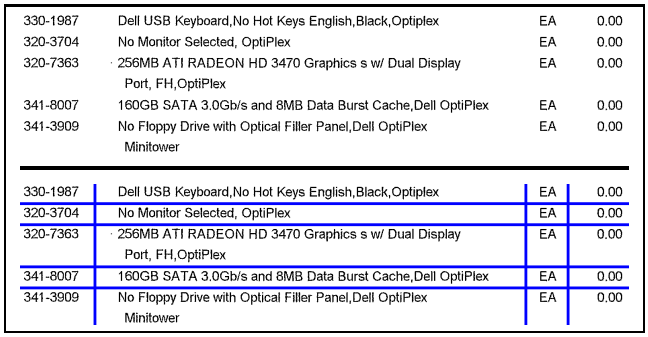
2012 10th IAPR International Workshop on Document Analysis Systems

Parsing tables by probabilistic modeling of perceptual cues

Evgeniy Bart
Intelligent Systems Lab
Palo Alto Research Center
Palo Alto, CA 94304
bart@parc.com

***Abstract*—In this paper, we propose a method for automatically parsing images of tables, focusing in particular on 'simple' matrix-like tables with rectilinear layout. Such tables account for over 50% of tables in business documents. The main novelty of the proposed method is that it combines intrinsic properties of table cells with properties of cell separators, as well as table rows, columns, and layout, in a single global objective function. This is in contrast to previous methods which focused on either separators alone or intrinsic cell properties alone. Our method uses a variety of perceptual cues, such as alignment and saliency, to characterize these properties. Candidate parses are evaluated by comparing their likelihoods, and the parse that optimizes the likelihood is selected. The proposed approach deals successfully with a wide variety of tables, as illustrated on a dataset of over 1,000 images.**

***Keywords*-table parsing, document analysis**



330-1987	Dell USB Keyboard,No Hot Keys English,Black,Optiplex	EA	0.00
320-3704	No Monitor Selected, OptiPlex	EA	0.00
320-7363	256MB ATI RADEON HD 3470 Graphics s w/ Dual Display Port, FH,OptiPlex	EA	0.00
341-8007	160GB SATA 3.0Gb/s and 8MB Data Burst Cache,Dell OptiPlex	EA	0.00
341-3909	No Floppy Drive with Optical Filler Panel,Dell OptiPlex Minitower	EA	0.00

330-1987	Dell USB Keyboard,No Hot Keys English,Black,Optiplex	EA	0.00
320-3704	No Monitor Selected, OptiPlex	EA	0.00
320-7363	256MB ATI RADEON HD 3470 Graphics s w/ Dual Display Port, FH,OptiPlex	EA	0.00
341-8007	160GB SATA 3.0Gb/s and 8MB Data Burst Cache,Dell OptiPlex	EA	0.00
341-3909	No Floppy Drive with Optical Filler Panel,Dell OptiPlex Minitower	EA	0.00

Figure 1. An example table parsed by the proposed algorithm. Top: the original table image. Bottom: the parse obtained by the algorithm. The blue lines denote the parsed structure. Note that the table is parsed correctly even though different rows occupy different number of text lines.

I. Introduction

Tables are widely used to arrange data for display. They are especially prevalent in business documents (such as invoices) and scientific literature. Many web sites also use tabular layout to provide access to an underlying database (such as a database of cars for sale or a database of gene sequences). Parsing such tables automatically would allow access to large quantities of data and has therefore a significant practical value. This access could also become the first step towards a more intelligent use of data, for example, for the purposes of data mining and business analytics.

Tables may have different structure and may be displayed in a variety of layouts [1]. In this paper, we focus on the class of 'simple' tables. These are matrix-like tables where cells are laid out on a rectilinear two-dimensional grid. In some of these tables, neighboring cells may be merged into 'compound cells'; tables with such compound cells are not considered here. A consequence of not having compound cells is that cell separators span (uninterrupted) the entire extent of the table. Several examples of such simple tables are shown in Figures 1, 3. Although not all tables are simple, simple tables do account for a significant fraction of tables in business documents. We surveyed invoices received by one medium and two large companies and found that in each company, over 50% of all invoices contain only simple tables. Large businesses may process tens of thousands of invoices per day, and manually processing a single invoice can cost up to 9 Euro [2]. Automating even a fraction of this processing could therefore provide significant cost savings.

