

Trainable table location in document images

F. Cesarini*, S. Marinai*, L. Sarti⁺, G. Soda*

* *DSI - Università di Firenze*
Via S.Marta, 3 - Firenze - Italy
{cesarini,simone,giovanni}@dsi.unifi.it

+ *DII - Università di Siena*
Via Roma, 56 - Siena - Italy
sarti@dii.unisi.it

Abstract

We describe an approach for table location in document images. The documents are described by means of a hierarchical representation that is based on the MXY tree. The presence of a table is hypothesized by searching parallel lines in the MXY tree of the page. This hypothesis is afterwards verified by locating perpendicular lines or white spaces in the region included between the parallel lines. Lastly, located tables can be merged on the basis of proximity and similarity criteria.

The use of an optimization method, that relies on the definition of an appropriate table location index, allows us to identify the optimal values of thresholds involved in the algorithm. In this way the algorithm can be adapted to recognize tables with different features by maximizing the performance on an appropriate training set.

The algorithm has been evaluated on two data-sets containing more than 1500 pages, and comparing its results with the tables identified by two commercial OCRs.

1. Introduction

Nowadays, document image analysis covers a broad range of documents. In all these documents, special tasks are required for recognizing and managing specific structures. Some of the most relevant structures that can be found in several document classes are tables. Two main processing steps are useful for table analysis: table location and table understanding. The purpose of table location is the identification of regions in document images that correspond to tables. Table understanding is performed in order to extract the table structure, for instance by discovering the number of rows and columns in the table.

Table location methods are usually designed to recognize some specific table categories. However, several table layouts can be found in different collections of documents. Ta-

bles can be delimited by both horizontal and vertical lines, only by parallel lines (either horizontal or vertical), or by no lines at all. Table location methods can be roughly split into two main classes: methods based on separator identification and methods based on table layout knowledge.

Methods looking for separators are characterized by the data structure used for document representation. For instance, the method described by Chandran and Kasturi [2] identifies lines and spaces delimiting tables into the connected component list representing the document. This data structure does not contain much information about the mutual position of components. Another approach based on line identification was more recently proposed by Hu et al. [3]. The essence of the approach is the assignment of a score to each line, proportional to the probability that the line is member of a table. Tables are afterwards identified by maximizing the score associated to nearest lines. An approach not relying on the presence of lines is presented in [5]. This method groups together words by considering their adjacency relationship. Table location is afterwards performed with a bottom-up approach.

Knowledge-based methods define a document model which describes the main features of tables belonging to a given domain [4, 6]. During table location, a comparison between a given document and the document model is needed in order to properly extract document information. Performance of these methods are heavily affected by the document model. In fact, satisfactory result can only be achieved when dealing with documents well described by the model.

The method that we propose can be assigned to the search for separators class. In contrast with flat representations we use all the additional information contained in hierarchical data structures (the Modified X-Y tree [1]). One important objective is to reduce at most as possible the prior knowledge needed for table location. The main requirement of our approach, that defines also a constraint on the kind of tables that can be detected, is the presence of at least two

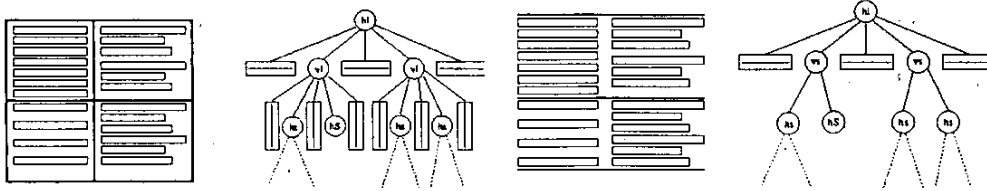


Figure 1. Left: MX Y-subtree representing a table delimited by either horizontal and vertical lines. Right: MX Y-subtree representing a table delimited by parallel lines.

parallel lines in the table structure. The algorithm can be trained to recognize some classes of tables with an optimization method, that is used for the choice of thresholds needed by the table location algorithm. This approach is somehow similar to the parameter selection approach proposed in [7] in the case of segmentation algorithms. However, the table location problem presents some peculiarities that are reflected in the definition of an appropriate “table detection index” which specifies the function to be maximized during the optimization step.

