Table Detection in Online Ink Notes

Zhouchen Lin, *Member*, *IEEE*, Junfeng He, Zhicheng Zhong, Rongrong Wang, and Heung-Yeung Shum, *Fellow*, *IEEE*

Abstract—In documents, tables are important structured objects that present statistical and relational information. In this paper, we present a robust system which is capable of detecting tables from *free style* online ink notes and extracting their structure so that they can be further edited in multiple ways. First, the primitive structure of tables, i.e., candidates for ruling lines and table bounding boxes, are detected among drawing strokes. Second, the logical structure of tables is determined by normalizing the table skeletons, identifying the skeleton structure, and extracting the cell contents. The detection process is similar to a decision tree so that invalid candidates can be ruled out quickly. Experimental results suggest that our system is robust and accurate in dealing with tables having complex structure or drawn under complex situations.

Index Terms—Table detection, table recognition, graphics recognition, handwriting recognition, document analysis, pen-based computing.

- +

1 INTRODUCTION

IN the past decades, great effort (e.g., the numerous references in [1], [2]) has been spent on offline document analysis. With the advent of pen-based devices such as Tablet PCs and Electronic White-boards, where whole-page or even multiple-page ink notes can be produced easily, the demand for analysis of online documents, as an indispensable part of pen-based computing, has become greater than ever. Online documents contain more information than offline document images, such as stroke order and connectivity between stroke points. Such extra information is very helpful for document analysis, e.g., waiving the need for binarization and extracting connected components [1], [2] and better for isolating overlapping strokes. Therefore, although there has been notable success in offline document analysis, it may be more efficient and robust to develop online approaches for online documents.

Table analysis plays an important role in document processing because tables are common in documents for their power in describing statistical and relational information. Moreover, many applications can be built upon table extraction, such as populating databases which can then be queried or be converted into charts. These would help to accelerate office automation.

There have been many papers focusing on table detection (e.g., [6], [7], [8], [9], [10], [12] and the references in [3], [4], [5], to name just a few). Generally speaking, besides ASCII tables in electronic documents, there are two kinds of tables in document analysis: printed tables and handwritten tables. The latter can further be classified as offline or online. A great deal of handwritten tables

- Z. Lin and H.-Y. Shum are with Microsoft Research Asia, 5th Floor, Sigma Building Zhichun Road #49, Haidian District, Beijing 100080, P.R. China. E-mail: {zhoulin, hshum} @microsoft.com.
- J. He is with the Department of Automation, Tsinghua University, Haidian District, Beijing 100084, P.R. China.

E-mail: heroson98@mails.tsinghua.edu.cn.

- Z. Zhong is with the Department of Electronics Science and Technology, Nankai University, Weijin Road #94, Nankai District, Tianjin 300071, P.R. China. E-mail: zhongzhicheng@vip.sina.com.
- R. Wang is with the Department of Computer Science and Engineering, Fudan University, Handan Road #220, Shanghai 200433, P.R. China. E-mail: rrwang@fudan.edu.cn.

Manuscript received 9 June 2005; revised 5 Dec. 2005; accepted 28 Dec. 2005; published online 13 June 2006.

Recommended for acceptance by D. Lopresti.

For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0304-0605.

exist in spite of many table creating/editing user interfaces (UIs) provided by pen-based devices. Dedicated UIs, mainly driven by gestures or menus, may be more effective in certain cases, but the free-form table drawing has a significant advantage from the user standpoint in that it does not require switching modes—the user can simply write without concerning for whether the correct drawing mode is enabled.

Most of the existing algorithms deal with printed tables, where the detection usually starts from spotting the ruling lines and separating space [7], [10] or block segmentation [8], [12], or using some prior knowledge of layout style of the tables [6]. Much less work has targeted the detection and recognition of handwritten tables in handwritten documents, either offline or online. This kind of detection and recognition is often more difficult than that in printed documents, because the drawing styles are usually far more irregular (Actually, the difficulty depends on the complexity of the document, e.g., detecting semiruled [2] printed tables in a noisy document image might be harder than detecting fully ruled handwritten tables in a neat ink note.) Moreover, handwritten tables may be closely surrounded or even overlaid by other contents so that white space is no longer a robust feature to isolate tables. Perhaps due to these reasons, we have not seen a paper that claims satisfactory performance when the handwritten tables are complex in logical structure and drawn in complex ways, e.g., having skewed orientation (Fig. 5a), overtraced or concatenated ruling lines (Figs. 4a, 4b, 4c, and 4d and Fig. 5a), and several ruling lines drawn in a single stroke (Figs. 4a, 4b, 4c, and 4d and Fig. 5a). Nonetheless, offline and online table detection can share some algorithms. For example, after the logical structure of a table is determined, the cell content can be identified similarly.