In this paper, we propose a method for automatically parsing images of simple tables. The main novelty of the proposed method is that it combines intrinsic properties of table cells with properties of cell separators, as well as table rows, columns, and layout, in a single global objective function. This is in contrast to previous methods which focused on either separators alone or intrinsic cell properties alone. Our method uses a variety of perceptual cues, such as alignment and saliency, to characterize these properties. The Naive Bayes model is used to estimate the likelihood of a given parse. The likelihoods of multiple candidate parses are compared, and the parse that optimizes the likelihood is selected. The proposed approach deals successfully with a wide variety of tables, as illustrated on a dataset of over 1,000 images.

The remainder of this paper is organized as follows. In section II, we survey the relevant previous work. In section III, we describe the proposed table parsing method. The experimental evaluation of this method is presented in section IV, followed by conclusions in section V.

II. A survey of previous work

In this section, we discuss several approaches to table parsing that are most relevant to our proposed method. Additional discussion can be found in [3], [4], [1].

978-0-7695-4661-2/12 $26.00 © 2012 IEEE
DOI 10.1109/DAS.2012.67

One of the most widely used approaches to table parsing is based on identifying row and column separators. Once separators are identified, the location and content of each cell is uniquely determined. Cues such as alignment of text, presence of whitespace, and rule lines, are often used to detect separators [5], [6], [7], [8]. These cues exploit the observation that text in rows and columns of a table is often aligned, and cell boundaries are often indicated by rule lines and whitespace. Additional cues may be used for specialized applications. For example, in [9], the focus is on tables found in invoices. Such tables typically list products and the price for each. In many cases, the price is the only real number in each row; in this case, finding a real number is a good indication of where one row ends and another begins.

A general criticism of approaches that rely on separator finding is that separators are often ambiguous. For example, in Figure 1 the spaces between multiple lines within a single row are similar to the spaces between different rows. Distinguishing row separators from text line separators based on properties of these spaces alone is therefore difficult.

In addition to finding cell boundaries, characterizing the properties of cells themselves is useful, particularly in situations where boundaries are ambiguous. For example, using word bigrams to estimate coherence of a block of text was used in [10]. Other examples include [11], where cells that produce satisfiable constraints are favored; and [12], where formatting similarity is favored across cells. An additional novelty of the method proposed here is that it introduces several additional cell properties that significantly improve performance.

There exist many table styles and formatting conventions. Case-based reasoning may be used to deal with each style separately; this refers to assigning a given input table (explicitly or implicitly) to a particular style and then invoking a style-specific parser. Styles may be defined at different levels of granularity. For example, in [7], a database of vendor-specific table structures is used to parse tables in invoices; each vendor thus has a narrowly-defined table style. In [13], [5], wider style definitions are used: tables are grouped by their periodicity structure in [13], and by the separator type in [5]. A benefit of case-based reasoning is that it may be easier to tune an algorithm for a subset of inputs. The main disadvantage is that it may be difficult to encompass all cases of interest. In contrast, generic methods such as [6], [11] are able to deal with unknown or unanticipated table styles.

III. The Proposed Method

In this section, we describe the proposed table parsing method in detail.

First, note that the input image is assumed to only contain the table to be parsed. In a complete system, this would be achieved by a separate table detection stage; see e. g. [6], [8], as well as the discussion in section V.

330-1987	Dell USB Keyboard,No Hot Keys English,Black,Optiplex	EA	0.00
320-3704	No Monitor Selected, OptiPlex	EA	0.00
320-7363	256MB ATI RADEON HD 3470 Graphics s w/ Dual Display Port, FH,OptiPlex	EA	0.00
341-8007	160GB SATA 3.0Gb/s and 8MB Data Burst Cache,Dell OptiPlex	EA	0.00
341-3909	No Floppy Drive with Optical Filler Panel,Dell OptiPlex Minitower	EA	0.00

Figure 2. Candidate cell separators in a table, denoted by green lines. Note that some of these candidates are spurious and do not correspond to any real separator.

