

Table Extraction from Document Images using Fixed Point Model

Anukriti Bansal^{*}
IIT Delhi
anukriti1107@gmail.com

Gaurav Harit
IIT Jodhpur
gharit@iitj.ac.in

Sumantra Dutta Roy
IIT Delhi
sumantra@ee.iitd.ac.in

ABSTRACT

The paper presents a novel learning-based framework to identify tables from scanned document images. The approach is designed as a structured labeling problem, which learns the layout of the document and labels its various entities as table header, table trailer, table cell and non-table region. We develop features which encode the foreground block characteristics and the contextual information. These features are provided to a fixed point model which learns the inter-relationship between the blocks. The fixed point model attains a contraction mapping and provides a unique label to each block. We compare the results with Condition Random Fields(CRFs). Unlike CRFs, the fixed point model captures the context information in terms of the neighbourhood layout more efficiently. Experiments on the images picked from UW-III (University of Washington) dataset, UNLV dataset and our dataset consisting of document images with multi-column page layout, show the applicability of our algorithm in layout analysis and table detection.

Keywords

Table recognition, Fixed Point Model, Structured labeling, Conditional Random Fields, Layout analysis

1. INTRODUCTION

Tables present in documents are often used to compactly communicate important information in rows and columns. To automatically extract this information by digitization of paper documents, the tabular structures need to be identified and the layout and inter-relationship between the table elements need to be preserved for subsequent analysis. The problem of table detection is challenging due to a wide range of layouts and random positioning of table elements. Algorithms for table detection have been proposed by authors in the past, but the problem of correctly localizing the tabular structure from a wide variety of documents, remains a

challenging task.

In this work we learn the layout of a document image by extracting the attributes of foreground and background regions and modeling the correlations between them. Using these attributes, a fixed point model captures the context and learns the inter-relationships between different foreground and background document entities to assign them a unique label which can be, table header, table trailer, table cell and non-table region. Regions which get table related labels are clustered together to extract a table.

The Fixed Point Model as proposed by Li *et al* [18] has been used for the task of structured labeling by capturing the correlation between the observed data. The structured input is denoted as a graph with nodes and edges. The objective of structured labeling task is to jointly assign the labels to all the nodes of a graph. In computer vision, the structured input comprises the set of inputs of all the pixels and the structured output constitutes the set of labels assigned to those pixels. Edges between the nodes are used to model the correlations among the nodes. The Fixed point model captures the neighborhood information and models the correlation between the different nodes to predict the label of each node. Markov random fields (MRF) [10] and conditional random fields (CRF) [17] are also used to model the inter-relationships of structural labels. However, due to heavy computational burden in the training and testing stages, MRF and CRF are often modeled to capture a few neighborhood interactions, limiting their modeling capabilities. The motivation to use fixed point model for the problem of table detection arises from the need to model the spatial inter-dependencies of different elements of a document image. The fixed point model utilizes the context information and attains a contraction mapping to assign a unique label to each element of document image. The final labeling helps extract the table regions. This can facilitate applications such as searching, indexing and information retrieval. A subset of the authors have previously used a fixed point model for article extraction [1].

1.1 Related Work

Several interesting survey papers [11] [7] [23] [20] [28] [36] [27] have been published on table structure analysis and layout analysis related work in the last two decades. Layout analysis is a major step in identifying any physical or logical document entity. In this section, we review the literature related to the use of machine learning-based methods for layout analysis, specifically for extracting tables. Table extraction has been attempted on scanned images [13] [32]

^{*}Corresponding author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
ICVGIP '14, December 14-18 2014, Bangalore, India
Copyright 2014 ACM 978-1-4503-3061-9/14/12...\$15.00
<http://dx.doi.org/10.1145/2683483.2683550>

[31] [34] [3] [26] [21] [14] [29], OCR'd images [25] [30] and electronic text documents [6] [19] [16]. Various approaches for table detection and layout analysis can be categorized as machine-learning based [33] [24] [15] [35] [6] [9] [5] [14], rule-based [12] [21] and model/template based [25] [30].

Fang *et al.* [9] proposed a method to detect table headers from an already extracted table. They designed a set of features which differentiate table header and table rows. Based on these features similarity between consecutive table rows is calculated to classify them as table header or table row. Wang *et al.* [31, 32, 34] have used binary decision tree to develop a top-down approach for localizing tables. Document zones delimited by blank areas are classified as table candidates. The table candidate zones are then segmented and the probability of the table being consistent is estimated, based on attributes such as, percentage of area occupied by non-blank parts of the table, its width, and the justification of cells within their columns. The table boundary is then refined by considering the improvement in probability if the table's upper and lower neighboring zones were included with it. Liu [19] has used the sparse line property of table rows to identify tables in PDF documents. The table border is validated by combining the sparse lines with the table keyword. The extracted table is represented with a unique table metadata file, and is made indexable so that the end-users can search for tables. Their system, named TableSeer, extracts tables from online documents, such as PDF, HTML, Word, Powerpoint, etc. It can be used for document images only if the OCR'd text is available. However, in the case of old degraded documents, OCR results may be wrong and may not identify table caption related keywords.

Ramel *et al.* [25] proposed a cognitive approach which hypothesizes that readers detect the presence of tables by perceiving different types of regularities. They analyze rectangular grid lines and assess regularity of text blocks, e.g. non-overlapping rectangles, alignment, and relative proximity of blocks.

