Word Extraction from Table Regions in Document Images

Chang-Bu Jeong[1], Sang-Cheol Park[2], Hwa-Jeong Son[2], and Soo-Hyung Kim[2]

[1] Department of Internet Software, Honam University,
59-1 Sebong-dong, Gwangsan-gu, Gwangju 506-714, Korea
cbjeong@honam.ac.kr
[2] Department of Computer Science, Chonnam National University,
300 YongBong-dong, Buk-Gu, Gwangju 500-757, Korea
{sanchun, sonhj}@iip.chonnam.ac.kr, shkim@chonnam.ac.kr

Abstract. This paper describes a method to extract words from table regions in document images. The proposed approach consists of two stages: cell detection and word extraction. In the cell detection module, a table frame is extracted first by analyzing connected components and then intersection points are detected by a method using masks in the table frame. We correct false intersections, and detect the location of the cells within the table. In the word extraction module, a text region in each cell is located by using the connected components information that was obtained during the cell extraction module, and segmented into text lines by using projection profiles. Finally we divide the segmented lines into words using gap clustering and special symbol detection. The method correctly included character components touching the table frame with words, so experimental results show that more than 99% of words were successfully extracted from table regions.

1 Introduction

With the continuous development of computer technology and the Internet environment, we can efficiently produce, store, process, and transmit document images. However, as more and more documents are stored in the image format for full-text image retrieval services or digital libraries, it is impossible to retrieve information from document images with text-based search engines that do not recognize text from images. The retrieval of relevant documents is usually based on indices (e.g. title, author, keyword, and so on) of pre-catalogued documents in the text format, so that it becomes necessary for the user to download part of or the entire document images and confirm the contents when retrieving relevant documents with information not specified in the indices. Over the past few years, a considerable number of studies have focused on keyword spotting through automatic indexing of document images to compensate for this drawback. Such a keyword spotting approach makes it possible to retrieve relevant word images regardless of the language of the document and character segmentation errors because it uses word image features [1-3].

There are a lot of technologies in the document image processing for keyword spotting, such as skew correction, layout analysis, word extraction, and etc. Jeong et

E.A. Fox et al. (Eds.): ICADL 2005, LNCS 3815, pp. 214 – 223, 2005.

al. [4] presented a document image preprocessing system for keyword spotting, which segments and classifies a document image into text regions and non-text regions (table, figure, etc) and then performs word extraction to decompose words from the text regions. Therefore the system is limited in that words in the non-text regions cannot be extracted. The words in the table regions, however, may be more useful for keyword spotting than words from other non-text regions because they may contain more meaningful words. Thus, it is necessary to decompose words from table regions in this respect.

In this paper, we propose a method to extract words from table regions in document images. The proposed approach consists of two stages: cell detection and word extraction (Fig. 1). In the cell detection module, a table frame is extracted first by analyzing CCs (connected components), and then intersection points are detected by a method using masks in the table frame. We correct false intersections using the correlation between neighboring intersections, and use the information of intersections to extrapolate the location of the cells within the table. In the word extraction module, a text region in each cell is located by using the CCs information that was obtained during the cell extraction module, and segmented into text lines by using projection profiles. Finally we divide the segmented lines into words using gap clustering and special symbol detection.

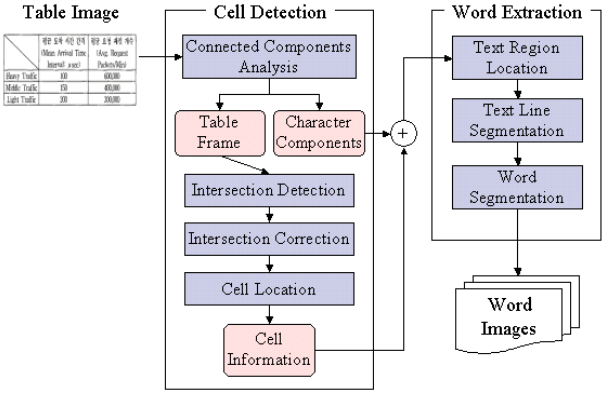


Fig. 1. Block diagram of the proposed method

2 Related Works

Previous research on extracting tables from document images can be divided into two types according to how tables are formed. One method is to analyze or recognize the table images delimited by line-art boundaries, and the other is to recognize the table images formed by the vertical alignment of fixed-width fields without line-art boundaries [5]. The proposed method is related to the former, but most of the previous research has dealt with form document images. Certainly, both tables and forms are sometimes used interchangeably, but a clear distinction exists. Tables are tabular structures, machine-printed for output, and their frame and content are created

simultaneously. But, forms are rectilinear structure, machine- or hand-printed for input, and their frames are created prior to creating the content. While the aim of the research on table images is to vectorize the table frame by analyzing the line components comprising the table and to extract character components, the existing research on form images sought to classify unknown input forms by utilizing the extracted structural information from the forms, querying the input to a form types database, selecting matching forms, and extracting content from form types that do not match.