To date, very few papers discuss online approaches for general online document analysis. As far as we know, only Jain et al. [9] presents a hierarchical method for extracting homogeneous regions in online documents, in which table detection is considered. His approach is quite efficient, but it is focused on tables with simple physical and logical structure.¹ As a result, it can only detect ruled or unruled horizontal tables that are properly drawn without overtracing, concatenation, etc., and with a rectilinear structure, i.e., consisting of exactly *m* columns and *n* rows.

In contrast, there has been much research on the recognition of handwritten graphical objects (e.g., [13], [14], [16]). However, in the existing systems there are often some assumptions that restrict the freedom of user input. For example, a math recognizer [16] often assumes that all input are parts of mathematical formulae, while a diagram analyzer [13], [14] usually assumes that every stroke is related to some predefined shape. Therefore, the user can neither draw a diagram in a math sketch pad nor write a mathematical formula in a diagram sketch editor. Unfortunately, in free style ink notes it is not known in advance whether an object of interest is present or whether a stroke belongs to the object. This makes object detection and recognition in free style ink notes a more challenging task.

In this paper, we present a robust system that detects tables in *free style* online documents. It is designed to be robust in dealing with the following cases:

- 1. The table can have very complex logical structure that have more than one row or column, but the table must be fully ruled and the four corners of the table bounding box are the only four L-type intersection points (Fig. 2d) among its ruling lines.
- 2. The physical structure could be irregular and free-form. The ruling lines can be nonstraight, overtraced, concatenated, or finished in one or multiple strokes. A table can still be detected after modification as long as it is still a table.
- 3. Multiple tables may exist on the same page. They can be side by side, so that their bounding boxes are ambiguous, or can be nonhorizontal.

1. In this paper, many terminologies comply with [4].



Fig. 1 (a) Factors to compute the concatenation strength of two line segments. (b) Factors to compute how likely two line segments are the opposite sides of a rectangle. (c), (d), and (e) Factors to compute the rectangularity of a possible bounding box. (c) The minimum distance between the line segments and the bounding rectangle. (d) The acute angles between successive line segments. (e) The "valid" length and "total" length of line segments. (f) The dirty strokes are those that are outside the bounding rectangle B_1 and touch the sensitive box B_2 .

Basically, the system first detects primitive structure, i.e., the ruling lines and bounding boxes and then identifies the logical structure, including table skeleton structure and table cell contents. A decision-tree like detection scheme (Fig. 3) is adopted so that invalid candidates can be ruled out quickly. The detector also extracts the physical structure information so that table manipulation can be supported. The methodology introduced in this paper can be adapted for the detection of other structured graphical objects. Tested on our online documents written on Tablet PCs, the accuracies evaluated at both the cell level and the table level are all well above 90 percent.

Our methodology is similar to the blackboard architecture proposed in [14]. In theory, our system could be roughly viewed as a realization of the architecture, which is a rather general and intuitive framework. However, in implementation details, our system does not exactly fit the architecture and is much more optimized for table detection. For example, their primitive object detection does not consider overtraced or concatenated strokes, our table grammar is implicitly embodied in structure identification (Section 3.2) and our candidate pruning exists in every step of the detection (Section 4).

The rest of this paper is organized as follows: Sections 2 and 3 introduce the steps in structure detection and the features for candidate selection and classification. Section 4 summarizes the detection scheme and presents the classifier. Experimental results are shown in Section 5. Finally, Section 6 provides our conclusions and plans for future work.

2 PRIMITIVE STRUCTURE DETECTION

A graphical object usually has some primitive structure. For a table, the primitive structure includes ruling lines and the bounding box.

2.1 Ruling Line Candidate Detection

In this step, strokes which could serve as the ruling lines of tables are detected. In printed table detection, the ruling lines are usually spotted by tracing black pixels along nearly horizontal or vertical directions [10]. Unfortunately, in online ink notes, the ruling lines are usually drawn flexibly. To achieve fast detection, writing/ drawing classification and primary grouping should be done beforehand. As our table detection system is a relatively independent module of the more complete advanced ink parser developed by Microsoft Research Asia, our system simply adopts the writing/ drawing classification and primary grouping results from previous modules. However, our system can work without writing/drawing classification and primary grouping results with some sacrifice in speed and accuracy.

