Converting Optically Scanned Regular or Irregular Tables to a Standardised Markup Format to be Accessible to Vision-Impaired

Azadeh Nazemi[1,*], Iain Murray[1], Chandrika Fernaando[1] & David A. McMeekin[2]

[1]Department of Electrical and Computing, Curtin University, Western Australia

[2]Department Spatial Science, Curtin University, Western Australia

*Correspondence: Department of Electrical and Computing, Curtin University, Western Australia. E-mail: azadeh.nazemi@curtin.edu.au

Received: August 5, 2016 Accepted: September 12, 2016 Online Published: September 22, 2016

doi:10.5430/wje.v6n5p9 URL: http://dx.doi.org/10.5430/wje.v6n5p9

Abstract

Documents use tables to communicate multidimensional information clearly, summarise and present data in an easy-to-interpret way. Tabular information in scanned PDF due to its nature without further processing is not accessible for vision-impaired people who use assistive technology such as screen readers. The lack of access to table contents limits educational and workplace opportunities for people with vision impairment. They require a complete equivalent to access table. This paper describes techniques which apply to scanned PDF document for table detection, extraction and cell segmentation to retrieve cell contents and represent them in a navigable manner to vision-impaired.The output is in mark-up format and provides navigation ability to access content of a table.

Keywords: *vision–impaired; Optical Character Recognition (OCR); document layout analysis; segmentation; accessibility; markup format; regular or irregular table; navigation*

1. Introduction

Tables play a very important role in storing numbers, words or items of any kind, in a definite and compact form for convenience of study, reference or calculation. The structure of a table consists of vertical and horizontal separators and their intersections create table cells. Cell positions can be extracted using image processing techniques. A table may contain different kinds of objects such as text, graphics and mathematical formula (Watanabe, Luo, & Sugie, 1995), (Tsuruoka, Takao, Tanaka, Yoshikawa, & Shinogi, 2001).

Although table cells can be extracted individually, the content cannot be accessed without OCR. The accuracy of recognized information extracted from the table cells strongly depends on performance of OCR which converts images to editable text to convey meaning. Keeping track of the order of cells during extraction and accurate information recognition from the cells are two major issues which must be considered during table information retrieval for accessibility.

Kieninger proposed a method for recognition of table structure based on robust block segmentation starting with the initial block clustering followed by post processing to correct errors and decomposition of columns into individual table cells (Kieninger, 1998)

Tupaj et al presented method to segment and analyze the document image using image processing techniques in order to detect potential table areas. They examined technical and financial tables (Tupaj, Shi, Chang, & Chang, 1996).

Yildiz et al developed a method for extracting table information by utilizing only the absolute position of text elements in a file, concretely in PDF files(Yildiz, Kaiser, & Miksch, 2005).

Ng et al applied Machine learning method and used purely surface features such as the relative locations of characters in a line and across lines. They only focused on detecting tables, columns, and rows and not on the content(Ng, Lim, & Koo, 1999)

In Seth method cell segmentation is performed via indexing of value cells by table header (Seth & Nagy, 2013). This method focused on cell segmentation and structure recognition for regular tables and CSV spread sheets.

This paper is based on the study that considers scanned PDF and focuses on regular/irregular and tables without ruled lines but not nested tables.

2. Table Structure and Navigtion Ability

Tabular data arrangement means arranging data in rows and columns. The use of tables is pervasive throughout all communication, research and data analysis. Tables appear in print media, handwritten notes, computer software, architectural ornamentation and many other places. Tables differ significantly in variety, structure, flexibility, notation, representation and use. In books and technical articles, tables are typically presented apart from the main text in numbered and captioned floating blocks. A table consists of rows and columns, the intersection of a row and a column is a cell. The table cells may be grouped, segmented, or arranged in many different ways, and even nested (Zielinski, 2006). To provide full access to tables, the titles of table and titles of each column or row must be presented before reading the data in cells. It should be left up to the user to decide the most logical one-dimensional way to read materials. The tables are mostly inaccessible in scanned PDF documents. Additionally, there is no guarantee for tables in untagged structured PDF documents to be represented in correct order due to lack of tags. Table structure often contains vertical lines as column separators and horizontal lines as row separators. In an optimal case, all horizontal and vertical lines which make table structure would be specified. Accessing each data cell individually is based on finding the related column and row intersection point.

The proposed method in this research is cells extraction by finding (Xmin, Ymin) and (Xmax,Ymax) for each cell which represent cell bounding box. All cell segments are tagged based on positions and sent individually to OCR software. Cell segmentation provides navigation ability by representing table cells by rows, by columns or both.

3. Methodology

This research developed an open source appliction runs under Linux and contains the following modules:

1) Block segmentation
2) Line segmentation of the document
3) Caption finding
4) Table detection and extraction (regular and irregular) from scanned documents using table layout
5) Cell segmentation
6) Cell labeling to make navigable Markup format

To facilitate OCR these steps are followed by line and symbol segmentation of cell contents. Figure 1 illustrated flowchart of system.