In Section 2 we describe the table location method, whereas in Section 3 we analyze how the thresholds involved in the algorithm are chosen by an optimization process. In Section 4 we report some experimental results, followed by concluding remarks in Section 5.

2. Table location

Top-down document image representations are very appropriate for table location. Examples of these descriptions are the XY tree decomposition, and its extension based on cuts along horizontal and vertical lines (MX Y tree [1]). In the work described in this paper we use MX Y trees as a hierarchical representation where to look for sub-images that can correspond to tables. MX Y trees are more appropriate than the classical XY decompositions, since tables contain very frequently horizontal and vertical lines. As an example, Figure 1 reports two simple line-delimited tables, together with their MX Y trees.

The table detection algorithm is based on a recursive analysis of the MX Y tree of a page, that is performed in order to identify regions that are surrounded by horizontal (vertical) lines. The search is refined by looking for further parallel lines that can be found in deeper levels of the tree. Starting from the siblings of the lines just found, the algorithm performs a depth-first search in the tree. This additional search is necessary because noisy document images can be cut along lines member of the same table at different levels of the tree (an example is shown in Figure 2). The hypothesis that a region corresponds to a table is verified by

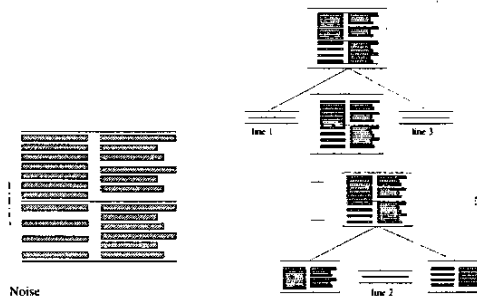


Figure 2. Decomposition of a table at different levels in the MX Y tree due to the presence of noise.

searching vertical (horizontal) lines or spaces in the regions included between the two parallel lines. If at least two vertical (horizontal) lines are found, then a sub-table (or a whole table) has been found, and it is added to the list of tables. Otherwise, the algorithm looks for perpendicular cuts along white strips that split the region into columns. A sub-table is located if some cuts (satisfying appropriate geometrical properties) are found. After the complete tree analysis, the main algorithm handles the table list by merging sub-tables belonging to one table and discarding tables smaller than a given threshold.

Many algorithms in the document processing field (and also in the broader domain of Pattern Recognition) are heavily based on appropriate values of some thresholds that define their behavior. The proposed table location algorithm is not an exception, since it uses five thresholds. The first threshold is considered in order to decide whether two parallel lines have the same length. The second threshold is used for checking the length of perpendicular lines during hypothesis verification, whereas the third threshold is involved in the search for white strips that separate table columns. The last two thresholds measure the minimum height and width allowed.

In the following section we describe the approach that

we propose for optimal threshold selection.

3. Threshold selection

In order to choose the “best” threshold values, we defined an optimization problem that depends on the identification of a function that measures the segmentation performance with an unique value.

In the most general case, table location algorithms can correctly locate tables, or give rise to four main error classes: split, merged, missed and false tables. Tables correctly detected are member of *Located* class. Partially located tables belong to *Split* class, whereas a single detected table corresponding to more original ones belongs to the *Merged* class. The *Missed* class contains “true” tables that are not located, the *False* class contains predicted tables which are not really present in the document image. In order to measure the performance of the table location algorithm, by taking into account all the previously listed classes of errors, we defined a *Table Location Index (TLI)*, as explained in the following.