The input image is first preprocessed by deskewing and denoising. Next, OCR is run and text tokens are extracted (these usually correspond to individual words or characters). Finally, projection profiles are calculated and thresholded to find candidate cell (row or column) separators. Typically, we obtain 5–15 candidate vertical separators per table, and one candidate horizontal separator between every pair of text lines. These are illustrated in Figure 2. Note that many of these candidates are spurious and do not correspond to any real cell separator. However, we do assume that the threshold is high enough that all real separators are captured.

The task now becomes to determine which of the candidate separators are real (i. e. correspond to a real cell separator) and which ones are spurious. This task is solved by optimizing an objective function over the set of separators. This objective function has one boolean variable for every candidate separator; a `true` value indicates that the corresponding separator is real and a `false` value indicates that it is spurious. The function measures the likelihood that the parse produced by the set of real separators is correct. The objective function is described next, and we describe the optimization at the end of this section.

The objective function includes several terms, each corresponding to a different aspect of a table. The aspects that are considered are the row/column separators; the individual cells; entire rows and columns; and repeatability of table structure. The terms for each aspect are detailed below.

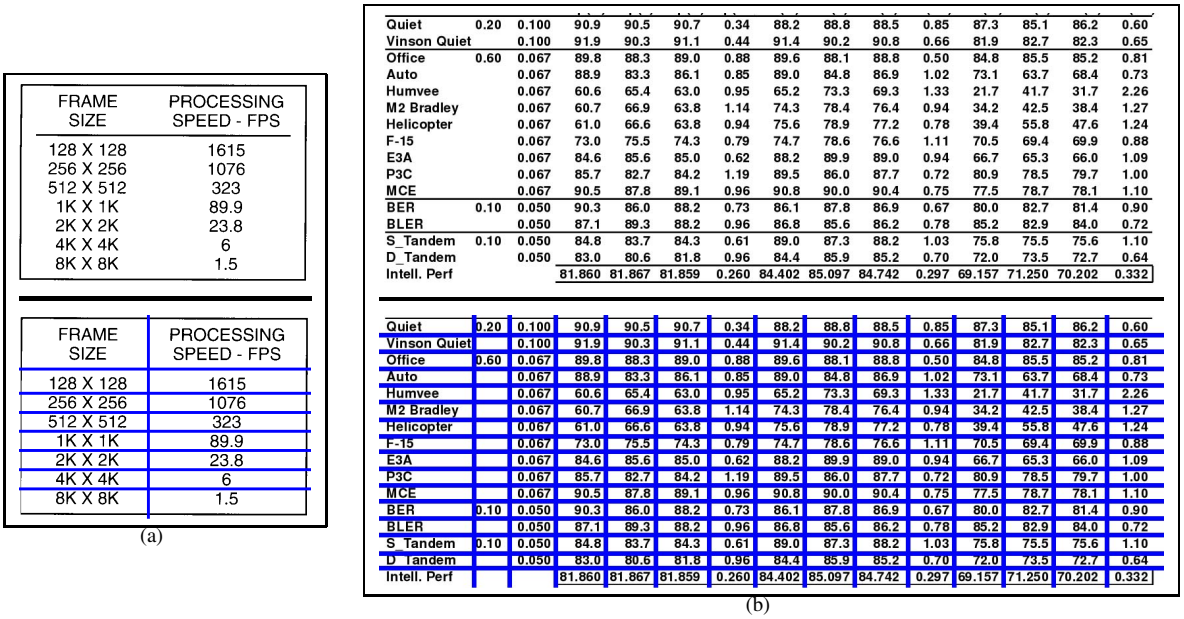
A. Separators

The purpose of this term is to evaluate the quality of the proposed cell separators. This is performed by extracting features for each separator, and using a Naive Bayes classifier to compute the log-likelihoods of these features. This process is detailed next.

The following features are extracted for each separator:

- Width and height.
- Width of rule lines (if any) within the current separator.
- Number of tokens aligned at endpoints of the separator.
- Number of tokens that intersect the separator.