First, all the drawing strokes that are close to line segments are collected. The linearity of a stroke is measured by:

$$f_l = \frac{B_w}{B_h + B_w} \cdot \frac{B_w}{L},$$

where B_w and B_h are the width and the height of the skewed bounding rectangle of the stroke, whose principal direction is computed via least squares, and L is the length of the stroke. Since the user may draw several ruling lines (such as the table bounding box) in a single stroke, we should also examine whether a stroke is composed of multiple line segments. If a stroke is not close to a line segment, we first apply Sklansky's polygonal approximation algorithm [11] to approximate the stroke with a polyline, then segment the stroke at high-curvature points so that each substroke can be approximated by a line segment. If the number of approximating line segments is less than 5 (because the user usually draws no more than four ruling lines in a stroke when drawing tables) and the turning angles between successive line segments are close to either 0 degree, 90 degrees, or 180 degrees, then we accept them as candidates for ruling lines. Thereafter, each accepted stroke is represented by its approximating line segments in order to reduce complexity and save computation. Our ruling line candidate detection algorithm is different from that in [9] where the Hough transform was used and, hence, short ruling lines might not be detected.

Next, the accepted line segments are grouped into blocks according to their spatial position in order to speed up detection, because there may be multiple tables that may not be close to each other. For each block, all line segments within it are stored in a line list. Then a histogram is built to count the number of line segments around each direction.

2.2 Bounding Box Candidate Detection

To detect the table bounding box, we first select line segments from the histogram built in the previous step whose directions are close to or perpendicular to the direction that corresponds to a peak. The peaks of the histogram will be tested one by one from the highest to the lowest. Rotation follows so that the selected line segments become nearly horizontal or vertical. Then, we build the horizontal line list and the vertical line list. As we need to deal with overtraced and concatenated ruling lines, the line lists also store all *composite* line segments that are a combination of some original line segments that are nearly parallel and can be merged or concatenated. For a composite line segment, its linearity f_l is defined as the product of the linearity of the component original line segments and the concatenation strength between successive line segments defined as:

$$f_{con} = 1 - \theta \cdot (A \cdot |p| + B) - C \cdot p \cdot (1 + |d|)^2, \tag{1}$$

where (assuming that the line segments are nearly horizontal, Fig. 1a): θ is the acute angle between the two line segments, p is the signed horizontal distance between the conjunction points, and d is 0 if the two line segments intersect each other; otherwise, d is the vertical distance between the ends of the two line segments. In our system, A, B, and C are chosen as 7, 0.5, and 3, respectively.

With the line lists, we first find all pairs of horizontal or vertical line segments, original or composite, and check whether they can be the opposite sides of a rectangle. This is done by computing their pairability f_p defined as:



Fig. 2. (a) An "invalid" line is a line that has only one intersection point. A "bad" intersection point is one that two line segments do not really intersect each other. An "erroneous" intersection point is an L-type intersection point that is not the corner of the table bounding box. (b), (c), and (d) Three types of intersection points: cross-type, T-type, and L-type. (e) An example of cell ordering. (f) The common area C_{ij} (the larger shaded area) of the *i*th cell and the *j*th stroke.



Fig. 3. The process of decision. The detection proceeds only when the value of the corresponding feature is above its threshold.

$$f_p = \left(1 - \frac{|\theta_1 - \theta_2|}{\pi/2}\right) \cdot \left(1 - \frac{|l_1 - l_2|}{l_1 + l_2}\right) \cdot \left(1 - \frac{|p_1 - p_2| + |p_3 - p_4|}{\max(l_1, l_2)}\right)$$

where (assuming that the line segments are nearly horizontal, Fig. 1b): $\theta_i \in (-\pi/4, \pi/4)$ is the acute angle of the *i*th line segment, l_i is the length of the *i*th line segment, and p_j is the horizontal coordinate of the end point of the line segment, ordered from left to right.

