algorithms

Article

A Proposal of Printed Table Digitization Algorithm with Image Processing

Chenrui Shi [1], Nobuo Funabiki [1,*], Yuanzhi Huo [1], Mustika Mentari [2], Kohei Suga [3] and Takashi Toshida [3]

[1] Department of Information and Communication Systems, Graduate School of Natural Science and Technology, Okayama University, Okayama 700-8530, Japan
[2] Department of Information Technology, State Polytechnic of Malang, Malang 65141, Indonesia
[3] Astrolab, Tokyo 107-0062, Japan
* Correspondence: funabiki@okayama-u.ac.jp

Abstract: Nowadays, *digital transformation (DX)* is the key concept to change and improve the operations in governments, companies, and schools. Therefore, any data should be digitized for processing by computers. Unfortunately, a lot of data and information are printed and handled on paper, although they may originally come from digital sources. Data on paper can be digitized using an *optical character recognition (OCR)* software. However, if the paper contains a table, it becomes difficult because of the separated characters by rows and columns there. It is necessary to solve the research question of "how to convert a printed table on paper into an *Excel* table while keeping the relationships between the cells?" In this paper, we propose a *printed table digitization algorithm* using image processing techniques and OCR software for it. First, the target paper is scanned into an image file. Second, each table is divided into a collection of *cells* where the topology information is obtained. Third, the characters in each cell are digitized by OCR software. Finally, the digitalized data are arranged in an *Excel* file using the topology information. We implement the algorithm on *Python* using *OpenCV* for the image processing library and *Tesseract* for the OCR software. For evaluations, we applied the proposal to 19 scanned and 17 screenshotted table images. The results show that for any image, the *Excel* file is generated with the correct structure, and some characters are misrecognized by OCR software. The improvement will be in future works.

Keywords: digitization; printed table; OCR; Python; OpenCV; Tesseract

check for updates

Citation: Shi, C.; Funabiki, N.; Huo, Y.; Mentari, M.; Suga, K.; Toshida, T. A Proposal of Printed Table Digitization Algorithm with Image Processing. *Algorithms* **2022**, *15*, 471. https://doi.org/10.3390/a15120471

Academic Editor: Frank Werner

Received: 28 September 2022
Accepted: 9 December 2022
Published: 11 December 2022

1. Introduction

Nowadays, *digital transformation (DX)* is the key concept to changing the working styles and improving the operations in governments, companies, and schools. In [1], DX is defined as the strategy of enabling business innovations predicated on the incorporation of digital technologies into operational processes, products, solutions, and customer interactions. DX can leverage the opportunities of new technologies and their impacts on businesses by focusing on the creation and monetization of digital assets. Therefore, any data should be digitized and processed by computers and networks.

Unfortunately, a lot of data and information are still handled on paper. People often feel comfortable accessing data and managing them on paper. Actually, a lot of data printed on paper were originally generated using computers digitally and were printed out using software fonts. Therefore, data on paper are often digitized using an *optical character recognition (OCR)* software. However, when the paper contains tables, it is difficult to digitalize them correctly, because, in a table, the characters are usually separated by rows and columns, called *cells*, and the characters in one cell are related to each other. Thus, the recognition of the characters in a table while keeping the relationships between the cells is far more difficult than that of normal texts.

Therefore, the research question in this paper is "how to convert a printed table on paper into an Excel table by keeping the relationships between the cells in the table?". Table formats can be diverse among documents, which further increases the difficulty of the solution to this question. To control the difficulty, we limit the table layouts to two that have been commonly used in this paper. For both layouts, it is possible to find and keep the relationships between the left-side cells and the right-side cells in each row for every column.

In the first layout, the table structure is a simple grid, which is the basic one. In this layout, the number of columns is constant from the first row to the last row in the table, and the number of rows is also constant from the leftmost column to the rightmost column, as shown in Figure 1. In the second layout, the table structure is a little bit more complex. The number of columns is constant from the top row to the bottom row in the table, but the number of rows may increase at some columns from the leftmost column to the rightmost column, as shown in Figure 2.

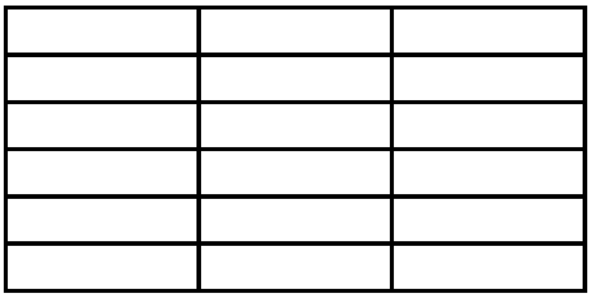


Figure 1. The first table mentioned in the text.

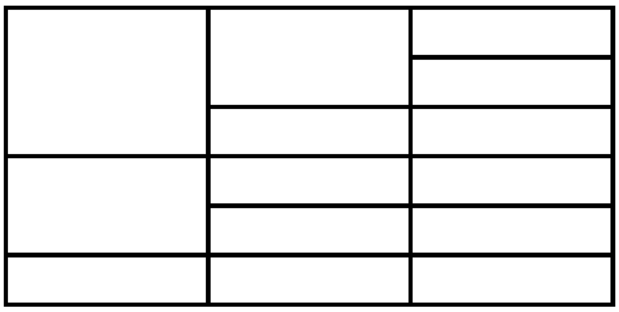


Figure 2. The second table mentioned in the text.

In this paper, we propose the *printed table digitization algorithm* for digitizing a printed table on paper using image processing techniques and an *optical character recognition (OCR)* software. The table layout is limited to either one of the abovementioned layouts. In this algorithm, first, the printed table is divided into a collection of *cells* where the topology information between the cells is extracted. Second, the characters in each cell are digitized by applying the OCR software. Finally, the digitized data are arranged into one *Excel* table using the topology information. We implement the proposed algorithm on *Python* using *OpenCV* for the image processing library and *Tesseract* for the OCR software.

For evaluations, we applied the proposal to 19 scanned and 17 screenshotted table images. The results show that for any image, the *Excel* file is generated with the correct structure, and some characters are misrecognized by the adopted OCR software. The improvement will be in future works.

The rest of this paper is organized as follows: Section 2 discusses related works in the literature. Section 3 introduces the adopted software in this paper. Section 4 describes the target table layout printed on paper. Section 5 presents the proposed algorithm. Section 6 evaluates the proposal. Finally, Section 7 concludes this paper with future works.

2. Related Works in the Literature

In this section, we introduce some related works in the literature.

In [2], Ohta et al. proposed a method based on the dataset provided by the ICDAR 2013 table competition and evaluation measures based on the adjacency relations between cells. They can be used to estimate implicitly ruled lines for recognizing the table structure. From the results of 156 tables, they found the structure for most tables by integrating the explicit and implicit ruled lines.

In [3], Phan et al. proposed an algorithm to digitize the tables in a document and the articles based on *Cascade R-CNN HRNet*. It can be used to detect and classify tables, and integrate image processing algorithms to improve the table data. The results show that this algorithm can identify the table of hydrological data with an accuracy rate of 98%. The algorithm was proved to be effective on real data of the hydrometeorological stations.

In [4], Sahoo et al. proposed an algorithm to find the tables in a document and the articles based on *machine learning*. It can be used to find tables, including bordered, borderless, or partially bordered tables in PDF files. From the results of extracted tables, they show that the tables extracted from PDF files can be well extracted into Excel files. The results showed that the algorithm is an efficient solution for table detection in diverse documents.

In [5], Amitha et al. proposed a web application based on *optical character recognition (OCR)*. It can be used to convert the attendance register image to Excel conversion. From the results of extracted tables, they showed that this web application can capture an image, upload it, and view the extracted text. The results showed that the web application is helpful for literacy education under different circumstances.

In [6], Nagy et al. proposed an algorithm based on *category hierarchies*, which can be used to convert the web tables into the structured collections of relational tables and *resource definition framework (RDF)* triples. They found that over 1.5 million table cells were classified into ten categories, where among the 1520 tables tested, only 16 were not analyzed. The results showed that the formalization and exploitation of intrinsic structural table constraints open the way for algorithmic conversions of vast amounts of tabulated data to uniform standard data analysis formats.

In [7], Minghao Li et al. proposed a new image-based table detection and recognition dataset built with novel weak supervision from Word and Latex documents on the internet. They used TableBank, which contains 417K high-quality labeled tables, to build several strong baselines using state-of-the-art models with deep neural networks. Experiments show that image-based table detection and recognition with deep learning is a promising research direction.

3. Adopted Technologies

In this section, we introduce two software used in this paper. First, *OpenCV* is used to recognize and extract the tables from the scanned image of a paper containing them, and separate the characters in each table into collection cells. Second, *Tesseract* is used to recognize the characters in these cells. *Python* is used to implement the algorithm, where the digitalized data is imported into *Excel* using the *pandas* library. The two pieces of software must be installed by the user of the algorithm.

