Automatic table detection and retention from scanned document images via analysis of structural information

Conference Paper · December 2017

DOI: 10.1109/ICIIP.2017.8313719

Varsha Ranka
University of Florida
1 PUBLICATION 4 CITATIONS
SEE PROFILE

Manish Kumar Gupta
Centre for Development of Advanced Computing
11 PUBLICATIONS 43 CITATIONS
SEE PROFILE

Automatic Table Detection and Retention from Scanned Document Images via Analysis of Structural Information

Kapil Mehrotra *[0], Varsha Ranka †[1], Shubham Patil ‡[1], Shubham Patni §[1]andTushar Raut ¶[1]

[0]Center for Development of Advanced Computing (C-DAC), Pune
[1]Department of Computer Engineering, PICT, Pune

Abstract: The problem of automatic table detection has always been a great topic of debate in the field of Document Analysis and Recognition (DAR). Digital documents are efficient than their printed counterparts for storage, maintenance and republishing. Being a non-textual object of a document, tables prevent OCR system to digitize a document perfectly and distorts layout and structure of digitized documents. There is no available algorithm or method which solves this problem for all possible types of tables. This paper tackles the problem of table detection and retention by proposing a bi-modular approach based on structural information of tables. This structural information includes bounding lines, row/column separators and space between columns. Through analysis of these properties, our experiments on a dataset of above 600 images have produced an accuracy of 90% on average.

1 Introduction

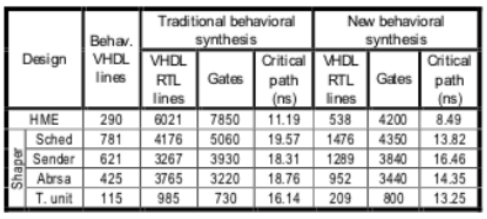
Documents are a great way to collect and store information from historical to current time. But these documents are not suitable for storage and safekeeping of information for a very long duration due to their materialistic nature. This problem can be solved by converting a printed document into a digital form, the process being called "digitization" of a document in the domain of Document Analysis and Recognition (DAR). Digitizing even a single document manually is a very tedious task and hence there is a need of automating such process. With the proliferation in the use of digital documents for information retrieval and processing, the present methodologies of DAR are not yet fully capable of recognizing them. Digitization of a document not only allows to store a document safely and efficiently, but also enable editing and republishing.

The best technology for digitizing a document is called Optical Character Recognition (OCR) systems. But the OCR systems mainly focus on identification and retrieval of textual patterns with its features and fails when the document contains complex non-textual objects like tables, images, graphs, charts, equations etc. OCRs are not yet capable of maintaining the layout and structure of the doc-

*kapilm@cdac.in
†rankavarsha@ymail.com
‡patil.sm17@gmail.com
§patnishubham07@gmail.com
¶rauttushar444@gmail.com

ument when such non-textual objects are present. Hence the problem of their identification arises. This paper deals with problem of automatic table detection and retention from scanned document images where emphasis is given on tables as a non-textual object.

The main challenge in this problem lies in the variety of table layouts known which makes table detection problem a topic of open research. A simple example can be a regular table with complete bounding information. Whereas, a complex table can contain multiple row or column spans with no or partial bounding information. In this paper, we propose two strategies based on the structural analysis of tables for the table detection problem. The term "structural analysis" represents the process of identifying how the tables are laid out and then retaining them for future uses.



Design		Behav. VHDL lines	Traditional behavioral synthesis			New behavioral synthesis		
			VHDL RTL lines	Gates	Critical path (ns)	VHDL RTL lines	Gates	Critical path (ns)
HME		290	6021	7850	11.19	538	4200	8.49
Shaper	Sched	781	4176	5060	19.57	1476	4350	13.82
	Sender	621	3267	3930	18.31	1289	3840	16.46
	Abrsa	425	3765	3220	18.76	952	3440	14.35
	T. unit	115	985	730	16.14	209	800	13.25

Figure 1: An example of heterogeneous table

The further paper is organized as follows. Section II describes the previous research done by other authors in this area. Section III and Section IV deals with formulation and description of methods proposed by us. Section V describes the results obtained from our work and section VI describes the future scope and conclusion of our work.

2 Previous Research

There is a lot of previous research conducted on table detection in scanned document images, camera document images, PDFs etc. Gaurav Harit and Anukriti Bansal [1] proposed a method for table detection based on identifying unique table header and trailer patterns. It finds white space separators and rulings and analyzes attributes like thickness, color, presence of rulings and text blobs characteristics. But its accuracy decreases for multiple tables aligned side by side and sometimes due to inadequate distinguishing capability of the identified header pattern.