Next, test all possible combinations of pairable horizontal line segments and pairable vertical line segments to check whether they can form a rectangle. This is done by checking the shape, angle, and length relationship among the four line segments, producing three features:

$$\begin{split} f_{shp} &= \left(1 - \frac{\bigtriangleup t + \bigtriangleup b}{H}\right) \cdot \left(1 - \frac{\bigtriangleup l + \bigtriangleup r}{W}\right), \\ f_{ang} &= \frac{1}{2\pi} \sum_{i=1}^{4} \theta_i, \ f_{len} = \frac{\sum_{i=1}^{4} '' \text{valid}'' \text{ length } i}{\sum_{i=1}^{4} '' \text{total}'' \text{ length } i}, \end{split}$$

where $\triangle b$ is the minimum distance between the end points of the bottom line segment and the bottom of the bounding rectangle ($\triangle t$, $\triangle l$, and $\triangle r$ are defined similarly (Fig. 1c)), *H* and *W* are the height and width of the bounding rectangle of the four line segments, respectively, θ_i is the acute angle between successive line segments (Fig. 1d), the "valid" length is the smaller of the length of line segment and the length of the part between successive line segments, and the "total" length is the whole length of the line segment plus the possible gap between its ends and its successive line segment (Fig. 1e).

The linear combination of the above three features produces the rectangularity f_r of the four line segments. If f_r is above a threshold, we further check whether the rectangle is well isolated from other outside strokes. This is measured by the saliency f_s defined as:

$$f_s = 1 - \frac{\text{number of dirty strokes}}{\text{number of all strokes in } B_2}$$

where the dirty strokes are those that are outside the bounding rectangle B_1 of the four line segments and touch the sensitive area (Fig. 1f) defined as the area between B_1 and the sensitive box B_2 , where B_2 and B_1 are concentric and the ratio of their sizes is greater than 1. The computation of saliency could incorporate some contextual information because usually tables should not be densely enclosed by outside strokes.

If f_s is also above a threshold, then the linearity, pairability, rectangularity, and saliency are linearly combined to produce P_{BF} as the likelihood of four line segments forming a bounding box of a table.

Next, we extract the four line segments with the largest P_{BF} and decide that they form the bounding box of a table candidate if P_{BF}

exceeds a threshold. Our bounding box detection algorithm does not require the grouping of the four sides of a rectangle in advance as done in [13]. Finally, all the line segments inside the bounding box are collected as candidate ruling lines.

3 LOGICAL STRUCTURE DETECTION

3.1 Table Skeleton Normalization

In order to obtain the logical structure from the table candidate and simplify the subsequent computation, this step decides whether a nearly vertical line segment intersects a nearly horizontal one and normalizes the tentative table skeleton so that the nearly horizontal or vertical line segments are replaced by exactly horizontal or vertical ones. After normalization, we can compute the feature P_{norm} , which is the linear combination of the following two subfeatures that measure the quality of intersection and the cleanness of the table skeleton (Fig. 2a), respectively:

$$f_{int} = 1 - \frac{\text{number of "bad" intersection points}}{\text{number of all intersection points}},$$

$$f_{cln} = 1 - \frac{\text{number of "invalid" line segments}}{\text{number of all line segments}}$$

 f_{cln} is computed because the system should reject a table candidate if there are too many strokes overlaying the ruling lines. After normalization, the invalid line segments and those that do not intersect with any other line segment are discarded because they do not contribute to the table structure.

3.2 Table Skeleton Structure Identification

Now, the intersection relationship among all ruling lines is available. The detector then recognizes the logical structure of the table by examining the type of intersection points.

There are three types of intersection points: cross-type, T-type, and L-type (Figs. 2b, 2c, and 2d). Similar classification of crossings has been used in [15] to register form images. Note that if an intersection point is not one of the four corners of the bounding box, then it can only be of cross-type or T-type. Therefore, the detector will remove all "erroneous" intersection points (Fig. 2a), i.e., those L-type intersection points that are not any of the four corner points of the table bounding box, by abandoning some line segments that form the intersection points until the remaining ruling lines form a valid table skeleton structure. The erroneous intersection points can also be fixed by inferring virtual ruling lines as done in [10]. However, this strategy is more reasonable for offline tables but less reasonable for online tables because for offline tables the ruling lines may be missing due to scanning and binarization, while for online tables every stroke forming the table should be available.

detected table skelet





Fig. 4. Examples of detected tables with different structure. Please notice the overtracing, concatenation, modification, and several-in-one-stroke phenomena of the ruling lines. Some degree of misclassification of writing/drawing from previous modules does not affect the table detection results.