A Naive Bayes model is trained to categorize separators as either real or spurious. A manually annotated set of tables is used for this training. All candidate separators are extracted; separators which correspond to a ground truth cell boundary become positive examples, and the remaining



FRAME SIZE	PROCESSING SPEED - FPS
128 X 128	1615
256 X 256	1076
512 X 512	323
1K X 1K	89.9
2K X 2K	23.8
4K X 4K	6
8K X 8K	1.5

(a)

Quiet	0.20	0.100	90.9	90.5	90.7	0.34	88.2	88.8	88.5	0.85	87.3	85.1	86.2	0.60
Vinson Quiet		0.100	91.9	90.3	91.1	0.44	91.4	90.2	90.8	0.66	81.9	82.7	82.3	0.65
Office	0.60	0.067	89.8	88.3	89.0	0.88	89.6	88.1	88.8	0.50	84.8	85.5	85.2	0.81
Auto		0.067	88.9	83.3	86.1	0.85	89.0	84.8	86.9	1.02	73.1	63.7	68.4	0.73
Humvee		0.067	60.6	65.4	63.0	0.95	65.2	73.3	69.3	1.33	21.7	41.7	31.7	2.26
M2 Bradley		0.067	60.7	66.9	63.8	1.14	74.3	78.4	76.4	0.94	34.2	42.5	38.4	1.27
Helicopter		0.067	61.0	66.6	63.8	0.94	75.6	78.9	77.2	0.78	39.4	55.8	47.6	1.24
F-15		0.067	73.0	75.5	74.3	0.79	74.7	78.6	76.6	1.11	70.5	69.4	69.9	0.88
E3A		0.067	84.6	85.6	85.0	0.62	88.2	89.9	89.0	0.94	66.7	65.3	66.0	1.09
P3C		0.067	85.7	82.7	84.2	1.19	89.5	86.0	87.7	0.72	80.9	78.5	79.7	1.00
MCE		0.067	90.5	87.8	89.1	0.96	90.8	90.0	90.4	0.75	77.5	78.7	78.1	1.10
BER	0.10	0.050	90.3	86.0	88.2	0.73	86.1	87.8	86.9	0.67	80.0	82.7	81.4	0.90
BLER		0.050	87.1	89.3	88.2	0.96	86.8	85.6	86.2	0.78	85.2	82.9	84.0	0.72
S_Tandem	0.10	0.050	84.8	83.7	84.3	0.61	89.0	87.3	88.2	1.03	75.8	75.5	75.6	1.10
D_Tandem		0.050	83.0	80.6	81.8	0.96	84.4	85.9	85.2	0.70	72.0	73.5	72.7	0.64
Intell. Perf			81.860	81.867	81.859	0.260	84.402	85.097	84.742	0.297	69.157	71.250	70.202	0.332

(b)

Figure 3. Additional examples of tables parsed by the proposed algorithm. In each box, the top part is the original table image, and the bottom part is the parse produced by the algorithm. Note that the tables are parsed correctly despite lack of clear separators, lack of periodicity, mixed separator types, and presence of empty cells.

(spurious) separators become negative examples. The log-likelihood ratio is then

$$L(\{f_i\}_{i=1}^n) = \log \frac{p(\{f_i\}_{i=1}^n|\text{real})}{p(\{f_i\}_{i=1}^n|\text{spurious})} = \sum_{i=1}^{n} w_i[f_i], \quad (1)$$

where f_i is the value of the i'th feature and w_i is the weight assigned to that value. These weights are learned by computing the probabilities $p(f_i = f_0|\text{real})$ and $p(f_i = f_0|\text{spurious})$ from the training data and setting the weight to $w_i[f_0] = \log \frac{p(f_i=f_0|\text{real})}{p(f_i=f_0|\text{spurious})}$. Note that this process is completely automatic. We compute a separate set of weights for the horizontal and vertical separators to capture any systematic differences in their properties. To compute the overall quality of all separators in a table, the log-likelihoods for each separator are simply added. The resulting quantity is denoted $L_{\text{separators}}$.

