1-1-2003

Recognition and identification of form document layouts

Kai Luo

University of Nevada, Las Vegas

Follow this and additional works at: <https://digitalscholarship.unlv.edu/rtds>

Repository Citation

Luo, Kai, "Recognition and identification of form document layouts" (2003). *UNLV Retrospective Theses & Dissertations*. 1602.

<http://dx.doi.org/10.25669/gdhk-py26>

This Thesis is protected by copyright and/or related rights. It has been brought to you by Digital Scholarship@UNLV with permission from the rights-holder(s). You are free to use this Thesis in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you need to obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/or on the work itself.

This Thesis has been accepted for inclusion in UNLV Retrospective Theses & Dissertations by an authorized administrator of Digital Scholarship@UNLV. For more information, please contact digitalscholarship@unlv.edu.

RECOGNITION AND IDENTIFICATION OF FORM DOCUMENT LAYOUTS

by

Kai Luo

Bachelor of Science
Southeast University, Nanjing, P.R.China
1998

A thesis submitted in fulfillment
of the requirements for the

**Master of Science Degree in Electrical Engineering
Department of Electrical and Computer Engineering
Howard R Hughes College of Engineering**

**Graduate College
University of Nevada, Las Vegas
December 2003**

UMI Number: 1417779

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI[®]

UMI Microform 1417779

Copyright 2004 by ProQuest Information and Learning Company.

All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

Thesis Approval

The Graduate College
University of Nevada, Las Vegas

November 20, 2003

The Thesis prepared by

Kai Luo

Entitled

Recognition and Identification of Form Documents Layouts

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering

Examination Committee Chair

Dean of the Graduate College

Examination Committee Member

Examination Committee Member

Graduate College Faculty Representative

ABSTRACT

Recognition and Identification of Form Document Layouts

by

Kai Luo

Dr. Shahram Latifi, Examination Committee Chair
Professor of Electrical and Computer Engineering Department
University of Nevada, Las Vegas

In this thesis, a hierarchical tree representation is introduced to represent the logical structure of a form document. But different forms might have the same logical structure, so the representation will be ambiguous. In this thesis, an improvement is proposed to solve the ambiguity problem by using the physical information of the blocks. To fulfill the application of hierarchical tree representation and extract the physical information of blocks, a pixel tracing approach is used to extract form layout structures from form documents. Compared with hough transform, the pixel tracing algorithm requires less computation. This algorithm has been tested on 50 different table forms. It effectively extracts all the line information required for the hierarchical tree representation, represents the form by a hierarchical tree, and distinguishes the different forms. The algorithm applies to table form documents.

TABLE OF CONTENT

ABSTRACT	iii
TABLE OF CONTENT	iv
LIST OF FIGURES	v
ACKNOWLEDGMENTS	vi
CHAPTER 1 INTRODUCTION	1
CHAPTER 2 BACKGROUND AND RELATED WORK.....	4
Review of Frame Line Detection Method	4
Hierarchical Representation of Form Document	6
Related Research	7
CHAPTER 3 LINE EXTRACTION AND PREPROCESSING OF THE LAYOUT	12
Preprocessing	12
Line Extraction Algorithm	14
A Special Case.....	24
CHAPTER 4 HIERARCHICAL REPRESENTATION OF FORM DOCUMENTS	27
Definitions and Approach	27
The Heuristic Algorithm.....	28
Experiments	33
Application Areas of the Algorithm	38
Improvement of the Algorithm.....	41
CHAPTER 5 CONCLUSIONS AND FUTURE WORKS.....	45
BIBLIOGRAPHY.....	50
VITA.....	52

LIST OF FIGURES

Figure 1	The Form Document Processing System Diagram	2
Figure 2	Typical elements in the sample table form.....	13
Figure 3	A sample form after skew correction	14
Figure 4	Attaching lines after skew correction	15
Figure 5	The form after attaching line segments deletion algorithm.....	17
Figure 6	The form after thinning procedure	19
Figure 7	Line crossing as a result of thinning.....	19
Figure 8	Hanging lines.....	20
Figure 9	The form after deleting hanging lines and redrawing crosses	21
Figure 10	A picture in form document	22
Figure 11	The final form after processing	24
Figure.12	A special case with a box structure	25
Figure.13	The form document after the box deletion processing	26
Figure 14	An ambiguity case	31
Figure 15	The hierarchical tree of sample forming Figure 3	36
Figure 16	Another sample of form document.....	37
Figure 17	The extracted layout for the sample in Figure 16.....	37
Figure 18	The hierarchical tree of the sample in Figure 16.....	38
Figure 19	A form only with frame lines	39
Figure 20	A form whose layout lines have no connection with frame lines.....	40
Figure 21	A form that contains a table and other elements	41
Figure 22	Two different form presented by the same structure.....	42
Figure 23	The form have the same logical structure with sample 3	43
Figure 24	A document that has only frame structure.....	46
Figure 25	The blocks creation of a non-block structure form	47
Figure 26	The hierarchical tree of the form which has a pseudo block structure.....	48

ACKNOWLEDGEMENTS

I want to thank Dr. Latifi, my advisor for his incessant guidance and precious advice on my study and research at University of Nevada, Las Vegas.