After all erroneous intersection points are removed, the logical structure of the table candidate is determined. Then, the table cells are ordered according to the coordinates of their top-left corner, left to right and then top to bottom (Fig. 2e).

This step produces the confidence P_{str} on the logical structure:

$$P_{str} = 1 - \frac{\text{number of erroneous intersection points}}{\text{number of all intersection points}}.$$

This will be used for the final table detection decision in Section 4.

Table Cell Content Extraction 3.3

This step finds the content of each cell in order to complete the table information. It checks the bounding rectangle of each cell and those of other objects by the proportion of their common area in the bounding rectangle of an object (such as paragraph, line, word, and stroke. Note that now the rotation for table bounding box (Section 2.2) is also applied to the other strokes that are spatially close to the bounding box.). This is done from the top level to lower levels. Namely, if a writing paragraph is judged to be in a cell, the test stops. Otherwise, we continue to test the writing line. The process may continue to the stroke level. In [9], the cell content is identified by

collecting the *writing* strokes within each cell boundary and this is done at the stroke level only.

This step produces the confidence P_{con} on the content extraction defined as:

$$P_{con} = \sum_{i=1}^{N_{coll}} \sum_{j=1}^{N_{stroke}} \mu_{ij} \cdot \left(1 - \frac{C_{ij}}{\min(A_i, B_j)}\right)$$

where $C_{i,j}$ is the common area of the *i*th cell and the bounding rectangle of the *j*th stroke (Fig. 2f), A_i is the area of the *i*th cell, B_i is the area of the bounding rectangle of the *j*th stroke, μ_{ij} is 0.7 when the *j*th stroke is close to the corners of the *i*th cell and 1.0 when otherwise, and N_{cell} and N_{stroke} are the number of cells and the number of strokes that intersect the ruling lines of cells, respectively. We introduce μ_{ii} due to the consideration that if a stroke is on a ruling line, the confidence should decrease more when it is near the ends of the ruling line than when it is near the center, because the stroke is often an arrow head (Fig. 2f). Pcon is computed because a table skeleton should not be overlaid with too many strokes, as already mentioned in Section 3.1. Note that Pcon can be made more accurate if the convex hulls or the skewed bounding rectangles of strokes are used instead, at some price of the speed.

Sabatage

Other

の用

(4.1.26

百

第

る

Tot -1



Fig. 5. Examples of table manipulation. (a) An ink note with a skewed table. (b) The re-orientation and cell right-justification of the table. (c) Resizing and cell leftjustification of the table. (d) Converting the table into a chart. Note that the ruling lines in (b) and (c) are still natural ink strokes.

 TABLE 1

 Performance of Our Table Detection System at the Cell Level and the Table Level

Eval.	total	detected	correct	splitting	merging	missing	false	spurious	correct/	correct/
Level							alarm		detected	total
Cell	3547	3479	3297	6	52	91	114	2	94.77%	92.95%
Table	232	230	223	·					96.96%	96.12%

4 OVERALL DETECTION AND CLASSIFICATION SCHEME

As one may have seen, our detection scheme is hierarchical and procedural. A higher structure is detected only when lower structure exists, and only when all the features P_{BF} , P_{norm} , P_{str} , and P_{con} are above their corresponding thresholds can the system reach the final classification step, which uses a linear classifier that combines the four high-level features. If the classifier decides that it is a table, then both the physical and logical structure information is output, including the number of rows and columns, the sizes of the table bounding box, the transform matrix between the local table coordinate (the top-left corner of the table bounding box is the origin) and the global coordinate, the coordinates of intersection points among the ruling lines in the local table coordinate, a matrix indicating whether a cell has content, and which part of a stroke that forms the table skeleton corresponds to which ruling line, etc. Such information is valuable for further table manipulation. Conceptually, our detection procedure resembles a decision tree (Fig. 3), which is common in table detection and recognition [4].

5 EXPERIMENTAL RESULTS

Our testing data set has 378 ink notes, which contains 195 handwritten documents written in English, 84 in Chinese, 44 in French, and 55 in German. Many ink notes are of multiple pages. They were collected from many people with different nationalities, by asking them to rewrite, without any restriction on their writing styles (e.g., not necessary to write exactly in the horizontal direction), pages that were selected by several people from books or magazines across various categories. They may contain texts, graphics, diagrams, flow charts, tables, etc. Tables can be drawn at random, large or small, skewed or not, ruled or unruled. We only deem fully-ruled tables as real tables. Those that are unruled or semiruled are currently regarded as nontables because our system is not designed to detect such kinds of tables.