Costa e Silva [6] used Hidden Markov Models (HMMs) for extracting table regions from PDF documents. Text is extracted from PDF using pdftotext Linux utility. Features are computed using white space between the extracted text. Cesari *et al.* [3] analyzed modified X-Y trees to identify sub-images that correspond to tables. Regions surrounded by lines are considered as candidates and then parallel lines and white spaces are searched. Sub-tables that satisfy a set of geometric properties are searched and merged to form tables. Mandal *et al.* [21] have identified tables on the basis of substantially larger gaps between columns and rows. Their approach works even in the absence of any horizontal or vertical rulings. However, they have not addressed the issue of tables present in documents with multi-column page layout.

Shaifat *et al.* [26] and Smith [29] have used tab-stops for doing the layout-analysis of document images and then used the alignment information (left, right or center) of columns for table detection. Their algorithm works fine for images containing both table and text but gives erroneous results when full-page tables are present. Tersteegen and Wenzel [30] have presented a system ScanTab for table recognition in scanned and OCR'd German business letters. The system identifies a table header by comparing it with the table headers of known reference tables. The table header detection tries to locate the table header by relating the positions

of the keywords (obtained from OCR) with a reference table header. Horizontal lines, if present, verifies the boundary of the table header. Hu *et al.* [13] have presented a table recognition system for single column documents (images or text) on the basis of a merit function. Merit functions analyze a group of lines and identify it as a table if it maximizes the merit value above a certain limit. These functions consider the resemblance among the lines by considering a weighted average of the number of white spaces with the same position and the number of overlapping word bounding boxes with similar sizes.

Kieninger *et al.* [16] proposed a bottom-up approach of table analysis. They developed a system called T-Recs which works for ASCII file. The system takes word bounding box information as input. These word boxes are clustered with a bottom-up approach into distinct regions which are designated as tables if they satisfy certain heuristic rules.

Harit and Bansal [12] detected tables in document images by searching for table-header and trailer patterns. The approach is applicable to multi-column page layouts, but it assumes that the header and trailer patterns do not vary within a single document. So, if a document has tables with varying header and trailer patterns, then the different types of tables present on a document image may not be identified properly. Chen *et al.* [5] proposed a learning-based framework for table detection in Handwritten documents. Handwritten text regions are first identified using SVM and then tables are identified on the basis of correlation score of neighboring text lines. Kasar *et al.* [14] proposed a method to detect tables from document images using the properties of intersecting row and column line separators. SVM classifier is used to classify line separators if they belong to a table or not. The applicability of this method is restricted to documents which have line separators. Pinto *et al.* [24] have used CRF to extract tables from HTML pages. They provide labels to table elements such as line heading, sub-heading, caption, etc.

1.2 Contributions of this work

From the past work, we see that the approaches which work on recognizing tables from scanned document images are mostly rule-based. For each new type of layout, new rules need to be designed and implemented. Furthermore, there are so many different and varied ways in which tables can be created. Thus, rule based methods fail in terms of scalability and robustness. Context based machine learning methods learn the layout of a document more efficiently and are capable of recognizing unseen patterns and layouts. Besides this, these methods are comparatively scalable and robust. Machine learning based approaches which are reported in literature mostly work on OCR'd scanned document images, PDFs or HTML pages.

In the present work, we develop a set of foreground and background features to be used for training a fixed point model, which learns the layout and contextual features for identifying tables, and assigns labels to different document elements. The foreground features are the features extracted from the text blocks and the background features describe attributes related to white spaces and ruling lines. We use a context-based model to extract tables from the scanned document images. It learns the contextual layout of a variety of tabular structures without using any user-defined heuristic rules. The design of appearance and contextual features for

training the fixed point model to learn the layout for identifying the tabular structures is the key contribution of this paper.

Our work motivates from [24] in the sense that we also use context dependent learning based framework to recognize tables and assign labels to different document elements. But our input is a scanned image, not web pages. Therefore, instead of using the text-content related features, we exploit the layout of document entities and context information to recognize tables from document images. Our approach is able to extract tables present in documents with non-aligned multi-column page layout. A comparison of our results with that obtained using Conditional Random Fields (CRF) which shows that the fixed point model is more efficient in capturing the contextual relationships between the neighboring blocks.

1.3 Overview of the System

The input to the system is a scanned document image having single or multi-column layout. As a pre-processing step, we binarize the image and create blocks of foreground and background regions using morphological image processing techniques [2]. Our approach can be divided into two stages. In the first stage, we extract the features for each block by combining its attributes (appearance features) and the correlation with neighboring blocks (contextual features). Fixed point model uses these features and labels the foreground text blocks as table header, table trailer, table cell and non-table region. In the final stage, these labeled blocks are clustered together to identify the set of blocks which belong to the same table.

The organization of the paper is as follows. In Section 2 we describe the mathematical model for table recognition using fixed point model. Section 3 discusses our approach in detail. The results of the proposed approach are discussed in Section 4. Finally we conclude the paper in Section 5.