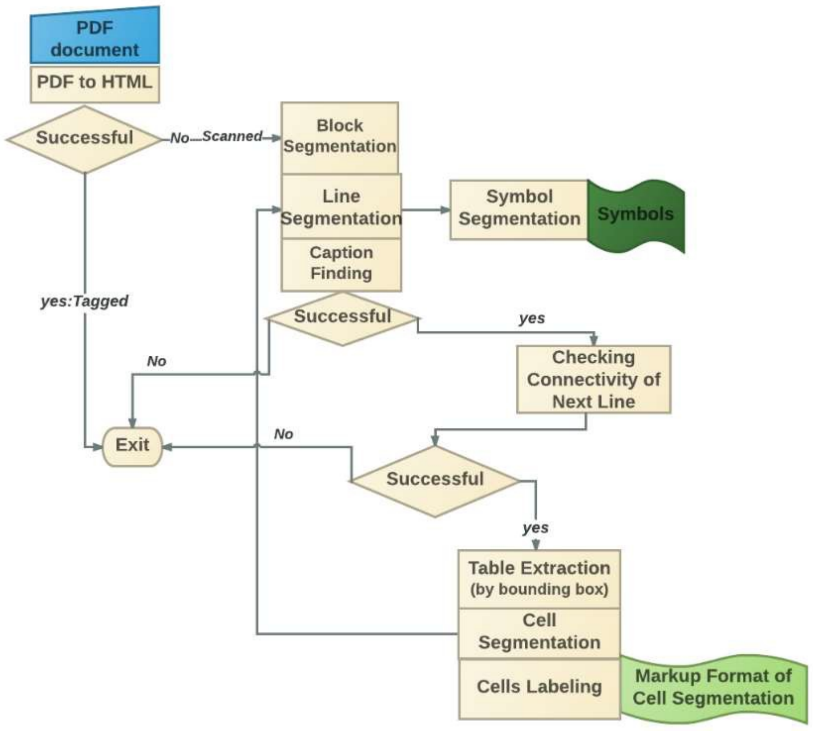


Figure 1. System Overview

3.1 Block Segmentation

If the scanned PDF page contains only one column, lines in the OCR result will be presented in order. However in cases where the PDF includes more than one column, there is no guarantee that the OCR result will convey the correct line ordering. In a double column document the OCR results do not keep the reading order. Block segmentation divides the page into logical blocks, preserving the reading order.

This research utilizes morphological operations for block segmentation. By applying morphology repetitively over a whole image, specific shapes can be found, removed or modified. If a pixel is white and completely surrounded by other white pixels, then that pixel is obviously not on the image edge. The entire process actually depends on the definition of a structuring element or kernel, which defines what pixels are to be classed as neighbours for each specific morphological method. The dilate operation returns the maximum value in the neighbourhood. The erode operation returns the minimum value in the neighborhood. Eroded morphology helps to remove non-interested white spaces between text words and recognises remaining white space area as block separators.

Since PDF document may contain different logical layouts such as footer and multiple columns, block segmentation must be done both vertically and horizontally using the following steps:

- Considering the horizontal white space area;
- Running horizontal segmentation;
- Considering vertical white space area in each extracted horizontal segment; and
- Applying vertical segmentation for all extracted horizontal segments.

3.2 Line Segmentation of the Blocks

In this research, line segmentation is performed based on specifying the rectangular background (white space) as horizontal gap between text lines in the image by finding the position of all lines which meet the following conditions and assuming BL and FL are background lines (gaps) and foreground lines (text) respectively:

$$BL_i is\ a\ horizontal\ gaps\ between\ text\ lines:$$

$$If\ length\ (BL_i)\ =\ width(block) = W\ \&\ L_i - 1\ is\ not\ similar\ BL_i$$

$$Assuming\ BL_i and\ BL_j are\ two\ consecutive\ horizontal\ gaps$$

$$Height\ (FL_i) = Y\left(BL_j\right) - Y\ (BL_i)$$

$$FL_i\ is\ obtained\ by\ cropping\ original\ block\ image.$$

The line segmentation module performs based on detecting the consecutive foreground pixels which have following conditions:

```
$Assuming\ n\ background\ Pixels\ as$:
$BP_i; :::::: BP_n$
$if\ n = Width(Page)$
  for (i = 1::n)
$if\ (j = i + 1\ \&\ X(BP_j) = X(BP_{i+1})\ \&\ Y\left(BP_j\right) = Y(BP_i)$
$Y(BP_i)$ is a segmentation position.
```

3.3 Table Extraction

The requirement of detection and identification of tables from document images is crucial to any document image analysis (Mandal, Chowdhury, Das, & Chanda, 2006). Mandal used the algorithm to observe tables that have distinct columns which implies that gaps between the fields are substantially larger than the gaps between the words in text lines.

Seth and Nagy has segmented the CSV table using only “logical layout analysis” without resorting to any appearance features or natural language understanding (Seth & Nagy, 2013). Handley described a table analysis system which reconstructs table formatting information from table images whether or not the cells are explicitly delimited. Inputs to the system are word bounding boxes and any horizontal and vertical lines that delimit cells(Handley, 2000)

Table detection in this research is based on caption finding. Bounding boxes of line segmentation results are checked for caption finding, each line satisfies in $Left\ margin = Right\ margin$ could be a caption.