I would also like to thank to my Examination Committee: Dr. Regentova, Dr. Selvaraj, and Dr. Taghva for their valuable suggestions and comments on this thesis. I would like to give special thanks to Dr. Regentova, who gave me so much help on the research.

I wish to express my great appreciation to Dr. Chensong Dai, Cheng De and Dr. Jun Zheng for their help and suggestion in my research work.

In addition, this work was supported by Dr Taghva. I want to show my special thanks to him, who provided me the financial support during the last year.

Finally, I would like to dedicate this thesis to my parents, for their support and encouragement.

CHAPTER 1

INTRODUCTION

Forms are widely used in many business and government organizations. Although paper-based forms are still the most widely used way to get information, today electronic filings are becoming prevalent. The traditional way to get the data from the form is time-consuming and inefficient. Consequently, some new technologies are developed to automatize the procedure, such as OCR (Optical Character Recognition).

Several form document processing systems have been developed in recent years. Taylor, Fritzson and Pastor presented [14] a system for manipulating preprinted forms. They used 10 corners as features for form registration and field description. In their system, a form image was first partitioned into nine equal rectangular blocks, and the numbers of each type of corner in each block were counted. The feature vectors were then fed into a neural network for recognition. Fujisawa, Nakaro and Kurino [8] presented a pattern-oriented segmentation method for separating touching handwritten characters. Fan, Lu, Wang [7] presented a clustering-based form recognition method.

Normally in an OCR form document recognition system there are six major steps in the system (Figure 1). In this thesis, the algorithm for layout extraction and representation procedures is presented.

Figure 1 The Form Document Processing System Diagram

There are several challenging problems in form processing such as: layout extraction, input form registration and characters recognition, which entwine each other. Before OCR is used to recognize the characters, the form layout is needed to make the OCR procedure more effective.

In this thesis, the following problem is considered: given a form image database and a query image, how does one find the form in the database which has the same or similar layout structure as the query?

Normally a form has these elements: horizontal and vertical layout lines, printed data, and user filled-in data. Printed data includes characters, symbols, checkbox and pictures. Our task is to get the information of the layout lines and at the same time get rid of these printed data as well as user filled-in data. Then with the lines of information the structure of the form in a hierarchical tree is represented.

The major objectives of the thesis are to develop an effective algorithm for line extraction from the form documents and to apply a hierarchical tree on these lines of data to represent the form structures without ambiguity.

The thesis is organized into five chapters. The background and related research are described in Chapter 2. The line detection and extraction are derived in Chapter 3. The application of the hierarchical tree structure is described in Chapter 4. Finally, discussion and future directions are summarized in Chapter 5.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, an overview of form document line extraction method and representation of form structure are presented. In section 2.1, a brief review of form document line extraction is given. An introduction to a hierarchical representation of form document is presented in section 2.2.

2.1 A Review of Frame Line Detection Method

There are many kinds of widely used line detection methods and every method is customized to handle some specific cases. For example, Hough transform is a popular method to detect lines. It is widely used to detect lines; one of its properties is to detect dash and broken lines. Especially when noise exists, Hough transform can eliminate noise and get the real line information by setting the threshold.

But there are some weaknesses for this method. First of all, Hough transform [9] is a very time consuming algorithm. Form structure extraction is just one step in form document processing system; thus practical systems should spend little time on it. Secondly, it is hard to locate the position of some short lines in the forms because all the line segments become points in Hough space after the Hough transformation. When these points are retrieved to our original coordinate system, only endless lines can be obtained. So if a line segment is short, it is hard to locate its position. Lastly, sometimes characters

connect each other and form a long string, Hough transform will recognize the string as a line. User filled-in data and pictures in the form also cause the same effect.

Because of these limits, some improved Hough transform are proposed, such as strip projection approach [3]. The frame lines on forms are horizontal or vertical. In the transformed space, the search range can be narrowed to a small area of around 0 and 90 degrees. This way, the efficiency of the algorithm is improved. By stripping a form document, positions of short line segments can be located and projections can be used to detect the length of the line segment. This approach fits some situations very well, but it still faces the problem of how to differentiate character strings and pictures from lines in some cases. For example, when the frame lines and the character strings have almost the same length, the projection of the lines and character strings will have nearly the same length of projections. Another example is when a form is striped, it is possible that the printed pictures in the form stride a strip and lines in the picture have the length with the layout lines. In such a case, it is hard to differentiate these lines.