We adopt the method developed in [17] to evaluate the detection results quantitatively. This is done at the cell level, i.e., we count the correspondence between cells of ground truth and those detected, including correct (one-to-one match), splitting (one-to-many match), merging (many-to-one match), missing (one-to-zero match), false alarm (zero-to-one match), and spurious (many-to-many match). It is easy for human to count the occurrence of different kinds of cell-level correspondence by directly overlaying the detected table skeleton over the ink note (Fig. 4 and Fig. 5). The cell-level detection results are shown in the second row of Table 1. We also test our system at the table level, i.e., as long as a table is detected, it is regarded as correct, no matter whether the cells of the table are all detected correctly. The results are shown in the third row of Table 1. It can be seen that the accuracy is quite high at both the cell level and the table level.

Figs. 4 and 5a shows some concrete examples of detected tables, where the overlaid straight lines are the detected table skeletons.



Fig. 6. Examples of failure cases. (a) Failure caused by overly casual drawing of table skeletons. (b) Failure caused by incorrect grouping or writing/drawing classification of the strokes. (c) Whether an ink object is a table is highly dependent on the semantic context.

Please notice the overtracing, concatenation, modification, and multiple ruling lines that are drawn in one stroke in those figures. Moreover, some misclassified writing/drawing strokes do not affect the table detection results.

After table physical and logical structure information is extracted, table editing can be performed, such as rotation, scaling, moving the whole table, and alignment of cell content. Some table editing examples of our ink parser demo system are shown in Figs. 5b and 5c. Note that the ruling lines therein are still natural ink strokes. Working with a handwriting recognizer, a handwritten table can also be converted to a printed table or other graphical objects, e.g., charts, as shown in Fig. 5d.

Although our system is robust and accurate, there are still several factors that can cause failure. For example, overly casual drawing of the table skeletons (Fig. 6a) may make the bounding box or other ruling lines undetectable, and incorrect grouping or writing/ drawing classification (Fig. 6b) may cause spurious or misdetection of ruling lines. It is also interesting to note that semantic context also matters. It is possible that locally an ink object is a table, but globally it should not be regarded as a table (Fig. 6c). Similar phenomenon also occurs in other situations, e.g., [16].

CONCLUSIONS AND FUTURE WORK 6

We have presented a robust table detection system that is capable of extracting free style tables from online handwritten documents. This is a nontrivial task in that the ruling lines and layout of hand drawn tables are often far more irregular and complex than those of printed tables. Our methodology can also be extended and applied to other graphical objects, such as flowcharts, organization charts, bar or pie graphs, and annotations, whose structure can be defined without ambiguity. For example, when the primitive structure of a graphical object is lines, rectangles, and circles (e.g., the diagram in Fig. 6c), our system could detect it with some changes in detecting the primitive structure and logical structure.

As there are many parameters and thresholds in our system, currently only a portion of them have been manually turned on our specially designed test pages which have a dense population of tables/nontables on each page to facilitate quick human examination of detection accuracy and speed. They are believed to have significant influence on the accuracy and the speed. The rest are fixed to our estimated values. As long as the thresholds are not too high, it is the final classifier that mainly determines the detection accuracy. We plan to apply genetic algorithms to tune the thresholds and parameters in full scale, where the detection speed will also be considered in order to make our system practical. We would also like to try other kinds of classifiers other than the linear classifier in use, such as support vector machines. Moreover, other kinds of tables should also be considered, such as unruled or semiruled tables² [2]. This problem has been tackled for printed tables [6], [8], [12], but is still not completely solved. Jain et al. [9] also present an algorithm to detect unruled online tables by assuming that the white

bounding boxes.

2. Our current system can also detect ruled tables with incomplete

space is sufficient for table isolation. However, for general handwritten tables in handwritten documents, this problem is not trivial, because the space that isolates the tables may not be salient and the space between columns or rows may not be uniform since the estimation of the directions of columns and rows is usually inaccurate. For semiruled tables, the ruling lines may be hard to distinguish from underlines if no a priori knowledge is available. Hurst [18] also points out that without integrating linguistic information, some inherent ambiguity in the table structure cannot be resolved. Finally, the integration of offline or perceptual grouping [19] techniques is also valuable, e.g., for improving the treatment of overtracing and faster spatial grouping and searching.