The properties of tabular lines which are different than text lines like number of words in line, height, word gaps etc are considered by MAC Akmal Jahan and Roshan G Ragel in [2]. In [3], Gatos et al. proposed a novel technique that neither requires any training phase nor uses domain-specific heuristics. This paper focuses on detection of lines which describe both physical and logical structure using morphological operations. But it is restricted for tables bounded by ruled lines. Kasar et al. [4] implemented a SVM based table detection using 26 low-level features, extracted from each group of intersecting horizontal and vertical lines.

S. Mandal et al. [5] introduced a method based the observation that table has distinct columns and white-space between them is larger than word-gap. But it fails when columns don't have significant gap or the document has multiple columns. In [6], S.Deivalakshmi et al. proposed a fast table detection and extraction technique based on morphological operations, connected components and labeling. In [7], Tian et al. proposed a method for low quality images. First skew is corrected using Radon transform, followed by detection of lines longer than predefined threshold using RLSA. Finally, frame lines are detected using Hough transform. Anukriti Bansal et al. [8] identified tables based on a novel learning-based framework which learns layout of document. A fixed point model is used to get features which learns the interrelationship between the blocks and then attains a contraction mapping by providing a unique label to each block.

In [9], Ray Smith and Faisal Shafait, based on the tab-lines, inferred the column layout of the page and grouped connected components into column partitions. Table was detected using this column layout and column partition. This paper also considers detecting page column split. Thomas

G. Kieninger [10] proposed method based on initial block segmentation. Cluster of words are formed by expanding in bottom up direction, which isolate the columns. But it is unable to identify heterogeneous tables. The system produces erroneous results, when the document contains header or footer section, document title or any small modification in layout.

3 Problem Formulation

A most general categorization of the tables can be done in two types based on its structure. Type I tables are formed using bounding lines or column and row separators. These tables are said to have line information. Type II tables are formed using space separated columns with no/partial line information.These tables are said to have space information.

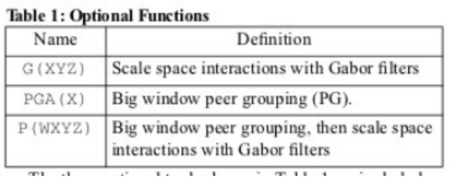
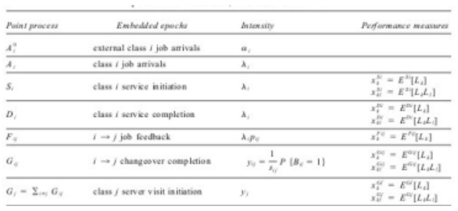


Table 1: Optional Functions

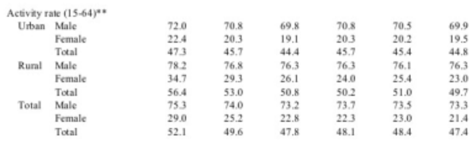
Name	Definition
G(XYZ)	Scale space interactions with Gabor filters
PGA(X)	Big window peer grouping (PG).
P(WXYZ)	Big window peer grouping, then scale space interactions with Gabor filters

Figure 2: An example of table with complete line information



Point process	*Embedded epochs*	*Intensity*	*Performance measures*
A_i^0	external class i job arrivals	α_i	
A_i	class i job arrivals	λ_i	
S_i	class i service initiation	λ_i	$x_i^{Si} = E^{Si}[L_i]$ $x_{il}^{Si} = E^{Si}[L_iL_l]$
D_i	class i service completion	λ_i	$x_i^{Di} = E^{Di}[L_i]$ $x_{il}^{Di} = E^{Di}[L_iL_l]$
F_{ij}	$i \rightarrow j$ job feedback	$\lambda_i p_{ij}$	$x_i^{Fij} = E^{Fij}[L_i]$
G_{ij}	$i \rightarrow j$ changeover completion	$y_{ij} = \frac{1}{s_{ij}} P\{B_{ij} = 1\}$	$x_i^{Gij} = E^{Gij}[L_i]$ $x_{il}^{Gij} = E^{Gij}[L_iL_l]$
$G_j = \Sigma_{i\neq j} G_{ij}$	class j server visit initiation	y_j	$x_i^{Gj} = E^{Gj}[L_i]$ $x_{il}^{Gj} = E^{Gj}[L_iL_l]$

Figure 3: An example of table with partial line information



Activity rate (15-64)**							
Urban	Male	72.0	70.8	69.8	70.8	70.5	69.9
	Female	22.4	20.3	19.1	20.3	20.2	19.5
	Total	47.3	45.7	44.4	45.7	45.4	44.8
Rural	Male	78.2	76.8	76.3	76.3	76.1	76.3
	Female	34.7	29.3	26.1	24.0	25.4	23.0
	Total	56.4	53.0	50.8	50.2	51.0	49.7
Total	Male	75.3	74.0	73.2	73.7	73.5	73.3
	Female	29.0	25.2	22.8	22.3	23.0	21.4
	Total	52.1	49.6	47.8	48.1	48.4	47.4

Figure 4: An example of table with space information

The problem statement is to detect the physical and logical structure of a table and to retain it for future purpose, given its structural information.