3.1. OpenCV

OpenCV (Open Source Computer Vision Library) is the open-source library of programming functions that are mainly aimed at real-time computer vision [8]. In the algorithm implementation, *opencv-python version 4.6.0.66* is adopted to call *OpenCV* functions from the *Python* program. The following four tasks are completed by *OpenCV* functions in this algorithm:

1. Recognize every intersection of the horizontal line and the vertical line to find the location of the cells in the table.
2. Divide the table into a collection of rectangles using the intersections.
3. Arrange the rectangles from left to right and top to bottom by recognizing the boundary of the horizontal and vertical lines.

4. Recognized rectangles sometimes are too large or too small. The rectangle will have multiple cells if it is too large, and it will have none if it is too small. In order to prevent both of these situations, we restrict the rectangle's size range.

3.2. Tesseract

Tesseract is open-source software to recognize the characters in an image file. It can be executed from the command-line interface. For this algorithm, *Tesseract* is imported into *Python* through the *pytesseract* package. After using *OpenCV* to divide the table into a collection of cells, *Tesseract* will recognize the characters in each cell and return the digitalized result.

4. Target Table Layout

In this section, we describe the layout of the target printed table on paper.

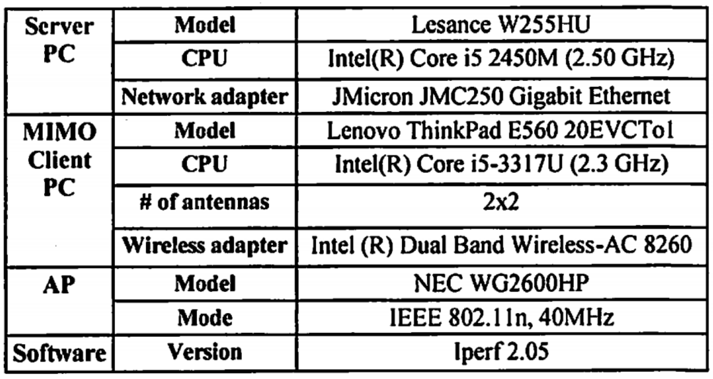
4.1. Table Layout

The proposed algorithm assumes that the table has the following features:

- The table basically has a grid structure of multiple rows and multiple columns.
- Two adjacent horizontal lines make one row, and two adjacent vertical lines make one column in the table.
- All the vertical lines connect the top horizontal line and the bottom horizontal line.
- Some horizontal lines including the top one and the bottom one connect the leftmost vertical line and the rightmost vertical line.
- Other horizontal lines connect the intermediate vertical line and the rightmost vertical line.
- A *cell* represents the rectangle area surrounded by two adjacent horizontal lines and two adjacent vertical lines.
- The number of cells in every row is equal.
- The number of cells in one column may be different by the column.
- A cell has one or multiple cells at the right neighbor and has one cell at the top/ bottom neighbors.

4.2. Example Table

Figure 3 illustrates an example target printed table of the algorithm. This table structure is often used. Four vertical lines make three columns. Five horizontal lines connect the leftmost vertical line and the rightmost one. Six horizontal lines connect the second vertical line and the rightmost one. Thus, the number of cells in the second column and third column is ten.



Server PC	**Model**	**Lesance W255HU**
	CPU	**Intel(R) Core i5 2450M (2.50 GHz)**
	Network adapter	**JMicron JMC250 Gigabit Ethernet**
MIMO Client PC	**Model**	**Lenovo ThinkPad E560 20EVCTo1**
	CPU	**Intel(R) Core i5-3317U (2.3 GHz)**
	# of antennas	**2x2**
	Wireless adapter	**Intel (R) Dual Band Wireless-AC 8260**
AP	**Model**	**NEC WG2600HP**
	Mode	**IEEE 802.11n, 40MHz**
Software	**Version**	**Iperf 2.05**

Figure 3. Example of a target printed table.

4.3. Topology Information

In the proposed algorithm, after dividing the printed table into a collection of cells and recognizing the characters in each cell, the digitized data for each cell are arranged into one table according to the topology information of the cells. This topology information is actually described by a *tree* for each row. The *root* node of the tree represents the cell bounded by the leftmost vertical line and the second vertical line. Then, its *child* node in the tree represents its right neighbor cell in the table. Figure 4 shows the topology information tree for the cells at the first row of the table in Figure 3.

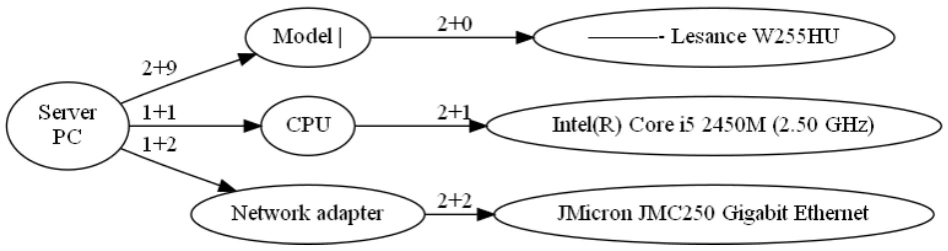


Figure 4. Topology tree for the first row of example table.

In order to obtain the table structure accurately, we use the cells' position, width, and height. The columns in this type of table are exactly the same as in a regular table, so we can classify all the cells into columns by finding their locations. Next, start with the first grid in the leftmost column, and find the child nodes of the grid by determining the height of the column and the latter column of this column. In the next step, find the child nodes of the first cell in the leftmost column by determining the height of the column and the height of the latter column. It can be seen that if the height of the cell in the leftmost column is fuzzy and equal to the height of the two cells in the latter column, they are the child nodes of the cell in the leftmost column. Specific algorithms are described in the next section.

5. Proposed Algorithm

In this section, we present the proposed *printed table digitization algorithm*.

5.1. Symbols in Algorithm

First, we define the symbols for the algorithm.

- I: the scanned color image of the paper.
- I_{bin}: reversed binary image of I.
- k_{ver}: the definition of a vertical kernel for detecting all vertical lines in an image.
- k_{hor}: the definition of a horizontal kernel for detecting all horizontal lines in an image.
- I_{ver}: the binary image with only vertical lines of I.
- L_{ver}: the vertical lines of I_{ver}.
- I_{hor}: the binary image with only horizontal lines of I.
- L_{hor}: the horizontal line of $Ihor$.
- I_{vh}: combine L_{ver} and L_{hor}.
- L_{cnts}: the contours of I_{vh}.
- l_i: each contour of L_{cnts}.
- r_i: the rectangles of l_i.
- R: the array of r_i.
- L_{cnts1}: the sorting result of L_{cnts}.
- R_1: the R after sorting.
- R_2: the array of rectangular which only contains a single cell.
- c_i: each *Cell* class.
- C: the array of c_i
- x_i: the horizontal coordinate of c_i.
- y_i: the vertical coordinates of c_i.
- w_i: the width of c_i.
- h_i: the height of c_i.
- $name_i$: the name of c_i that uses pytesseract to recognize.
- row: the number of rows in I.

- *Root*: the root node of the cardinal tree.
- row_{ci}: the c_i of each column, for example, when $row = 1$, we have $row1_{ci}$, when $row = 2$, we have $row2_{ci}$
- row_{ci+1}: the c_i of each column, for example, $row1_{ci+1}$ is the next class of $row1_{ci}$.
- *Tree*: the structure of I's table.
- E: the output Excel table.

5.2. Procedure in Algorithm

Next, we describe the procedure of the algorithm.

