11] B. Klein, S. Gokkus, T. Kieninger, A. Dengel, Three approaches to "industrial" table spotting, in: Sixth International Conference on Document Analysis and Recognition (ICDAR01), Seattle, WA, September 2001, pp. 513–517.

[12] J. Hu, R. Kashi, D. Lopresti, G. Wilfong, Table structure recognition and its evaluation, in: SPIE Document Recognition and Retrieval VIII, San Jose, CA, January 2001.

[13] J.C. Handley, Table analysis for multi-line cell identification, in: SPIE Document Recognition and Retrieval VIII, San Jose, CA, January 2001.

[14] J. Hu, R. Kashi, D. Lopresti, G. Wilfong, Evaluating the performance of table processing algorithms, Int. J. Document Anal. Recogn. 4 (3) (2002) 140–153.

[15] J. Hu, R. Kashi, D. Lopresti, G. Nagy, G. Wilfong, Why table ground-truthing is hard, in: Proceedings of the Sixth International Conference on Document Analysis and Recognition, Seattle WA, USA, September 2001, pp. 129–133.

[16] I. Phillips, S. Chen, R. Haralick, Cd-rom document database standard, in: Proceedings of the Second International Conference on Document Analysis and Recognition, Tsukuba Science City, Japan, October 1993, pp. 478–483.

[17] I. Phillips, Users' Reference Manual, CD-ROM, UW-III Document Image Database-III, 1995.

[18] Y. Wang, I.T. Phillips, R. Haralick, Automatic table ground truth generation and a background-analysis-based table structure extraction method, in: Sixth International Conference on Document Analysis and Recognition (ICDAR01), Seattle, WA, September 2001, pp. 528–532.

[19] G. Liu, R.M. Haralick, Flir atr using location uncertainty, J. Electron. Imaging 9 (2000) 178–193.

[20] Y. Wang, Document analysis: table structure understanding and zone content classification, Ph.D. Thesis, University of Washington, Seattle, WA, 2002.

[21] J. Liang, R. Rogers, R.M. Haralick, I.T. Phillips, Uw-isl document image analysis toolbox: an experimental environment, in: Proceedings of the International Conference on Document Analysis and Recognition (ICDAR) '97, Ulm, Germany, August 1997, pp. 984–988.

[22] J. Ha, I.T. Phillips, R.M. Haralick, Recursive x–y cut using bounding boxes of connected components, in: Proceedings of the Second International Conference on Document Analysis and Recognition, Tsukuba, Japan, October 1993, pp. 952–955.

[23] J. Liang, I.T. Phillips, R.M. Haralick, Consistent partition and labeling of text blocks, J. Pattern Anal. Appl. 3 (2000) 196–208.

[24] Y. Wang, R. Haralick, I.T. Phillips, Improvement of zone content classification by using background analysis, in: Fourth IAPR International Workshop on Document Analysis Systems (DAS2000), Rio de Janeiro, Brazil, December 2000.

[25] W. Press, B. Flannery, S. Teukolsky, W. Vetterling, Numerical Recipes in C, Cambridge University Press, Cambridge, 1988.

[26] Y. Wang, http://students.washington.edu/ylwang/auttabgen.html.

[27] M. Goossens, F. Mittelbach, A. Samarin, The LATEX Companion, Addison-Wesley Publishing Company, Reading, MA, 1994.

[28] T. Kanungo, Dvi2tiff user manual, UW English Document Image Database-(I) Manual, 1993.

[29] RAF Technology Inc., DAFS: Document Attribute Format Specification, 1995.

[30] J. Liang, Document structure analysis and performance evaluation, Ph.D. Thesis, University of Washington, Seattle, WA, 1999.

[31] Y. Wang, I.T. Phillips, R. Haralick, Statistical-based approach to word segmentation, in: 15th International Conference on Pattern Recognition, ICPR2000, Vol. 4, Barcelona, Spain, September 2000, pp. 555–558.

[32] RAF Technology Inc., Illuminator User's Manual, 1995.

[33] R. Haralick, L. Shapiro, Computer and Robot Vision, Vol. 1, Addison-Wesley, Reading MA, 1997.

[34] H. Wasserman, K. Yukawa, B. Sy, K. Kwok, I.T. Phillips, A theoretical foundation and a method for document table structure extraction and decomposition, in: D. Lopresti, J. Hu, R. Kashi (Eds.), Document Analysis Systems V, Fifth IAPR International Workship on Document Analysis Systems, Princeton, NJ, USA, August 2002, pp. 291–294.

About the Author—YALIN WANG received the BS and MS degrees from Tsinghua University, Beijing, China, in computer science, in 1994 and 1997, and received the Ph.D. degree from the University of Washington, Seattle, in 2002, in electrical engineering. His research interests include image processing, pattern recognition, document image analysis, medical imaging, computer graphics and geometric modeling. He has published a dozen papers in the area of document image analysis. Dr. Wang is currently working as an assistant researcher in Mathematics Department, UCLA where he has been involved in projects concerning medical imaging, geometric modeling and computer graphics. He was the best student paper award winner in "Fifth IAPR International Workshop on Document Analysis System", Princeton, NJ, USA, August 2002.

About the Author—IHSIN T. PHILLIPS received the BS, MS and Ph.D. degrees in 1979, 1981, and 1984, respectively, all in computer science, from the University of Maryland, College Park. Currently, she is the chair of the Department of Computer Science at Queens College, the City University of New York. Her research areas include image processing, pattern recognition, document image understanding, document image database design, and performance evaluation of document image analysis, and recognition systems. Her most significant contribution to the field of document image analysis and recognition has been the leadership role she has in the design and creation of the three sets of document image databases: UW-I, UW-II, and UW-III. She has served as program committee member for several IEEE and IAPR conferences and workshops. She is currently the chairperson of the IAPR technical committee on performance evaluation. She is a senior member of IEEE and a member of the IEEE computer Society.

About the Author—ROBERT M. HARALICK is a distinguished professor in the Department of Computer Science, Graduate Center, City University of New York. He is responsible for developing the gray scale cooccurrence texture analysis technique and the facet model technique for image processing. In high level computer vision, he has worked on robust methods for photogrammetry and developed fast algorithms for solving the consistent labeling problem. He has developed shape analysis and extraction techniques using mathematical morphology, the morphological sampling theorem, and fast recursive morphology algorithms. In the area of document image understanding, Dr. Haralick, along with Dr. I. Phillips, developed a comprehensive ground-truthed set of some 1600 document image pages most in English and some 200 pages in Japanese. He has also developed algorithms for document image skew angle estimation, zone delineation, and word and text line bounding box delineation. His most recent research is in the area of computer vision performance characterization and covariance propagation. He is a fellow of IEEE for his contributions in computer vision and image processing and a fellow of IAPR for his contributions in pattern recognition, image processing, and for service to IAPR. He has published more than 500 papers and recently completed his term as the president of the International Association for Pattern Recognition.