ขั้นตอนวิธีสำหรับการบ่งชี้การสกัดและการแปลงโครงสร้างตารางจากภาพเอกสารเป็นรูปแบบลาเท็กซ์

นายสัณห์ เศรษฐโศภณ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาการคณนา ภาควิชาคณิตศาสตร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2545

ISBN 974-17-3318-6

AN ALGORITHM FOR IDENTIFYING, EXTRACTING AND CONVERTING A TABLE STRUCTURE FROM A DOCUMENT IMAGE INTO LATEX FORMAT

Mr. San Sethasopon

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Computational Science
Department of Mathematics
Faculty of Science
Chulalongkorn University
Academic Year 2002

ISBN 974-17-3318-6

Thesis Title	AN ALGORITHM FOR IDENTIFYING, EXTRACTING AND CONVERTING A TABLE STRUCTURE FROM A DOCUMENT IMAGE INTO LATEX FORMAT
By	Mr. San Sethasopon
Field of Study	Computational Science
Thesis Advisor	Professor Chidchanok Lursinsap, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

....Dean of Faculty of Science

(Associate Professor Wanchai Phothiphichitr, Ph.D.)

THESIS COMMITTEE

....Chairman

(Assistant Professor Peraphon Sophatsathit, Ph.D.)

....Thesis Advisor

(Professor Chidchanok Lursinsap, Ph.D.)

....Member

(Associate Professor Jack Asavanant, Ph.D.)

สัณห์ เศรษฐโศภณ : ขั้นตอนวิธีสำหรับการบ่งชี้การสกัดและการแปลงโครงสร้างตารางจากภาพเอกสารเป็นรูปแบบลาเท็กซ์. (AN ALGORITHM FOR IDENTIFYING, EXTRACTING AND CONVERTING A TABLE STRUCTURE FROM A DOCUMENT IMAGE INTO LATEX FORMAT) อ. ที่ปรึกษา : ศาสตราจารย์ ดร. ชิดชนก เหลือสินทรัพย์, 52 หน้า. ISBN 974-17-3318-6.

การวิเคราะห์ตารางเป็นส่วนหนึ่งของปัญหาการวิเคราะห์ภาพเอกสารที่น่าสนใจ ประกอบด้วยวิธีการบ่งชี้ตารางซึ่งอยู่บนพื้นฐานของเทคนิคการแบ่งภาพและแยกประเภทออกเป็นส่วน และวิธีการรู้จำตาราง วิทยานิพนธ์นี้เสนอขั้นตอนวิธีใหม่สำหรับการวิเคราะห์ตาราง เริ่มจากการแบ่งภาพเอกสารออกเป็นส่วนๆ ส่วนที่ไม่ใช่ตารางจะถูกกำหนดโดยการเรียงตัวของก้อนข้อมูลและตำแหน่งของเส้น แล้วส่วนที่เป็นตารางจะถูกแปลงเป็นรูปแบบลาเท็กซ์ ซึ่งเหมาะสำหรับการแก้ไข การจัดเก็บ การนำมาใช้ใหม่ และการส่งข้อมูล

ขั้นตอนวิธีนี้ถูกทดสอบกับตัวอย่างที่เป็นส่วนที่สกัดมาจากภาพเอกสารจริง และจากการสร้างขึ้นเอง ตารางที่มีการเรียงตัวของข้อมูลและเส้นหลายรูปแบบถูกบ่งชี้และวิเคราะห์ได้อย่างถูกต้อง ขั้นตอนวิธีที่ใช้นี้ให้ผลที่ดีกับตัวอย่างที่เอียงไม่มากและมีสิ่งรบกวนน้อย

ภาควิชา	**คณิตศาสตร์**	ลายมือชื่อนิสิต……………………………...
สาขาวิชา	**วิทยาการคณนา**	ลายมือชื่ออาจารย์ที่ปรึกษา…………………..
ปีการศึกษา	2545	

4272423023 : MAJOR COMPUTATIONAL SCIENCE
KEYWORD: TABLE ANALYSIS / DOCUMENT IMAGE ANALYSIS

SAN SETHASOPON : AN ALGORITHM FOR IDENTIFYING, EXTRACTING AND CONVERTING A TABLE STRUCTURE FROM A DOCUMENT IMAGE INTO LATEX FORMAT. THESIS ADVISOR : PROFESSOR CHIDCHANOK LURSINSAP, Ph.D., 52 pp. ISBN 974-17-3318-6.

Table analysis is one of the attractive and challenging problems in document image analysis that encompasses table identification and table recognition. Table identification is based on the techniques of page segmentation and classification, whereby the results so extracted are analyzed and stored in some prearranged structures. This study proposes an algorithm for table analysis that starts from separating a document image into individual blocks. A non-tabled block is determined by the arrangement of data inside the block and the position of lines. Then, the recognized table blocks are converted into LaTeX formatted tables suitable for subsequent modification, storage, retrieval and transmission.

The algorithm was tested with image blocks extracted from actual document images and synthesis samples. Various styles of tabled block-lines and data arrangement were correctly identified and analyzed. The algorithm gave good results for input samples having less skewed angle and noise.

Department	**Mathematics**	Student’s signature……………………....
Field of study	**Computational Science**	Advisor’s signature……………………..
Academic year	**2002**	

Acknowledgements

I am greatly indebted to my supervisor, Professor Dr. Chidchanok Lursinsap, at The Advance Virtual and Intelligent Computing (AVIC) Research Center, for his suggestions, untiring help and guidance of this thesis. Without him my thesis would have never been accomplished.

I also wish to express my special thanks to the thesis committee, Assistant Professor Dr. Peraphon Sophatsathit and Associate Professor Dr. Jack Asavanant, for their valuable advise, reading and considering the manuscript.

Furthermore, I would like to thank all my past and present instructors for their valuable lecture and instructions. Finally, I would like to express my sincere gratitude to my parents for their love, understanding and constant support during my study.

I truly believe that all people whom I have not personally mentioned here are aware of my deep appreciation.

Table of Contents

List of Tables

List of Figures

CHAPTER I

INTRODUCTION

The most information available today is on paper form document that continues to increase the volume everyday. This is because human prefer to use paper form documents in daily life for several centuries. It is very difficult to store and retrieve the large and every increasing number of paper form documents. On the other hand, an electronic system has several advantages in storage, and retrieval. As a result, transformation of paper form document to its electronic version has become an important and a challenging domain for computer vision and pattern recognition researchers.

The rest of the thesis is organized into five chapters. Chapter I define the problem identification, the objective of this study, the scope of work and constraints and the contributions and applications. The next chapter reviews the literatures related to page segmentation and table analysis. Chapter III discusses the proposed algorithm for identifying, extracting and converting a table structure from a document image into LaTeX format. Chapter IV explains and compares all the experimental results. Conclusion of this study and discussion for the future work are shown in chapter V.

1.1 Problem Identification

The paper form documents can be easily converted to their electronic forms via a scanner. These scanned documents are stored as image data, so they have two disadvantages: they require a large amount of storage space, and it is difficult to search the area that the required data are stored and to reuse these data. For efficient retrieving and storage of documents, it is necessary to generate a description of graphical elements in the document to decrease the storage space and processing time. The document images need page segmentation method to be segmented into individual regions of text area and non-text area. Then, the characters in the regions of text area should be recognized and stored as coded data using Optical Character Recognition (OCR) technique. And the regions of non-text area should be stored as image data. However, there are many characters in some regions of the non-text area such as the numbers in graphed area, the descriptions for figure, the characters in each cell of table, etc., which should be stored as coded data.

In tabled areas, the columns and rows in which the characters are located are very important pieces of information. To store these information, we must detect the tabled area and then analyze its structure. The result of table analysis and its characters recognition should be stored together with some arrangement formats. This study proposes to use LaTeX format that can store both of table structure and its characters as the code data. It uses less space storage than the other word processing so it is widely used for the most researchers in the world.

1.2 Objectives of This Work

The main objectives of this study are:

1. To develop an algorithm for identifying and extracting a tabled area from a document image.

2. To convert the table structure into LaTeX format.

To obtain the result of this work, the algorithm was applied on gray scale document image without any skewed angles. As a result, the algorithm can be integrated as a part of an OCR (Optical Character Recognition) system to enhance the system capability.

CHAPTER II

LITERATURE REVIEW

This chapter describes the survey of papers related to this study. It consists of two sections. The first section explains the algorithms for page segmentation and the last one reviews the table analysis.

2.1 Review of Literatures Related to Page Segmentation

Earlier approaches of page segmentation method were based on two basic types. The first approach is a bottom-up method that started by grouping all interested pixels together and merged them into larger homogeneous regions. Another approach is a top-down method adopted by dividing a document image into large regions that were recursively segmented into smaller sub-regions.

Most bottom-up methods divide text from components in a document image. L. A. Fletcher and R. Kasturi [1] described an algorithm for separating text strings from mixed texts and graphic images in 1988. This algorithm grouped each connected black pixels together to generate a connected component. Then, it used Hough transform [2] to group the connected components into text strings which should be separated from the graphics. This algorithm could be used with various font styles, sizes and orientations of texts.

In 1992, F. Lebourgeois, Z. Bublinski, and H. Emptoz [3] presented a bottom-up method for extracting text paragraphs and graphics from document images. The method used a horizontal smoothing from a Run Length Smoothing Algorithm (RLSA) [4] for linking characters and graphics to form blocks. In addition, height and density of blocks are utilized for identifying blocks as text lines or graphics that subsequently merged the text lines into paragraphs.

Instead of considering black pixels in almost bottom-up methods, A. Antonacopoulos and R. T. Ritchings [5] considered the white pixels of background spaces. They presented an algorithm that linked the white pixels to build white regions in 1995. The other regions that were surrounded by the white regions were extracted into individual regions.

All the above approaches required some suitable threshold values for linking the pixels, merging or classifying the regions. In 1996, D. P. Mital and G. W. Leng [6] developed a technique for text segmentation that did not require any predefined parameter values. This technique used two arrays to store the connected black and white pixel length values. The differences of nearby values were considered for text regions.

Typically, a document image contained a very large amount of pixels. In order to segment image to blocks, it was not necessary to analyze every pixel. E. Trupin and Y. Lecourtier [7] compressed the image with a compression factor, F. Each pixel of the compressed image was set to black when there was a black pixel in an area of F pixels width and F pixels height. The connected black pixels of the compressed image were linked to form blocks. The application for page segmentation could be done in both bottom-up and top-down methods. A low value of the compression factor would segment the image into words or pieces of word for processing with the bottom-up method. On the other hand, a result of a height compression factor value should process with the top-down method.

T. Saitoh and T. Pavlidis [8] presented an algorithm for page segmentation that could handle skewed pages and not rectangle columns. This algorithm would make a compressed image by considering each eight pixels width and four pixels height area. Then, each connected black pixel was linked to form blocks. Each block was classified into six classes by its height, width, and halftone. The text class blocks that connected together were merged to form one text block. After the skew angles were estimated using of the least square technique, some blocks were classified as tables with reference to the ruled lines. The blocks of text area were merged again into columns.

In 2001, S. Chuai-aree, C. Lursinsap, P. Sophatsathit and S. Siripant [9] presented a technique for page segmentation. This technique separated a document image to blocks with the same size. Each block was classified as text, image and background area using Fuzzy C-mean [10] with statistical features such as mean and standard deviation. Then, the class of each block and its neighborhoods were used to define its true class.