(1) Make the color image I of each table by scanning the paper.
(2) Transform the color image I into the black-and-white binary image I_{bin} by applying the *thresholds* function in *OpenCV* to I.
(3) Reference [9], obtain the vertical kernel k_{ver} and the horizontal kernel k_{hor} by applying the *getStructuringElement* function in *OpenCV* to I_{bin}.
(4) Obtain the binary image I_{ver} that only has vertical lines by applying the *erode* function in *OpenCV* to I_{bin}. Figure 5 shows the binary image with only vertical lines for the printed table in Figure 3.
(5) Extract the vertical lines L_{ver} by applying the *dilate* function in *OpenCV* to I_{ver}.
(6) Reference [9], obtain the binary image I_{hor} that only has horizontal lines by applying the *erode* function in *OpenCV* to I_{bin}. Figure 6 shows the binary image with only horizontal lines for the printed table in Figure 3.
(7) Extract the horizontal lines L_{hor} by applying the *dilate* function in *OpenCV* to I_{hor}.
(8) Make the binary image I_{vh} that represents the table frame by combining the vertical lines L_{ver} and the horizontal lines L_{hor} into one image, applying the *addWeighted* function in *OpenCV* to L_{ver} and L_{hor}. Figure 7 shows the table frame image.
(9) Obtain the contour information of each cell in the table and store them in the array L_{cnts} by applying the *findContours* function in *OpenCV* to I_{vh}.
(10) Find the rectangular r_i for each contour l_i of L_{cnts} by applying the *boundingRect* function in *OpenCV* to every contour of L_{cnts}, and make the array R of r_i.
(11) Sort r_i in R from the leftmost one to the rightmost one according to the coordinates of r_i by applying the *zip* method and the *sort* method in *Python* to R, and make the array L_{cnts1} and the array R_1.
(12) Remove all excessively large or small rectangles in R_1, to avoid rectangles that contain multiple cells or have no cells. Produce the R_2 with every rectangle containing a cell.
(13) Create *Cell*. Create class c_i for each rectangle in R_2, collect x_i, y_i, and w_i of each rectangle in R_2, and put them into c_i as member variables.
(14) Obtain the character recognition result $name_i$ by applying the *imagetostring* function in *pytesseract* to each cell.
(15) Sort the c_i, in which the x_i and y_i of each c_i are arranged from left to right and from top to bottom. Arrange c_i with similar x_i into a column in the order of y_i from small to large to output row_{ci}
(16) Create the multi-tree class *Tree* to describe the structure of the table with the root node *Root*. The child nodes of this root node are all c_i with a row equal to 1.
(17) Confirm the child node of $row2_{ci}$, if $row1_{ci}$'s h_i is approximately equal to $row2_{ci}$'s h_i, the child node of $row1_{ci}$ is $row2_{ci}$. Otherwise, let $row2_{ci}$'s h_i plus $row2_{ci+1}$'s h_{i+1}, and make the sum of $row2_{ci}$'s h_i and $row2_{ci+1}$'s h_{i+1} compare with $row1_{ci}$'s h_i. If the two are approximately equal, the child node of $row1_{ci}$ is $row2_{ci}$ and $row2_{ci+1}$. If the two are not approximately equal, it can be done again in the above way.
(18) Follow the previous step to get the child node of $row2_{ci}$, the child node of $row2_{ci}$, up to the last column, and obtain the tree *Tree* composed of c_i, which represents the structure of the table cell structure. Figure 8 shows the tree.
(19) Preorder the *Tree* and put each c_i's $name_i$ in the leaf node's $name_i$ in the order from the root node to the leaf node. Each leaf node's $name_i$ contains one row of the table, and then output each leaf node's $name_i$ in order to get the list of table information.

(20) Convert the list to the *Excel* table *E* using *openpyxl* package and output it. Figure 9 shows this Excel file.

(21) Merge the cells corresponding to the same node in the tree in step (18) into one cell in the *Excel* table *E* using the *openpyxl* package, and output it. Figure 10 shows the final output Excel file.

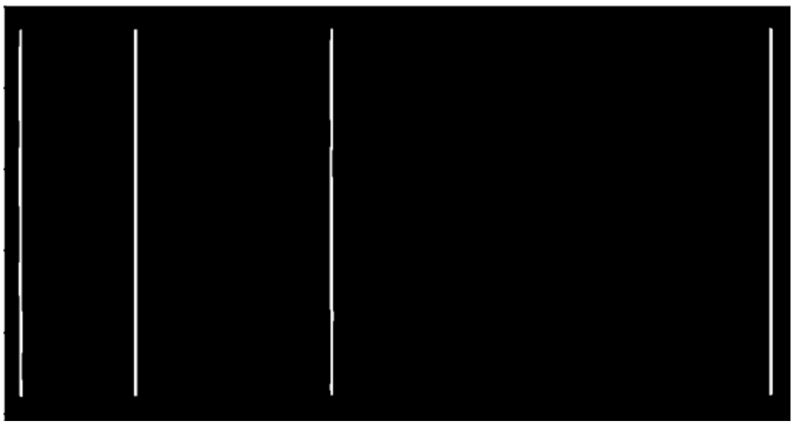


Figure 5. Binary image with vertical lines.

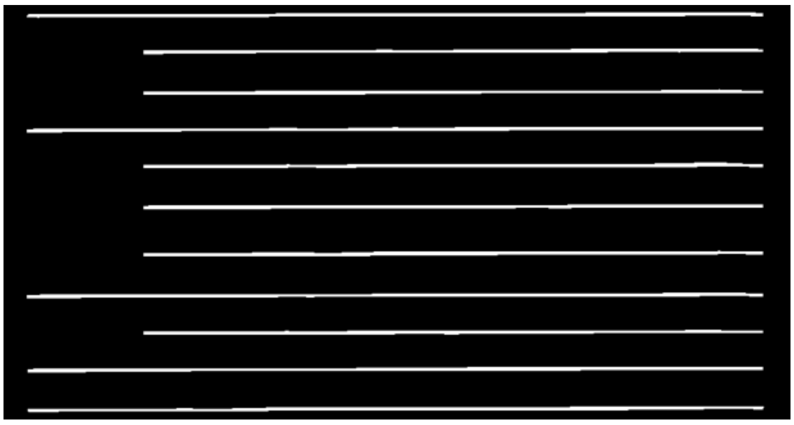


Figure 6. Binary image with horizontal lines.

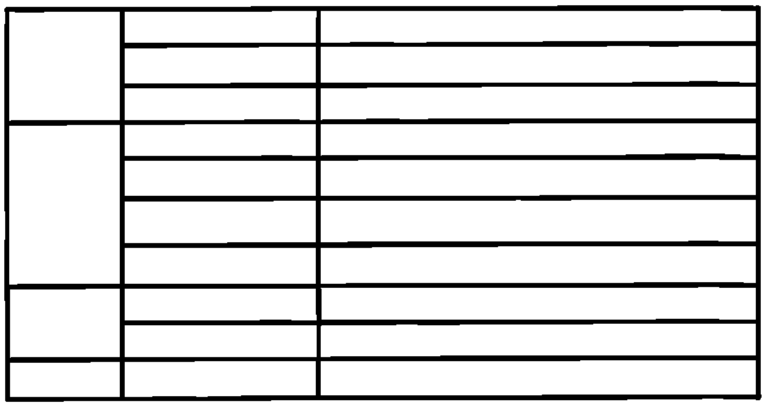


Figure 7. Table frame image.

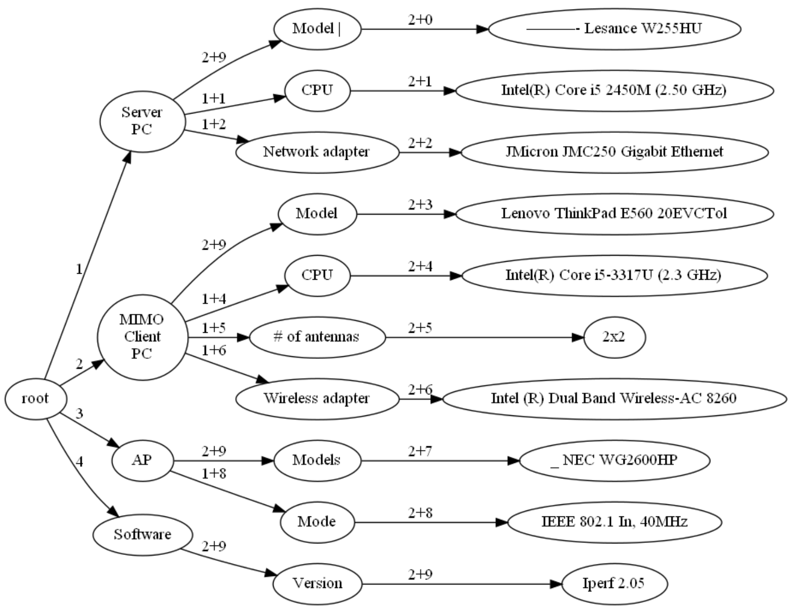
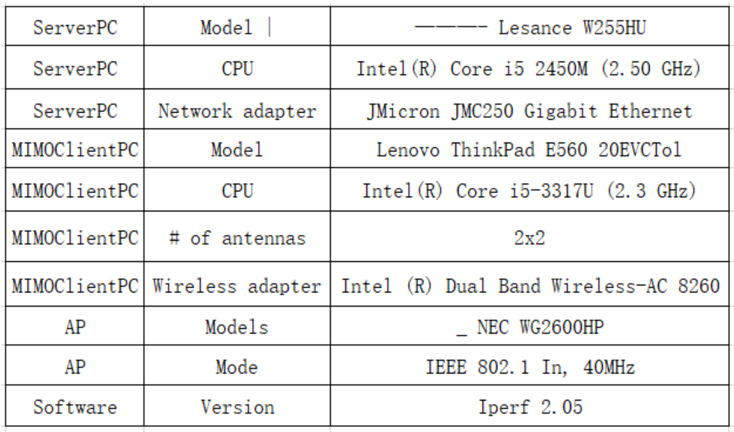
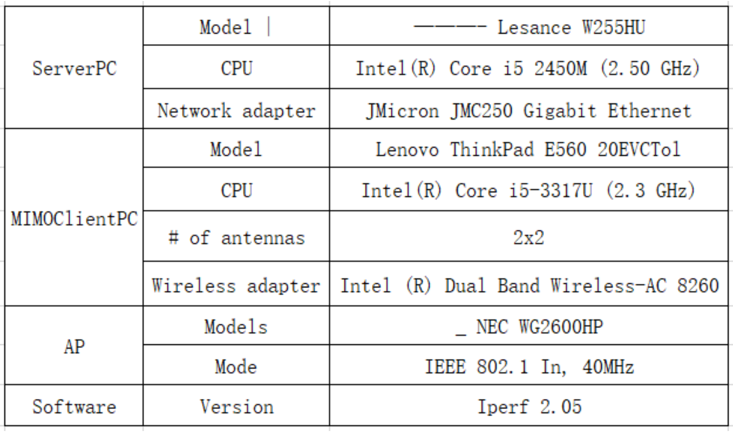


Figure 8. Tree structure of the table.