ACKNOWLEDGMENTS

Junfeng He, Zhicheng Zhong, and Rongrong Wang were visiting students at Microsoft Research Asia when this work was done.

REFERENCES

- G. Nagy, "Twenty Years of Document Image Analysis in PAMI," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 22, no. 1, pp. 38-62, Jan. 2000.
- J.C. Handley, "Document Recognition," Electronic Imaging Technology, [2] E.R. Dougherty, ed., Bellingham, Wash.:IS & T/SPIE Optical Eng., chapter 8, pp. 289-316, 1999.
- D. Lopresti and G. Nagy, "A Tabular Survey of Automated Table Processing," Proc. Third Int'l Workshop Graphics Recognition, Recent Advances, [3]
- pp. 93-120, 1999. R. Zanibbi, D. Blostein, and J.R. Cordy, "A Survey of Table Recognition: [4] Models, Observations, Transformations, and Inferences," Int'l J. Document Analysis and Recognition, vol. 7, no. 1, pp. 1-16, 2004.
- [5]
- [6]
- Anarysis and Recognition, vol. 7, no. 1, pp. 1-10, 2004.
 D. Lopresti and G. Nagy, "Automated Table Processing: An (Opinionated) Survey," Proc. Third Int'l Workshop Graphics Recognition, pp. 109-134, 1999.
 J.H. Shamilian, H.S. Baird, and T.L. Wood, "A Retargetable Table Reader," Proc. IEEE Int'l Conf. Document Analysis and Recognition, pp. 158-163, 1997.
 E. Green and M. Krishnamoorthy, "Model-Based Analysis of Printed Tables," Proc. IEEE Int'l Conf. Document Analysis and Recognition, pp. 214-217, 1000 [7] 217, 1999.
- [8]
- T.G. Kieninger, "Table Structure Recognition Based on Robust Block Segmentation," *Proc. Fifth SPIE Conf. Document Recognition*, pp. 22-32, 1998. A.K. Jain, A. Namboodiri, and J. Subrahmonia, "Structure in On-Line Documents," *Proc. IEEE Int'l Conf. Document Analysis and Recognition*, [9] pp. 844-848, 2001.
- A. Laurentini and P. Viada, "Identifying and Understanding Tabular Material in Compound Documents," Proc. 11th Int'l Conf. Pattern Recogni-[10] tion, pp. 405-409, 1992.
- [11] J. Sklansky and V. Gonzalez, "Fast Polygonal Approximation of Digitized Curves," *Pattern Recognition*, vol. 12, pp. 327-331, 1980. Y. Wang, I.T. Phillips, and R.M. Haralick, "Table Structure Understanding
- [12] and Its Performance Evaluation," Pattern Recognition, vol. 37, no. 7, pp. 1479-1497, 2004.
- L.B. Kara and T.F. Stahovich, "Hierarchical Parsing and Recognition of [13] Hand-Sketched Diagrams," Proc. 17th ACM Symp. User Interface Software and Technology, pp. 13-22, 2004.
 C. Alvarado, "A Framework for Multi-Domain Sketch Recognition," Proc.
- AAAI Spring Symp. Sketch Understanding, AAAI Technical Report SS-02-08, Stanford Univ., pp. 1-8, 2002. S.L. Taylor, R. Fritzson, and J.A. Pastor, "Extraction of Data from
- [15] Preprinted Forms," Machine Vision and Applications, vol. 5, pp. 211-222, 1992
- J.J. LaViola Jr. and R.C. Zeleznik, "MathPad2: A System for the Creation [16] and Exploration of Mathematical Sketches," ACM Trans. Computer Graphics, vol. 24, no. 3, pp. 432-440, 2004.
- J. Liang, "Document Structure Analysis and Performance Evaluation," PhD [17] thesis, Univ. of Washington, Seattle, 1999.
- [18] M. Hurst, "Layout and Language: An Efficient Algorithm for Detecting Text Blocks Based on Spatial and Linguistic Evidence," Proc. Document Recognition and Retrieval VIII (IS & T/SPIE Electronic Imaging), vol. 4307, pp. 56-67, 2001.
- E. Saund, "Finding Perceptually Closed Paths in Sketches and Drawings, [19] IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 25, no. 4, pp. 475-491, Apr. 2003.

> For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.