The bottom-up method was effective for every document styles, but it required considerable computation time and memory. While the top-down method was faster, it gave a good result for the documents with fixed and known structure. One of the most powerful technique of top-down method was RLSA first introduced by F. M. Wahl, K. Y. Wong and R. G. Casey [4] in 1982. The smoothing rule would change a white run to a black run if its length was less than or equal to a predefined threshold value, when a run was a sequence of the same value pixels. This algorithm contained the following four steps:

1. A horizontal smoothing was applied to the original document image.

2. A vertical smoothing was applied to the original document image.

3. The results of steps 1 and 2 were combined by a logical AND operation, and

4. A horizontal smoothing was applied to the output of step 3.

This four-step RLSA required scanning the whole image four times. In 1992, it was improved into a two-steps RLSA by F. Y. Shih, S. S. Chen, D. D. Hung and P. A. NG [11]. It scanned the whole image only two times. F. Y. Shih et al. used this algorithm to extract a document image into individual blocks. Each block was classified into one of text, horizontal or vertical line and graphic by means of properties such as the height, the aspect ratio, the density and the number of changed pixels per unit length. The results of their algorithm experiments were presented in 1996 [12].

The last RLSA gave a good result when the input was a horizontally written document. However, its results were not good for a vertically written document. Therefore, N. Amamoto, S. Torigoe and Y. Hirogoki [13] developed an algorithm for the both horizontally and vertically written documents. The algorithm extracted a group of black pixels that was surrounded by a white space as a block. The block, which did not have any long black runs, was decided as the horizontal or vertical writing. Then, every block was classified using similar properties to the previous approach and the number of thin and long black lines.

In 1996, N. Papamarkos, J. Tzortzakis and B. Gatos [14] developed a technique that calculated the threshold values for the RLSA automatically. This technique was based on the values of the mean character length and the mean text line distance of a document that were estimated by the distributions of the horizontal and vertical black and white runs.

Another top-down method for page segmentation widely used was called a Recursive X-Y Cut (RXYC) algorithm. The RXYC cut an image recursively into blocks. At each step of the recursive process, it determined a sum of black pixels along both horizontal and vertical for projection profile computing. Thus, the document projection profile was a waveform whose deep valleys corresponded to blank areas of the document. A deep valley with a width greater than a threshold could be cut as the edge of a block. H. Wang, S. Z. Li and S. Ragupathi [15] used this algorithm for document segmentation and explained a classification process with tested results in 1997.

The page segmentation by RLSA or RXYC used a fixed threshold for a whole document, so there were some problems with some document styles such as a text lines with different font size, or font styles, etc. To solve these problems, K. C. Chan, X. Huang and P. Bao [16] presented a page segmentation method based on the concept of fuzzy set theory. This method determined the thresholds in any positions in the document automatically.

Both the bottom-up and top-down methods had their own advantages. As such, attempts were made to combine these two methods. Y. Hirayama [17] presented a block segmentation method that started by linking black connected pixels to form rectangles. They were classified as character strings, horizontal and vertical lines, and picture elements by considering their properties. The character strings were merged into text groups. The threshold of the merging was determined by analyzing the height and distance of character strings. Next, the borderlines of columns were detected by linking the edges of text groups. Then, the whole page was segmented into blocks based on borderlines. These blocks were segmented into text and pictured areas by means of projection profile method.

In 1998, A. K. Jain and B. Yu [18] presented an algorithm for page segmentation based on a top-down method which was generated through the bottom-up method. They defined a block of regular small connected components as a text region and a composed of large connected components as a non-text region.

2.2 Review of Literatures Related to Table Analysis

Table analysis process consists of table identification and table recognition methods. The goal of table identification is to separate tabled areas from non-tabled areas in a document image. Each tabled area contains important data that are related by positions (column and row). The table recognition method is used to determine the structure of the tabled area and stored with some arrangement formats.

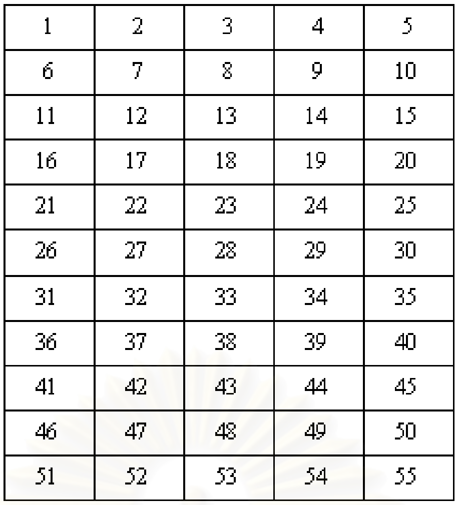
Typical page segmentation methods encompass table identification methods. Thus, they identify a tabled area by considering some blocks from the page segmentation process. For example, T. Saitoh and T. Pavlidis [8] classified some blocks as table areas based on ruled lines. They defined a block as a tabled area if it had at least two horizontal lines (at top and bottom sides) and one vertical line (not so close to the side edges).

N. Amamoto et al. [13] defined a block as a tabled area according to properties of the block. The table block must have the number of thin and long solid black lines higher than other classes.

A. K. Jain and B. Yu [18] identified tabled areas from non-text blocks. They first extracted horizontal and vertical lines. The horizontal top and bottom lines should have the same length without any skew lines. The height of all connected components should be small.

Y. Hirayama [19] presented a process for table identification from non-text blocks that had both horizontal and vertical lines. The block was assumed to be the tabled area and was divided into small areas by the horizontal and vertical lines. Each small area was classified as a celled area or non-celled area. If the non-celled area had less space than one thirds of the whole block area, the block should be a tabled area.

Y. Wang, I. T. Phillips and R. Haralick [20] presented a table identification algorithm that analyzed background. They used the large horizontal blank blocks to construct tabled candidates. Then, the tabled candidates were identified based on some predefined parameters such as the ratio of total large vertical blank block areas over the tabled candidate area, the maximum difference of the cell baselines



1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25
26	27	28	29	30
31	32	33	34	35
36	37	38	39	40
41	42	43	44	45
46	47	48	49	50
51	52	53	54	55

Figure 2.1: An example of a table with cells labeled by numbers.

in a row, and the maximum difference of the justification in a column. These parameters could be estimated from the real table instances.

Most approaches did not concentrate on the table identification but focused on the table recognition. S. Chandran and R. Kasturi [21] used the horizontal and vertical projection profiles of a table image for extracting a data as cells in the table. Each cell was labeled by a number and arranged in order of positions (left to right and top to bottom) as shown in Figure 2.1. The data and its corresponding label were stored with the number of columns in the table.

However, the above method could not be used for complicated tables. K. Itonori [22] presented a method that was applicable for all styles of table. It used the projection profiles to assign row and column numbers in a table image. Each cell could be stored with its right and bottom coordinates as shown in Figure 2.2.

E. Green and M. Krishnamoorthy [23] gave each cell a label by a series of a

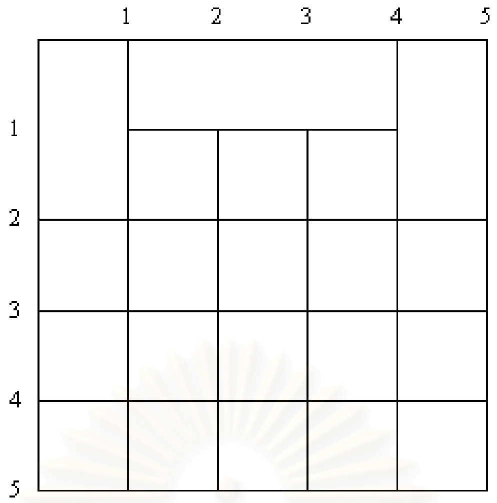


Figure 2.2: An example of the numbers assigning in each column and row.

letter for column and a number for row. The label, consisting of one letter followed by a number was used for each level of cell in the table image. An example of a complete labeling appears in Figure 2.3.

J. F. Arias, A. Chhabra and V. Misra [24] used the information about horizontal and vertical lines to extract the structure of table image. They used bit strings for labeling each cell of table. The label indicated the columns and rows to which the cell belonged. The length of the labels was equal to the number of the columns and rows that contained in the table. Each bit indicated an order of column and row. The bit was 1 if the cell corresponded to a given column and row. An example of the labeling of cells in a table is shown in Figure 2.4.

T. Watanabe, Q. Luo and N. Sugie [25] used a binary tree to represent the structure of table. Each node corresponded to a block of one or more cells. The nodes were arranged in vertical node v, horizontal node h, and the terminal node

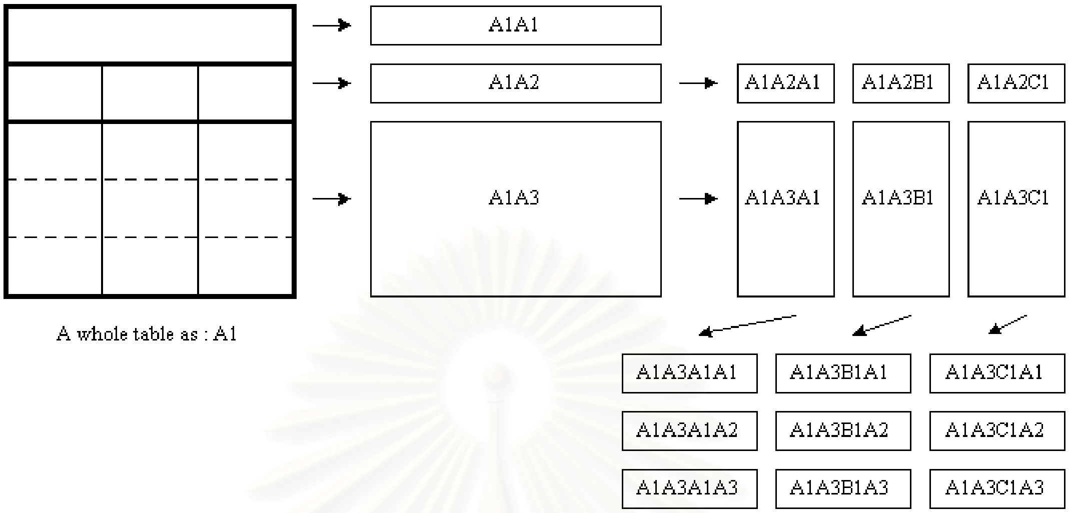
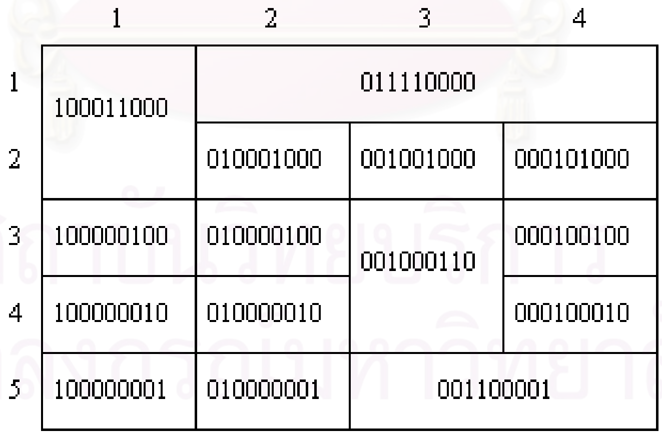


Figure 2.3: Illustration of the process for labeling by letters and numbers.