Assuming:

$$I)\ j > i$$

$$II)\ L_i, L_{i+1}\ , \ldots\ L_{j-1}\ are\ \ captions$$

$$III) L_j\ \ is\ connected\ component$$

Then next block is table and extracted as described in block segmentation

3.4 Cell Segmentation

A table in a document is a rectilinear arrangement of cells where each cell contains a sequence of words. Several lines of text may compose one cell. Cells may be delimited by horizontal or vertical lines, but often this is not the case (Handley, 2000)

This research performs cell segmentation based on identification of intersections of vertical and horizontal borders as cell’s bounding box. In order to navigate through cells, the structure recognition is necessary for both regular and irregular tables. Unlike regular tables which contain fixed number of cells for all rows, number of cells in a given row of an irregular table varies.

This research investigates the tables which include horizontal and vertical line borders as separators of cells in the table. The following steps shows how to obtain the positions of the borders using image processing techniques such as applying morphology, compose and composite as the following snippet shows .

```
convert $1 -threshold 75% bin.png
#convert to binary image
height=$( identify -format "%h" bin.png )
width=$( identify -format "%w" bin.png )
#obtain    size of image
```

```
   convert bin.png -alpha off -channel A -transparent white -ransparent black
-negate +channel
    -write mpr:ORG +clone -negate –morphology Erode rectangle:60x1 -mask mpr:ORG
-mology Dilate rectangle:60x1 +mask -compose Lighten -composite -negate txt:-| grep
-Ev '#FFFFFF '| sed '1d;s/:.*/ /g;s/,/ /g'| awk '{ print $2 ,$1 }'| sort -b -k1n,1|
awk 'p{ print $1 -p,$1 ,$2 }{p=$1 }{ if (NR ==1) print $1 ,$1 ,$2 }'| awk '$1 >1
'| awk '{print $2}'|sort -b -k1n,1> hbox.dat
```

#obtain x coordinate value of intersections

#Morphology dilate rectangle: Nx1 remove vertical lines and keep horizontal lines

```
   convert bin.png -rotate 90 -alpha off -channel A -transparent white -transparent
black -negate +channel -write mpr:ORG +clone -negate -morphology Erode
rectangle:60x1 -mask mpr:ORG -morphology Dilate rectangle:60x1 +mask -compose
Lighten -composite -negate -rotate -90 txt:-| grep -Ev '#FFFFFF '| sed '1d;s/:.*/
/g;s/,/ /g'| awk '{ print $1 ,$2 }'| sort -b -k1n,1|awk 'p{ print $1
-p,$1 ,$2 }{p=$1 }{ if (NR ==1) print $1 ,$1 ,$2 }'| awk '$1>1 '| awk '{ print $2}'|sort
-b -k1n,1> vbox.dat
```

# obtain y coordinate value of intersections

#Morphology dilate rectangle: Nx1 remove vertical lines and keep horizontal lines

```
   nv=$( cat vbox.dat |wc -l)
   nh=$( cat hbox.dat |wc -l)
   for (( j=1 ; j <$nh;j++)) do
   for (( i=1 ; i <$nv;i++)) do
   xs=$(cat vbox.dat | awk 'NR=='$i'')
   xe=$(cat vbox.dat | awk 'NR=='$i'+1')
   ye=$(cat hbox.dat | awk 'NR=='$j'+1')
   ys=$(cat hbox.dat | awk 'NR=='$j'')
   x=$(($xe-$xs))
   y=$(($ye-$ys))
   convert $1 -shave 0x0 -repage $width"x"$height"+0+0" png:-|convert png:- -crop
$x"x"$y"+"$xs"+"$ys  'colum'$i'row'$j.png
   done
   done
```

#Cell extraction using top left and bottom right coordinate values and cropping

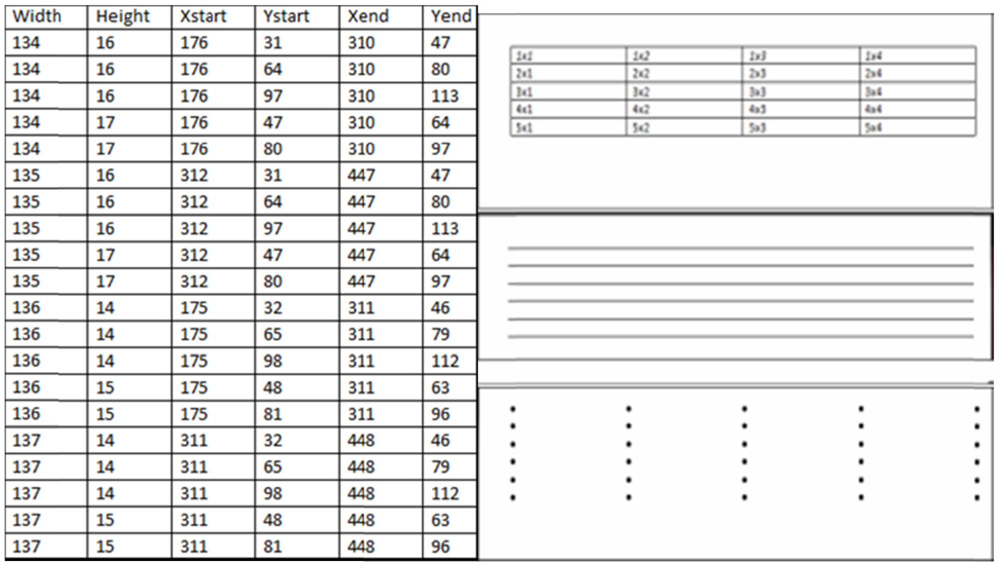
In terms of cell segmentation the tables are divided into two categories, namely, regular and irregular tables.