Dataset

We collected a dataset of 644 images from C-DAC and split it into two parts. These parts are described below.

1. The dataset I contained 300 images with complete line information.
2. The dataset II contained 344 images with no/partial line information and space information.

We have manually identified whether a table contains line information or space information but this process can be further automated using text/non-text classification.

Assumptions

There are few assumptions that are needed to solve the problem of automatic table detection efficiently and with high accuracy. These are as follows.

1. Each table is considered to have minimum of two rows and two columns.
2. A Block within a table can't further contain any non-textual objects like tables, image, graphs, charts etc.
3. The input dataset received from C-DAC is assumed to be binarized and skew-corrected, which is not always the case. General binarization algorithms like Otsu Method (refer [3]) and skew correction algorithms using Hough Transform (refer [7]) can be used as pre-processing steps to deal with this problem.

4 Models or Methods

In this paper, we have proposed a bi-modular approach to cover almost all types of table layouts ranging from simple tables to heterogeneous tables. This bi-modular approach has two strategies to detect the physical and logical structure of a table. However, only the first strategy cover up the problem of table retention and republishing. Retention using second strategy is still under study. Both of these strategies are explained below.

Method I : Table Detection using Line Information

This method is an extension of the method mentioned in [3]. It includes the process of identifying tables by extracting its layout, bounding lines and row/column separators from the original document and then reconstructing it by finding intersection points (Refer [3]). The emphasis is given on bounding structure instead on internal features like space between columns or tab stops. We have tweaked this method by optimizing intermediate steps and by adding heuristic methods for reconstructing the table from its identified intersection points. We have also added a methodology for retention of table with its contents.

This method is explained below as follows.

Step 1: Calculation of Average Character Height (ACH) of the document

It is a general observation that a document contains more text objects than non-text objects. To remove the lines formed due to textual strokes, Average Character Height (ACH) of the document is required. Steps to calculate ACH are briefly explained below.

1. Find heights of all connected components in the document using graph algorithms.

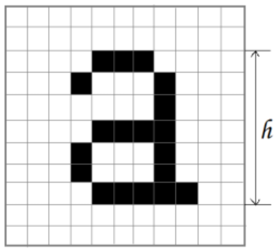


Figure 5: Example representing height of a connected component

2. Create a histogram and plot heights of all connected components on this histogram.

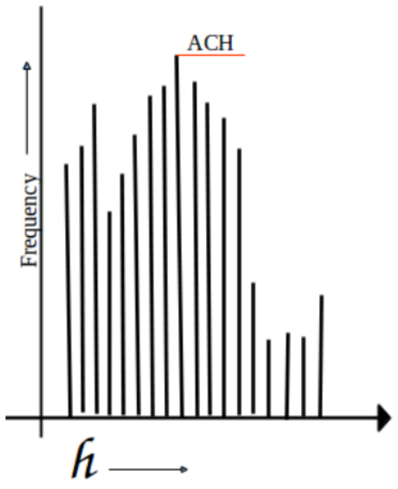
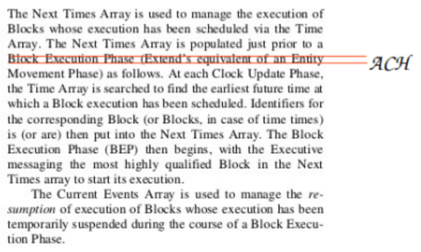


Figure 6: Histogram to find ACH

3. The peak value of this histogram will give the ACH of document.



The Next Times Array is used to manage the execution of Blocks whose execution has been scheduled via the Time Array. The Next Times Array is populated just prior to a Block Execution Phase (Extend's equivalent of an Entity Movement Phase) as follows. At each Clock Update Phase, the Time Array is searched to find the earliest future time at which a Block execution has been scheduled. Identifiers for the corresponding Block (or Blocks, in case of time times) is (or are) then put into the Next Times Array. The Block Execution Phase (BEP) then begins, with the Executive messaging the most highly qualified Block in the Next Times array to start its execution.