Watanabe et al. [6] described tables by using the upper-left corners of cells. Firstly, the proposed approach detects vertical and horizontal line segments from the binarized document images by using two extraction filters, and then applies two corner detection filters to the edge-extracted document images for the detection of upper-left corners. However, it is difficult to detect the intersection errors because it detects only the upper-left corners and the corner detection filters may not perform well in case line segments are distorted.

Horizontal and vertical lines can intersect each other to form any of the nine structures: ┌, ┐, ┘, └, ├, ┬, ┤, ┴, ┼. Taylor et al. [7] use four 9×9 pixel templates to detect the intersections. Each template finds one of the basic intersections (┌, ┐, ┘, └), and then the detected corners are combined into an appropriate way for detecting the extensions (├, ┬, ┤, ┴, ┼). However, Arias et al. [8] did not use templates but the leg length of the basic and extended intersections in the corresponding horizontal and vertical directions, and defined their relationship through the hierarchical representation of the nine intersections. This method can reduce the computing complexity spent on detecting the extension intersections in [7] because it obviates the need to visit all of the pixels in the templates of [7] and detects intersections by using their relations.

Neves et al. [9] use binary mathematical erosion to locate intersections and the hierarchical representation of [8] to save calculation time. For the detection and correction of identification errors, it also defines the tenth intersection (virtual intersection), which is represented by a type 0 intersection that is not a real intersection but a part of a horizontal or vertical line. To improve the cell extraction, it detects and corrects identification errors by comparing each neighboring intersection to reference neighborhoods. These reference neighborhoods are congregated in two manners, the rejection tables and the acceptance tables.

While the research mentioned above deal with extracting cells from table images, [4] presents an approach to segment text images into word images. This approach separates text regions into text lines by a horizontal projection profile analysis, and then utilizes gaps and special symbols as delimiters between words by a CC analysis in order to separate a text line into word units. As it combines the top-down approach (projection profile analysis) and the bottom-up approach (CC analysis), it is more efficient than other methods using a single approach.

3 Proposed Algorithm

3.1 Cell Detection

In table images, line components, which compose the table frame, and character components coexist. The line with the largest width is determined as a table frame by

virtue of the 8-CC analysis. Thus, in order to extract cells from a table image, we deal with only the table frame while excluding character components. The intersection extraction module determines the position and the type of intersections on the table frame.

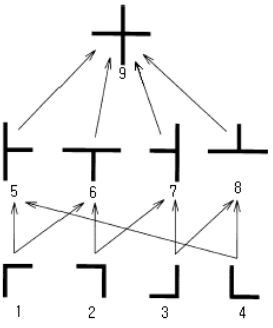


Fig. 2. Hierarchy of intersections

For extracting intersections, we employ the technique in [8] to reduce the extraction time. The intersections can be classified into 9 types: four with two legs, four with three legs, and one with four legs. They can be arranged in a hierarchy as shown in Fig. 2. Each type of intersection is assigned a number from 1 to 9. The proposed approach consists of trying to fit the largest possible intersection type in every black pixel of the table frame and assigning the corresponding label to the pixel. The fit of an intersection consists of finding runs of consecutive black pixels in the corresponding horizontal and vertical directions with the length equal to the leg length of the intersection. The process of labeling the pixels is performed in a hierarchical manner: Corners (┌, ┐, └, ┘) are tried first: if one can be fitted, the corresponding intersections (├, ┬, ┤, ┴) are tried, and so on. While around 30 pixels are checked for each leg in [8], 10 pixels are checked in our work so that the operation of searching for intersections is performed more efficiently. Although the time to check the length of the leg checked is reduced by one-third, there need not be any concern about the number of candidates for intersections amounting to larger than in [8] because character components are excluded from the intersection extraction.

When a vertical line and a horizontal line intersect, one intersection is generated. Two more candidates corresponding to the intersection can be detected according to the thickness of the lines of the table frame so that it becomes necessary to choose from among them by analyzing the information of the candidates. The position of the intersection is located in the center of the candidates, and the type is decided by a majority vote.