3.4.1 Regular Tables

In regular tables, all vertical borders meet all horizontal borders so all intersections are the coordinate points of cell bounding boxes. It means obtaining intersections leads to access cell bounding boxes. In regular table of size , mxn where *m* is number of rows and *n* is number of columns, each row has *n* number of cells.

```
Total Number of cells $m \times n$
m=Number   of  rows
n= Number  of  columns
for( i=1 to m): row(i)has n cells
$X_i, Y_j$ is top left coordinate point
$X_{i+1}, Y_j$ is top right coordinate point
$X_i, Y_{j+1}$ is bottom left coordinate point
$X_{i+1}, Y_{j+1}$ is bottom right coordinate point
```

Figure 2 illustrates an image of a regular table, its vertical borders, horizontal borders, points of intersection (right) and the table of cell bounding boxes (left)



Width	Height	Xstart	Ystart	Xend	Yend
134	16	176	31	310	47
134	16	176	64	310	80
134	16	176	97	310	113
134	17	176	47	310	64
134	17	176	80	310	97
135	16	312	31	447	47
135	16	312	64	447	80
135	16	312	97	447	113
135	17	312	47	447	64
135	17	312	80	447	97
136	14	175	32	311	46
136	14	175	65	311	79
136	14	175	98	311	112
136	15	175	48	311	63
136	15	175	81	311	96
137	14	311	32	448	46
137	14	311	65	448	79
137	14	311	98	448	112
137	15	311	48	448	63
137	15	311	81	448	96

1x1	1x2	1x3	1x4
2x1	2x2	2x3	2x4
3x1	3x2	3x3	3x4
4x1	4x2	4x3	4x4
5x1	5x2	5x3	5x4

Figure 2. A regular Table, Its Vertical Borders, Horizontal Borders, Points of Intersection (Right) and the Table of Cell Bounding Boxes (Left)

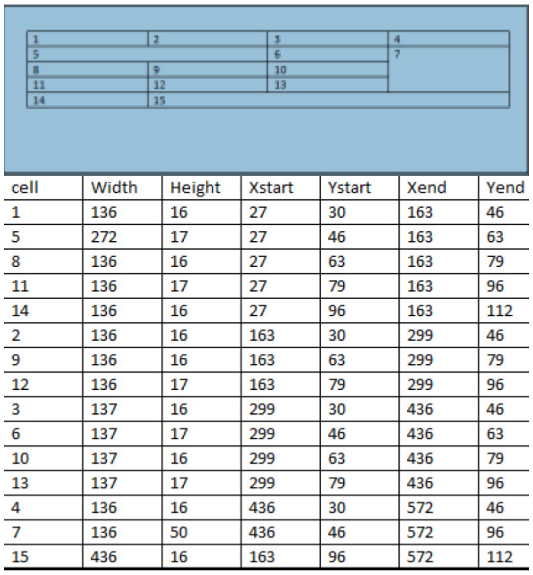
3.4.2 Irregular Tables

There is no particular relationship between number of rows and number of columns in irregular tables. Therefore, there is no guarantee that every vertical border meets all horizontal borders. All points of intersection must be checked to make sure whether it is a coordinate point of a cell bounding The point (X_i, Y_i) is considered as the coordinate of the top left of a cell bounding box if and only if there are two intersection points such that (X_j, Y_j) is the top right coordinate value and (X_k, Y_k) is the bottom left coordinate values which meet the following conditions:

$$(X_j > X_i) \quad \& \quad (Y_i = Y_J)$$
$$(X_K = X_i) \quad \& \quad (Y_K > Y_i)$$

As a result (X_j, Y_k) becomes the bottom right coordinate value of the cell which is extracted and labelled as Cell(i) using the top left and bottom right coordinate values of the cell bounding box.