Note that by comparing the log-likelihood of each separator to a threshold, we could obtain binary 'real/spurious' decisions. However, binary decisions at this early stage would lead to poor performance, especially in cases where the separators are ambiguous and cannot be determined by their visual appearance alone. Therefore, we use the log-likelihood as a component in a global optimization criterion instead of making binary decisions.

B. Cells

The purpose of this term is to evaluate the coherence of the table cells. Overall, this term is handled similarly to the term above.

First, the cell locations are determined from the boundaries with the corresponding variables set to `true` (these are the hypothesized real separators). For each cell, features that measure the cell's perceptual coherence are extracted. These features are:

- The size of the largest horizontal whitespace within the cell. The idea is that a coherent block of text is usually typeset without large gaps; a large gap therefore indicates that the candidate is in fact an aggregation of multiple cells.
- The size of the largest vertical whitespace within the cell.
- Whether the cell is 'properly terminated'. The value of this feature is 0 if the cell text is terminated improperly, defined in our case as ending with a dash or a comma. Otherwise, the cell is considered to be terminated properly, and the value of this feature is 1. The idea is that text rarely ends with these characters; therefore, improper termination may indicate that the cell was oversegmented. Note that a much more elaborate way to measure proper continuation was proposed in [10]. It is, however, difficult to apply it in our case. The reason is that invoices often include product names, codes, and other nonstandard words, and their statistics is quite different from that of natural language.
- The number of text lines in the cell that only include numeric characters (i. e., digits, periods, commas, and dashes). The idea here is that data in a purely numerical cell usually spans just one text line, and multi-line cells

usually include text rather than just numbers.
- The size of the largest unfilled space within a cell. A space at the end of a text line is considered 'unfilled' if the first token on the subsequent text line could have fit within it. The idea is, again, that multi-line cells are typeset so that no large gaps are present; a large unfilled space therefore indicates that the candidate is in fact an aggregation of multiple cells.

The latter two features, while relatively straightforward, are rarely used in the literature. However, they contribute quite significantly to the method's performance. When those features are removed, the performance drops by 31 percentage points.

As before, a Naive Bayes model is trained and used to compute the log-likelihood for each cell. These individual log-likelihoods are aggregated over all cells and the resulting quantity is denoted L_{cells}.

C. Row and column coherence

One additional term is used to evaluate the coherence of an entire row of the table. As above, the cells for a candidate parse are determined. An entire row is considered at a time, and the following features are measured:

- The number of empty cells in the current row
- The number of non-empty cells in the current row
- The degree of cell alignment (using top-, bottom-, or center-alignment, whichever is best for the current candidate).

The likelihood per row is learned using the Naive Bayes model. The likelihood of all rows is aggregated in a term denoted L_{rows}.

A similar term for columns (denoted L_{columns}) is computed, based on similar features (the only difference is that top-, bottom-, and center-alignment are replaced by left-, right, and center-alignment, respectively).

D. Layout consistency

Finally, one more term is used to evaluate the consistency of layout across the table rows. We use the model developed in [14] for this purpose. The likelihood returned by this method is denoted L_{layout}.

E. The global objective function

The terms described above are used to specify a global objective function. This objective function provides a numerical score for a proposed set of cell separators. The simplest way to combine all terms would be by adding them. In practice, the terms corresponding to individual cells (L_{cells}) and to the layout (L_{layout}) were found to be the most important. These terms were therefore given (manually) a larger weight. The resulting objective function is thus

$$\begin{aligned} O &= L_{\text{separators}} + 10 \cdot L_{\text{cells}} + L_{\text{rows}} \\ &\quad + L_{\text{columns}} + 10 \cdot L_{\text{layout}}. \end{aligned} \tag{2}$$

Train on \ Test on	invoices 1	invoices 2
invoices 1	91% ± 1%	86%
invoices 2	88%	86% ± 2%

Table I
THE TABLE PARSING PERFORMANCE. SHOWN: % CORRECT (SEE SECTION IV-A). HIGHER VALUES INDICATE BETTER PERFORMANCE.

In the future we plan to investigate ways of learning these weights automatically.

F. Optimization

The objective function specified above provides a numerical score for a proposed set of cell separators. The parsing is performed by selecting a subset of the candidate separators which maximizes this objective function. This optimization process is detailed below.