<table>
<tr><th></th><th>1</th><th>2</th><th>3</th><th>4</th></tr>
<tr><td>1</td><td rowspan="2">100011000</td><td colspan="3">011110000</td></tr>
<tr><td>2</td><td>010001000</td><td>001001000</td><td>000101000</td></tr>
<tr><td>3</td><td>100000100</td><td>010000100</td><td rowspan="2">001000110</td><td>000100100</td></tr>
<tr><td>4</td><td>100000010</td><td>010000010</td><td>000100010</td></tr>
<tr><td>5</td><td>100000001</td><td>010000001</td><td colspan="2">001100001</td></tr>
</table>

Figure 2.4: An example of the cells labeling using the bit-strings.

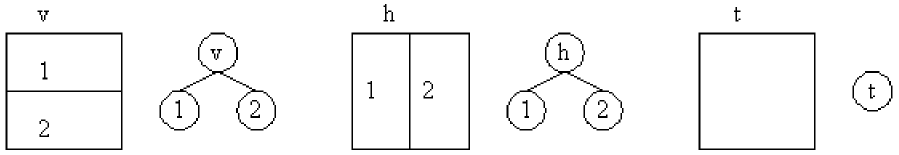


Figure 2.5: Node types in the binary tree for representing a table structure.

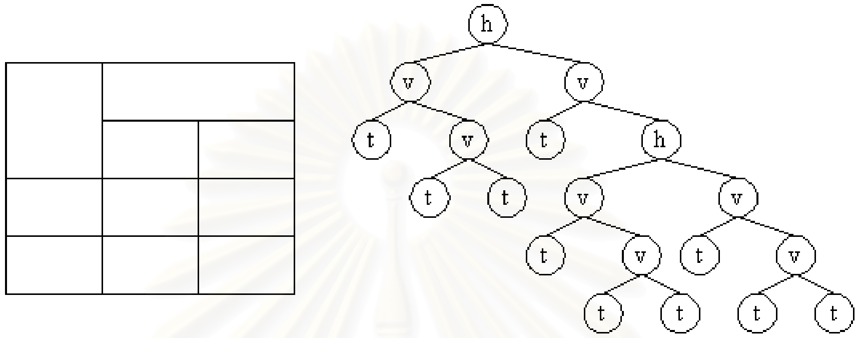


Figure 2.6: An example of a table and its corresponding binary tree.

t. The type of node represented the relation to its left and right child nodes as illustrated in Figure 2.5. Figure 2.6 presents a sample of table structure and its corresponding binary tree.

H. Saiga, Y. Kitamura and S. Ida [26] extracted all dotted and solid lines in a table image and defined their names. Then, they constructed cells from the horizontal and vertical lines interceptions. Each cell was stored as names of the horizontal and vertical lines that constructed it. This algorithm was easily combined with character recognition.

Y. Hirayama [19] presented a dynamic programming matching method to detect the correspondence between strings in two columns. And they applied this method for the table with more than two columns. As a result, the information arrangement was easily detected and transformed into other data formats.

K. Zuyev [27] introduced a concept of tabled grid which represented a complex structure table as an uncomplicated table. The grid of table was formed in every row and column using the horizontal or vertical lines, and the space between rows or columns.

In the case of a table with the large amount of data, It was difficult to represent as the labeling method. S. Tsuruoka, K. Takao, T. Tanaka, T. Yoshikawa and T. Shinogi [28] described an algorithm of segmentation and conversion for a table image. First, they segmented a table by the ruled lines into some regions. Then, the segmented regions were further divided into cells by the projection profiles. These results were converted into a HyperText Markup Language (HTML) file.

A. Amano, N. Asada, T. Motoyama, T. Sumiyoshi and K. Suzuki [29] designed a table form document analysis and synthesis system that cooperated between users and the computers. The computer detected the boxes formed by horizontal and vertical lines. These boxes were classified semi-automatically into four types, namely, blank, insertion, indication, and explanation. Then, the relationship between each box and its neighbors was analyzed. Finally, the system generated Latex code of the synthesized document with blank and insertion boxes that were filled with text or image data given by the user.

CHAPTER III

MODEL AND ALGORITHM

This chapter explains the algorithm of this study in three steps. Firstly, a document image is extracted to individual blocks. Next, each block is identified as a tabled block or non-tabled area. Lastly, the tabled blocks are converted into a LaTeX format.

The document image analyzed in this study is a gray scale image. Each pixel has a value between 0-255 that represents the color intensity. The pixels whose value close to 0 are called the black pixels. On the other hand, the white pixels have the values close to 255. The size of the image can be represented by a two dimensional array. The number of columns and rows define the width and height of a table, respectively.

3.1 Blocks Extraction Approach

This study focuses on the tabled areas in a document image. All of tabled areas have rectangular shapes. Thus, the proposed algorithm will extract a document image to rectangular blocks that are easy for processing in the later steps. For extracting the blocks, an image smoothing algorithm is applied to a document image for merging the near black pixels. The result of image smoothing is a blurred image from the document image. Each pixel of the blurred image is determined by the adjacent pixels as in the following equation:

$$B_{i,j} = \frac{\sum_{k=i-I}^{i+I} \sum_{l=j-I}^{j+I} P_{k,l}}{(2I+1)^2} \tag{3.1}$$

where $B_{i,j}$ is the value of pixel in the ith row and jth column of the blurred image.

$P_{k,l}$ is the value of pixel in the kth row and lth column of the document image.

$I \in \{1, 2, 3, ..., N | N \text{ is the number of columns}\}$

The value of I depends on the space between text lines in the document image.

We consider on the blurred image to construct cutting lines that are used for extracting the document image to rectangular blocks. When the blurred image is scanned horizontally or vertically from one edge to the opposite edge, we classify the scanned results into two types. The first one that has only white pixels is called space line, and the one that has some black pixels in it is called data line. The cutting line is a data line that is next to the space line. The detail of the algorithm for constructing the horizontal cutting lines is as follows.

Cutting Lines Construction Algorithm

```
If the first row is a data line then
    Set this row as the top edge of a new block.
For $i$=2 to number of row do
    If this row is a data line And the ($i$-1)th row is a space line then
        Set this row as the top edge of a new block.
    If this row is a space line And the ($i$-1)th row is a data line then
        Set the ($i$-1)th row as the bottom edge of a block.
End for
If the last row is a data line then
    Set the last row as the bottom edge of a block.
```

In this study, we apply this algorithm three times. The first time is mainly for constructing the horizontal cutting lines that extract the blurred image into smaller blocks. The second time is applied on each column in each block for vertical cutting lines. The last time is for the horizontal cutting lines that lie inside the vertical block. All of the cutting lines are used for extracting the document image into rectangular blocks. Figure 3.1 illustrates an example of the process in the blocks extraction method.

3.2 Table Identification Method

Each block from the last step is assumed as tabled block. The groups of black pixels in every block represent lines and data of a table. Data and lines are then analyzed and stored in a text file that is suitable for identifying as well as converting into LaTeX format. In the process for preparing this text file, it starts by detecting lines in each block to store the positions of lines for the later processing. The block is then separated into small rectangular areas using the lines as their borderlines. The coordinates of each area consisting of four values; top, left, bottom, and right positions are defined with the line informations.

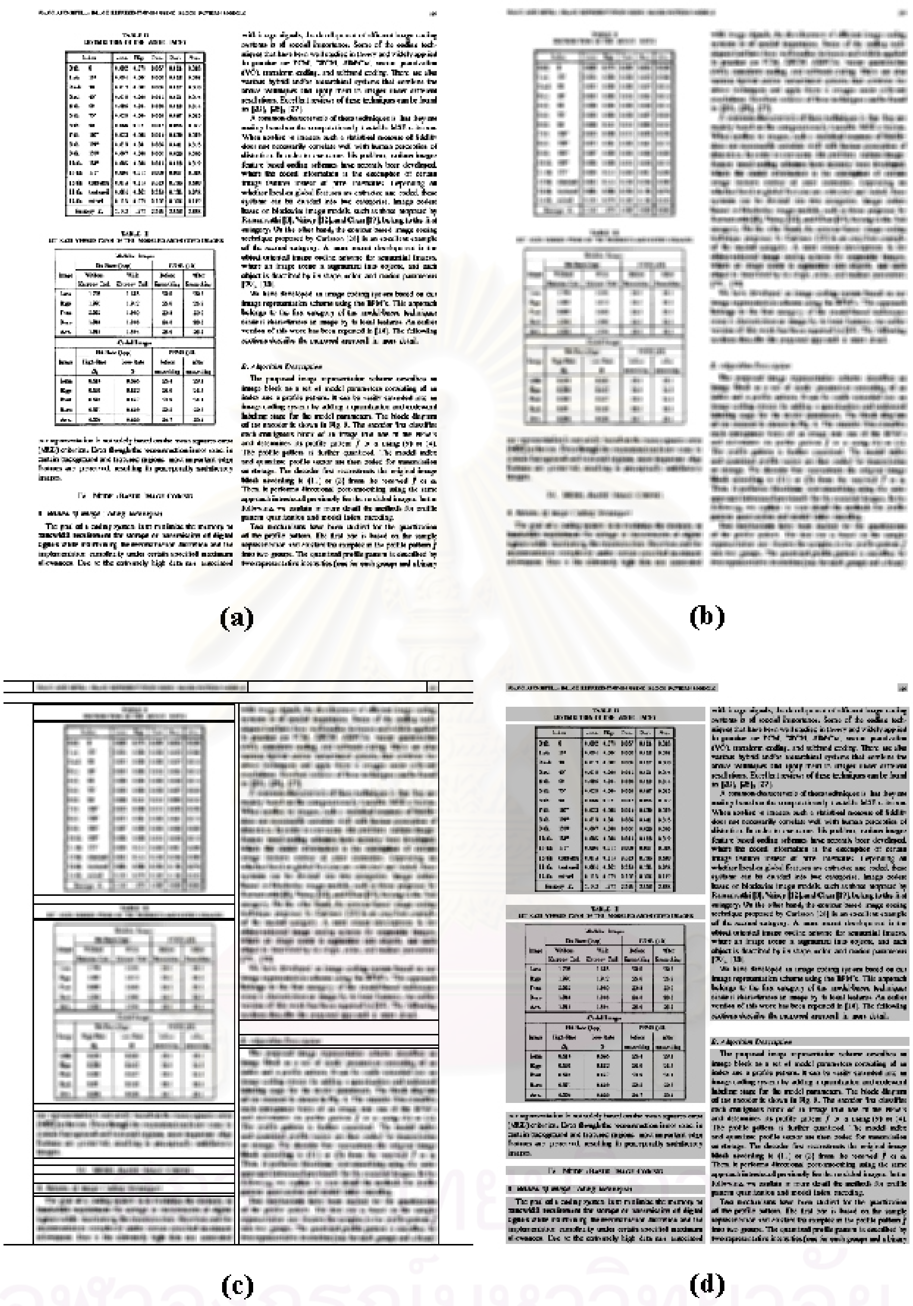


Figure 3.1: An example of process in the block extraction method: (a) a document image, (b) a blurred image, (c) a blurred image with the cutting lines, (d) the result of appling the cutting lines on the document image.