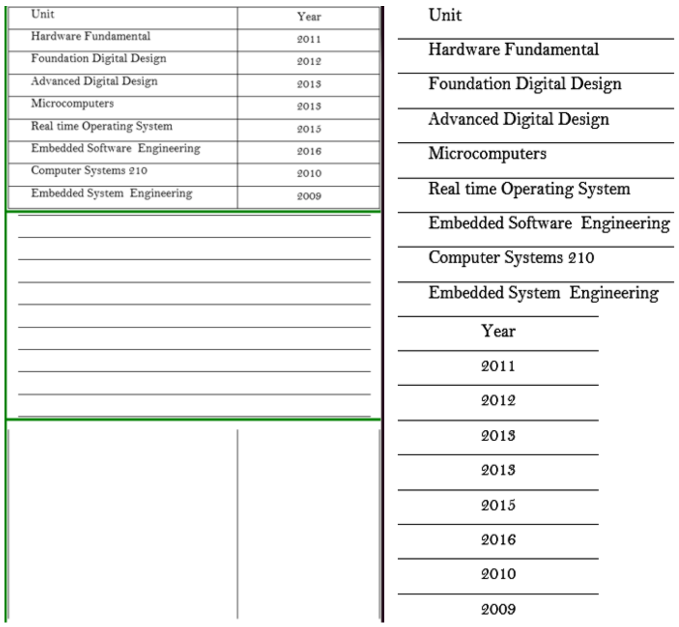
Figure 3 shows an irregular table with vertical and horizontal intersections and extracted cells (not in order).



cell	Width	Height	Xstart	Ystart	Xend	Yend
1	136	16	27	30	163	46
5	272	17	27	46	163	63
8	136	16	27	63	163	79
11	136	17	27	79	163	96
14	136	16	27	96	163	112
2	136	16	163	30	299	46
9	136	16	163	63	299	79
12	136	17	163	79	299	96
3	137	16	299	30	436	46
6	137	17	299	46	436	63
10	137	16	299	63	436	79
13	137	17	299	79	436	96
4	136	16	436	30	572	46
7	136	50	436	46	572	96
15	436	16	163	96	572	112

Figure 3. Irregular Table

Figure 4 shows different steps for cell segmentation including sample table (top left) horizontal borders extraction (middle left) vertical borders extraction (bottom left) and segmented cells (right)



Unit	Year
Hardware Fundamental	2011
Foundation Digital Design	2012
Advanced Digital Design	2013
Microcomputers	2013
Real time Operating System	2015
Embedded Software Engineering	2016
Computer Systems 210	2010
Embedded System Engineering	2009

Figure 4. Sample Table (Top Left) Horizontal Borders Extraction (Middle Left) Vertical Borders Extraction (Bottom Left) and Segmented Cells (Right)

3.5 Cell Labelling

Each cell is represented by two coordinate values of top left and bottom right corners.

$$Top\ \ left\ \ corner\ =\ TL$$
$$Bottom\ \ right\ \ corner\ =\ BR$$
$$cell_i\ and\ cell_j\ locate\ in\ same\ column\ if\ \ xTL_i\ =\ xTL_j\ \ \&\ xBR_i\ \ = xBR_j$$
$$cell_i\ and\ cell_j\ locate\ in\ same\ row\ if\ \ yTL_i\ =\ yTL_j\ \ \&\ yBR_i = yBR_j$$
$$Assuming, cell_i\ and\ cell_j\ :\ if\ x_{TL_i} =\ x_{TL_j}\ \ \&\ x_{BR_i} <\ x_{BR_j}$$
$$Then\ they\ locate\ in\ same\ column\ but\ table\ is\ irregular\ and$$
$$cell_j\ may\ also\ locates\ in\ another\ column.\ cell_i\ and\ cell_j\ are$$
$$located\ in\ same\ row\ if\ y_{TL_i} =\ y_{TL_j}\ \ \&\ y_{BR_i}\ =\ y_{BR_j}$$
$$Assuming\ cell_i\ and\ cell_j\ if\ y_{TL_i}\ =\ y_{TL_j}\ \ \&\ y_{BR_i}\ <\ y_{BR_j}$$
$$Then\ they\ locate\ in\ same\ row\ but\ table\ is\ irregular\ and$$
$$cell_j\ may\ also\ locates\ in\ another\ row.$$

3.6 Markup Presentation and Navigation Ability

In marks-up format table is presented with the <table></table> tag.

Table row is defined with the <tr> </tr>tag.

First row indicates table header and is defined with the <th></ th> tag.

Table cell is defined with the <td></td> tag

Number of rows in table is number of <tr> tag between <table></table>

Number cells in each rows is number of <td>between<tr></tr>

Marks-up format supports navigation ability. Navigation through the table means finding by Row, by Column or by Cell.

$cell_{ij}$ Indicates cell(j) which is located row(i), Thus to navigate through it needs to extract the part of marks-up file located between i^{th} of <tr></tr> then in extracted part of

find the part located between j^{th} of <td></td>

The following snippet shows how to navigate through

- The specific row_i

```
START_ROW=location-of-ith<tr>=$(cat marks-up.html|grep –bo $(cat  merks-up.html|
grep  -m$i   “<tr>”   |tail –n1))
END_ROW=location-of-ith</tr>=$(cat marks-up.html|grep –bo $(cat  merks-up.html|
grep  -m$i   “</tr>”   |tail –n1))
LENTGH=$[$END_ROW-$START_ROW]
 File=$( cat marks-up.html)
ROW=$(File:$START_ROW:$LENTGH)
```