The column separators and considered in the order of decreasing separator score (section III-A). For each such subset of column separators, the best subset of row separators is selected as described below. The best (in terms of the overall score) table is selected from this set as the optimal parse.

The selection of the optimal subset of row separators for the given set of column separators is performed essentially by a brute force search. One slight optimization is that this search is done progressively from top to bottom. This is possible because the bottommost rows only weakly affect the decisions of the topmost rows.

Note again that both the parameter setting (tuning the weights w) and the optimization process are completely automatic. The only manual input into the system is the set of ground truth table parses at the training stage.

IV. EXPERIMENTS

Two datasets were used in the experiments. Each of the datasets contained tables from invoices received by a particular company. The dataset called 'invoices 1' contained 111 tables, and the dataset called 'invoices 2' contained 976 tables. All tables had between two and 25 rows and between two and 11 columns.

A. Evaluation

Several methods for evaluating table parsing exist [15]. Of these, graph probing seems to measure accuracy in terms that most closely reflect our target application. Originally, graph probing methods included several types of queries. It was therefore necessary somehow to set relative weights between different query types. We have further adapted the graph probing method to suit our target application. This adaptation, described below, also eliminates the need for weighting, since only one type of queries is used.

Our target application involves two scenarios. In one, items from a database are looked up in the parsed invoice. In another, the parsed table is used as an index into the

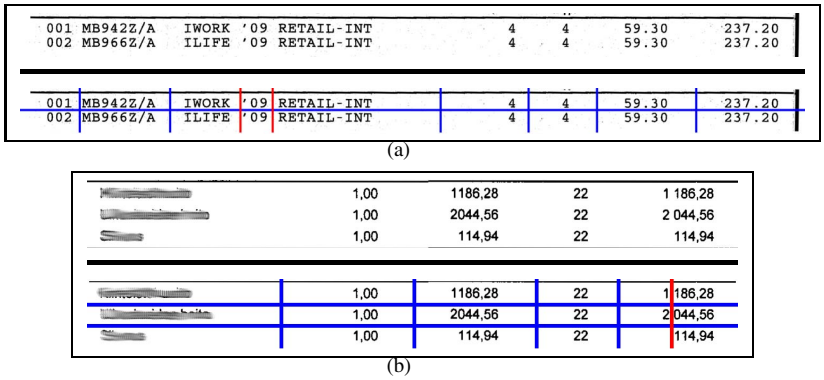


Figure 4. Typical examples of errors made by the algorithm. In each box, the top part is the original table image, and the bottom part is the parse produced by the algorithm. The incorrect separators are shown in red. In both cases, spurious alignment of text has caused the algorithm to generate incorrect separators. Some text in the bottom example is redacted for confidentialily.

database. It therefore is appropriate to pair one row (or column) of the ground truth table with the most similar row (or column) of the proposed parse, since this pairing will occur when the database is matched to the parsed table or vice versa. Once this matching has occured, the target metric is the number of mismatches between the ground truth and the parse. We use edit distance between the text content of corresponding cells to calculate this number of mismatches, and express it as percent of correct matches.

B. Results

The evaluation method outlined above was used in the experiments. Four experiments were performed, where one of the datasets was used for training the model and another (possibly the same) dataset was used for testing. The results are reported in Table I. Higher numbers indicate better performance. When the same dataset was used for training and testing (the diagonal entries in Table I), the training and testing sets were disjoint; i. e., the model was trained on a randomly selected subset of the data and tested on a disjoint subset of the data. This process was repeated five times; the average performance and the standard deviation are shown. When the training and testing datasets were different (the off-diagonal entries), all images in the training and testing sets were used; therefore, only one number is shown.

As can be seen, the proposed method achieves above 85% correct performance under all training and testing conditions. Note in particular that the generalization performance (i. e. the performance when training on one dataset and testing on a different dataset) is good. This indicates that the model generalizes easily to unfamiliar tables.