Lines Detection Algorithm

```
1. List $L$ is empty.
2. For each row of a block do
3.     If there are some black runs that is longer than $\alpha$
(The value of $\alpha$ depends on the size of the character string in the block being considered)
4.         Store their position as the row number, beginning and ending point
           in list $L$.
5. End for
6. For each column of a block do
7.     If there are some black runs that is longer than one fifths of the number
       of rows then
8.         Store their position as the column number, beginning and ending
           point in list $L$.
9. End for
```

Block Separation Algorithm

```
1. List $B$ is empty.
2. For $i$= 1 + $\beta$ to number of rows - $\beta$ do
3.     For $j$= 1 + $\beta$ to number of columns - $\beta$ do
(The value of $\beta$ depends on the thickness of block border from the extraction step)
4.         If $W_{i,j}$is a white pixel and not in the ranges of the coordinates in
           list $B$ then
5.             Set $W_{i,j}$ to be the point of a new borderline.
```

```
6.            Direction = 1.
7.            **If** there are some white pixel around $W_{i,j}$ **then**
8.                **Do**
9.                    Find the next pixel of this borderline.
9.1                       Start at the opposite of the last direction as
                          in Figure 3.2.
9.2                       Find a white pixel around the point in
                          clockwise direction.
9.3                       Stop when a white pixel is found.
10.                   Set this pixel to be the new point of consideration.
11.               **Until** the point is $W_{i,j}$
12.               Define the coordinate of this retangular area from the
                  pixel positions in this borderline.
13.               Add the coordinate of area to list $B$.
14.     **End for**
15. **End for**
16. **For** all rectangular areas in the list **do**
17.     **For** each edge of the area **do**
18.         **If** the edge is close to the border of tabled block and there are no
            line positions from list $L$ lying between the border and the edge
            **then**
19.             Define this edge to have no line.
20.         **Else**
21.             Define this edge to have the line.
22.     **End for**
```

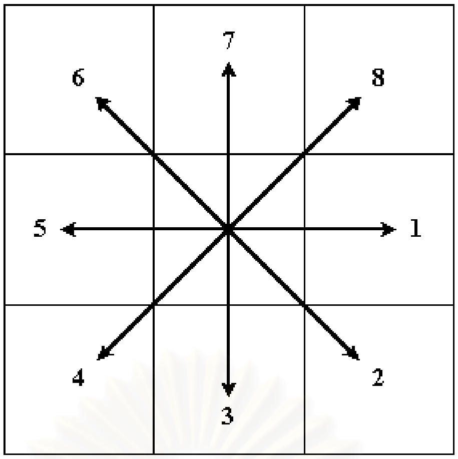


Figure 3.2: Illustration of the directions of the middle pixel.

```
End for
```

Next, the algorithm deletes the lines from a block and, then, uses the image smoothing algorithm on the block without line. Afterward, the coordinates of the data inside each rectangular area and the line informations are defined according to their borders as in the following algorithm:

Data Block Coordinate Defining Algorithm

```
List $D$ is empty.
For each rectangular area in list $B$ do
    For $i$ = the first row to the last row of this area do
        For $j$ = the first column to the last column of this area do
            If $B_{i,j}$is a black pixel and not in the ranges of the
            coordinates in list $D$ then
                Set $B_{i,j}$ to be the point of a new borderline.
```

```
7.                      Direction = 1.
8.                      If there are some black pixel around $B_{i,j}$ then
9.                          Do
10.                             Find the next pixel of this borderline.
10.1                                Start at the opposite of the last
                                    direction as in Figure 3.2.
10.2                                Find a black pixel around the point in
                                    clockwise direction.
10.3                                Stop when a black pixel is found.
11.                             Set this pixel to be the new point of
                                consideration.
12.                         Until the point is $B_{i,j}$
13.                         Define the coordinate of this data block from
                            the pixel positions in this borderline.
14.                         Add the coordinate of data block to list $D$.
15.             End for
16.     End for
17.     For all data blocks in the rectangular area do
18.             If the size is too small then
19.                     Remove it from list $D$.
20.     End for
21.     If there are more than one data blocks in the rectangular area then
22.             For all data blocks in the rectangular area do
23.                     For each edge of the data block do
24.                             If the edge position close to the border of
```

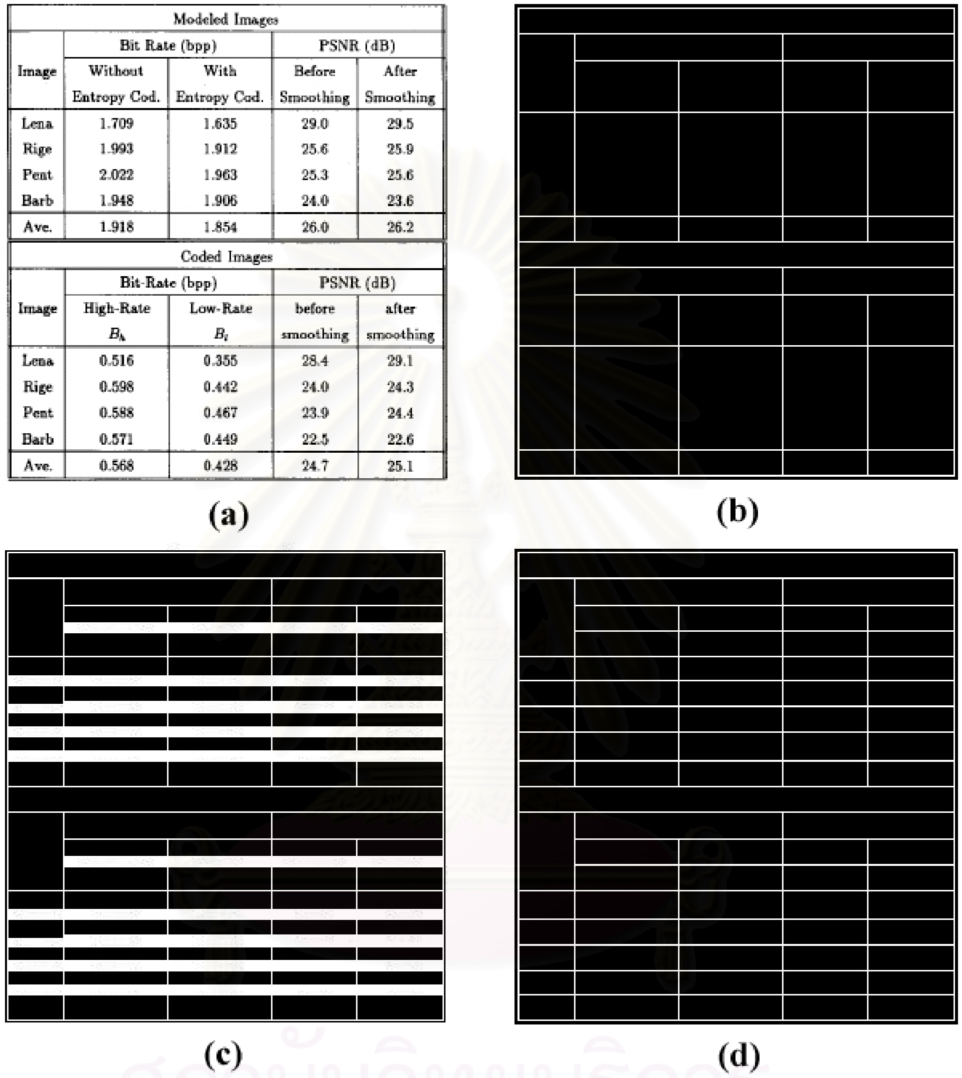
```
            rectangular area **then**
                Set the border as the edge.
            **Else**
                Define this edge to have no line.
        **End for**
    **End for**
**End for**
```

Every data block expands to form a rectangular cell that fits in the table. The boundary of the data block is designated by the adjacent data blocks in four directions (top, left, bottom, and right). Next, we find the relation of each data block to set the row or column. The data blocks with the same value of the left, right, or center position are assigned to the same column. Hence, the data blocks in the same column must have the same left and right boundary values. Similary, the data blocks in the same row must have the same top and bottom boundary values. There may be some areas in the block which are not covered by any table cells. In which case, we add those areas to the adjacent cells, provided that the edges of the adjacent cells do not have lines separating them. Figure 3.3 presents an example of tabled cells formation.

Table Cell Forming Algorithm

```
**For** each data block **do**
    **If** there are some data block over it **then**
        Set its top boundary = the bottom edge of the nearest data block
        on the top.
    Else
        Set its top boundary = 1.
    If there are some data block under it then
        Set its bottom boundary = the top edge of the nearest data block
        on the bottom.
    Else
        Set its bottom boundary = the number of pixel height.
    If there are some data block on the left of it then
        Set its left boundary = the right edge of the nearest data block on
        the left.
    Else
        Set its left boundary = 1.
    If there are some data block on the right of it then
        Set its right boundary = the left edge of the nearest data block on
        the right.
    Else
        Set its right boundary = the number of pixel width.
End for
Construct the rows of the table.
Construct the columns of the table.
For each row of the table do
    For each data block in this row do
        Set its top boundary = max (all top boundaries in this column).
        Set its bottom boundary = min (all bottom boundaries in this
        column).
    End for
End for
For each column of the table do
    For each data block in this column do
        Set its left boundary = max (all left boundaries in this column).
        Set its right boundary = min (all right boundaries in this column).
    End for
End for
While some edges are not equal to their boundary do
    For each data block do
        Decrease the value of its top edge.
        Decrease the value of its left edge.
        Increase the value of its bottom edge.
        Increase the value of its right edge.
    End for
End do
For each area is not covered by any table cells do
    If the edge of the cell that connects to the area does not have the line
    then
        Add this area to that cell.
End for
```

After table cells formation is completed, the cell coordinates, line and data information are stored as a text file. Figure 3.4 shows an example of this text file. The first row represents the number of cells and size of the block. The remaining rows store the coordinates line and data information for each cell. Each column is



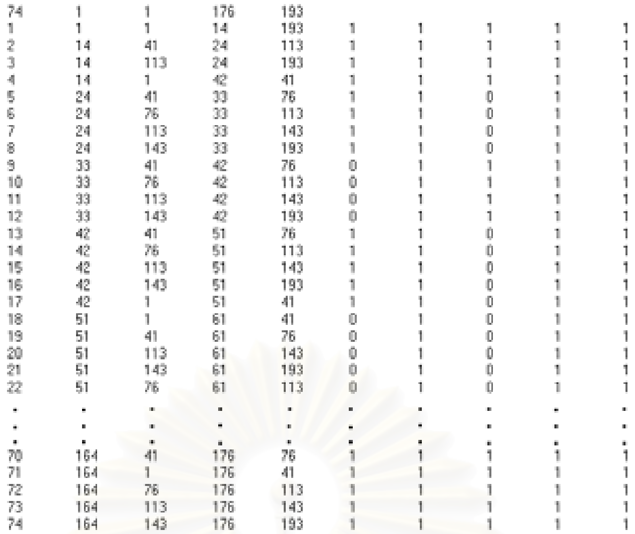
Modeled Images				
Image	Bit Rate (bpp)		PSNR (dB)	
	Without Entropy Cod.	With Entropy Cod.	Before Smoothing	After Smoothing
Lena	1.709	1.635	29.0	29.5
Rige	1.993	1.912	25.6	25.9
Pent	2.022	1.963	25.3	25.6
Barb	1.948	1.906	24.0	23.6
Ave.	1.918	1.854	26.0	26.2