The Current Events Array is used to manage the *resumption* of execution of Blocks whose execution has been temporarily suspended during the course of a Block Execution Phase.

Figure 7: ACH of a document

Step 2: Extracting horizontal and vertical lines using ACH and discarding textual strokes

A conclusion that can be derived from the previous step is that a sequence of black pixels having length near or equal to ACH are character strokes and more than ACH may be candidates of table structure. To improve the accuracy, ACH can be used with an integer multiplication factor. This factor is different for different languages and can be found using trial and error method. For example, 7 as multiplication factor works well for English documents.

Using the ACH obtained from the previous step, we can extract the candidate lines of table structure from the document. To improve the results this step is executed in two phases. This helps in handling the lines of varying width. Also, these two phases remove all the images present in the document.

In first phase, if value of multiplication factor is X then use X/3 as multiplication factor for the document. This helps in discarding lines of very short length.

In second phase, if value of multiplication factor is X then use X as multiplication factor for the document. This helps in retaining lines of longer length and shorter width.

For each phase, the steps of processing are given as below.

1. Horizontal processing of horizontal lines :
   For each sequence of black pixels in horizontal direction, if length of sequence is greater than X*ACH, then keep it in the result else flush the sequence.
2. Vertical processing of horizontal lines :
   For each sequence of black pixels in vertical direction, if length of sequence is less than X*(ACH/2), then keep it in the result else flush the sequence.

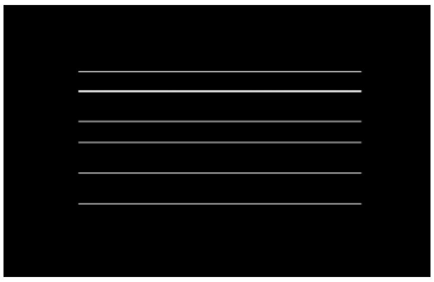


Figure 8: Result of Horizontal Processing

3. Vertical processing of vertical lines :
   For each sequence of black pixels in vertical direction, if length of sequence is greater than X*ACH, then keep it in the result else flush the sequence.
4. Horizontal processing of vertical lines :
   For each sequence of black pixels in horizontal direction, if length of sequence is less than X*(ACH/2), then keep it in the result else flush the sequence.

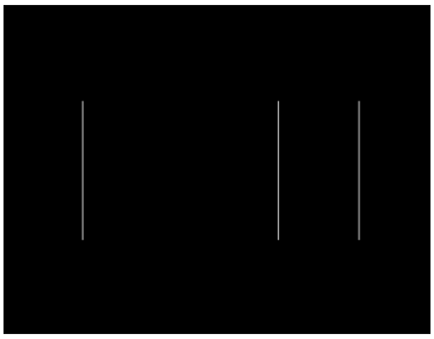


Figure 9: Result of Vertical Processing

The results obtained from horizontal and vertical processing of the document can be merged to obtain an approximate tabular structure.

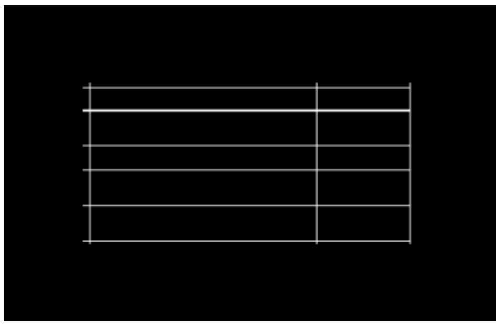


Figure 10: Merged image obtained from horizontal and vertical processing

This merged image contains all possible candidate lines but may also contain some extra lines which are not part of table. To improve the results further, some heuristics are applied in the next step.

Step 3: Identification of intersection points

The results from previous steps give us all candidate lines that may constitute a table. One important characteristic of a table having line information is that the tabular lines intersect each other. These intersection points will belong to the table. When only outer boundary is absent non-intersecting points tend to occur as well. These points can be very well used to classify candidate lines from last step into valid table lines or non-table lines.

Consider the diagram below. It represents the types of intersections points that can be found in a table. Point A, B, C and D represent corner L joints. While E, G and H represent T joints.