ServerPC	Model |	——— Lesance W255HU
ServerPC	CPU	Intel(R) Core i5 2450M (2.50 GHz)
ServerPC	Network adapter	JMicron JMC250 Gigabit Ethernet
MIMOClientPC	Model	Lenovo ThinkPad E560 20EVCTol
MIMOClientPC	CPU	Intel(R) Core i5-3317U (2.3 GHz)
MIMOClientPC	# of antennas	2x2
MIMOClientPC	Wireless adapter	Intel (R) Dual Band Wireless-AC 8260
AP	Models	_ NEC WG2600HP
AP	Mode	IEEE 802.1 In, 40MHz
Software	Version	Iperf 2.05

Figure 9. Excel table before merging the same cells.



ServerPC	Model |	——— Lesance W255HU
	CPU	Intel(R) Core i5 2450M (2.50 GHz)
	Network adapter	JMicron JMC250 Gigabit Ethernet
MIMOClientPC	Model	Lenovo ThinkPad E560 20EVCTol
	CPU	Intel(R) Core i5-3317U (2.3 GHz)
	# of antennas	2x2
	Wireless adapter	Intel (R) Dual Band Wireless-AC 8260
AP	Models	_ NEC WG2600HP
	Mode	IEEE 802.1 In, 40MHz
Software	Version	Iperf 2.05

Figure 10. Excel table after merging the same cells.

6. Evaluations

In this section, we evaluate the proposed algorithm by applying it to 19 scanned table images and 17 screenshotted table images that are obtained from websites. They satisfy the table layout features in Section 4.1.

6.1. Accuracy Index

To evaluate the accuracy of the proposal, we consider the rate of correctly recognizing the table cell structure and the rate of correctly recognizing the characters in each cell.

For the former index of recognizing table structure, the proposed algorithm achieved the 100% accuracy. It recognized the cell structure correctly for any image.

For the latter index of recognizing characters, the rate p is calculated by the following equation:

$$p = \frac{ch_1}{ch_2} \times 100\% \quad (1)$$

where ch_1 represents the total number of characters in the tested table images except for spaces and tabs, and ch_2 does the total number of correctly recognized characters.

Because it is a very fuzzy concept to identify the quality of a table, a core comparison value is needed to determine the quality of the table identified by the algorithm.

In this paper, the evaluation standard of our algorithm is p. The closer p is to 100%, the better the recognition result is. The closer p is to 0%, the worse the recognition result is. In this paper, we believe that forms with $p \geq 80\%$ have a qualified recognition effect.

The output result of the algorithm is only related to the algorithm itself and the input table. Therefore, we control the algorithm unchanged, only change the input table, and observe the relationship between the input table and p.

6.2. Character Recognition Accuracy Results

Figures 11 and 12 show the character recognition rates for the 19 scanned table images and the 17 screenshotted table images, respectively. From them, the rates for the scanned table images at 2, 5, 10, 16, 18, and 19 are lower than 90%. We will analyze the reasons.

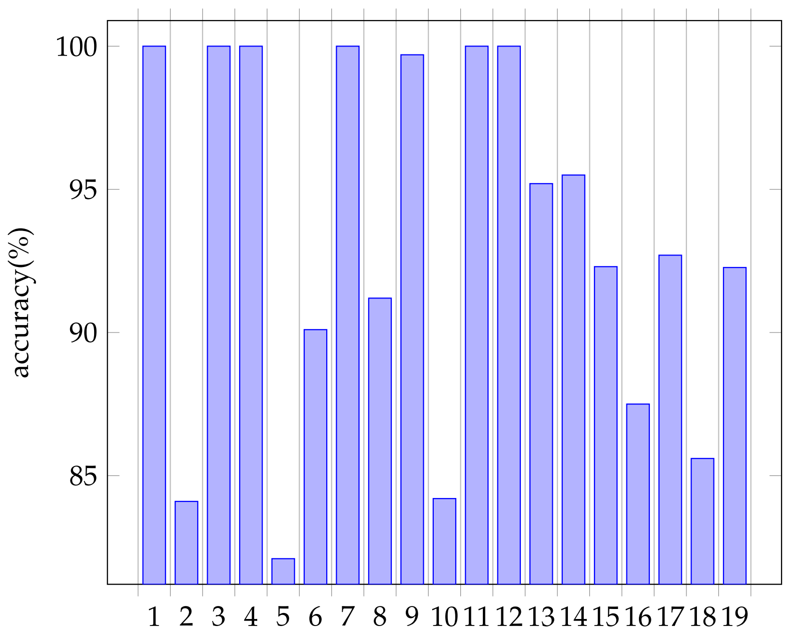


Figure 11. Character recognition rates for scanned table images.

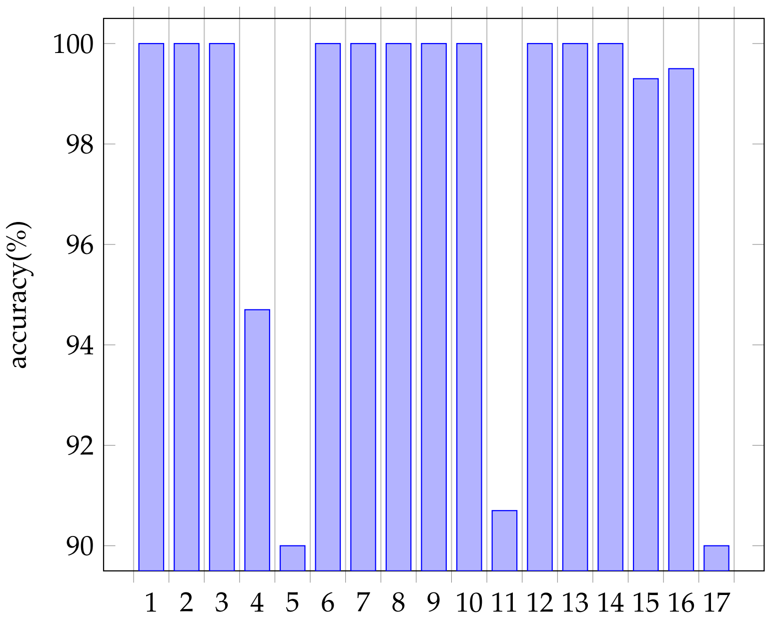


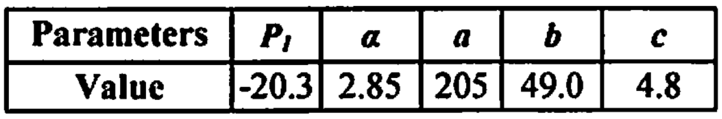
Figure 12. Character recognition rates for screenshotted table images.

Here, we discuss reasons why the character recognition rates for the scanned table images at 2, 5, 10, 16, and 18 are lower than 90%.

6.2.1. Table Image at 2

Figures 13–15 show the scanned table image, the table structure by the algorithm, and the output Excel table. The character recognition rate is given by:

$$p = \frac{\frac{10}{10} + \frac{1}{2} + \frac{0}{1} + \frac{1}{1} + \frac{1}{1} + \frac{1}{1} + \frac{5}{5} + \frac{5}{5} + \frac{4}{4} + \frac{3}{3} + \frac{4}{4} + \frac{2}{3}}{12} \times 100\% =\approx 84.66\%. \quad (2)$$



Parameters	P_l	α	a	b	c
Value	-20.3	2.85	205	49.0	4.8

Figure 13. Printed table of scanned table image 1.