Coded Images				
Image	Bit-Rate (bpp)		PSNR (dB)	
	High-Rate B_h	Low-Rate B_l	before smoothing	after smoothing
Lena	0.516	0.355	28.4	29.1
Rige	0.598	0.442	24.0	24.3
Pent	0.588	0.467	23.9	24.4
Barb	0.571	0.449	22.5	22.6
Ave.	0.568	0.428	24.7	25.1

(a) (b)

(c) (d)

Figure 3.3: An example of tabled cells formation: (a) a sample block, (b) the rectangular areas, (c) the coordinate boxes of the data blocks, (d) the coordinate boxes of the tabled cells.



```
74     1      1      176    193
1      1      1      14     193    1      1      1      1      1
2      14     41     24     113    1      1      1      1      1
3      14     113    24     193    1      1      1      1      1
4      14     1      42     41     1      1      1      1      1
5      24     41     33     76     1      1      0      1      1
6      24     76     33     113    1      1      0      1      1
7      24     113    33     143    1      1      0      1      1
8      24     143    33     193    1      1      0      1      1
9      33     41     42     76     0      1      1      1      1
10     33     76     42     113    0      1      1      1      1
11     33     113    42     143    0      1      1      1      1
12     33     143    42     193    0      1      1      1      1
13     42     41     51     76     1      1      0      1      1
14     42     76     51     113    1      1      0      1      1
15     42     113    51     143    1      1      0      1      1
16     42     143    51     193    1      1      0      1      1
17     42     1      51     41     1      1      0      1      1
18     51     1      61     41     0      1      0      1      1
19     51     41     61     76     0      1      0      1      1
20     51     113    61     143    0      1      0      1      1
21     51     143    61     193    0      1      0      1      1
22     51     76     61     113    0      1      0      1      1
.      .      .      .      .      .      .      .      .      .
.      .      .      .      .      .      .      .      .      .
.      .      .      .      .      .      .      .      .      .
70     164    41     176    76     1      1      1      1      1
71     164    1      176    41     1      1      1      1      1
72     164    76     176    113    1      1      1      1      1
73     164    113    176    143    1      1      1      1      1
74     164    143    176    193    1      1      1      1      1
```

Figure 3.4: An example of the text file for cell coordinates and line informations.

denoted by cell number, top position, left position, bottom position, right position, line information for top edge, left edge, bottom edge, right edge, and appearance of data in side the cell.

This text file is used for identifying the tabled blocks based on the arrangement of the cells and lines position. We initially assume every block is a tabled block. We employ two conditions to validate any non-tabled blocks. In the first condition, the cell of table should not overlap each other as in Figure 3.5. The second condition checks the appearance of lines. Cell C_i overlaps cell C_j in four cases.

1. The top position of C_i is between the top and bottom position of C_j while the bottom position of C_i is greater than the bottom position of C_j.

2. The bottom position of C_i is between the top and bottom position of C_j while the top position of C_i is less than the top position of C_j.

3. The left position of C_i is between the left and right position of C_j while the

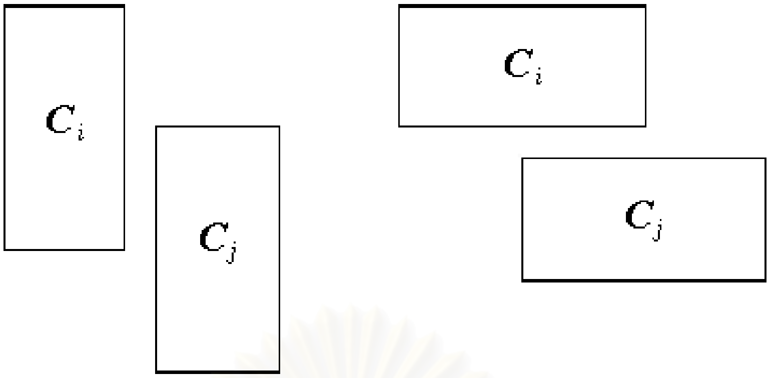


Figure 3.5: Illustration of the possibilities of cell overlappings.

right position of C_i is greater than the right position of C_j.

4. The right position of C_i is between the left and right position of C_j while the left position of C_i is less than the left position of C_j.

When we finish checking the cells overlapping, we will use the second condition for separating a matrix block from a tabled block. A matrix has two vertical lines on the left and right side of block and the arrangement of cells in the matrix is regular. The block identifying from its text file is explained as follows.

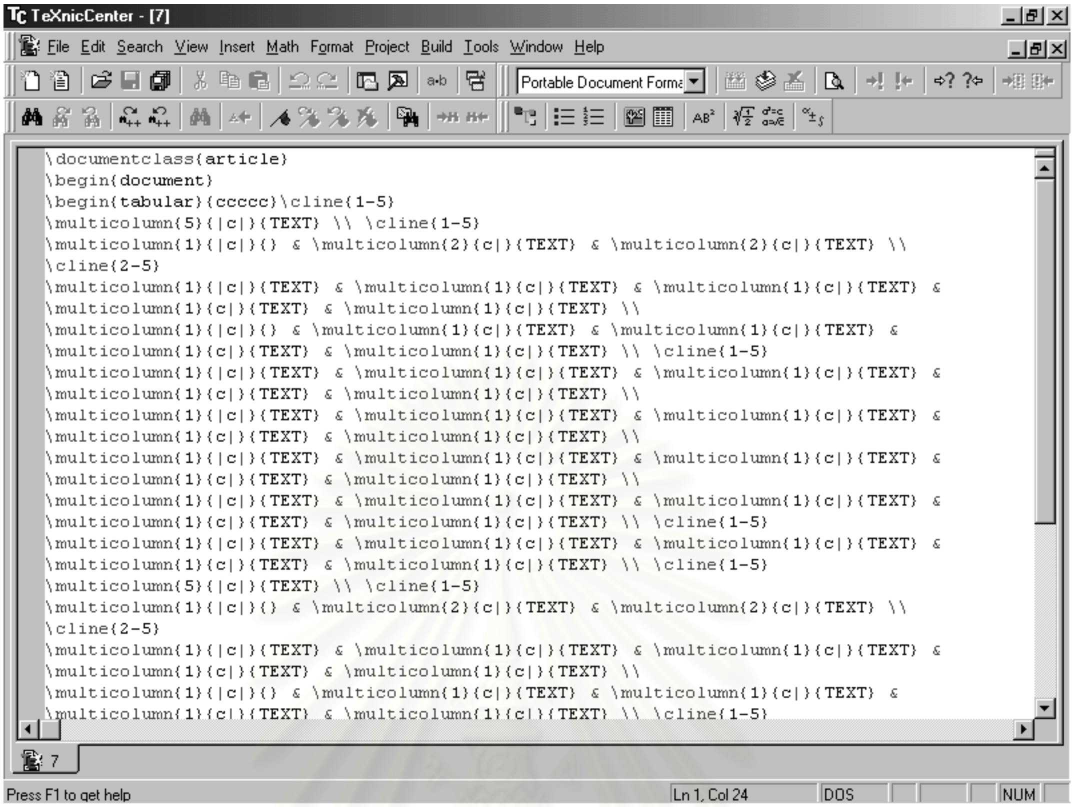
Table Identification Algorithm

```
1. Let $T = 1$.
2. For each tabled cell do
3.     If it overlaps the other tabled cells then
4.         $T = 0$.
5. End for
```

```
If $T = 0$ then
    This block is not a tabled block.
Else
    If there are only two lines on the left and right sides then
        This block is not a tabled block (Matrix).
    Else
        This block is a tabled block.
```

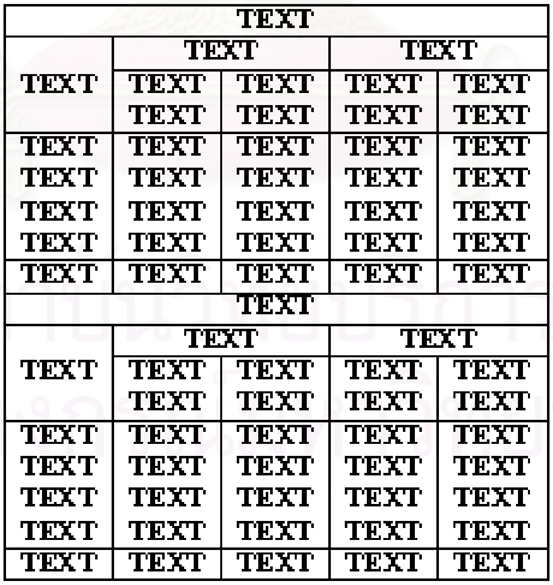
3.3 Converting Table into LaTeX Format

The tabled block, which is identified in the early step, will be processed in this step. The structure information in its text file is converted into LaTeX format. LaTeX can align text in columns as a table by using the tabular environment. From the left and right position in text file, we can determine number of columns that is necessary at the beginning of the tabular environment. We start a LaTeX code with "`\begin {tabular}{cccc}`" and follow with the command for the line on the top of table. The number of c in ccc is equal to the number of columns. The command for the line consists of two commands. We use the "`\hline`" command if there is a line along the row. If there are the lines in some cell in this row, we use the "`\cline`" command. Next, we consider the cells in each row for the number of columns and positions of lines. For each cell, we use the "`\multicolumn`" command that can present the cell width as the number of columns. A "`|`" is put between two cells if there is a vertical line in that position. A "`&`" is used for separating each cell. Each row is separated by the "`\\`" command. Figure 3.6 and Figure 3.7 shows an example of a LaTeX file and a table from it that is a result of this algorithm.



```
\documentclass{article}
\begin{document}
\begin{tabular}{ccccc}\cline{1-5}
\multicolumn{5}{|c|}{TEXT} \\ \cline{1-5}
\multicolumn{1}{|c|}{} & \multicolumn{2}{c|}{TEXT} & \multicolumn{2}{c|}{TEXT} \\
\cline{2-5}
\multicolumn{1}{|c|}{TEXT} & \multicolumn{1}{c|}{TEXT} & \multicolumn{1}{c|}{TEXT} &
\multicolumn{1}{c|}{TEXT} & \multicolumn{1}{c|}{TEXT} \\
\multicolumn{1}{|c|}{} & \multicolumn{1}{c|}{TEXT} & \multicolumn{1}{c|}{TEXT} &
\multicolumn{1}{c|}{TEXT} & \multicolumn{1}{c|}{TEXT} \\ \cline{1-5}
\multicolumn{1}{|c|}{TEXT} & \multicolumn{1}{c|}{TEXT} & \multicolumn{1}{c|}{TEXT} &
\multicolumn{1}{c|}{TEXT} & \multicolumn{1}{c|}{TEXT} \\
\multicolumn{1}{|c|}{TEXT} & \multicolumn{1}{c|}{TEXT} & \multicolumn{1}{c|}{TEXT} &
\multicolumn{1}{c|}{TEXT} & \multicolumn{1}{c|}{TEXT} \\
\multicolumn{1}{|c|}{TEXT} & \multicolumn{1}{c|}{TEXT} & \multicolumn{1}{c|}{TEXT} &
\multicolumn{1}{c|}{TEXT} & \multicolumn{1}{c|}{TEXT} \\
\multicolumn{1}{|c|}{TEXT} & \multicolumn{1}{c|}{TEXT} & \multicolumn{1}{c|}{TEXT} &
\multicolumn{1}{c|}{TEXT} & \multicolumn{1}{c|}{TEXT} \\ \cline{1-5}
\multicolumn{1}{|c|}{TEXT} & \multicolumn{1}{c|}{TEXT} & \multicolumn{1}{c|}{TEXT} &
\multicolumn{1}{c|}{TEXT} & \multicolumn{1}{c|}{TEXT} \\ \cline{1-5}
\multicolumn{5}{|c|}{TEXT} \\ \cline{1-5}
\multicolumn{1}{|c|}{} & \multicolumn{2}{c|}{TEXT} & \multicolumn{2}{c|}{TEXT} \\
\cline{2-5}
\multicolumn{1}{|c|}{TEXT} & \multicolumn{1}{c|}{TEXT} & \multicolumn{1}{c|}{TEXT} &
\multicolumn{1}{c|}{TEXT} & \multicolumn{1}{c|}{TEXT} \\
\multicolumn{1}{|c|}{} & \multicolumn{1}{c|}{TEXT} & \multicolumn{1}{c|}{TEXT} &
\multicolumn{1}{c|}{TEXT} & \multicolumn{1}{c|}{TEXT} \\ \cline{1-5}
```

Figure 3.6: An example of the LaTeX file.