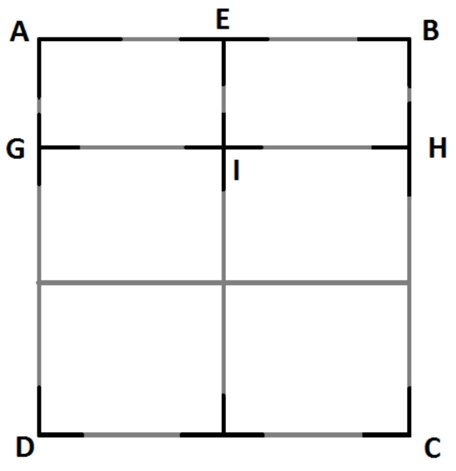


Figure 11: Pattern for intersection points

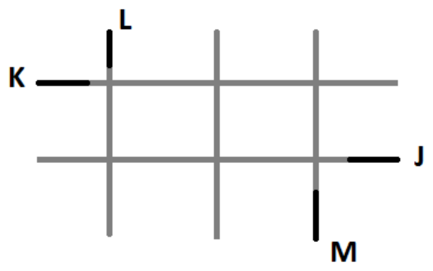


Figure 12: Pattern for non-intersection points

Steps to identify these intersection points using previous candidate lines are described below.

1. Identify intersecting points : Initially we identified A,B,C,D patterns using hit and miss transform and labeled separately as shown in Fig 11. Structural element used for pattern A is as shown in Fig 13 for hit and miss transform. We also identified intersection patterns of variable width and height with constraints that height must be greater than ACH and width must be less than ACH.

2. Hit and Miss Transform : The structural element used in the hit and miss transform is slightly an extension of the type that has been introduced for erosion and dilation, in that it can contain both foreground and background pixels. Black represents foreground pixel, gray represents background pixels and white represents don't care.

3. Various combinations of initial patterns A, B, C and D are merged to get different patterns. For example A and B are combined to get E. Similarly A and D are combined to get G.

4. Identify non intersecting points: Next we identified patterns J, K, L and M which are non-intersecting points. Here we have possibility of having points which may not form table for example end points of single line.

5. Align points: For further processing, we aligned all these obtained points horizontally and vertically. Here while aligning points horizontally, points having vertically distance <= 0.5*ACH are placed on same horizontal line. Similarly we aligned points vertically.

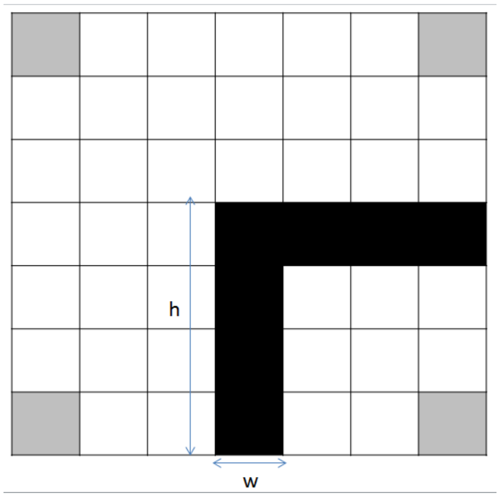


Figure 13: Example of corner pattern A

The result of these steps give us all points identified using candidate lines which may belong to table.

Figure 14: Image with intersection points

Step 4: Reconstruction of Table using identified intersection points.

The intersection points identified in the last step can be used to reconstruct the tables as present in the original document that gives lines which belongs to the tables and the reconstructed image can be used to identify table area.

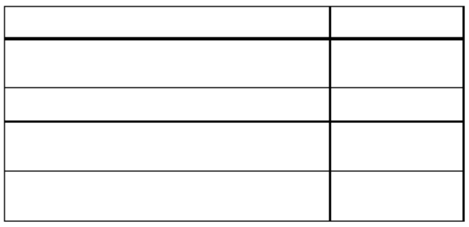


Figure 15: Example of reconstructed image

Refer assumption 1. Hence, any intersection point with only one point parallel in either of horizontal or vertical direction can be discarded as it would make a table with only one row or one column.

After identification of intersection points, the heuristic rules in the following table can be used for the reconstruction of tables using intersection points are described below.

	Horizontal	Vertical
A	Yes	Yes
B	No	Yes
C	No	No
D	Yes	No
E	Yes	Yes
F	No	Yes
G	Yes	Yes
H	Yes	No
I	Yes	Yes
J	No	No
K	Yes	No
L	No	Yes
M	No	No

Table 1: Heuristic Rules for reconstruction

As per the heuristics, Yes under Horizontal or Vertical column indicates that the intersection pattern should be connected to next point in that direction. For example, consider point A in Figure 12. This point can be joined to next point in horizontal direction as well as vertical direction. While B can only be extended in the vertical direction. Similarly from point M we can not draw any line.