- The specific $column_j$
- The specific $cell_{ij}$ using marks-up format file and regular expression in Linux

```
START_CELL=location-of-jth<td>=$(echo $ROW|grep –bo $(echo $ROW| grep -m$j   “<td>”
|tail –n1))
END_CELL=location-of-jth</td>=$(echo $ROW|grep –bo $(echo $ROW| grep -m$j   “</td>”
|tail –n1))
LENTGH=$[$END_CELL-$START_CELL]
CELL=$(ROW:$START_ROW:$LENTGH)
```

3.7 Symbol Segmentation

Symbol Segmentation module processes binary image of cell and converts content into a sequence of images. Each output image contains at least one unconnected black segment on a transparent background. It finds first non-transparent pixel in input image, extracts segment at this coordinate value and removes that segment from image. This task is repeated until all non-transparent pixels are processed. In order to improve performance of segmentation, a distance morphological function is used to create larger space between symbols.

In this module, the segmented lines of a cell obtained in previous step are converted into a series of isolated symbols and sent to OCR considering their orders.

3.8 Table in Spread Sheet

Tables in spread sheet usually are presented without border lines. This research treat them such as table with border lines. Applying line segmentation horizontally and vertically makes boarder lines then cell segmentation is performed as stated in section 3.4. This method can be used for information retrieval and navigate through ‘table of contents” The following snippet is used for making border lines as it is shown in Figure 5.

```
in="$1"
height=$(convert "$in" -format '%h' info:)
width=$(convert "$in"  -format '%w' info:)
#obtain table dimensions
convert "$in" -threshold 60% -fuzz 1% -trim +repage -scale 50% bin.mpc
#convert to binary image
convert bin.mpc -morphology erode diamond:2  bbin.mpc
#remove non-interested white spaces by eroding
max=$(convert bbin.mpc txt:-|sed 's/:.* #/ /g;1d;s/,/ /g'|grep 'FFFFFF'| awk
'{ print $1 ,$2 }'| sort -b -k2n,2| awk '{ a[$2 ]++} END { for (i in a) print i,a[i]}
'|sort -b -k2n,2| awk 'END { print $2 }')
```

#find maximum height of horizontal white spaces

```
x=($( convert bbin.mpc txt:-| sed 's/:.* #/ /g;1d;s/,/ /g'|grep 'FFFFFF'| awk
'{ print $1 ,$2  }'| sort -b -k2n,2| awk '{ a[$2 ]++} END { for (i in a) print i,a[i]}
'|sort -b -k2n,2| awk ''$max'-$2<5'| sort -b -k1n,1| awk '{print $1 }'| awk 'p{ print
$1,p}{p=$1 }{ if (NR ==1) print $1 ,$1 }'|awk '$1 -$2 !=1 '| awk '{ print $1 }'))
nox=${#x[@]}
```

#find locations of extended horizontal white spaces

```
for (( e=0;e< $nox ;e++ )) do
l=$(( $e +1))
if [[ $e -ne $(($nox-1)) ]] ; then
convert bin.mpc -crop "0x"$((${x[$l]}-${x[$e]}))"+0+"${x[$e]} hor$l.png
else
convert bin.mpc -crop "0x"$(($height-${x[$e]}))"+0+"${x[$e]} hor$l.png
fi
done
convert bin.mpc -crop "0x"${x[0]}"+0+0"  hor0.png
convert $(ls hor*.png) -background red -splice 1x1+0+0 -append row.png
```

#insert horizontal border lines in proper locations

```
max=$(convert bbin.mpc txt:-|sed 's/:.* #/ /g;1d;s/,/ /g'|grep 'FFFFFF'| awk
'{ print $2 ,$1 }'| sort -b -k2n,2| awk '{ a[$2 ]++} END { for (i in a) print i,a[i]}
'|sort -b -k2n,2| awk 'END { print $2 }')
```