2. MODEL DESCRIPTION

Li *et al.* [18] has proposed the fixed point model for the problem of structured labeling. They introduced a contextual prediction function for each node which takes its features and labeling of neighboring nodes as input and gives the labeling of all the individual nodes as output. The fixed point function is the vector form of the contextual prediction function of all the nodes and it is trained with the property of contraction mapping so that a unique label is obtained during the iterative prediction process.

The input to our system is the set of blocks obtained after morphological image processing operations to a document page, as illustrated in Section 3.1. This structured input can be denoted as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each node $v_i \in \mathcal{V}$ corresponds to a block in the image with its features denoted as \mathbf{x}_i . Our approach for table recognition is a structured labeling approach, in which our objective is to jointly assign the labels $\mathbf{y} = (y_i : i = 1 \dots |\mathcal{V}|)$ ($y_i \in \mathcal{L}$, where \mathcal{L} is the label space) to all blocks $\mathcal{V} = (v_i : i = 1 \dots |\mathcal{V}|)$ as a joint output. The edges \mathcal{E} decide the neighborhood of each block. We represent an image as a 2D graph. The neighborhood \mathcal{N}_i of block v_i is specified by m blocks in each of the 4 directions: top, bottom, left and right, i.e., $\mathcal{N}_i = \{v_{(x-m/2, y)}, v_{(x-m/2+1, y)}, \dots, v_{(x-1, y)}, v_{(x+1, y)}, \dots, v_{(x+m/2, y)}, v_{(x, y-m/2)}, \dots, v_{(x, y-1)}, v_{(x, y+1)}, \dots, v_{(x, y+m/2)}\}$. Here, m denotes the number of neighbors a block can have. The la-

bel space for our model is $\mathcal{L} = \{table-header, table-trailer, table-cell, non-table\}$. Note that these labels correspond to the foreground text content.

Contextual prediction function f is learned to provide labels for the nodes. The function f takes in both v_i 's appearance features \mathbf{x}_i and contextual features $\mathbf{q}_{\mathcal{N}_i}$. Appearance features associate each block to a label using the characteristics of that block. Contextual features, on the other hand, associate a block to a label using the label of neighboring blocks. The contextual prediction function is described as (following the scheme proposed in [18])

$$q_i = f(\mathbf{x}_i, \mathbf{q}_{\mathcal{N}_i}; \theta) \quad (1)$$

where f is a classification function with parameter θ . Using Equation 1, the labeling \mathbf{q} of all the blocks in a vector form is given by,

$$\mathbf{q} = \mathbf{f}(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n, \mathbf{q}; \theta), \quad (2)$$

where $\mathbf{q} = [q_1, q_2, \dots, q_n]^T$ $\mathbf{f}(\cdot) = [f(\mathbf{x}_1, \mathbf{q}_{\mathcal{N}_1}; \theta), f(\mathbf{x}_2, \mathbf{q}_{\mathcal{N}_2}; \theta), \dots, f(\mathbf{x}_n, \mathbf{q}_{\mathcal{N}_n}; \theta)]^T$

From Equation 2, we see that the vector \mathbf{q} appears both as the output and as part of the input. The parameter set θ is learned using labeling \mathbf{q} and the features $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ of the training set.

There are two approaches to obtain the labeling of structured input \mathcal{G} . In the first approach, one has to solve a set of non-linear equations $\mathbf{q} = \mathbf{f}(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n, \mathbf{q}; \theta)$. This approach is difficult, so, we look for a function \mathbf{f} that satisfies the property of contraction mapping, i.e., having a stable status for each structured input. When ground-truth labeling is used during the training process, that label configuration considered to be the stable status. This stable status leads to the fixed-point iterations in the prediction process:

$$\mathbf{q}^t = \mathbf{f}(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n, \mathbf{q}^{t-1}; \theta) \text{ and } \mathbf{q}^t \rightarrow \mathbf{q} \text{ as } t \rightarrow \infty.$$

The function \mathbf{f} with such property is termed as a fixed point function.

We have performed experiments using two different contextual prediction functions: SVM (Support Vector Machines) and KLR (Kernel Logistic Regression). We train the function using the appearance features \mathbf{x}_i and contextual features $\mathbf{q}_{\mathcal{N}_i}$ of each block. In the testing phase, we apply the learned contraction mapping iteratively to the new structured inputs. The iterations begin with a randomly initialized label q_i for each block $v_i \in \mathcal{V}$. Note that q_i is not sensitive to the choice of initialization and so we have initialized it with zero in our experiments.

3. METHODOLOGY

3.1 Extraction of Blocks

The input image is converted into gray scale and is binarized. Horizontal and vertical ruling lines are then removed by replacing them with the background pixels. Next we segment the binarized image into text and graphics regions. We use the leptonica library¹ designed by Bloomberg [2] for segmenting the document image into homogeneous regions of text and graphics. A morphological closing operation is performed on the text regions with a line structuring element of length equal to the maximum inter-word gap. This

¹<http://www.leptonica.com/>

results in the formation of text blobs [21], i.e., the characters and words in a text line get coalesced to form a single rectangular block. Since normally the column width in a table is more than the inter-word spacing, the text blobs do not get merged across the columns. This is the reason for getting multiple text blocks in the same row for a table. The structure we obtain after morphological closing can be considered as a graph with each block as its node. Now we compute the features as described in next section.

3.2 Feature Set

For the context-based table extraction we compute two types of features: appearance features and contextual features.