TEXT				
	TEXT		TEXT	
TEXT	TEXT	TEXT	TEXT	TEXT
	TEXT	TEXT	TEXT	TEXT
TEXT	TEXT	TEXT	TEXT	TEXT
TEXT	TEXT	TEXT	TEXT	TEXT
TEXT	TEXT	TEXT	TEXT	TEXT
TEXT	TEXT	TEXT	TEXT	TEXT
TEXT	TEXT	TEXT	TEXT	TEXT
TEXT				
	TEXT		TEXT	
TEXT	TEXT	TEXT	TEXT	TEXT
	TEXT	TEXT	TEXT	TEXT
TEXT	TEXT	TEXT	TEXT	TEXT
TEXT	TEXT	TEXT	TEXT	TEXT
TEXT	TEXT	TEXT	TEXT	TEXT
TEXT	TEXT	TEXT	TEXT	TEXT
TEXT	TEXT	TEXT	TEXT	TEXT

Figure 3.7: An example of the table from LaTeX.

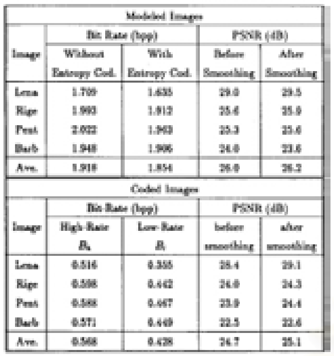
CHAPTER IV

EXPERIMENTAL RESULTS

The proposed algorithm is implemented by using C programming language. The tested samples consist of the blocks extracted from some actual document images and the synthesis sample blocks. Seven document images with less skewed angle and noises are extracted into rectangular blocks. These blocks consist of text paragraphs, picture, graphs, tables, and matrix that are identified into tabled block and non-tabled block.

The tabled blocks with the various styles of lines and data arrangement are synthesized for testing. Table 4.1 presents the tested samples that are correctly identified. The samples consisting of tabled areas and non-tabled areas are presented in Table 4.2 and Table 4.3. Table 4.4 presents the tested results of identification method. The most tabled blocks are correctly identified and converted. The comparisions between the tabled blocks and the table structures from LaTeX are shown in Table 4.5. The tables in Figure 4.1 is identified as a non-tabled block because of cells overlapping shown in the circles.

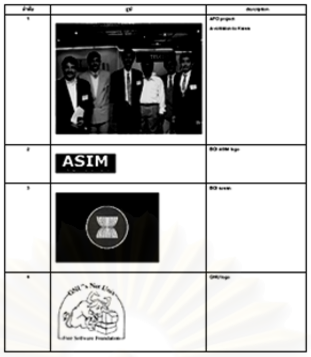
Table 4.1: Examples of the tested samples and their identification results.



Modeled Images				
Image	Bit Rate (bpp)		PSNR (dB)	
	Without Entropy Cod.	With Entropy Cod.	Before Smoothing	After Smoothing
Lena	1.709	1.635	29.0	29.5
Rige	1.993	1.912	25.6	25.9
Pent	2.022	1.963	25.3	25.6
Barb	1.948	1.906	24.0	23.6
Ave.	1.918	1.854	26.0	26.2

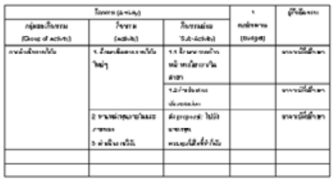
Coded Images				
Image	Bit Rate (bpp)		PSNR (dB)	
	High-Rate R_h	Low-Rate R_l	before smoothing	after smoothing
Lena	0.516	0.355	28.4	29.1
Rige	0.598	0.442	24.0	24.3
Pent	0.588	0.467	23.9	24.4
Barb	0.571	0.449	22.5	22.6
Ave.	0.568	0.428	24.7	25.1

Table

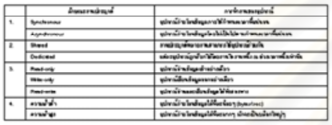


ASIM

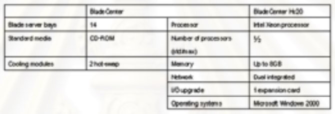
Table



Table



Table



	BladeCenter		BladeCenter HS20
Blade server bays	14	Processor	Intel Xeon processor
Standard media	CD-ROM	Number of processors (std/max)	½
Cooling modules	2 hot-swap	Memory	Up to 8GB
		Network	Dual integrated
		I/O upgrade	1 expansion card
		Operating systems	Microsoft Windows 2000

Table

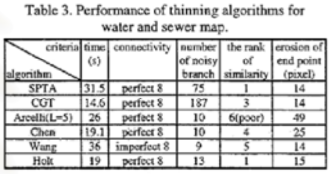


Table 3. Performance of thinning algorithms for water and sewer map.

criteria / algorithm	time (s)	connectivity	number of noisy branch	the rank of similarity	erosion of end point (pixel)
SPTA	31.5	perfect 8	75	1	14
CGT	14.6	perfect 8	187	3	14
Arcelli(L=5)	26	perfect 8	10	6(poor)	49
Chen	19.1	perfect 8	10	4	25
Wang	36	imperfect 8	9	5	14
Holt	19	perfect 8	13	1	15

Table

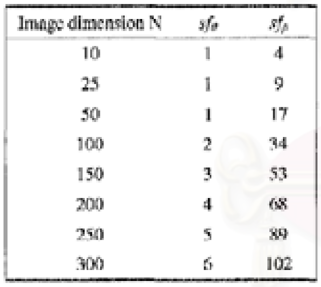
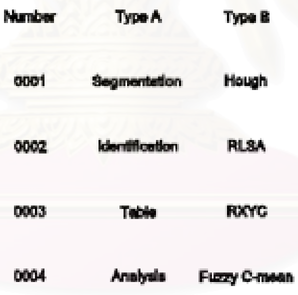


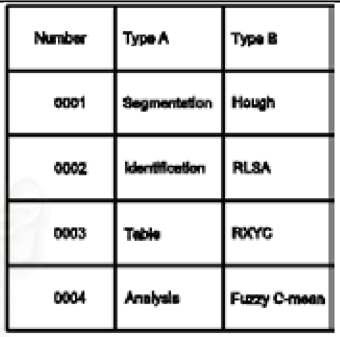
Image dimension N	sf_a	sf_b
10	1	4
25	1	9
50	1	17
100	2	34
150	3	53
200	4	68
250	5	89
300	6	102

Table



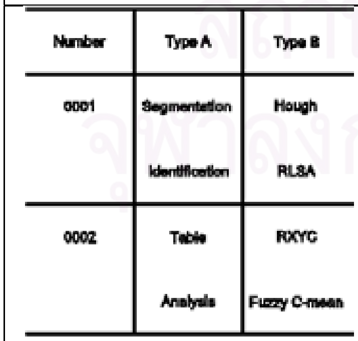
Number	Type A	Type B
0001	Segmentation	Hough
0002	Identification	RLSA
0003	Table	RXYC
0004	Analysis	Fuzzy C-mean

Table



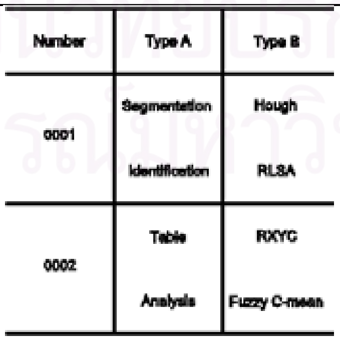
Number	Type A	Type B
0001	Segmentation	Hough
0002	Identification	RLSA
0003	Table	RXYC
0004	Analysis	Fuzzy C-mean

Table



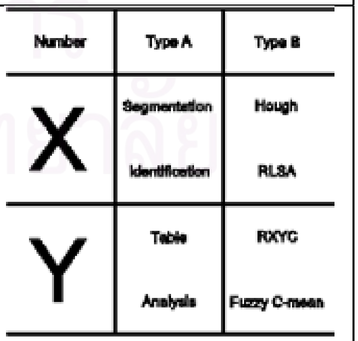
Number	Type A	Type B
0001	Segmentation Identification	Hough RLSA
0002	Table Analysis	RXYC Fuzzy C-mean

Table



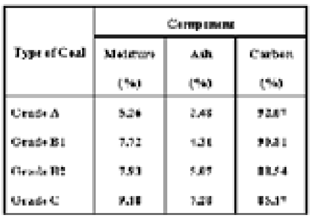
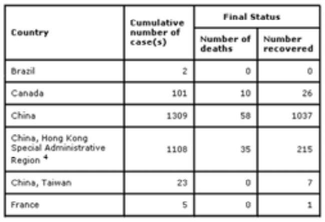
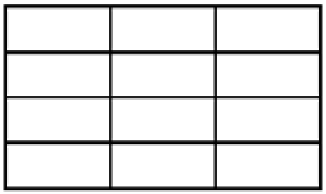
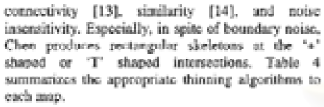
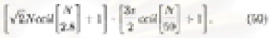
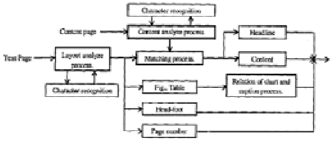
Number	Type A	Type B
0001	Segmentation Identification	Hough RLSA
0002	Table Analysis	RXYC Fuzzy C-mean

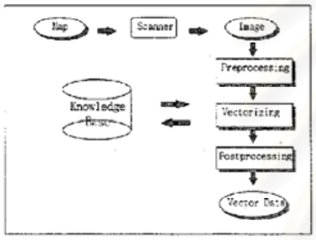
Table

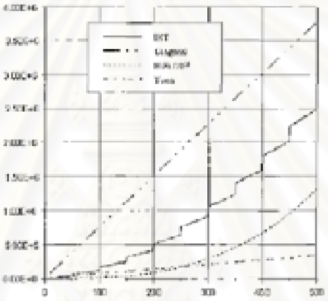


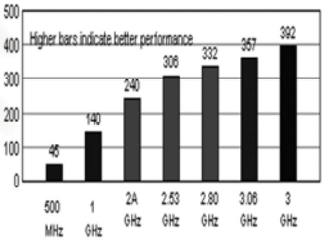
Number	Type A	Type B
X	Segmentation Identification	Hough RLSA
Y	Table Analysis	RXYC Fuzzy C-mean