Let us define two sets of tables: G and T . G contains all the tables that are actually present in the image set (the ground truth). The tables found by the algorithm described in Section 2 belong to the set T , which can be partitioned into four subsets: L (*Located*), F (*False*), S (*Split*), M (*Merged*). In addition, we define also the tables in G that are not found by the algorithm (Mis): In order to identify the tables in G that give rise to tables in M , we define the following subset of G :

$$\bar{M} = \{t \in G \mid \exists t' \in M, Int(t, t') \neq \emptyset\}, \quad (1)$$

where $Int(t, t')$ denotes the intersection of the bounding boxes corresponding to t and t' . Let us define also two functions: $S : S \rightarrow G$, $\mathcal{M} : \bar{M} \rightarrow M$. $S(t)$ associates to each table $t \in S$, the table in G that is split by the algorithm. $\mathcal{M}(t)$ associates to each table $t \in \bar{M}$ the corresponding table in M . Both functions are surjective but are not injective. Lastly we define the function A , that associates to a table the corresponding area. Using the previous notation, we can define the *Table Location Index*:

$$TLI(T, G) = \frac{|L| + \sum_{t \in S} \frac{A(t)}{A(S(t))} + \sum_{t \in \bar{M}} \frac{A(t)}{A(\mathcal{M}(t))}}{|G| + |F|} \quad (2)$$

It can be easily verified that $0 \leq TLI \leq 1$. Greater values of TLI correspond to better location performance. In particular the index increases if the number of located tables grows up and decreases if missed or false classes raise. When split tables are found, then TLI increases proportionally to the ratio between the split table area and the area of the table actually present in the document image. Similar considerations hold for merged tables.

3.1. Optimization problem

The optimal threshold values ($Th_i, i = 1, \dots, 5$) are chosen in order to maximize the performance of the table location algorithm on a training set of images I , with ground truth G . In the following, $T = TabFind(I, Th_1, \dots, Th_5)$ will denote the use of the table location algorithm for the location of tables T on the image set I with thresholds Th_1, \dots, Th_5 . The performance are measured by TLI , that indirectly depends on the five thresholds:

$$TLI(T, G) = TLI(TabFind(I, Th_1, \dots, Th_5), G). \quad (3)$$

By using an optimization method, we can maximize $TLI(T, G)$. At the end of the process the optimal values Th_1, \dots, Th_5 are computed. The choice of the optimization method is fundamental for threshold selection and it directly depends on the function TLI . We can describe the optimization problem as follows:

$$\begin{aligned} \max \quad & TLI(TabFind(I, Th_1, \dots, Th_5), G) \\ & Th_i \in \mathbb{N}_0 \\ & \begin{cases} Th_1 \leq Max_1 \\ \dots \\ Th_5 \leq Max_5 \end{cases} \end{aligned} \quad (4)$$

The solution of the problem (4) is a point in an \mathbb{N}^5 subset. This subset is defined by the threshold constraints. This problem belongs to the class of nonlinear optimization problems with integer constraints. Unfortunately we can make no assumption on the function behavior because it depends only indirectly on the threshold value. We compared the well-known simplex method proposed by Nelder and Mead [8] and an algorithm that we designed to solve this problem that is described in the following.

3.2. Iterative decreasing step method

Since the thresholds of the table location algorithm have discrete values, we could use a very simple algorithm for optimal threshold selection that is based on an exhaustive measure of TLI for each combination of the five thresholds. Obviously, this approach is unfeasible from a computational point of view, and we tried to reduce the computational burden by exploring the space of threshold values with decreasing resolution iterations.

We describe in the following the proposed algorithm, that is also sketched in Algorithm 1. At the beginning we set all the thresholds to their minimum value. In each iteration we analyze the values of TLI that are obtained when varying one threshold at a time, whereas the other thresholds are fixed. At the end of the analysis of one threshold,

we keep its value that provide the maximum of TLI , and we process the next threshold. When all the thresholds have been analyzed one iteration is terminated and we start another iteration by analyzing again the first threshold. In the first iteration the steps considered during threshold variation are quite large, and all the range of allowed values for the threshold is explored. At every iteration we reduce the step and the search interval as well. The exploration interval is centered in the threshold value corresponding to the maximum in the previous iteration. In this way an higher resolution is considered only for small intervals, and this trick allows us to reduce the computational burden of the approach. The process is stopped when there is no increase in TLI value from one iteration and the previous one.