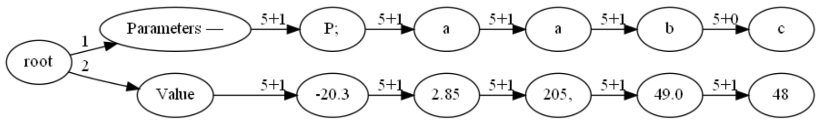
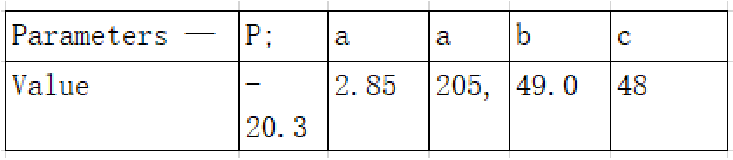


Figure 14. Tree for scanned table image 1.



Parameters —	P;	a	a	b	c
Value	– 20. 3	2. 85	205,	49. 0	48

Figure 15. Excel output for scanned table image 1 by our algorithm.

This table contains several subscripts, decimal points, and Greek letters, which are not correctly recognized by *Tesseract* OCR software.

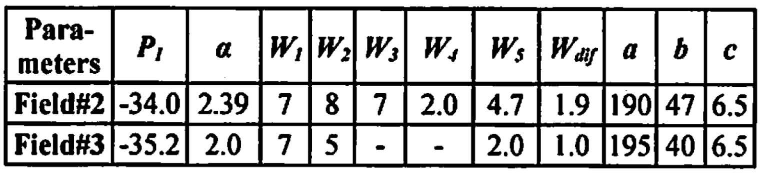
6.2.2. Reasons for Table Image at 5

Figures 16–18 show the scanned table image, the table structure by the algorithm, and the output Excel table.

The character recognition rate is given by:

$$p = \frac{\frac{10}{10} + \frac{1}{2} + \frac{0}{1} + \frac{1}{2} + \frac{1}{2} + \frac{1}{2} + \frac{1}{2} + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{3}{3} + \frac{2}{2} + \frac{3}{3}}{36} \times 100\% \tag{3}$$

$$p \approx 82.13\% \tag{4}$$



Para-meters	P_1	α	W_1	W_2	W_3	W_4	W_5	W_{dif}	a	b	c
Field#2	-34.0	2.39	7	8	7	2.0	4.7	1.9	190	47	6.5
Field#3	-35.2	2.0	7	5	-	-	2.0	1.0	195	40	6.5

Figure 16. Printed table of scanned table image 2.

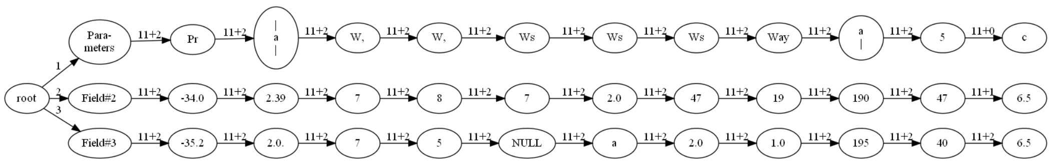
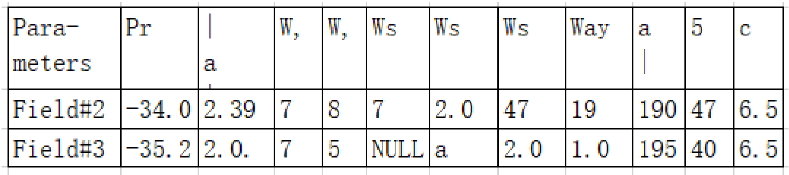


Figure 17. Tree for scanned table image 2.



Para-meters	Pr	| a	W,	W,	Ws	Ws	Ws	Way	a |	5	c
Field#2	-34. 0	2. 39	7	8	7	2. 0	47	19	190	47	6. 5
Field#3	-35. 2	2. 0.	7	5	NULL	a	2. 0	1. 0	195	40	6. 5

Figure 18. Excel output for scanned table image 2 by our algorithm.

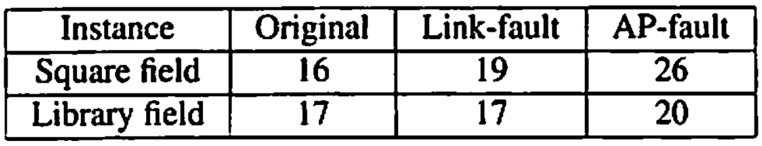
This table contains several subscripts, decimal points, and Greek letters, which are not correctly recognized by *Tesseract* OCR software.

6.2.3. Reasons for Table Image at 10

Figures 19–21 show the scanned table image, the table structure by the algorithm, and the output Excel table. The character recognition rate is given by:

$$p = \frac{\frac{8}{8} + \frac{8}{8} + \frac{9}{9} + \frac{7}{7} + \frac{11}{11} + \frac{1}{2} + \frac{2}{2} + \frac{1}{2} + \frac{12}{12} + \frac{0}{2} + \frac{2}{2} + \frac{2}{2}}{12} \times 100\% \tag{5}$$

$$p \approx 83.33\% \tag{6}$$



Instance	Original	Link-fault	AP-fault
Square field	16	19	26
Library field	17	17	20

Figure 19. Printed table of scanned table image 3.

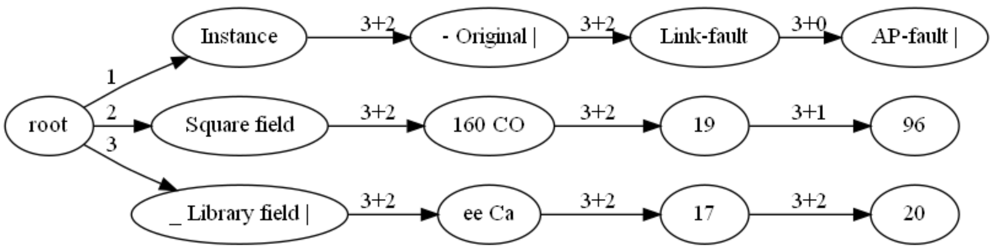
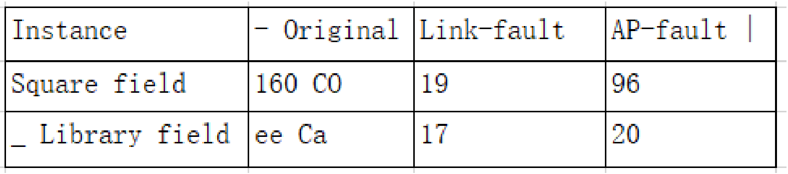


Figure 20. Tree for scanned table image 3.



Instance	- Original	Link-fault	AP-fault |
Square field	160 CO	19	96
_ Library field	ee Ca	17	20

Figure 21. Excel output for scanned table image 3 by our algorithm.

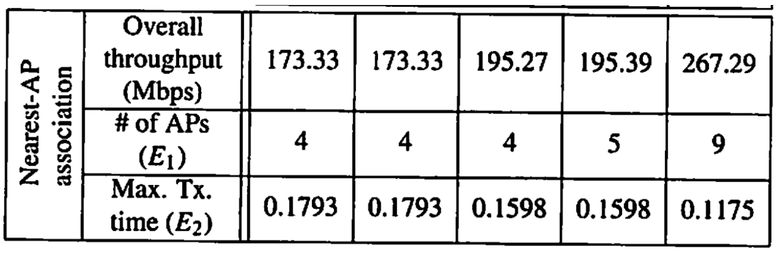
This table contains several cells which contain characters that are very close to the cell edge, which are not correctly recognized by *Tesseract* OCR software.

6.2.4. Reasons for Table Image at 16

Figures 22–24 show the scanned table image, the table structure by the algorithm, and the output Excel table. The character recognition rate is given by:

$$p = \frac{\frac{0}{20} + \frac{6}{6} + \frac{6}{6} + \frac{6}{6} + \frac{6}{6} + \frac{6}{6} + \cdots + \frac{11}{12} + \frac{6}{6} + \frac{6}{6} + \frac{6}{6} + \frac{6}{6} + \frac{6}{6}}{19} \times 100\% \tag{7}$$

$$p \approx 83.73\% \tag{8}$$



Nearest-AP association	Overall throughput (Mbps)	173.33	173.33	195.27	195.39	267.29
	# of APs (E_1)	4	4	4	5	9
	Max. Tx. time (E_2)	0.1793	0.1793	0.1598	0.1598	0.1175

Figure 22. Printed table of scanned table image 4.