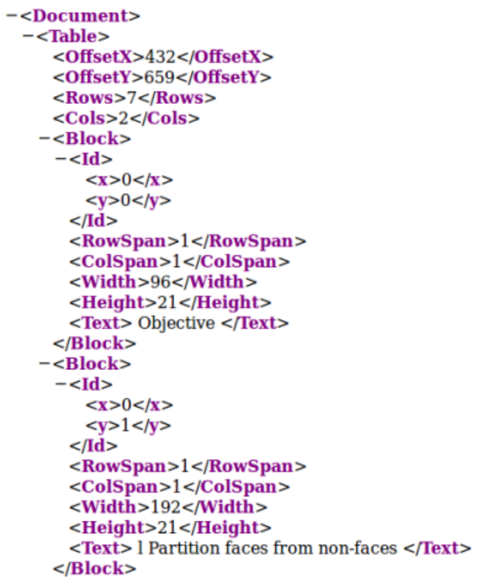
Furthermore, we will extend intersection point only if it has at least two other points present in the same direction. This ensures we eliminate independent lines, random points and lines which do not belong to table.

After this step, we will get all tabular structures reconstructed in the output image.

These tables will only contain the line information without any textual data inside it.

Method I: Table Retention

Tables with all of its line information is generated in the last step. But this table is still an image and can't be used for editing or republishing. Retention is a necessary task to convert a table from an image form to an editable document like Microsoft Word. Retaining a table doesn't only deal with its structure but also with its inner contents. Hence, here OCR is an integral part of the process. The basic idea is to convert a graphical representation of a table into an XML schema. This XML schema can then be used by another methods like Python pydocx library, to reproduce or republish a table into an editable form by parsing the XML schema of the table.



```
−<Document>
  −<Table>
      <OffsetX>432</OffsetX>
      <OffsetY>659</OffsetY>
      <Rows>7</Rows>
      <Cols>2</Cols>
    −<Block>
      −<Id>
          <x>0</x>
          <y>0</y>
        </Id>
        <RowSpan>1</RowSpan>
        <ColSpan>1</ColSpan>
        <Width>96</Width>
        <Height>21</Height>
        <Text> Objective </Text>
      </Block>
    −<Block>
      −<Id>
          <x>0</x>
          <y>1</y>
        </Id>
        <RowSpan>1</RowSpan>
        <ColSpan>1</ColSpan>
        <Width>192</Width>
        <Height>21</Height>
        <Text> l Partition faces from non-faces </Text>
      </Block>
```

Figure 16: Example of XML schema

Following steps are included in the process of retaining a table.

Step 1: Per table page segmentation

Sometimes a document may contain multiple tables on a single page. To reduce the complexity of the problem, we separate these tables into individual entities and deal with them one by one.

The method of segmenting a document into individual tables is as follows.

1. Take the output image from reconstruction step and find all connected components in it.
2. Label background pixels with 0 and all other connected components in the increasing sequence.
3. For all labels other than 0, find the bounding rectangular box and obtain its height and width.
4. Use this height and width to crop the table from original image and treat it as an individual entity for further use.

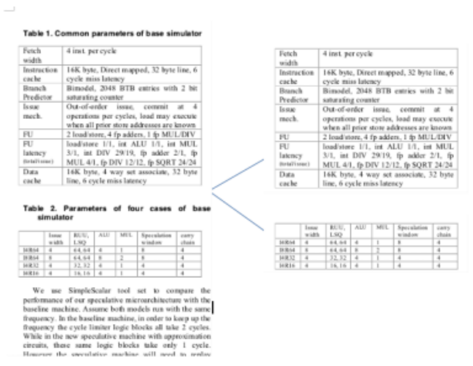


Table 1. Common parameters of base simulator

Fetch width	4 inst. per cycle
Instruction cache	16K byte, Direct mapped, 32 byte line, 6 cycle miss latency
Branch Predictor	Bimodel, 2048 BTB entries with 2 bit saturating counter
Issue mech.	Out-of-order issue, commit at 4 operations per cycles, load may execute when all prior store addresses are known
FU	2 load/store, 4 fp adders, 1 fp MUL/DIV
FU latency (total/issue)	load/store 1/1, int ALU 1/1, int MUL 3/1, int DIV 29/19, fp adder 2/1, fp MUL 4/1, fp DIV 12/12, fp SQRT 24/24
Data cache	16K byte, 4 way set associate, 32 byte line, 6 cycle miss latency

Table 2. Parameters of four cases of base simulator

	Issue width	RUU, LSQ	ALU	MUL	Speculation window	carry chain
[illegible]	4	64, 64	4	1	8	4
[illegible]	8	64, 64	8	2	8	4
[illegible]	4	32, 32	4	1	4	4
[illegible]	4	16, 16	4	1	4	4

We use SimpleScalar tool set to compare the performance of our speculative microarchitecture with the baseline machine. Assume both models run with the same frequency. In the baseline machine, in order to keep up the frequency the cycle limiter logic blocks all take 2 cycles. While in the new speculative machine with approximation circuits, these same logic blocks take only 1 cycle.