3.2.1 Appearance Features

Appearance features represent the characteristics of a foreground text block. The following features are computed to create the appearance feature vector:

1. Normalized block height.
2. Normalized block width.
3. Average Character Height: Taken as the median of the histogram of the heights of all connected components.
4. Background color: Colored or white. Here, we assume that foreground color is always darker than the background.
5. Foreground/font color: Colored or black.
6. Font Phase: Boldness of a word blob [4].

3.2.2 Contextual Features

The contextual features capture the neighborhood information and help in better prediction of block labels. Consider an example of a table present in a single column document. Trailer blocks and table cells have similar appearance features but the trailer block will be followed by a line with larger white space, which in turn is followed by a single text block. Thus, the context in the form of the thicker white space below, helps in identifying the trailer block. In other words, the labels of neighboring blocks help in correctly predicting the label of a given block. We formulate the contextual feature vector, of a given block, to be composed of the set of labels of the neighboring blocks along four directions: top, bottom, left and right. All the document entities, foreground and background, contribute in designing the contextual feature vector. We formulate a label space for document entities and represent it as $\mathcal{L}_D = \{\text{Table-header, Table-Trailer, Table-cell, Non-table, horizontal white space, ruling lines}\}$. Note that this label space also includes the background entities – the horizontal white space and the ruling lines.

3.2.3 Identifying White Space Separators

The white space information is analyzed to identify contiguous runs of white pixels along both the horizontal and vertical directions. For each white pixel p_{ij} its distance dp_{ij}^+ to the closest black pixel towards the right side and its distance dp_{ij}^- to the closest black pixel towards the left side along the same horizontal scan line is computed. The pixel is then labeled with the distance $[dp_{ij}^+ + dp_{ij}^-]$ measured in terms of number of pixels constituting the white space run. We form a *horizontal run map matrix* which comprises the horizontal run values for all the pixels. Each horizontal run is a horizontal separator. The horizontal run map is refined

Table 1: Distribution of the number of points and frames						
	2 Groups			3 Groups		
	# Sec. Points Frames			# Sec. Points Frames		
Check.	78	291	28	26	437	28
Traffic	31	241	30	7	332	31
Articul.	11	155	40	2	122	31
All	120	266	30	35	398	29
Point Distr.	35%-65%			20%-24%-56%		

Figure 1: Neighborhood Estimation: Current block marked in blue, neighbors at $m = 1$ marked in red, neighbors at $m = 2$ marked in red and green

in two pruning steps:

1. A separator with a run value higher than the run value of its immediate neighboring separators on both sides is retained and all others are ignored.
2. If two adjacent horizontal separators have a significant overlap and if no text blocks are present between them, then the smaller one is pruned out.

The prominent horizontal separators are considered as rectangular blocks. White space blocks are characterized as thick or thin by using the method given in [21].

3.2.4 Identifying Horizontal and Vertical Ruling Lines

To identify the ruling lines, colored images are converted into grayscale images and binarized. Horizontal and vertical runs of black pixels are identified. A run of black pixels is considered as a part of a ruling line if the number of contiguous black pixels in the run is greater than or equal to the average character height. In poor quality document images, binarization may result in broken lines. A dilation step is used to join the gap and get complete ruling line. Ruling lines are also considered as a rectangular block which also includes the surrounding white space.

3.3 Neighborhood Estimation

The neighborhood of the i^{th} block is defined by all the adjacent blocks to its right, left, top and bottom. It represents inter-relationship between different blocks and plays a significant role in determining the correct label. As seen in the first image in Figure 3, the header region is represented by a green colored block, followed by a ruling, followed by multiple blocks in green color, followed by ruling lines. The header boundary is defined by black colored text blocks in the next row. Therefore, to correctly predict the labels of document elements, considering the immediate neighborhood is not sufficient. We need to extend the neighborhood to a suitable number so that rich context information can be captured. We use parameter m to specify the span of the neighborhood. This can be understood using Figure 1. The text block highlighted in blue represent the i^{th} block whose neighborhood has to be estimated. For $m = 1$, neighborhood is defined by the labels of all the blocks highlighted in red. Let us denote this set of blocks by ℓ_1 . Note that the white space separators are blocks of background pixels. For $m = 2$, the neighborhood consists of all the elements of ℓ_1 and the labels of the blocks highlighted in green, i.e., neighbors of red blocks. Similarly, we keep on extending the neighborhood as specified by parameter m . In our experiments, we

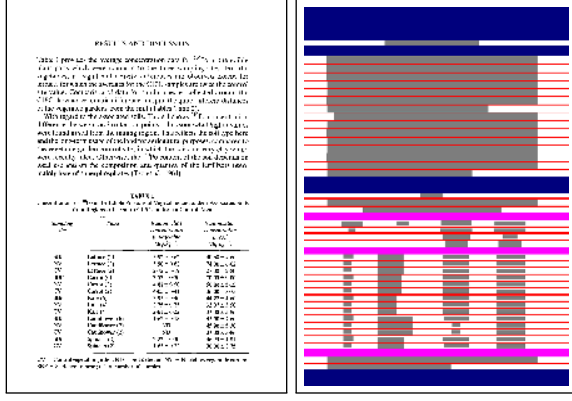


Figure 2: The set of blocks forming input to the block labeling process: text blocks (gray), thick whitespace without ruling (blue), thin whitespace without ruling (red), thick whitespace with ruling (pink)

empirically choose $m = 13$.