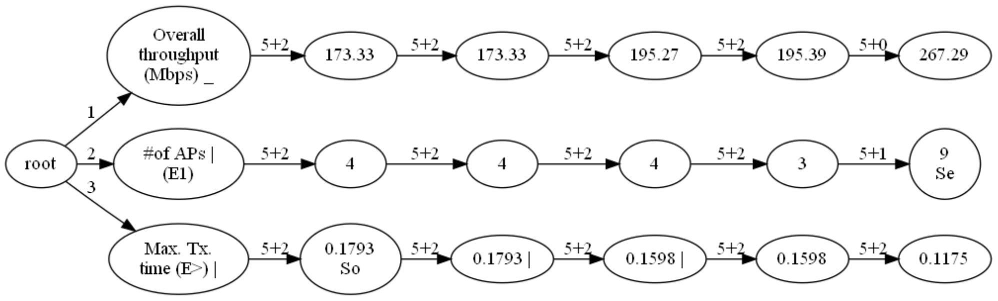
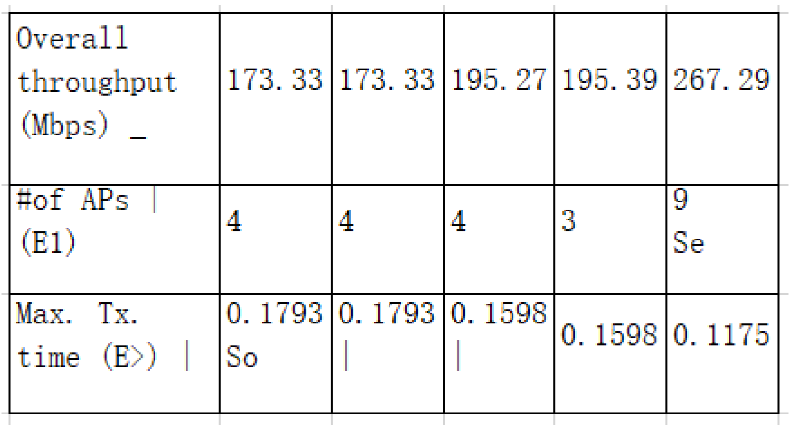


Figure 23. Tree for scanned table image 4.



Overall throughput (Mbps) _	173. 33	173. 33	195. 27	195. 39	267. 29
#of APs | (E1)	4	4	4	3	9 Se
Max. Tx. time (E>) |	0. 1793 So	0. 1793 |	0. 1598 |	0. 1598	0. 1175

Figure 24. Excel output for scanned table image 4 by our algorithm.

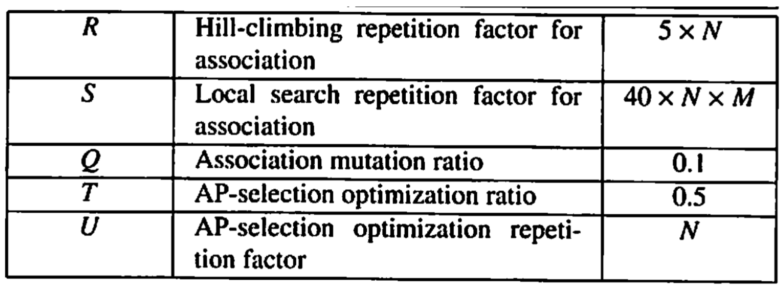
This table contains several subscripts and vertical characters, which are not correctly recognized by *Tesseract* OCR software.

6.2.5. Reasons for Table Image at 18

Figures 25–27 show the scanned table image, the table structure by the algorithm, and the output Excel table. The character recognition rate is given by:

$$p = \frac{\frac{1}{1} + \frac{42}{42} + \frac{2}{3} + \frac{41}{41} + \frac{6}{6} + \frac{1}{1} + \frac{24}{24} + \frac{2}{3} + \cdots + \frac{1}{1} + \frac{29}{29} + \frac{39}{39} + \frac{1}{1}}{15} \times 100\% \tag{9}$$

$$p \approx 86.53\% \tag{10}$$



R	**Hill-climbing repetition factor for association**	**$5 \times N$**
S	**Local search repetition factor for association**	**$40 \times N \times M$**
Q	**Association mutation ratio**	**0.1**
T	**AP-selection optimization ratio**	**0.5**
U	**AP-selection optimization repeti-tion factor**	***N***

Figure 25. Printed table of scanned table image 5.

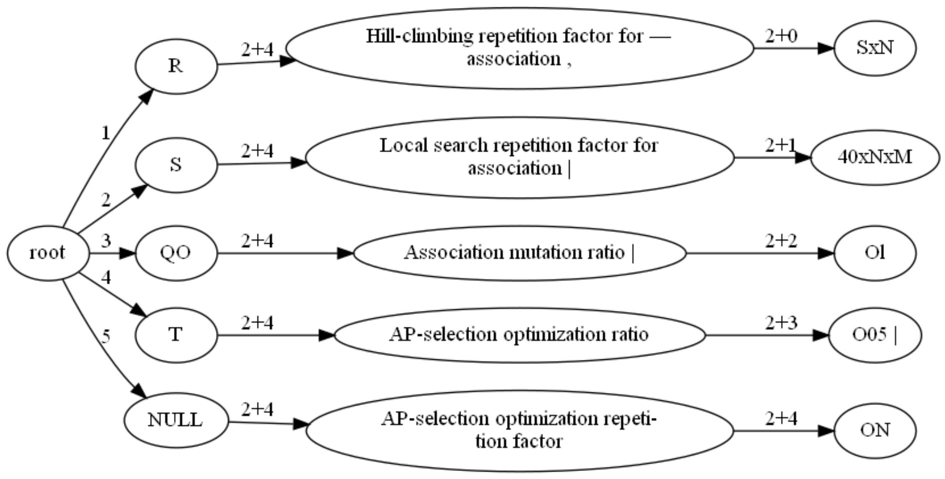
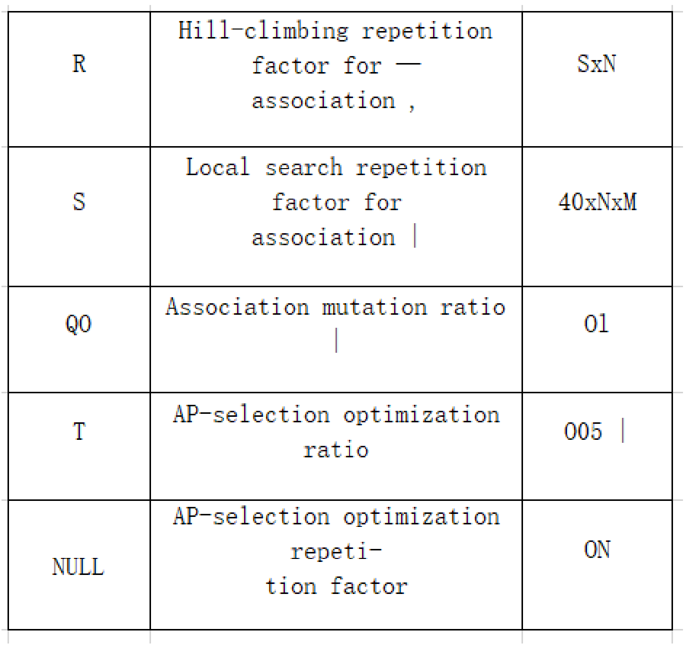

root
1
2
3
4
5
R
S
QO
T
NULL
2+4
2+4
2+4
2+4
2+4
Hill-climbing repetition factor for — association ,
Local search repetition factor for association |
Association mutation ratio |
AP-selection optimization ratio
AP-selection optimization repeti-tion factor
2+0
2+1
2+2
2+3
2+4
SxN
40xNxM
OI
O05 |
ON

Figure 26. Tree for scanned table image 5.



R	Hill-climbing repetition factor for — association ,	SxN
S	Local search repetition factor for association |	40xNxM
QO	Association mutation ratio |	01
T	AP-selection optimization ratio	005 |
NULL	AP-selection optimization repeti- tion factor	ON

Figure 27. Excel output for scanned table image 5 by our algorithm.

This table contains several cells which contain characters that are very close to the cell edge, which are not correctly recognized by *Tesseract* OCR software.

6.3. Conclusion of This Section

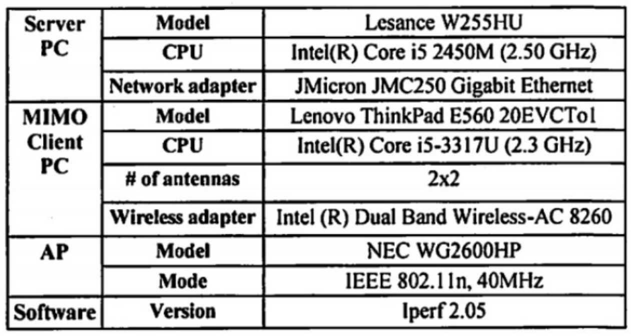
Through applications, we found the following weakness of the adopted OCR software *Tesseract*:

- When a cell size is small, it often recognizes the border as a character.
- It is difficult to correctly recognize vertical texts, subscripts, and Greek letters.
- It is more difficult to recognize a scanned image than a screenshot because it is usually blurry and has more noise.