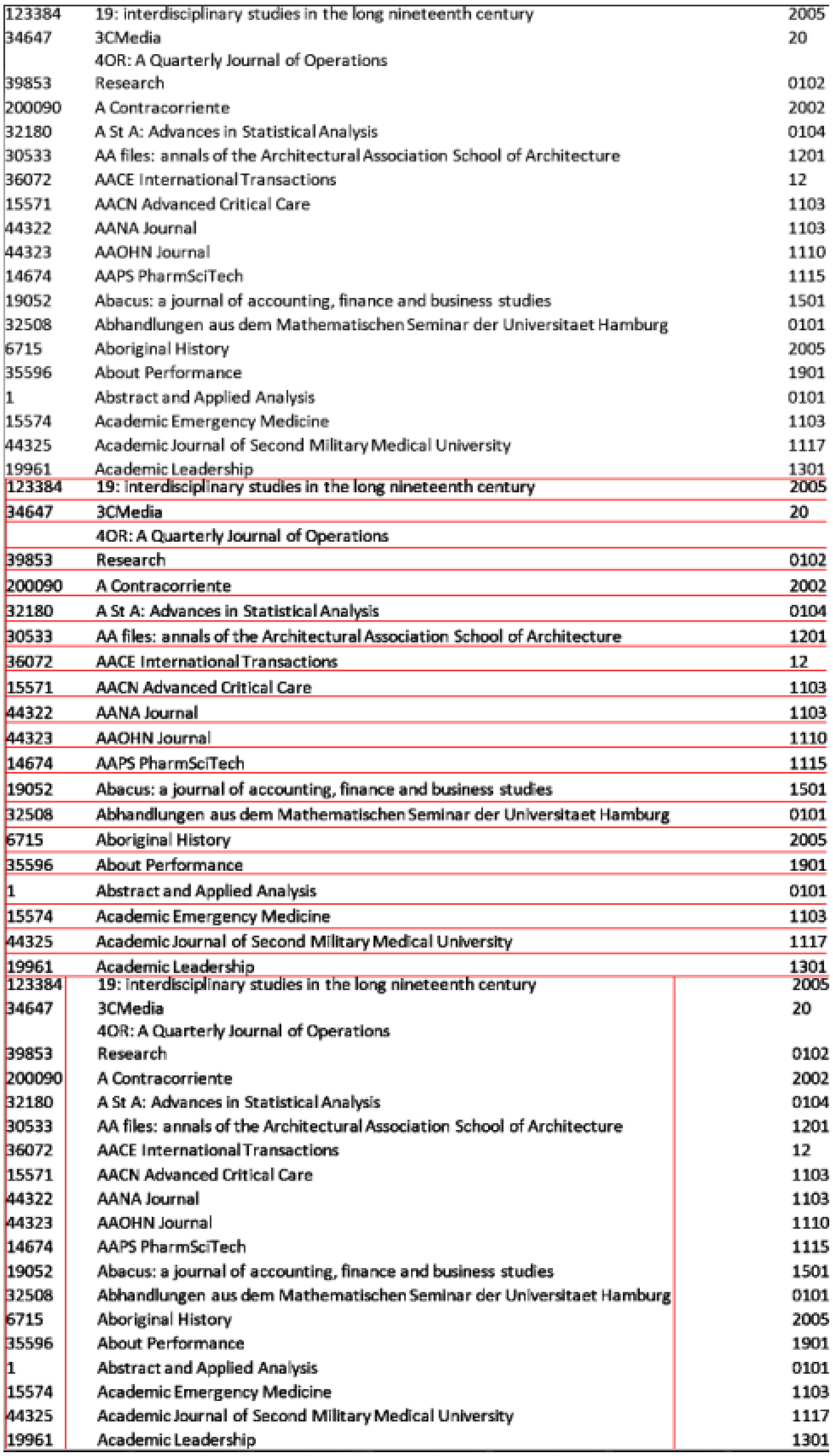
#find maximum width of vertical white spaces

```
x=($( convert bbin.mpc txt:-| sed 's/:.* #/ /g;1d;s/,/ /g'|grep 'FFFFFF'| awk
'{ print $2,$1  }'| sort -b -k2n,2| awk '{ a[$2 ]++} END { for (i in a) print i,a[i]}
'|sort -b -k2n,2| awk ''$max'-$2<5'| sort -b -k1n,1| awk '{print $1 }'| awk 'p{ print
$1,p}{p=$1 }{ if (NR ==1) print $1 ,$1 }'|awk '$1 -$2 !=1 '| awk '{ print $1 }'))
nox=${#x[@]}
```

#find locations of extended vertical white spaces

```
for (( e=0;e<$nox ;e++ )) do
l=$(( $e+1 ))
if [[ $e -ne $(($nox-1)) ]] ; then
convert bin.mpc -crop $((${x[$l]}-${x[$e]}))"x0+"${x[$e]}"+0" ver$l.png
else
convert bin.mpc -crop $(($width-${x[$e]}))"x0+"${x[$e]}"+0"  ver$l.png
fi
done
convert bin.mpc -crop ${x[0]}"x+0+0"  ver0.png
convert $(ls ver*.png) -background red -splice 1x1+0+0 +append column.png
```

#insert vertical border lines in proper locations



123384	19: interdisciplinary studies in the long nineteenth century	2005
34647	3CMedia	20
39853	4OR: A Quarterly Journal of Operations Research	0102
200090	A Contracorriente	2002
32180	A St A: Advances in Statistical Analysis	0104
30533	AA files: annals of the Architectural Association School of Architecture	1201
36072	AACE International Transactions	12
15571	AACN Advanced Critical Care	1103
44322	AANA Journal	1103
44323	AAOHN Journal	1110
14674	AAPS PharmSciTech	1115
19052	Abacus: a journal of accounting, finance and business studies	1501
32508	Abhandlungen aus dem Mathematischen Seminar der Universitaet Hamburg	0101
6715	Aboriginal History	2005
35596	About Performance	1901
1	Abstract and Applied Analysis	0101
15574	Academic Emergency Medicine	1103
44325	Academic Journal of Second Military Medical University	1117
19961	Academic Leadership	1301