Figure 3 shows an example of the behavior of TLI when varying one threshold. Each line corresponds to one iteration, and we can observe that smaller intervals are considered at each iteration.

Algorithm 1 IterativeDecreasingStep(I, G)

```

Input:
   $I$ : image set;  $G$ : ground truth.
begin
   $step_i \leftarrow step_i^0, Th_i \leftarrow 0 \quad i = 1, \dots, 5$ 
   $start_i \leftarrow 0, stop_i \leftarrow max_i \quad i = 1, \dots, 5$ 
   $MaxValue \leftarrow 0$ 
  repeat
    for  $i = 1$  to 5
      begin
         $OldMaxValue \leftarrow MaxValue$ 
        for  $Th_i = start_i$  to  $stop_i$  step  $step_i$ 
           $Value \leftarrow TLI(TabFind(I, Th_1, \dots, Th_5), G)$ 
          if  $Value > MaxValue$  then
             $MaxValue \leftarrow Value$ 
             $ThMax_i \leftarrow Th_i$ 
          end
        end
         $Th_i \leftarrow ThMax_i$ 
      end
       $step_i \leftarrow step_i/2 \quad i = 1, \dots, 5$ 
       $start_i \leftarrow Th_i - step_i \quad i = 1, \dots, 5$ 
       $stop_i \leftarrow Th_i + step_i \quad i = 1, \dots, 5$ 
    until ( $MaxValue \leq OldMaxValue$  or  $step_i > 1$ )
  end

```

3.3. Nelder and Mead method

The original Nelder and Mead method is defined to solve an optimization problem without constraints. Our optimization problem is not a member of this class, and we adapted this method to our aims. When the method finds a value external to the N^5 subset, we replace the threshold not allowed with the nearest threshold member of the subset. To select the nearest value we use the Euclidean distance. A Nelder and Mead method application used to find optimal parameters for segmentation algorithms is described in [7].

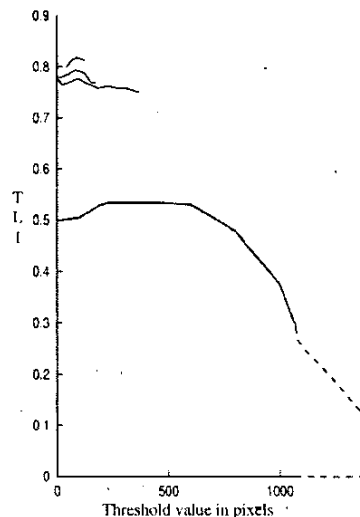


Figure 3. Iterative decreasing step method. Each line report the TLI values obtained when varying the threshold values at each iteration.

4. Experimental results

The proposed table location method (herein after called Tabfinder) has been evaluated by taking into account two data-sets containing 1551 pages, and comparing its performance with the tables identified by two commercial OCRs.

The documents we are dealing with in the experiments are technical papers. These documents often contain tabular structures and other items, like charts and graphics, that can be erroneously classified as tables. In order to allow easy comparison of the experiments we used data that are widely available to the document processing community. The first set of documents is made by all the papers published in *IEEE Transaction on Pattern Analysis and Machine Intelligence* in the first six months of 1999 (we will refer to this as TPAMI data-set). The second data-set contains the black and white images from *University of Washington - English Document Image Database I* (this data set is referred to as UWASH).

Images in TPAMI are gray-scale and are acquired with 300 dpi resolution. This set includes 575 images, 82 images have tables and 114 tables are contained. This data-set is particularly clean, and each image has no skew. The TPAMI set contains both Long Papers and Short Papers, that clearly present different layouts.