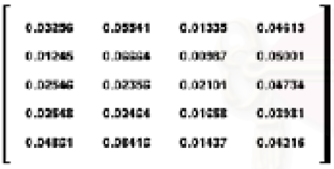
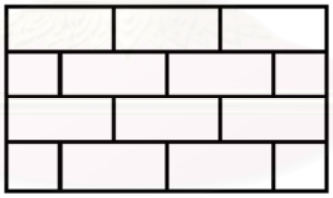
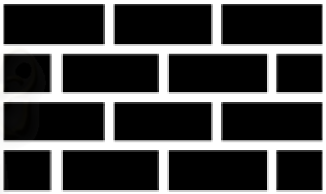
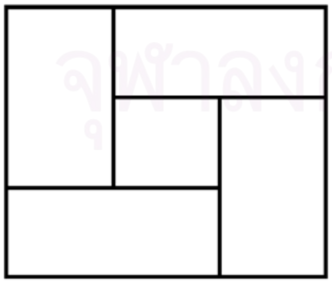
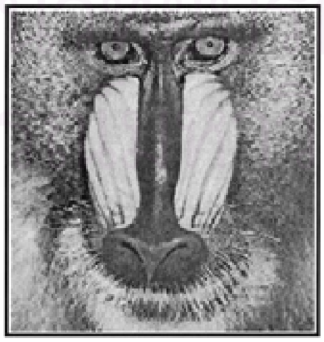
Table

<table>
<tr>
<td>

<table>
<tr><th rowspan="2">Type of Coal</th><th colspan="3">Component</th></tr>
<tr><th>Moisture
(%)</th><th>Ash
(%)</th><th>Carbon
(%)</th></tr>
<tr><td>Grade A</td><td>5.26</td><td>2.48</td><td>92.87</td></tr>
<tr><td>Grade B1</td><td>7.72</td><td>1.28</td><td>90.81</td></tr>
<tr><td>Grade B2</td><td>7.93</td><td>5.07</td><td>88.54</td></tr>
<tr><td>Grade C</td><td>8.18</td><td>7.28</td><td>85.17</td></tr>
</table>
</td>
<td>

<table>
<tr><th rowspan="2">Country</th><th rowspan="2">Cumulative number of case(s)</th><th colspan="2">Final Status</th></tr>
<tr><th>Number of deaths</th><th>Number recovered</th></tr>
<tr><td>Brazil</td><td>2</td><td>0</td><td>0</td></tr>
<tr><td>Canada</td><td>101</td><td>10</td><td>26</td></tr>
<tr><td>China</td><td>1309</td><td>58</td><td>1037</td></tr>
<tr><td>China, Hong Kong Special Administrative Region [4]</td><td>1108</td><td>35</td><td>215</td></tr>
<tr><td>China, Taiwan</td><td>23</td><td>0</td><td>7</td></tr>
<tr><td>France</td><td>5</td><td>0</td><td>1</td></tr>
</table>
</td>
<td>

</td>
</tr>
<tr><td>Table</td><td>Table</td><td>Table</td></tr>
<tr>
<td>

connectivity [13], similarity [14], and noise insensitivity. Especially, in spite of boundary noise, Chen produces rectangular skeletons at the '+' shaped or 'T' shaped intersections. Table 4 summarizes the appropriate thinning algorithms to each map.
</td>
<td>

$\left[\sqrt{2}N\mathrm{ceil}\left[\frac{N}{2.8}\right]+1\right]\cdot\left[\frac{3\pi}{2}\mathrm{ceil}\left[\frac{N}{50}\right]+1\right].$ (50)
</td>
<td>


</td>
</tr>
<tr><td>Non-table</td><td>Non-table</td><td>Non-table</td></tr>
<tr>
<td>


</td>
<td>


</td>
<td>


</td>
</tr>
<tr><td>Non-table</td><td>Non-table</td><td>Non-table</td></tr>
<tr>
<td>

$\begin{bmatrix} 0.03256 & 0.05541 & 0.01335 & 0.04613 \\ 0.01265 & 0.06664 & 0.00987 & 0.05001 \\ 0.02546 & 0.02356 & 0.02101 & 0.04734 \\ 0.03648 & 0.03424 & 0.01258 & 0.03921 \\ 0.04891 & 0.08416 & 0.01437 & 0.04216 \end{bmatrix}$
</td>
<td>

</td>
<td>

</td>
</tr>
<tr><td>Non-table</td><td>Non-table</td><td>Non-table</td></tr>
<tr>
<td>

</td>
<td>

</td>
<td>

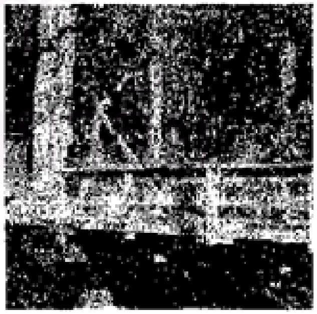



Fig. 4 Degraded image.
</td>
</tr>
<tr><td>Non-table</td><td>Non-table</td><td>Non-table</td></tr>
</table>

Table 4.2: The tabled samples.

Table	Components		
	Line	Data	Line and Data
Actual	-	-	17
Synthesis	1	2	16

Table 4.3: The non-tabled samples.

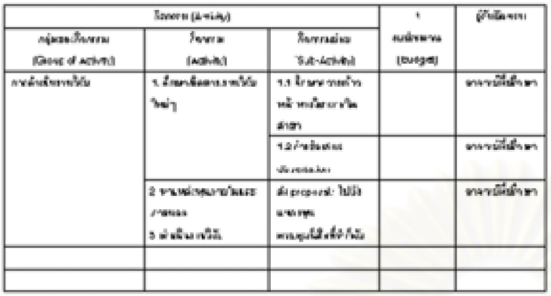
Non-table	Picture	Graph	Diagram	Matrix	Others
Actual	2	4	2	-	20
Synthesis	-	-	-	1	4

Table 4.4: The tested results of table identification.

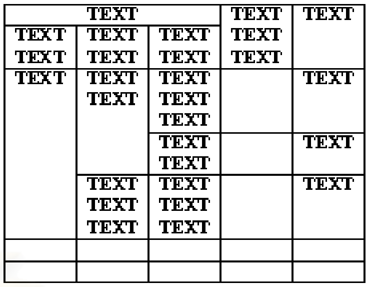
Samples		Number of tests	Results		Correct identification rate (%)*
			Correct	Miss	
Table	Actual	17	16	1	94.12
	Synthesis	19	19	-	100
Non-table	Actual	28	28	-	100
	Synthesis	5	5	-	100

*Correct identification rate = Number of correct results/Number of tests

Table 4.5: The comparision of the tabled blocks and their table structures from LaTeX.



[illegible]			[illegible] (budget)	[illegible]
[illegible] (Group of activity)	[illegible] (Activity)	[illegible] (Sub-Activity)		
[illegible]	1. [illegible]	1.1 [illegible]		[illegible]
		1.2 [illegible]		[illegible]
	2 [illegible] 3 [illegible]	[illegible] proposal: [illegible]		[illegible]



TEXT			TEXT	TEXT
TEXT TEXT	TEXT TEXT	TEXT TEXT	TEXT TEXT	
TEXT	TEXT TEXT	TEXT TEXT TEXT		TEXT
		TEXT TEXT		TEXT
	TEXT TEXT TEXT	TEXT TEXT TEXT		TEXT

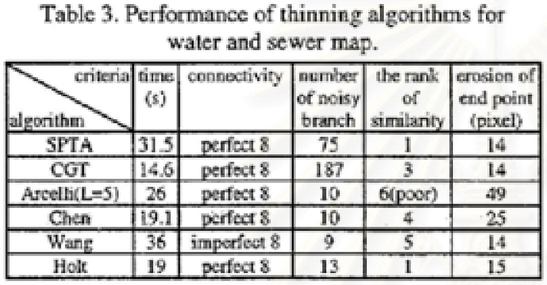
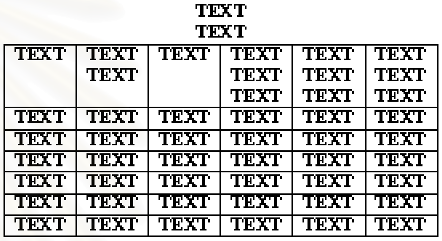


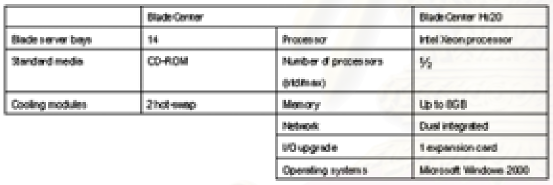
Table 3. Performance of thinning algorithms for water and sewer map.

criteria algorithm	time (s)	connectivity	number of noisy branch	the rank of similarity	erosion of end point (pixel)
SPTA	31.5	perfect 8	75	1	14
CGT	14.6	perfect 8	187	3	14
Arcelli(L=5)	26	perfect 8	10	6(poor)	49
Chen	19.1	perfect 8	10	4	25
Wang	36	imperfect 8	9	5	14
Holt	19	perfect 8	13	1	15

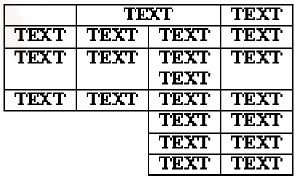


TEXT
TEXT

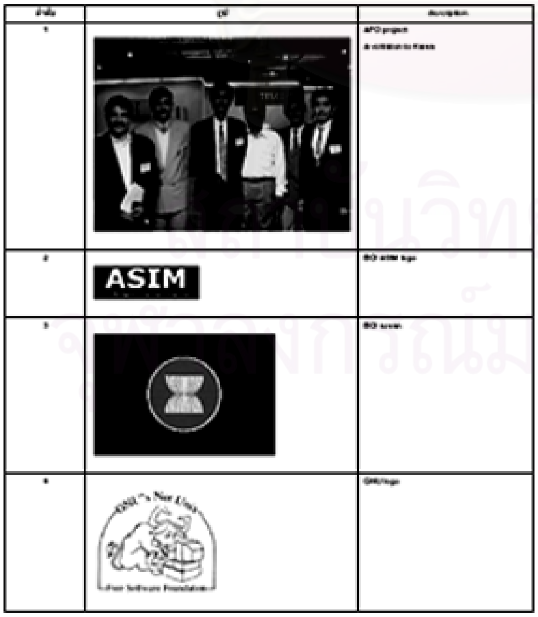
TEXT	TEXT TEXT	TEXT	TEXT TEXT TEXT	TEXT TEXT TEXT	TEXT TEXT TEXT
TEXT	TEXT	TEXT	TEXT	TEXT	TEXT
TEXT	TEXT	TEXT	TEXT	TEXT	TEXT
TEXT	TEXT	TEXT	TEXT	TEXT	TEXT
TEXT	TEXT	TEXT	TEXT	TEXT	TEXT
TEXT	TEXT	TEXT	TEXT	TEXT	TEXT
TEXT	TEXT	TEXT	TEXT	TEXT	TEXT