3.4 Block Labeling

In the block labeling stage all the foreground blocks (i.e. the blocks containing text) of a document image are assigned one of the 4 labels: table-header, table-cell, table-trailer and non-table. The labels of the background blocks (white space, rulings) are not updated. The set of all the segmented blocks (foreground and background blocks) of an image (Figure 2) constitute the structured input \mathcal{G} . For each block v_i , the features mentioned in Section 3.2.1 form the appearance feature vector \mathbf{x}_i . A normalized histogram of class labels of neighboring blocks is used as contextual feature \mathbf{q}_{N_i} . The number of bins in a histogram is equal to the number of document entities. The frequency of occurrence of a particular document entity within the neighborhood of a block is assigned to the corresponding bin. For each block v_i , this creates a $4 \times m \times \mathcal{L}_D$ dimensional contextual feature vector. Here, \mathcal{L}_D is the label space of document entities (see Section 3.2.2), m is the span of context to be captured and 4 specifies the neighborhood in the four directions (top, bottom, left and right).

In the training phase, we learn a contextual prediction function using both appearance features \mathbf{x}_i and contextual features \mathbf{q}_{N_i} . We use Kernel Logistic Regression (KLR) with RBF kernel as the contextual prediction function. We have also experimented with L1 regularized Support Vector Machine (SVM-L1), provided in the Liblinear software package [8] as the classifier.

During testing, at each iteration, contraction mapping is applied to the new structured inputs. When convergence is achieved, the block labels do not change in subsequent iterations. We begin the iterations by initializing the labels q_i for all the blocks to be 0. The steps in the testing phase of block labeling are summarized in Algorithm 1.

The blocks which get labelled with table components (table-header, table-cell, table-trailer) are clustered (block grouping) together to extract the tables. The header and trailer blocks provide the vertical bound of the table. The left and the right boundary of the table is established by looking for transitions in block labels from any of the table components to a non-table region.

Algorithm 1 The prediction phase of the block labeling

Input The text blocks of new block image $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; labels of background blocks (white spaces); the trained contextual prediction function f ; the number of iterations \mathcal{T} ; a threshold ϵ

Output The labeling \mathbf{q} of \mathcal{G}

Procedure

1. Initialize $t = 1$; for each $v_i \in \mathcal{V}$;
2. **Repeat**
 - (a) For each node v_i , compute the labeling q_i^t :
 $q_i^t = f(\mathbf{x}_i, \mathbf{q}_{N_i}^{t-1}; \theta)$;
 - (b) $t = t + 1$;
until $t \geq \mathcal{T}$ or $\mathbf{q}^t - \mathbf{q}^{t-1} \leq \epsilon$.
 $\mathbf{q} = [q_1^t, q_2^t, \dots, q_m^t]^T$

4. RESULTS AND DISCUSSION

Table 1: Experimental results of block labeling with four classes

Block Label	# blocks	Accuracy using SVM (in %)	Accuracy using KLR (in %)
Table-Header	1036	89.5	96.2
Table-Trailer	761	78.7	89.4
Table-cell	6053	91.3	95.7
Non-Table	5593	96.1	98.3
Total Blocks	13443	91.5	96.4

Table 2: Comparison of block labeling with CRF

# Blocks	Accuracy using SVM (in %)	Accuracy using KLR (in %)	Accuracy using CRF (in %)
13443	91.5	96.4	85.2

The proposed method for table detection is tested on a dataset of 50 images which were picked from UW-III dataset, UNLV dataset and our own dataset consisting of documents with multi-column page layout. The size of images varies from 2592 x 3300 to 1275 x 1650 pixels. Both single column and multi-column layout documents are included. Out of the 50 images, 44 images contain tables. For the purpose of block labeling, we manually created the ground truth at the block level (Figure 4(b)). From 50 images, we got 19334 blocks of both foreground and background pixels. Out of the 19334 blocks, 13443 are text blocks while the rest are non text and white space blocks. We do a 10-fold cross validation for evaluating the results of the block labeling method. Experimental results of our algorithm are summarized in Table 1.

We have achieved an overall accuracy of around 96% for assigning a label to each document region. Accuracy for each class is shown in Table 1. It can be seen that the accuracy for the table-trailer blocks is less in comparison to other blocks. We notice that when the table trailer comprises multiple rows (Figure 4), the blocks in the last row