Several examples of correctly processed tables are shown in Figures 1, 3. Since most of the images in our datasets contained proprietary data, an additional set of tables was used for these illustrations. This set contained some of the author's personal invoices, as well as tables from scientific publications. As can be seen, the proposed method can successfully deal with lack of clear separators (Figures 1, 3(a), where text line separators are identical to cell separators), lack of periodicity (Figures 1, 3(a), where different rows have different number of text lines), mixed separator types (Figures 3(a), 3(b), where some row separators have rule lines and others don't), and the presence of empty cells (Figure 3(b)). Traditionally such tables have been difficult to deal with [13], [7], [9].

Despite not using a sophisticated search method, the proposed approach is quite efficient computationally. Depending on table size, 3–10 seconds are needed to parse a typical table using our Java implementation on a 2.5 GHz CPU. Most of this processing time is taken up by OCR.

Typical errors in parsing are illustrated in Figure 4. As can be seen, repetition and spurious alignment of text created a structure that is visually similar to a table column. Note that the width of these spurious separators is similar to the width of real separators (e. g. in Figure 4(a), both of the spurious separators are wider than the leftmost correct separator). In some cases, higher-level knowledge seems necessary to deduce that these separators are spurious. Fortunately, in other cases a better feature set seems sufficient to deal with the problem. For example, in Figure 4(b), the thousands digits in prices created a spurious column. A feature set that models this and similar typesetting conventions could be easily incorporated into the proposed framework.

V. Discussion

We have presented a method for parsing simple tables. The method uses a wide variety of perceptual cues to characterize different aspects of the table, including the separators, cells, rows, and columns. A probabilistic model is trained to evaluate these characteristics and select the optimal parse. The method achieves over 85% correct performance on a variety of realistic datasets.

An important assumption in the proposed method is that the table has already been detected in the image. This is a common assumption in related methods (see section II).

Nevertheless, finding a more general table detection method is a desirable extension. Additional directions for future research include formulating better objective functions (for example, by using models more sophisticated than Naive Bayes) and better optimization techniques.

REFERENCES

[1] D. P. Lopresti and G. Nagy, "A tabular survey of automated table processing," in *GREC '99*, 2000, pp. 93–120.

[2] B. Klein, S. Agne, and A. Dengel, "Results of a study on invoice-reading systems in Germany," in *DAS*, 2004.

[3] R. Zanibbi, D. Blostein, and J. R. Cordy, "A survey of table recognition: Models, observations, transformations and inferences," *IJDAR*, vol. 7, no. 1, pp. 1–16, 2004.

[4] D. W. Embley, M. Hurst, D. Lopresti, and G. Nagy, "Table-processing paradigms: a research survey," *IJDAR*, vol. 8, no. 2, pp. 66–86, 2006.

[5] K. Zuyev, "Table image segmentation," in *ICDAR*, 1997.

[6] J. Hu, R. S. Kashi, D. P. Lopresti, and G. Wilfong, "Medium-independent table detection," in *DRR VII*, 2000.

[7] B. Klein, S. Gokkus, T. Kieninger, and A. Dengel, "Three approaches to "industrial" table spotting," in *ICDAR*, 2001.

[8] F. Shafait and R. Smith, "Table detection in heterogeneous documents," in *DAS*, 2010.

[9] Y. Belaid and A. Belaid, "Morphological tagging approach in document analysis of invoices," in *ICPR*, 2004.

[10] M. Hurst, "Layout and language: An efficient algorithm for detecting text blocks based on spatial and linguistic evidence," in *DRR VIII*, 2001.

[11] ——, "A constraint-based approach to table structure derivation," in *ICDAR*, 2003.

[12] I. A. Doush and E. Pontelli, "Detecting and recognizing tables in spreadsheets," in *DAS*, 2010.

[13] H. Hamza, Y. Belaïd, and A. Belaïd, "Case-based reasoning for invoice analysis and recognition," in *Proc. 7th Intl. Conf. on Case-Based Reasoning*, 2007.

[14] E. Bart and P. Sarkar, "Information extraction by finding repeated structure," in *DAS*, 2010.

[15] J. Hu, R. S. Kashi, D. Lopresti, and G. T. Wilfong, "Evaluating the performance of table processing algorithms," *IJDAR*, vol. 4, no. 3, pp. 140–153, 2002.