	Blade Center		Blade Center HS20
Blade server bays	14	Processor	Intel Xeon processor
Standard media	CD-ROM	Number of processors (std/max)	½
Cooling modules	2 hot-swap	Memory	Up to 8GB
		Network	Dual integrated
		I/O upgrade	1 expansion card
		Operating systems	Microsoft Windows 2000

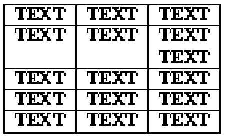


	TEXT		TEXT
TEXT	TEXT	TEXT	TEXT
TEXT	TEXT	TEXT TEXT	TEXT
TEXT	TEXT	TEXT	TEXT
		TEXT	TEXT
		TEXT	TEXT
		TEXT	TEXT

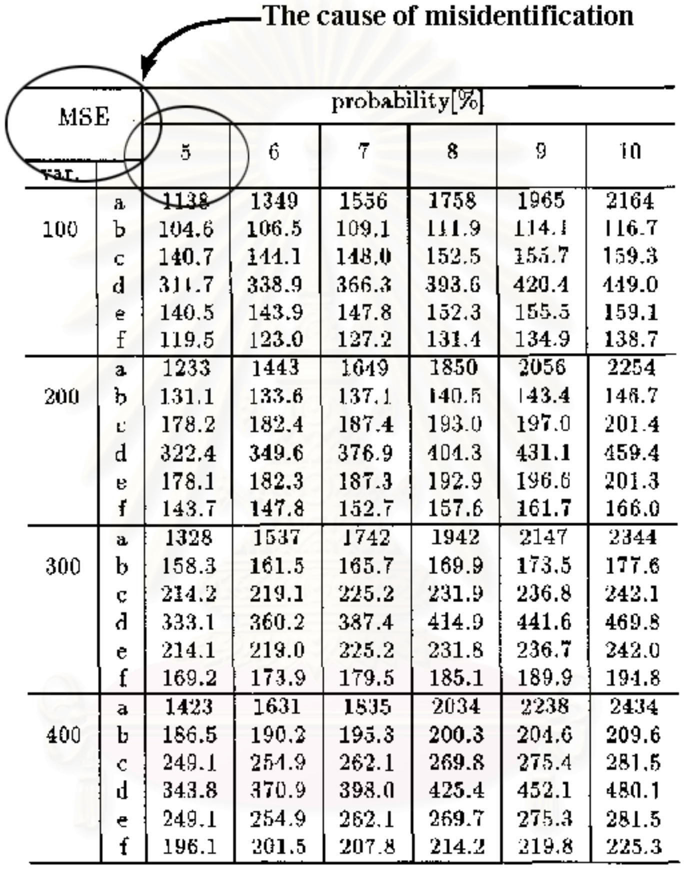


[illegible]	[illegible]	description
1		APO project exhibition in France
2		BOI ASIM logo
3		BOI [illegible]
4		GNU logo

ASIM
GNU's Not Unix
Free Software Foundation



TEXT	TEXT	TEXT
TEXT	TEXT	TEXT TEXT
TEXT	TEXT	TEXT
TEXT	TEXT	TEXT
TEXT	TEXT	TEXT



The cause of misidentification

MSE		probability[%]					
var.		5	6	7	8	9	10
100	a	1138	1349	1556	1758	1965	2164
	b	104.6	106.5	109.1	111.9	114.1	116.7
	c	140.7	144.1	148.0	152.5	155.7	159.3
	d	311.7	338.9	366.3	393.6	420.4	449.0
	e	140.5	143.9	147.8	152.3	155.5	159.1
	f	119.5	123.0	127.2	131.4	134.9	138.7
200	a	1233	1443	1649	1850	2056	2254
	b	131.1	133.6	137.1	140.5	143.4	146.7
	c	178.2	182.4	187.4	193.0	197.0	201.4
	d	322.4	349.6	376.9	404.3	431.1	459.4
	e	178.1	182.3	187.3	192.9	196.6	201.3
	f	143.7	147.8	152.7	157.6	161.7	166.0
300	a	1328	1537	1742	1942	2147	2344
	b	158.3	161.5	165.7	169.9	173.5	177.6
	c	214.2	219.1	225.2	231.9	236.8	242.1
	d	333.1	360.2	387.4	414.9	441.6	469.8
	e	214.1	219.0	225.2	231.8	236.7	242.0
	f	169.2	173.9	179.5	185.1	189.9	194.8
400	a	1423	1631	1835	2034	2238	2434
	b	186.5	190.2	195.3	200.3	204.6	209.6
	c	249.1	254.9	262.1	269.8	275.4	281.5
	d	343.8	370.9	398.0	425.4	452.1	480.1
	e	249.1	254.9	262.1	269.7	275.3	281.5
	f	196.1	201.5	207.8	214.2	219.8	225.3

Figure 4.1: An example of a non-tabled block.

CHAPTER V

CONCLUSION

A new algorithm is proposed in order to identify various styles of table and converts the table structure into a LaTeX format suitable for modification, storage, retrieval, and transmission. This algorithm considers the line appearance and the position of cells inside the table for identifying the tabled area. Thus, the matrix is identified as a non-tabled area. In the case of the table without any lines, the algorithm uses only the position of cells as the variable for decision.

From the experimental results, this algorithm can correctly identify many types of table. However, if there exist some small areas of a considered table containing some non-tabled areas such as that in Figure 4.1, this algorithm cannot identified that correctly.

Finally, it could be concluded that the algorithm for table identification by considering the arrangement of cells and the appearance of lines gives 94.12 % of the correct identification rate for the actual tabled areas and 100 % for the synthesis ones. And this algorithm is suitable for converting the table structure into LaTeX format. The further study can be considered in the following issues.

1. It is possible to develop a faster method for preparing the information of cell and line positions.

2. The algorithm should be developed for supporting the tables consisting of any non-tabled areas.

3. More information of block are needed for a good representation from LaTeX.

REFERENCES

[1] Fletcher L. A., and Kasturi R., A robust algorithm for text string separation from mixed text/graphic images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 10, no. 6, pp. 910-918, 1988.

[2] Gonzalez R. C., and Woods R. E., *Digital Image Processing*, Addison-Wesley Publishing Company, pp.432-438, 1992.

[3] Lebourgeois F., Bublinski Z., and Emptoz H., A fast and efficient method for extracting text paragraphs and graphics from unconstrained documents, *Proceedings of the 11th International Conference on Pattern Recognition*, Vol. 2, pp. 272-276, 1992.

[4] Wahl F. M., Wong K. Y., and Casey R. G., Block segmentation and text extraction in mixed text/image document, *Computer Graphics and Image Processing*, Vol. 20, pp. 375-390, 1982.

[5] Antonacopoulos A., and Ritchings R. T., Segmentation and classification of document images, *IEE Colloquium on Document Image Processing and Multimedia Environments*, pp. 1-7, 1995.

[6] Mital D. P., and Leng G. W., Text segmentation for automatic document processing, *IEEE Conference on Emerging Technologies and Factory Automation*, Vol. 2, pp. 642-648, 1996.

[7] Trupin E., and Lecourtier Y., A modified contour following algorithm applied to document segmentation, *Proceedings of the 11th International Conference on Pattern Recognition*, Vol. 2, pp. 525-528, 1992.

[8] Saitoh T., and Pavlidis T., Page segmentation without rectangle assumption, *Proceedings of the 11th International Conference on Pattern Recognition*, Vol. 2, pp. 277-280, 1992.

[9] Chuai-aree S., Lursinsap C., Sophatsathit P., and Siripant S., Fuzzy C-mean: a statistical feature classification of text and image segmentation method, *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, Vol. 9, No. 6, pp. 661-671, 2001.

[10] Klir G. J., and Yuan B., *Fuzzy Sets and Fuzzy Logic Theory and Applications*, Prentice Hall, 1995.

[11] Shih F. Y., Chen S. S., Hung D. D., and Ng P. A., A document segmentation, classification and recognition system, *Proceedings of the Second International Conference on Systems Integration*, pp. 258-267, 1992.

[12] Shih F. Y., and Chen S. S., Adaptive document block segmentation and classification, *IEEE Transactions on Systems Man and Cybernetics*, Vol. 26, pp. 797-802, 1996.

[13] Amamoto N., Torigoe S., and Hirogaki Y., Block segmentation and text area extraction of vertically/horizontally written document, *Proceedings of the*

Second International Conference on Document Analysis and Recognition, pp. 739-742, 1993.

[14] Papamarkos N., Tzortzakis J., and Gatos B., Determination of run-length smoothing values for document segmentation, *Proceedings of the Third International Conference on Electronics Circuits and Systems*, Vol. 2, pp. 684 -687, 1996.

[15] Wang H., Li S. Z., and Ragupathi S., Document segmentation and classification with top-down approach, *Proceedings of the First International Conference on Knowledge-Based Intelligent Electronic Systems*, pp. 243-247, 1997.

[16] Chan K. C., Huang X., and Bao P., Fuzzy segmentation for document image analysis, *IEEE International Conference on Systems Man and Cybernetics*, Vol. 2, pp. 997-982, 1997.

[17] Hirayama Y., A Block Segmentation Method for Document Images with Complicated Column Structures, *Proceedings of the Second International Conference on Document Analysis and Recognition*, pp. 91-94, 1993.

[18] Jain A. K., and Yu B., Document representation and its application to page decomposition, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 3, pp. 294-308, 1998.

[19] Hirayama Y., A method for table structure analysis using DP matching, *Proceedings of the Third International Conference on Document Analysis and Recognition*, Vol. 2, pp. 583-586, 1995.

[20] Wang Y., Phillips I. T., and Haralick R., Automatic table ground truth generation and a background-analysis-based table structure extraction method, *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, pp. 528-532, 2001.

[21] Chandran S., and Kasturi R., Structural recognition of tabulated data, *Proceedings of the Second International Conference on Document Analysis and Recognition*, pp. 516-519, 1993.

[22] Itonori K., Table structure recognition based on textblock arrangement and ruled line position, *Proceedings of the Second International Conference on Document Analysis and Recognition*, pp. 765-768, 1993.

[23] Green E., and Krishnamoorthy M., Model-based analysis of printed tables, *Proceedings of the Third International Conference on Document Analysis and Recognition*, Vol. 1, pp. 214-217, 1995.

[24] Arias J. F., Chhabra A., and Misra V., Interpreting and representing tabular documents, *Proceedings of International Conference on Computer Vision and Pattern Recognition*, pp. 600-605, 1996.

[25] Watanabe T., Luo Q., and Sugie N., Layout recognition of multi-kinds of table-form documents, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 4, pp. 432-445, 1995.

[26] Saiga H., Kitamaru Y., and Ida S., High-speed recognition of tabulated data, *Proceedings of the 12th International Conference on Pattern Recognition*, Vol. 2, pp. 577-579, 1994.

[27] Zuyev K., Table Image Segmentation, *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, Vol. 2, pp. 705-708, 1997.

[28] Tsuruoka S., Takao K., Tanaka T., Yoshikawa T., and Shinogi T., Region segmentation for table image with unknown complex structure, *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, pp. 709-713, 2001.

[29] Amano A., Asada N., Motoyama T., Sumiyoshi T., and Suzuki K., Table form document synthesis by grammar-based structure analysis, *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, pp. 533-537, 2001.

BIOGRAPHY

San Sethasopon was born in 1976. He received a Bachelor Degree of Science in Chemical Engineering from Chulalongkorn University.