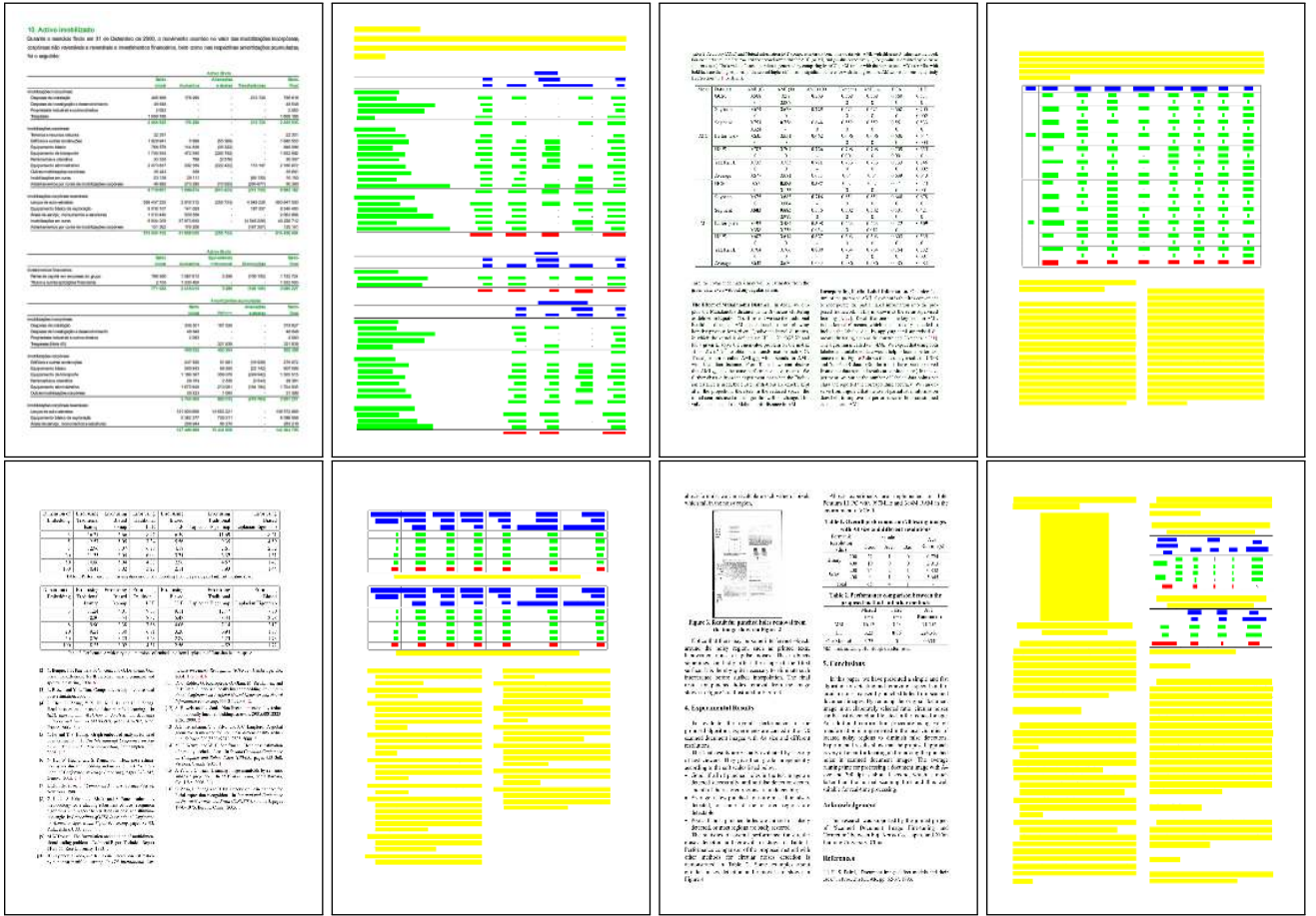


Figure 3: Results of labeling each document element as table-heading (blue), table-trailer (red), table-cell (green), non-table regions (yellow).

get labeled as table trailer and others get misclassified as table-cells. This happened possibly because our dataset has few tables with multiple rows in the table trailer. However, this will not have much effect on accurate localization of tables using header and trailer regions. Table-cells and non-table blocks are labeled with high accuracy due to their strong appearance and contextual features.

We compared our results with CRF for labeling document entities and extracting tables. For CRF, we used the UGM toolbox [22] and used loopy belief propagation for the inference. Table 2 shows the quantitative comparison of the performance of both the classifiers on our dataset. In Figure 5 we analyze the labeling results of CRF and the fixed point model on a document page from our dataset. CRF was not able to distinguish between table header, table trailer and table cell. The possible reason could be that since CRF is a probability-based classifier, it learns the context for table rows (which are more in number) better than header or trailer.

Figure 3 shows the result of our approach on few images picked from the test set. Single page tables, tables spanning multiple columns and tables confined to single columns are labeled correctly. Different feature combinations have shown effectiveness in case of different layouts. For example, in Figure 3(a) the font color of text region and the thickness of horizontal separators around text provide a strong clue

for the detection of table header and trailer regions. In the first table, a few rows have the same pattern as that of table trailer, but the thickness of the horizontal separator helps in determining the correct labels. Similarly, in Figure 3(e), presence of the ruling line between table rows differentiates the table-header blocks from the rest of the table and provides correct labeling to multi-row header blocks. Our approach gives correct labeling for tables present in different page layouts. Thus it is evident that the proposed method learns the layout with effectively and can be used on images with complex layouts.