The UWASH data-set includes 976 binary images (with

	TPAMI				UWASH			
	Tabfinder iterative	Tabfinder Nelder	Fine reader	Caere	Tabfinder iterative	Tabfinder Nelder	Fine Reader	Caere
Images	425	425	425	425	518	518	518	518
Tables	70	70	70	70	58	58	58	58
LOCATED	58	56	51	50	22	22	13	14
SPLIT	5	7	10	8	21	17	9	10
MERGED	0	0	1	2	1	1	10	9
MISSED	7	7	8	10	14	22	26	25
FALSE	16	16	52	86	19	18	29	86
<i>TLI</i>	0.709	0.705	0.508	0.414	0.481	0.403	0.277	0.183

Table 1. Tabfinder compared with OCR results.

resolution of 300 dpi), 103 images have tables and 125 tables are contained. This set contains technical papers which represent several layouts from different publishers. Analyzing this set, it is possible to find papers with very different layout and very noisy. Table location on this set is an harder task than on TPAMI set.

	Training images	Test images
TPAMI	150	425
UWASH	458	518

Table 2. Number of document images in the two data-sets.

Both TPAMI and UWASH are split in training set and test set (Table 2) to perform the optimization process and to evaluate the optimization result. Many OCR engines can identify tables during layout analysis. We chosen two of the most effective OCRs (Abby Fine Reader 4.0 and Scansoft Caere Development Kit 2000) and located tables on both test sets using their engines. Then, we performed four optimization processes by using the two optimization methods with both training sets. For each data-set we obtained two threshold sets. At the end we used the two optimal thresholds sets on test sets and compared Tabfinder and OCR results.

By analyzing the results shown in Table 1 we can observe that Tabfinder locates more tables and predict less false tables than the two OCRs. Moreover, a small difference between the optimization methods is shown. Iterative optimization process and Nelder method fix two different threshold sets. Each set is close to the other, but this difference has influences to the test set results. In this particular case, iterative decreasing step method is better than the other. However, it must be mentioned that the optimization time required by the Nelder method is only 15 % of the time required by the iterative method.

5. Conclusions

We described a table location algorithm that can be adapted to different tasks by learning optimal thresholds. This training is made by maximizing the recognition on a training set of document images.

There are three main contributions of the paper: the table location algorithm, the introduction of an index for table location evaluation, and an optimization algorithm that is used in order to select optimal threshold values. The proposed system has been compared with table location performance of two commercial OCRs on two different tasks: clean pages of one journal, and noisy images of several journals. In both cases the proposed algorithm clearly outperforms results of OCRs.

References

- [1] F. Cesarini, M. Gori, S. Marinai, and G. Soda. Structured document segmentation and representation by the modified X-Y tree. In *Proc. ICDAR'99*, pages 266 – 274, 1999.
- [2] S. Chandran and R. Kasturi. Structural recognition of tabulated data. In *Proc. ICDAR'93*, pages 516–519, 1993.
- [3] J. Hu, R. Kashi, D. Lopresti, and G. Wilfong. A system for understanding and reformulating tables. In *Proc. of the SPIE - Document Recognition*, 1998.
- [4] A. Jain and B. Yu. Document representation and its application to page decomposition. *IEEE Trans. PAMI*, 20(3):294–309, March 1998.
- [5] T. Kieninger. Table structure recognition based on robust block segmentation. In *Proc. of the SPIE - Document Recognition*, 1998.
- [6] K. Lee and Y. Choy. Geometric structure analysis of document images: A knowledge-based approach. *IEEE Trans. PAMI*, 22(11):1224–1240, November 2000.
- [7] S. Mao and T. Kanungo. Empirical performance evaluation methodology and its application to page segmentation algorithms. *IEEE Trans. PAMI*, 23(3):242–256, March 2001.
- [8] J. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, July 1965.