6.4. Comparison with Microsoft Software

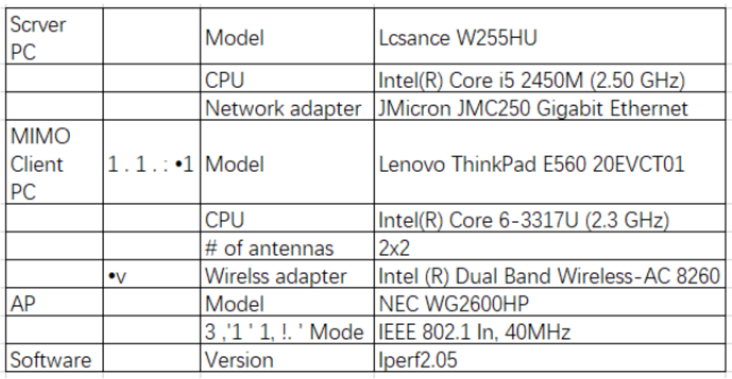
Microsoft offers software for the same purpose as our algorithm for Excel users [10,11]. Therefore, we applied this software to the same set of table images. Then, we found that the table cell structure was not correctly recognized in some images. For example, for the image in Figure 28 it obtains the result in Figure 29, which is the regular grid structure and has unnecessary cells. On the other hand, our algorithm obtains the correct structure in Figure 30.

When using the Office application, we also found that the software could not recognize tables as complex as in Figure 31. In the future, we will also try to make this algorithm directly identify this type of table.



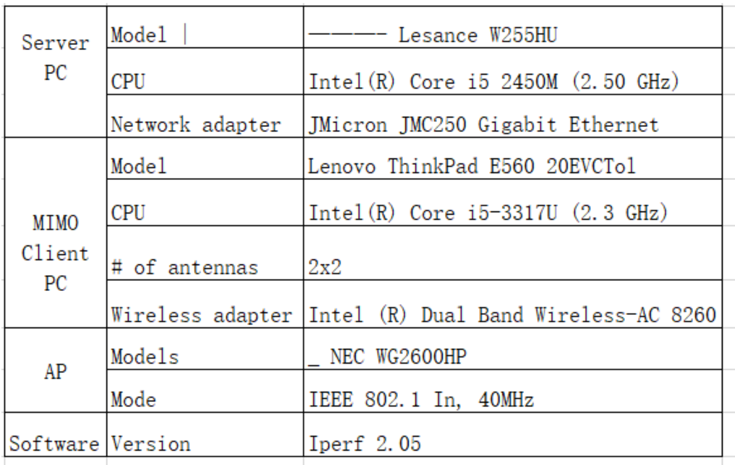
Server PC	**Model**	Lesance W255HU
	CPU	Intel(R) Core i5 2450M (2.50 GHz)
	Network adapter	JMicron JMC250 Gigabit Ethernet
MIMO Client PC	**Model**	Lenovo ThinkPad E560 20EVCTo1
	CPU	Intel(R) Core i5-3317U (2.3 GHz)
	# of antennas	2x2
	Wireless adapter	Intel (R) Dual Band Wireless-AC 8260
AP	**Model**	NEC WG2600HP
	Mode	IEEE 802.11n, 40MHz
Software	**Version**	Iperf 2.05

Figure 28. Printed table of scanned table image 6.



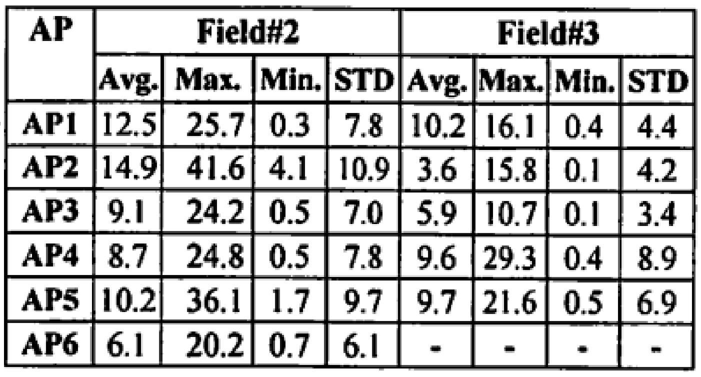
Scrver PC		Model	Lcsance W255HU
		CPU	Intel(R) Core i5 2450M (2.50 GHz)
		Network adapter	JMicron JMC250 Gigabit Ethernet
MIMO Client PC	1 . 1 . : •1	Model	Lenovo ThinkPad E560 20EVCT01
		CPU	Intel(R) Core 6-3317U (2.3 GHz)
		# of antennas	2x2
	•v	Wirelss adapter	Intel (R) Dual Band Wireless-AC 8260
AP		Model	NEC WG2600HP
		3 ,'1 ' 1, !. ' Mode	IEEE 802.1 In, 40MHz
Software		Version	Iperf2.05

Figure 29. Excel output for scanned table image 6 by Microsoft Office.



Server PC	Model |	———— Lesance W255HU
	CPU	Intel(R) Core i5 2450M (2. 50 GHz)
	Network adapter	JMicron JMC250 Gigabit Ethernet
MIMO Client PC	Model	Lenovo ThinkPad E560 20EVCTol
	CPU	Intel(R) Core i5-3317U (2. 3 GHz)
	# of antennas	2x2
	Wireless adapter	Intel (R) Dual Band Wireless-AC 8260
AP	Models	_ NEC WG2600HP
	Mode	IEEE 802. 1 In, 40MHz
Software	Version	Iperf 2. 05

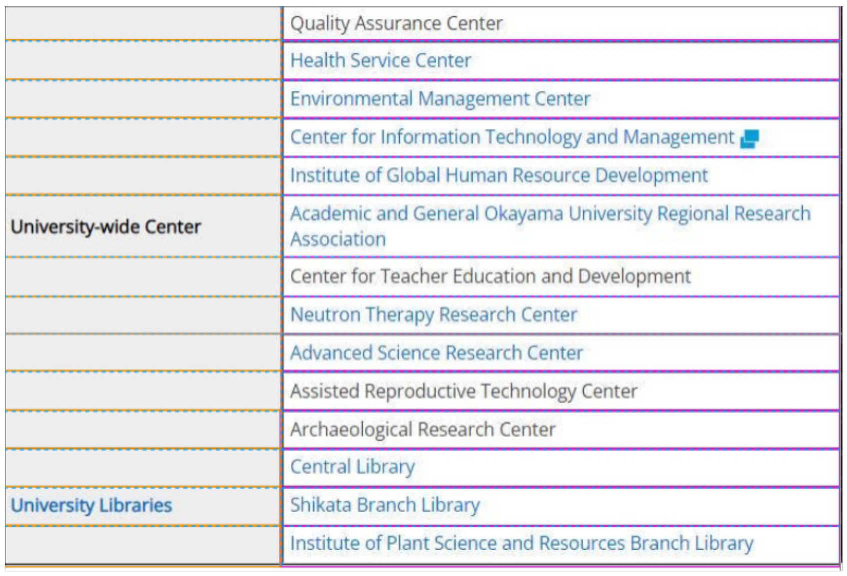
Figure 30. Excel output for scanned table image 6 by our algorithm.



AP	Field#2				Field#3			
	Avg.	Max.	Min.	STD	Avg.	Max.	Min.	STD
AP1	12.5	25.7	0.3	7.8	10.2	16.1	0.4	4.4
AP2	14.9	41.6	4.1	10.9	3.6	15.8	0.1	4.2
AP3	9.1	24.2	0.5	7.0	5.9	10.7	0.1	3.4
AP4	8.7	24.8	0.5	7.8	9.6	29.3	0.4	8.9
AP5	10.2	36.1	1.7	9.7	9.7	21.6	0.5	6.9
AP6	6.1	20.2	0.7	6.1	-	-	-	-

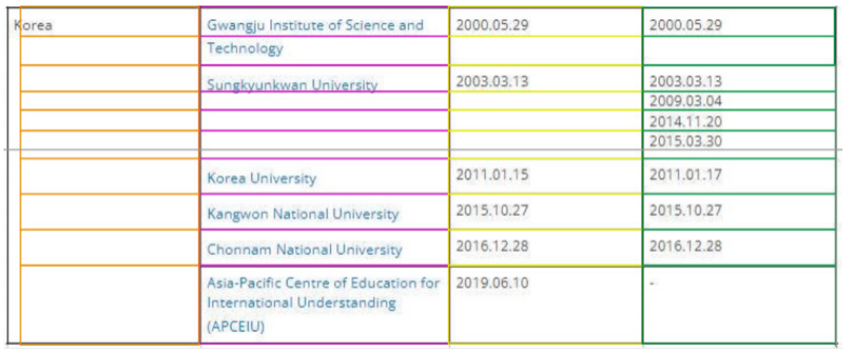
Figure 31. Printed table of the scanned table that Microsoft Office cannot recognize.

- Too many dotted lines are generated as in Figure 32.
- The location of the generated box is different from the original as in Figures 33–35.
- The dotted line is generated around the center of a cell as in Figure 36.