5. CONCLUSION

This paper presents a novel learning-based framework for the problem of layout analysis and table detection in document images with complex layouts. The method contributes an efficient way of analyzing and labeling different document elements which in turn defines the table boundary. A combination of foreground and background features is extracted and used with the fixed model for learning the context information. This helps in learning the document layout and labeling the different regions. The experimental results are promising and our approach works well on a heterogeneous collection of documents. Unlike other existing techniques, the method is general and does not rely on heuristic rules

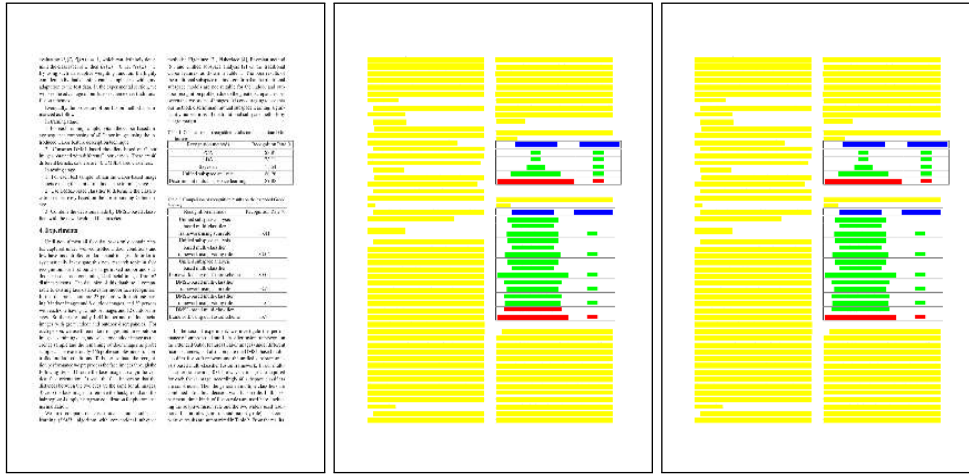


Figure 4: Results with incorrect labeling of table trailer blocks: (a) Original Image (b) Ground truth image with correct labeling, (c) Results of our approach which misclassified table-trailer blocks as table cell.

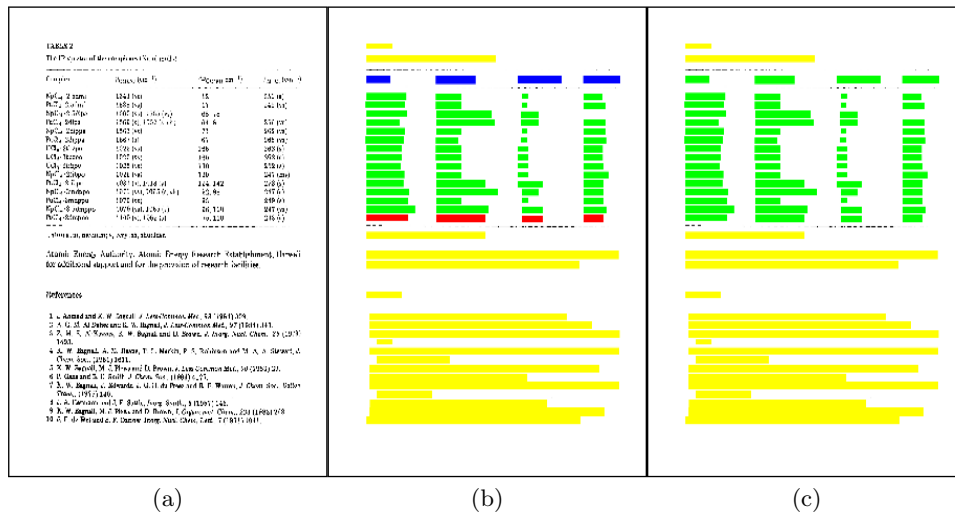


Figure 5: Comparison of Fixed point model and CRF: (a) Original Image (b) Labeling results of Fixed point model, (c) Labeling results of CRF

such as the presence of horizontal and vertical ruling lines and their intersection. It provides an alternate to most of the present rule-based and learning-based systems. While the method correctly locates the tables one above another, it merges the tables present side by side. Future work includes a comprehensive analysis of the applicability of the proposed approach on other types of images and using additional clues to handles tables present side by side. We shall also explore other context-handling approaches like Recurrent Neural Networks (RNNs) for this task.

Acknowledgements

We are grateful to Prof. Santanu Chaudhury and Prof. J. B. Srivastava for introducing us to the fixed point model, and its applications.

6. REFERENCES

- [1] A. Bansal, S. Chaudhury, S. Dutta Roy, and J. B.

Srivastava. Newspaper article extraction using hierarchical fixed point model. In *Proc. IAPR International Workshop on Document Analysis Systems, DAS'14*, pages 257 – 261, 2014.

- [2] D. S. Bloomberg. Multiresolution morphological approach to document image analysis. In *Proceedings of the 1991 International Conference on Document Analysis and Recognition, ICDAR '91*, 1991.
- [3] F. Cesari, S. Marinai, L. Sarti, and G. Soda. Trainable table location in document images. In *Proceedings of the 16th International Conference on Pattern Recognition (ICPR'02) Volume 3*, 2002.
- [4] B. B. Chaudhuri and U. Garain. Automatic detection of italic, bold and all-capital words in document images. In *Proceedings of the 14th International Conference on Pattern Recognition-Volume 1 - Volume 1*, ICPR '98, 1998.
- [5] J. Chen and D. Lopresti. Table detection in noisy