Since a cell is generally composed of four intersections, a representation model for extracting cells with intersections can be made with regard to the vertical position of intersections. Fig. 3 shows the representation model, whose elements determine the intersection type and position within a simple table. For example, "1-(3, 5)" corresponding to the element of (1, 1) in Fig. 3 means the intersection of type 1 (┌) located

in (3, 5) of table image. In general, it is possible to extract cells using this representation model with nine types of intersections, but it is necessary to modify the representation model in case two more cells are vertically or horizontally merged into a cell. If the distance between the horizontal positions of intersecting neighborhoods in a vertical direction is larger than the threshold, we consider that there is a merged cell in the table and then add a virtual intersection, which is type 0. In Fig. 3, (b) shows the representation model modified from (a) to add a virtual intersection.

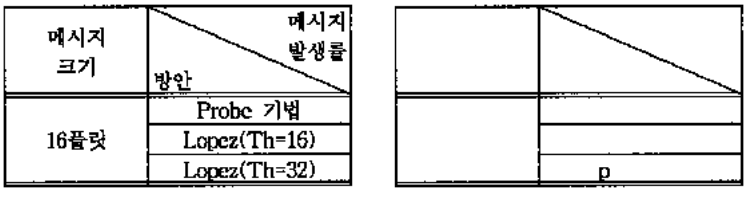
	1	2	3
1	1-(3, 5)	6-(203, 5)	2-(473, 4)
2	5-(3, 139)	9-(203, 138)	7-(474, 138)
3	5-(3, 143)	9-(203, 143)	7-(474, 143)
4	5-(203, 190)	7-(474, 190)	
5	5-(203, 243)	7-(474, 242)	
6	5-(3, 296)	9-(203, 296)	7-(474, 295)
7	4-(2, 300)	8-(202, 300)	3-(473, 300)

(a) Before inserting virtual intersections

	1	2	3
1	1-(3, 5)	6-(203, 5)	6-(473, 4)
2	5-(3,139)	9-(203, 138)	7-(474, 138)
3	5-(3,143)	9-(203, 143)	7-(474, 143)
4	0-(3,190)	5-(203, 190)	7-(474, 190)
5	0-(3,243)	5-(203, 243)	7-(474, 242)
6	5-(3,296)	9-(203, 296)	7-(474, 295)
7	4-(2,300)	5-(202, 300)	5-(473, 300)

(b) After inserting virtual intersections

Fig. 3. Representation model for a cell



(a) Original image (b) CC of the table frame

Fig. 4. An example that a character CC touches at a table frame

But when certain line intersections do not appear due to the poor quality of the acquisition, binarization problems, or skew correction, etc, it is difficult to correctly extract intersections with only the method described so far. Since those factors can distort document images and prevent the detection of real intersections, or even detect false intersections, it is necessary to detect and correct the error of intersections. Therefore, we adapt the technique in [9] to detect and correct the error of intersections by comparing each intersection neighborhood with reference neighborhoods. When a character component and the table frame touch as shown in Fig. 4, the character component is recognized as part of the table frame and false intersections can be detected around this area. The false intersections must be modified into virtual intersections by correcting the intersections.

After the intersection correction, each cell is determined with the four adjacent intersections excluding the virtual intersections appropriately chosen from the extracted intersections. But the meaningless cells formed by double lines must be excluded by considering the height and width of cells.

3.2 Word Extraction

The analysis of character components within the cells can be conducted faster and easier than the previous approaches by using the information from the character components that were excluded from the line component analysis. This method reduces or removes the influence of line components that may be located within the cells. Firstly, text regions within cells are determined by examining whether or not the CCs are located inside the appropriate cell. Since the character components touching the table frame are regarded as line components as mentioned above, we use the extracted cell information to separate these character components from the line components and add them to character components.