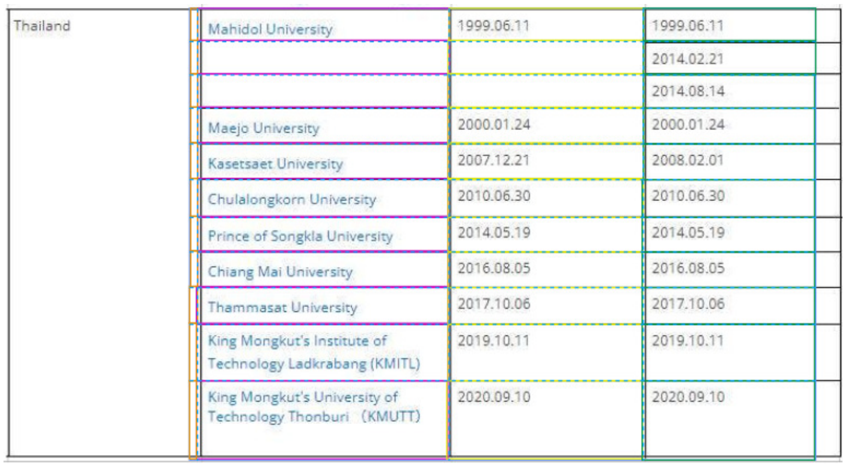
	Quality Assurance Center
	Health Service Center
	Environmental Management Center
	Center for Information Technology and Management
	Institute of Global Human Resource Development
University-wide Center	Academic and General Okayama University Regional Research Association
	Center for Teacher Education and Development
	Neutron Therapy Research Center
	Advanced Science Research Center
	Assisted Reproductive Technology Center
	Archaeological Research Center
	Central Library
University Libraries	Shikata Branch Library
	Institute of Plant Science and Resources Branch Library

Figure 32. Result with too many dotted lines.



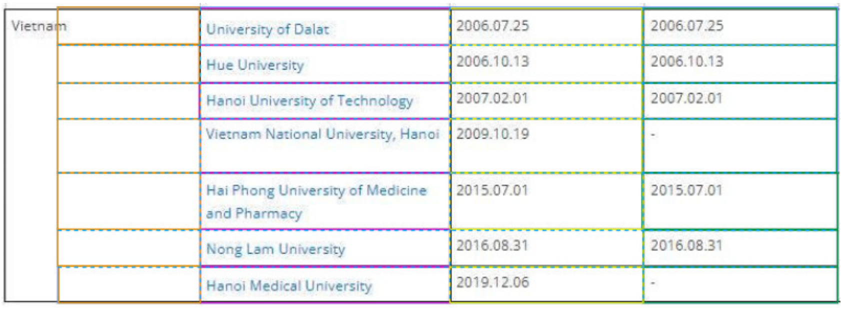
Korea	Gwangju Institute of Science and Technology	2000.05.29	2000.05.29
	Sungkyunkwan University	2003.03.13	2003.03.13
			2009.03.04
			2014.11.20
			2015.03.30
	Korea University	2011.01.15	2011.01.17
	Kangwon National University	2015.10.27	2015.10.27
	Chonnam National University	2016.12.28	2016.12.28
	Asia-Pacific Centre of Education for International Understanding (APCEIU)	2019.06.10	-

Figure 33. Result with wrong box locations (1).



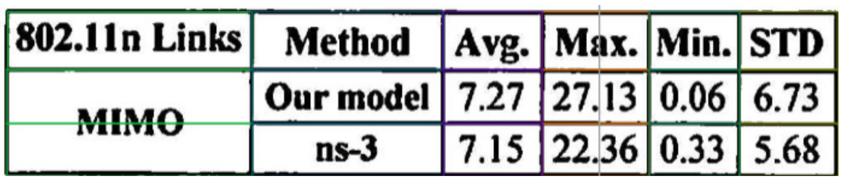
Thailand	Mahidol University	1999.06.11	1999.06.11
			2014.02.21
			2014.08.14
	Maejo University	2000.01.24	2000.01.24
	Kasetsaet University	2007.12.21	2008.02.01
	Chulalongkorn University	2010.06.30	2010.06.30
	Prince of Songkla University	2014.05.19	2014.05.19
	Chiang Mai University	2016.08.05	2016.08.05
	Thammasat University	2017.10.06	2017.10.06
	King Mongkut's Institute of Technology Ladkrabang (KMITL)	2019.10.11	2019.10.11
	King Mongkut's University of Technology Thonburi （KMUTT)	2020.09.10	2020.09.10

Figure 34. Result with wrong box locations (2).



Vietnam	University of Dalat	2006.07.25	2006.07.25
	Hue University	2006.10.13	2006.10.13
	Hanoi University of Technology	2007.02.01	2007.02.01
	Vietnam National University, Hanoi	2009.10.19	-
	Hai Phong University of Medicine and Pharmacy	2015.07.01	2015.07.01
	Nong Lam University	2016.08.31	2016.08.31
	Hanoi Medical University	2019.12.06	-

Figure 35. Result with wrong box locations (3).



802.11n Links	Method	Avg.	Max.	Min.	STD
MIMO	Our model	7.27	27.13	0.06	6.73
	ns-3	7.15	22.36	0.33	5.68

Figure 36. Unnecessary dotted line around the cell center.

6.5. Comparison with Nanonets Software

Here, we considered another table recognition software by Nanonets in [12]. This software generates the dotted lines from the lines in the table images and recognizes the cell boxes from them. We applied it to the same table images. Then, we found that the accuracy of this software is not sufficient, and the following problems appeared:

7. Conclusions

This paper presented the *printed table digitization algorithm* using image processing techniques and an *optical character recognition (OCR)* software. For evaluations, the proposal was applied to 19 scanned and 17 screenshotted table images. It was confirmed that the table structure was correctly recognized for all of them, whereas some characters were misrecognized by the adopted OCT software. In future works, we will improve the character recognition accuracy by refining the OCR software, extending the proposal to printed tables that have more complex structures, and applying it to various table images.

Author Contributions: Conceptualization, C.S. and N.F.; methodology, C.S.; software, C.S.; validation, N.F., C.S. and Y.H.; formal analysis, C.S. and M.M.; investigation, N.F., K.S. and T.T.; resources, C.S.; data curation, C.S.; writing—original draft preparation, C.S.; writing—review and editing, N.F.; visualization, C.S.; supervision, N.F.; project administration, N.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data available on request from the authors.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. What is Digital Transformation (DX)? Available online: https://www.netapp.com/devops-solutions/what-is-digital-transformation/ (accessed on 2 July 2022).
2. Ohta, M.; Yamada, R.; Kanazawa, T.; Takasu, A. A cell-detection-based table-structure recognition method. In Proceedings of the ACM Symposium on Document Engineering 2019, Berlin, Germany, 23–26 September 2019.
3. Phan, H.H.; Dai Duong, N. An Integrated Approach for Table Detection and Structure Recognition. *J. Inf. Technol. Commun.* **2021**, *1*, 41–50. [CrossRef]
4. Sahoo, R.; Kathale, C.; Kubal, M.; Malik, S. Auto-Table-Extract: A System To Identify And Extract Tables From PDF To Excel. *Int. J. Sci. Technol. Res.* **2020**, *9*, *217* .
5. Amitha, S.; Mithun, M.; Chandana, P.C.; Borkakoty, M.; Adithya, U. Conversion of Image To Excel Using Ocr Technique. *Int. Res. J. Mod. Eng. Technol. Sci.* **2020**, *4*, 3743–3747.

6. Nagy, G.; Seth, S. Table headers: An entrance to the data mine. In Proceedings of the 2016 IEEE 23rd International Conference on Pattern Recognition (ICPR), Cancun, Mexico, 4–8 December 2016.
7. Li, M.; Cui, L.; Huang, S.; Wei, F.; Zhou, M.; Li, Z. Tablebank: A benchmark dataset for table detection and recognition. *arXiv* **2019**, arXiv:1903.01949.
8. Pulli, K.; Baksheev, A.; Kornyakov, K.; Eruhimov, V. Realtime Computer Vision with OpenCV. *Queue* **2012**, *10*, 40–56. [CrossRef]
9. A Table Detection, Cell Recognition and Text Extraction Algorithm to Convert Tables in Images to Excel Files. Available online: https://towardsdatascience.com/ (accessed on 10 July 2022).
10. How to Convert a Table of Paper Data to Excel Data? We Can Scan Tables by Use Office Application in Smartphone. Available online: https://dekiru.net/article/21950/ (accessed on 10 September 2022).
11. Scan and Edit a Document. Available online: https://support.microsoft.com/en-us/office/scan-and-edit-a-document-7a07a4bd-aca5-4ec5-ba73-4589ac8b9eed (accessed on 10 September 2022).
12. Table Detection, Table Extraction & Information Extraction Using DL. Available online: https://nanonets.com/blog/table-extraction-deep-learning/ (accessed on 10 September 2022).