123384	19: interdisciplinary studies in the long nineteenth century	2005
34647	3CMedia	20
39853	4OR: A Quarterly Journal of Operations Research	0102
200090	A Contracorriente	2002
32180	A St A: Advances in Statistical Analysis	0104
30533	AA files: annals of the Architectural Association School of Architecture	1201
36072	AACE International Transactions	12
15571	AACN Advanced Critical Care	1103
44322	AANA Journal	1103
44323	AAOHN Journal	1110
14674	AAPS PharmSciTech	1115
19052	Abacus: a journal of accounting, finance and business studies	1501
32508	Abhandlungen aus dem Mathematischen Seminar der Universitaet Hamburg	0101
6715	Aboriginal History	2005
35596	About Performance	1901
1	Abstract and Applied Analysis	0101
15574	Academic Emergency Medicine	1103
44325	Academic Journal of Second Military Medical University	1117
19961	Academic Leadership	1301

123384	19: interdisciplinary studies in the long nineteenth century	2005
34647	3CMedia	20
39853	4OR: A Quarterly Journal of Operations Research	0102
200090	A Contracorriente	2002
32180	A St A: Advances in Statistical Analysis	0104
30533	AA files: annals of the Architectural Association School of Architecture	1201
36072	AACE International Transactions	12
15571	AACN Advanced Critical Care	1103
44322	AANA Journal	1103
44323	AAOHN Journal	1110
14674	AAPS PharmSciTech	1115
19052	Abacus: a journal of accounting, finance and business studies	1501
32508	Abhandlungen aus dem Mathematischen Seminar der Universitaet Hamburg	0101
6715	Aboriginal History	2005
35596	About Performance	1901
1	Abstract and Applied Analysis	0101
15574	Academic Emergency Medicine	1103
44325	Academic Journal of Second Military Medical University	1117
19961	Academic Leadership	1301

Figure 5. Table in Spread Sheet(Top), Inserting Horizotal Line Border Lines(Middle), Inserting Vertical Lines Border(Bottom)

4. Conclusion

This study focused on detection and extraction tables using document layout analysis. Furthermore, cell segmentation is performed to facilitate OCR. The markup format for the table is created by cell labelling based on bounding boxes. This output preserves the navigation ability which is required for information retrieval from tables. Cell segmentation was performed based on finding boundary boxes using horizontal and vertical border lines intersection recognition. Tables without border lines such as csv files, spared sheets or table of contents are investigated by finding extended vertical and horizontal white spaces and inserting border lines in interested location of white spaces. Then cell segmentation is performed as before.

Segmented cells are labelled based on location and sent to symbol segmentation module and OCR to be converted to plain text. The accuracy of final retrieved information from table cells severely depends on segmented cells.

This research did not involve to OCR algorithm and used TESSERACT and OCRopus two open sources OCR under Linux. For testing stage developed application applied for 25 double column scientific scanned article. 23 out of 25 (92%) regular tables were recognized and extracted. 18 out of 23 (78%) were segmented completely. TESSERACT performance showed 13 out of 18 (72%) converted to text successfully, however OCRopus accuracy was 10 of 18 (0.55). Text from each cell is tagged to make marks-up format which supports navigation and searching ability without any error.

Obtained result proved segmented cells must be preprocessed before sending to OCR. Further development will investigate segmented cells preprocessing before OCR and extracting cells from nested tables using a recursive algorithm.

References

Handley, John C. (2000). Table analysis for multiline cell identification. *Document Recognition & Retrieval VIII, 4307*, 34-43.

Kieninger, Thomas G. (1998). Table structure recognition based on robust block segmentation. *Proc. SPIE, 3305*, Document Recognition V, 22 (April 1, 1998),

Mandal, S., Chowdhury, S. P., Das, A. K., & Chanda, Bhabatosh. (2006). A simple and effective table detection system from document images. *International Journal of Document Analysis and Recognition (IJDAR), 8*(2), 172-182. http://dx.doi.org/10.1007/s10032-005-0006-5

Ng, Hwee Tou, Lim, Chung Yong & Koo, Jessica Li Teng. (1999). *Learning to recognize tables in free text*. Paper presented at the Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics, College Park, Maryland.

Seth, Sharad & Nagy, George. (2013). *Segmenting Tables via Indexing of Value Cells by Table Headers*. Paper presented at the Proceedings of the 2013 12th International Conference on Document Analysis and Recognition.

Tsuruoka, S., Takao, K., Tanaka, T., Yoshikawa, T., & Shinogi, T. (2001). *Region segmentation for table image with unknown comple//x structure.* Paper presented at the Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on.

Tupaj, Scott, Shi, Zhongwen, Chang, C. Hwa, & Chang, Dr. C. Hwa. (1996). Extracting Tabular Information From Text Files *EECS Department, Tufts University*.

Watanabe, Toyohide, Luo, Qin, & Sugie, Noboru. (1995). Layout Recognition of Multi-Kinds of Table-Form Documents. *IEEE Trans. Pattern Anal. Mach. Intell., 17*(4), 432-445. http://dx.doi.org/10.1109/34.385976

Yildiz, Burcu, Kaiser, Katharina & Miksch, Silvia. (2005). *pdf2table: A Method to Extract Table Information from PDF Files.* Paper presented at the IICAI.

Zielinski, K. (2006). *Software Engineering: Evolution and Emerging Technologies*. Amsterdam: IOS Press. ISBN 1-58603-559-2205