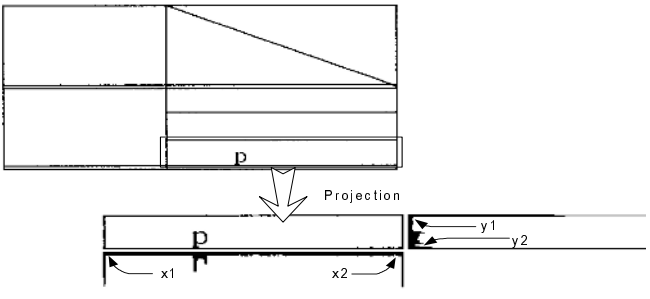


Fig. 5. Processing method for locating the internal region of the cell when character components touch the table frame

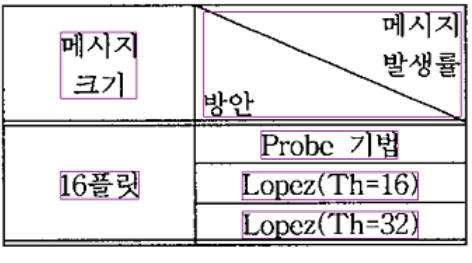


Fig. 6. Result of locating the internal region of the cell when characters components touch the table frame

When the false intersections detected in the intersection correction are located in the table frame, we assume that character components are touching the table frame. We detect the cell in which this contact occurs by analyzing the position and type of the false intersections, and then locate the internal region of the cell from the table frame by using the projection profile of the cell. By comparing with the neighborhood projection profile in the direction of center from both ends, the internal region of the cell (x1, x2, y1, y2) is decided if a current value is less than 10% of the previous value in Fig. 5. We analyze the CCs again for the part of the table frame located in the

internal region of the cell and then add it to the character components of the text region in the cell. Fig. 6 shows the result for locating the internal region of the cell when character components touch the table frame.

Basically we employ our algorithm proposed in [4] for separating text regions into words, and additionally perform a post-process because words are not located in a text region but a table. The post-processing for the word segmentation is needed when the characters within a word are segmented into words due to the special spacing of the cell.

4 Experiment

4.1 Experiment Environment

We used a set of 100 binary images of table to evaluate the performance of the proposed method. The data are constructed manually from document images, which are pre-processed by the system in [4]. They are digitized in 300 dpi with spatial resolutions from 849×117 to 1500×1770. The document images were provided from the full-text image retrieval services by the Korean Information Science Society. Table 1 shows the data classified into four types by the cell formation and the boundary of cell, etc. The experiment was run on a Pentium-4 2.0 GHz PC.

Table 1. Test data

Type of table	Number
Normal table	60
Table whose cells are merged	20
Table consisted of a cell	10
Table which all or a part of border lines are missed	10
Total	100

4.2 Experiment Results

The performance results from the proposed method were evaluated in terms of accuracy and speed. The results obtained by the proposed cell extraction were 100% accurate for cell extraction (the total number of cells extracted is 2,313 in 100 table images). There were 7 false intersections in 5 table images due to the contact between character components and a table frame, but our algorithm corrected them and extracted all cells correctly.

Table 2. The result of word segmentation

		Result-1		Result-2	
# of images	# of words	# of successes	# of failures	# of successes	# of failures
100	4,547	4,301 (94.59%)	246 (5.41%)	4,509 (99.16%)	38 (0.84%)

Table 2 shows the results of word segmentation using the extracted cell information: Result-1 was obtained by applying only a gap clustering and Result-2 was obtained by additionally applying the special symbol detection. In the word segmentation module, we obtained an accuracy of 94.59% with only the gap clustering, and improved the performance rate by 4.57% by additionally applying the special symbol detection. Finally, we achieved an accuracy of 99.16% in the word segmentation. Fig. 7 shows the final result of the proposed method.

트래픽 형태	메시지 발생률 / 메시지크기	0.01	0.1
균등분포	16 플릿	2.202	2.076
	32 플릿	2.391	2.146
	64 플릿	2.564	2.432
perfect-shuffle	16 플릿	1.947	1.95
	32 플릿	1.973	2.181

Fig. 7. The final result of the proposed method

There are several types of errors in the word segmentation and Fig. 8 shows a few examples. The first type is because an underline extends over two or more words and then between-word gaps cannot be computed by using horizontal projection profiles. So the words connected by an underline is segmented into a word as shown in Fig. 8(a). The second one is due to a split in a character. Some character can be split in two because of the image degradation or low quality and then classified as special symbols. Such a split character is used for an additional between-word gap shown in Fig. 8(b). The other is due to miss-classification of special symbols. When a special symbol touches neighbor characters as shown in Fig. 8(c) or has a property of the Italic fonts as shown in Fig. 8(d), our system cannot classifies them into special symbols.