Figure 17: Segmentation of document into individual cells

Step 2: Per table block segmentation.

Each table can be thought of as a constitution of individual rectangular blocks. In this step we divide a table into its constituent blocks and extract XML features from these blocks.

A simple method for segmenting a homogeneous table into its individual blocks is described below.

1. Take the set of intersection points of a table from the previous steps, sort them by x coordinate and create a 2D map of the table using this set.

2. For heterogeneous tables, sometimes a intersection point can't be mapped to this 2D map. For such points fill the map with (-1,-1)

3. For each valid point i.e the point which is not (-1,-1), find a valid point towards the right direction and towards the bottom direction. Use these three points to obtain a fourth valid point.

4. Use these four points to extract each block from the original image.

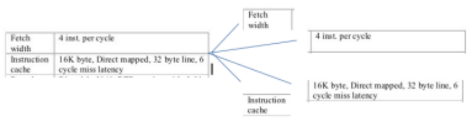

Fetch width
4 inst. per cycle
Instruction cache
16K byte, Direct mapped, 32 byte line, 6 cycle miss latency
Fetch width
4 inst. per cycle
16K byte, Direct mapped, 32 byte line, 6 cycle miss latency
Instruction cache

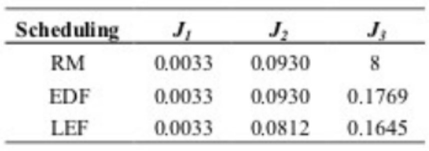
Figure 18: Example of table segmentation

Step 3: Passing individual blocks to OCR

Once individual blocks are obtained, based on assumption 1, these blocks are ready to be passed to OCR. Any open source OCR engine e.g. Tesseract can be used for research purpose. The OCR will extract the textual part from each block and insert into XML schema of the table at the corresponding block entry.

Method II : Table detection using space information

This methods works well for tables which do not contain any line information in them and is an extension over method given in [6]. These tables are mostly space separated columns or tables with partial line information. For example, refer the figure below. The basic idea is to obtain a threshold value of space which can be used to separate tabular lines from non-tabular lines.



Scheduling	J_1	J_2	J_3
RM	0.0033	0.0930	8
EDF	0.0033	0.0930	0.1769
LEF	0.0033	0.0812	0.1645

Figure 19: Tables with space information

In this method, instead of finding the physical and logical structure of table inside the document, each line of document is analyzed based on some heuristics and classified whether it belongs to tabular part or not. Also, table can't be classified as a whole because even a single line falling short from classification criteria in heuristic algorithms will cause whole table to be discarded. Hence, using this method table should be detected line by line.

The steps included in this method is described as follows.

Step 1: Identification and removal of partial line information

The main aim of this step is to convert the input document into required format by removing all existing line information in the document. So that, the document can be analyzed only on the basis on space information.

The steps in this process are as follows.

1. Use step 2 of method I to identify all horizontal and vertical lines in the document.

2. Subtract the output of previous step from the original document to get a document with only space information

Step 2: Splitting document into individual lines.

To analyze the document line by line, we first need to split it. This is done using horizontal projection of the document on a vertical axis. This projection is nothing but a histogram representing the count of black pixels in each row. The steps are as follows.

1. Obtain the horizontal projection of document on a vertical axis.
2. Analyze this projection vertically and obtain the position of gap in the histogram. A sequence of zeros is considered as a gap.
3. For each gap obtained, crop the original image from last starting point to generate strips of the documents. These strips represent the lines in the documents.

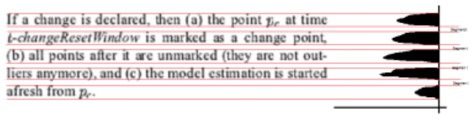

If a change is declared, then (a) the point p_c at time
t-changeResetWindow is marked as a change point,
(b) all points after it are unmarked (they are not outliers anymore), and (c) the model estimation is started
afresh from p_c.

Figure 20: Splitting document into individual lines

Step 3: Analyzing word gaps in each line to obtain space thresholds. After lines of document are obtained, each line is analyzed for word gap. This is done as follows.