- off-line handwritten documents. In *Document Analysis and Recognition (ICDAR)*, 2011 *International Conference on*, pages 399–403, 2011.
- [6] A. C. e Silva. Learning rich hidden markov models in document analysis: Table location. In *ICDAR*, pages 843–847. IEEE Computer Society, 2009.
- [7] D. W. Embley, M. Hurst, D. P. Lopresti, and G. Nagy. Table-processing paradigms: a research survey. *IJDAR*, 8(2):66–86, 2006.
- [8] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June 2008.
- [9] J. Fang, P. Mitra, Z. Tang, and C. L. Giles. Table header detection and classification. In *AAAI*, 2012.
- [10] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6:721–741, 1984.
- [11] J. C. Handley. *Electronic Imaging Technology*. Document Recognition, SPIE, 1999.
- [12] G. Harit and A. Bansal. Table detection in document images using header and trailer patterns. In *Proceedings of the Eighth Indian Conference on Computer Vision, Graphics and Image Processing, ICVGIP '12*, pages 62:1–62:8, 2012.
- [13] J. Hu, R. S. Kashi, D. P. Lopresti, and G. Wilfong. Medium-independent table detection. In *Proc. SPIE*, volume 3967, pages 291–302, 1999.
- [14] T. Kasar, P. Barlas, S. Adam, C. Chatelain, and T. Paquet. Learning to detect tables in scanned document images using line information. In *ICDAR*, pages 1185–1189, 2013.
- [15] D. Keysers, F. Shafait, and T. M. Breuel. Document image zone classification - a simple high-performance approach. In *2nd Int. Conf. on Computer Vision Theory and Applications*, pages 44–51, 2007.
- [16] T. Kieninger. Table structure recognition based on robust block segmentation. In *Document Recognition V SPIE*, pages 22–32, 1998.
- [17] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning, ICML '01*, pages 282–289, 2001.
- [18] Q. Li, J. Wang, Z. Tu, and D. P. Wipf. Fixed-point model for structured labeling. In *Proc. of the 30th International Conference on Machine Learning (ICML-13)*, volume 28:1, pages 214–221, 2013.
- [19] Y. Liu. *Tableseer: Automatic Table Extraction and Search and Understanding*. Ph.D. Thesis, The Pennsylvania State University, 2009.
- [20] D. P. Lopresti and G. Nagy. A tabular survey of automated table processing. In *Selected Papers from the Third International Workshop on Graphics Recognition, Recent Advances, GREC '99*, pages 93–120, 2000.
- [21] S. Mandal, S. P. Chowdhury, A. K. Das, and B. Chanda. A simple and effective table detection system from document images. *IJDAR*, 8:172–182, 2006.
- [22] S. Mark. Ugm: Matlab code for undirected graphical models, 2011. <http://www.cs.ubc.ca/~schmidtm/Software/UGM.html>.
- [23] G. Nagy. Twenty years of document image analysis in PAMI. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22:38–62, 2000.
- [24] D. Pinto, A. McCallum, X. Wei, and W. B. Croft. Table extraction using conditional random fields. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval, SIGIR '03*, pages 235–242, 2003.
- [25] J.-Y. Ramel, M. Crucianu, N. Vincent, and C. Faure. Detection, extraction and representation of tables. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 1, ICDAR '03*, 2003.
- [26] F. Shafait and R. Smith. Table detection in heterogeneous documents. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems, DAS '10*, pages 65–72, 2010.
- [27] A. Shahab, F. Shafait, T. Kieninger, and A. Dengel. An open approach towards the benchmarking of table structure recognition systems. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems, DAS '10*, pages 113–120, 2010.
- [28] A. C. Silva, A. M. Jorge, and L. Torgo. Design of an end-to-end method to extract information from tables. *International Journal Document Analysis Research*, 8:144–171, 2006.
- [29] R. W. Smith. Hybrid page layout analysis via tab-stop detection. In *Proceedings of the 2009 10th International Conference on Document Analysis and Recognition, ICDAR '09*, pages 241–245, 2009.
- [30] W. Tersteegen and C. Wenzel. Scantab: Table recognition by reference tables. In *in Proceedings of Document Analysis Systems, (DAS'98)*, 1998.
- [31] Y. Wang, R. Haralick, and I. T. Phillips. Improvement of zone content classification by using background analysis. In *In Fourth IAPR International Workshop on Document Analysis Systems. (DAS 2000)*, Rio de Janeiro, pages 10–13, 2000.
- [32] Y. Wang, R. M. Haralick, and I. T. Phillips. Automatic table ground truth generation and a background-analysis-based table structure extraction method. In *ICDAR*, pages 528–532, 2001.
- [33] Y. Wang and J. Hu. A machine learning based approach for table detection on the web. In *Proceedings of the 11th International Conference on World Wide Web, WWW '02*, pages 242–250, 2002.
- [34] Y. Wang, I. T. Phillips, and R. M. Haralick. Table detection via probability optimization. In *in Proceedings of Document Analysis Systems, (DAS'02)*, pages 272–282. Springer-Verlag, 2002.
- [35] Y. Wang, I. T. Phillips, and R. M. Haralick. Document zone content classification and its performance evaluation. *Pattern Recogn.*, 39(1):57–73, Jan. 2006.
- [36] R. Zanibbi, D. Blostein, and R. Cordy. A survey of table recognition: Models, observations, transformations, and inferences. *Int. J. Doc. Anal. Recognit.*, 7:1–16, 2004.