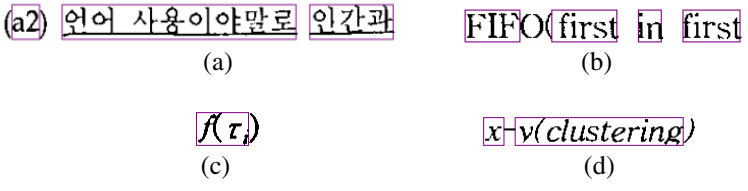


Fig. 8. Examples that the proposed system failed

The average processing time for cell detection and word segmentation was about 1.399 and 0.071 seconds respectively. Since the information of the CCs analyzed in the cell extraction module was repeatedly used in the word segmentation module, the

processing time for cell detection was significantly longer than the word segmentation module.

Table 3 shows the result obtained by combining the proposed method into the system in [4]. While the accuracy rate of word extraction was 90.84% for 50 Korean images before the combination, a 2.66% improvement was obtained after the combination. The performance improvement can be mainly attributed to the fact that the proposed method is able to extract words from the table regions, whereas the system in [4] is unable to achieve. For example, the accuracy rates of the 35th and 47th images were less than 70% before the table processing, but they were improved by more than 96% after the table processing.

Table 3. The result of the [4]'s system with the proposed method

Image ID	# of real words				# of extracted words				Accuracy(%)	
	TR	NTR		All	TR	NTR		All	Before table processing	After table processing
		Tab.	Fig.			Tab.	Fig.			
1	474	0	0	474	472	0	0	472	99.58	99.58
2	707	0	0	707	707	0	0	707	100.00	100.00
3	581	0	35	616	581	0	0	581	94.32	94.32
4	570	0	27	597	569	0	0	569	95.31	95.31
…							…			
35	374	208	0	582	353	208	0	561	60.65	96.39
36	546	74	0	620	537	71	0	608	86.61	98.06
…							…			
46	432	43	19	494	420	43	0	463	85.02	93.72
47	400	169	0	569	387	169	0	556	68.01	97.72
48	491	0	10	501	486	0	0	486	97.01	97.01
49	477	0	15	492	470	0	0	470	95.53	95.53
50	402	53	45	500	378	53	0	431	75.60	86.20
									90.84	93.50

5 Conclusion

We have proposed a method to extract words from table images and improve the word extraction system for document images. The proposed method is composed of cell detection and word extraction. The method correctly included character components touching the table frame with words using the cell information, so experimental results show that more than 99% of words were successfully extracted from table regions.

The proposed method in this paper can be used as a core technology for keyword spotting or retrieval systems for digital libraries.

Acknowledgement

This work was supported by grant No. R05-2003-000-10396-0 from the KOSEF.

References

1. Oh, I. S., Choi, Y. S., Yang, J. H., Kim, S. H.: A Keyword Spotting System of Korean Document Images. Lecture Notes in Computer Science, Vol. 2555. Springer-Verlag, Berlin Heidelberg New York (2002) 530
2. Marinai, S., Marino, E., Cesarini, F., Soda, G.: A General System for the Retrieval of Document Images from Digital Libraries. Proceedings of the First International Workshop on Document Image Analysis for Libraries (2004) 150-173
3. Lu, Y., Zhang, L., Tan, C. L.: Retrieving Imaged Documents in Digital Libraries Based on Word Image Coding. Proceedings of the First International Workshop on Document Image Analysis for Libraries (2004) 174-187
4. Jeong, C. B., Kim, S. H.: A Document Image Preprocessing System for Keyword Spotting. Lecture Notes in Computer Science, Vol. 3334. Springer-Verlag, Berlin Heidelberg New York (2004) 440-443
5. Lopresti, D., Nagy, G.: A Tabular Survey of Automated Table Processing. Lecture Notes In Computer Science, Vol. 1941. Springer-Verlag, Berlin Heidelberg New York (1999) 93-120
6. Watanabe, T., Luo, Q., Sugie, N.: Layout Recognition of Multi-Kinds of Table-Form Documents. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 17. (1995) 432-445
7. Taylor, S., Fritzson, R., Pastor, J.: Extraction of Data from Pre-printed Forms. Machine Vision and Applications, Vol. 5. No. 3. (1992) 211-222
8. Arias, J. F., Kasturi, R.: Efficient Extraction of Primitives from Line Drawings Composed of Horizontal and Vertical Lines. Machine Vision and Applications archive, Vol. 10. (1997) 214-221
9. Neves, L. A. P., Facon, J.: Methodology of Automatic Extraction of Table-Form Cells. XIII Brazilian Symposium on Computer Graphics and Image Processing, (2000) 15-21