1. For each line, obtain the vertical projection of line on a horizontal axis.
2. Analyze this projection to find word gaps in each line. Map each word gap onto an another histogram.
3. Analyze the last histogram. It contains two humps : one for letter gap and one for word gap.
4. The peak value of the second hump with some multiplication factor can be considered as the threshold value of gap

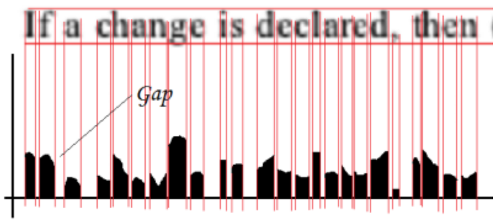


Figure 21: Analysis of word gap per line

Step 4: Classification of lines into tabular or non-tabular lines. The threshold value of gap obtained in the last step can be used as a heuristic to classify lines into tabular or non-tabular parts. This is done as follows.

1. For each line, analyze its vertical projection to find a gap greater than the threshold value of gap.
2. If such gap exists, the line can be classified as a tabular line. Else it is a non-tabular line.

5 Results

After running our proposed methods on dataset of 644 images, where 300 images belonged to dataset type I and 344 images belonged to dataset type II, we have the following result.

Method I:

Parameters	Total Images	Total Tables	Result
Correct Detection	240	340	340
Partial Detection	20	50	40
Missed Tables	20	31	25
False Positives	20	23	21

Table 2: Results of method I

Area Precision: (388/444)*100 = 87.38 %.

Method II:

Parameters	Hindi	English	Marathi	Mix
Total table rows	1047	267	293	271
Correct detection	961	257	290	257
Missed table rows	76	9	21	10
False Positives	300	28	4	8

Table 3: Results of method I

Area Precision: (1765/1878)*100 = 93.9 %.

6 Conclusion

In this paper, after a thorough literature survey and experiments, we have proposed two strategies which modulates the table detection problem as structural analysis problem. First strategy has shown an accuracy rate of 87% on 300 images. Second strategy has shown an accuracy rate of 93% on 344 images. These methods work pretty well on tables having line information or space information.

The future scope of this paper includes study and implementation of methods for table retention using method II. We are aiming to improve efficiency of method I by solving the problems faced in tables with bounding lines missing e.g. a table used to play tic-tac-toe game and to handle exceptions like multi-column layouts for method II. We are also aiming to automate the process of classification of table into table with line information or table with space information. After solving all these issues, proposed methods will be able to cover almost all types of tables.

7 References

1. Gaurav Harit, Anukriti Bansal, "Table detection in document images using header and trailer patterns," ICVGIP, December, 2012.
2. MAC Akmal Jahan, Roshan G Ragel, "Locating Tables in Scanned Documents for Reconstructing and Republishing," 7th International Conference on Information and Automation for Sustainability (ICIAFS), 2014
3. B. Gatos, D. Danatsas, I. Pratikakis and S. J. Perantonis, "Automatic Table Detection in Document Images," In Proceedings of the Third international conference on Advances in Pattern Recognition,ICAPR, pp. 609-618, 2005
4. T. Kasar, P. Barlas, S. Adam, C. Chetalain and T, Pacquet,"Learning to detect tables in scanned document images using line information," 12th International Conference on Document Analysis and Recognition, 2013.
5. S. Mandal, S. P. Chowdhury, A. K. Das, Bhabatosh Chanda, "A Very Efficient Table Detection System from Document Images,"
6. S.Deivalakshmi, K.Chaitanya, P.Palanisamy, "Detection of Table Structure and Content Extraction from Scanned Documents," International Conference on Communication and Signal Processing,April 3-5, 2014, India
7. Yangyang Tian, Chenqiang Gao, Xiaoming Huang, "Table Frame Line Detection in Low Quality Document Images Based on Hough Transform," 2nd International Conference on Systems and Informatics (ICSAI 2014)
8. Anukriti Bansal, Gaurav Harit, Samantra Dutta Roy, "Table Extraction from Document Images using Fixed Point Model," ICVGIP '14, December 14-18 2014, Bangalore, India.
9. Faisal Shafait, Ray Smith, "Table Detection in Heterogeneous Documents," DAS'10, June 9-11 2010, Boston, MA, USA

10. Thomas G. Kieninger, "Table Structure Recognition Based On Robust Block Segmentation," Document Recognition V, April 1998, San Jose, CA.

11. Tanushree Dhiran, Rakesh Sharma, "Table Detection and Extraction from Image Document," International Journal of Computer and Organization Trends, Volume 3, Issue 7, August 2013, Bangalore, India.