There are also some other approaches to detect the frame line segments. Connected component analysis is used to detect horizontal and vertical lines. X. N. Chen and D.C. Tseng [4] presented a method according to the lengths and aspect ratios of components. It is more efficient than hough transform in detecting the horizontal and vertical lines, but it cannot handle lines touching characters or other data. Bin Yu and A. K. Jain [17] present a vectorization algorithm which uses a block adjacency graph. But most form lines are horizontal and vertical lines. Algorithms that can detect lines in arbitrary directions and characters are not necessary in form-processing system.

The methods used to detect frame lines depend on specific cases. In some types of forms, some specific methods may be better than others, but in other situations, this method might not be efficient. It also depends on what kind of frame line information is desired. If only the horizontal and vertical frame lines information is needed, diagonal lines must be omitted.

In this thesis, only information of horizontal and vertical frame lines is needed based on the need of the hierarchical tree algorithm. So here the dash lines, diagonal lines, and non-frame lines will be removed. It is assumed that the form processed is non-distorted, that it is rotated to horizontal, and that all the frame lines are not cursive.

2.2 A Hierarchical Representation of Form Document

To perform an automatic recognition of form, user filled data must be extracted and associated with the corresponding preprinted field. In order to perform this task, the structure should be known first. Then the structure can be compared with the reference model form in the database.

When a reference model form is stored in the database and the input form is compared with reference model forms, these forms need to be represented in a structure. The direct images of these forms are the most straightforward representatives of these form documents. Unfortunately, there are many problems with this method. First of all, the storage space form image is big. If there are thousands of reference model forms in the database, the images will occupy a huge space. Secondly, the physical information of a form might be changed during the processing. Geometry of a form changes when documents are enlarged, shrunk and rotated. This situation happened a lot when people

process documents, such as printing and scanning. Lastly, there will be some noise when the document is processed. Noise becomes part of the image, which will mismatch the form with the reference model form.

A logical structure can represent a form structure. A geometrical structure will be mapped to a logical structure by considering the logical relation. So when a form structure is represented by a logical structure, the geometrical changes in the processing procedure will not influence the logical structure. But it will also bring a problem that sometimes physically different forms might have the same logical structure. This will cause the mismatch of the forms.

In this thesis, a hierarchical tree is used to represent the form documents structure. This method is proposed by Pinar Duygulu and Volkan Atalay [5, 6]. The heuristic algorithm is based on the XY-tree method. The geometrical structure of a form document is transformed to a hierarchical structure by using the horizontal and vertical layout lines which extracted through the form. So the geometrical modifications can be handled by this representation. In some cases, physically different forms will have the same logical representations.

2.3 Related Research

The main interest in form document systems is to extract user filled-in data and associate it with the corresponding preprinted field. In order to extract the user filled-in data, the structure of the form should be known in advance and the input instance form should be registered with one of the reference model forms in the database. A reference form is a blank form on which no user filled-in data exist. To store a reference model

form in the database and match an input form with a reference form in the database, a common representation is needed to perform the task. An input form is registered with one of the model forms in the form database if their structures are identical. Then required data can be automatically extracted following the specification given for the reference model form.

To effectively register the input form with the reference model form, the representation of the form structure becomes the key. Rather than its direct image, a form can be represented by the physical features of its components. For example, length, width, the position of the cross points [14] and the position of horizontal and vertical lines can be used. But these characters are not very convenient to represent the form. Slight variations produced in the processing of form cannot be handled very well. There are many studies in which the main aim is to classify the form documents. Liu [13] describe a method to use horizontal and vertical lines to extract the blocks of a given form. But they do not take into account the relationship among blocks. Further more, they do not give a representation scheme for the extracted blocks. Watanabe, Luo and Sugie [15] proposed an analytical method for recognizing the layout structures of table-form documents. In their approach, form structures were described according to mutual relationships among line segments. Three binary trees are constructed. Two of them are for the global and local structure. The last one is for classification. In addition to the horizontal and vertical lines, some of preprinted data are used. The method is found to be hard to apply to analysis of filled-in form, because it is considered to be limited to empty fields. Ishitani [10] uses a hierarchical matching strategy based on sub-graph matching, local matching stages by line matching, and the interactions between them. But the

similarity measure totally depends on the number of vertical and horizontal lines. Furthermore, there is no explicit representation of a form document.

Pinar Duygulu and Volkan Atalay [5, 6] describe a hierarchical representation of form documents. In this approach, no domain knowledge such as the preprinted data or user filled-in data is used. Slight modifications and variations are handled by the representation. In this thesis, the method is used to represent the form structure. But this property also causes a problem. Sometimes physically different forms might have the same logical structure. There might be several reference model forms with the same representation. The algorithm must be used in combination with form physical information in some cases. Except the hierarchical tree representation, the physical information about relation of the parent block and children block is added. When the representation of input form matches the unique reference model in the database, the matching procedure will be stopped. If not, the physical information will be compared to choose the real matched reference model. This method not only solves the problem of ambiguity, but also makes the system more efficient and save time.

To represent a form document with a hierarchical tree, layout of the form document needs to be known in advance. The purpose of form-structure extraction is to determine the construction lines in a particular form using image-analysis methods. A number of techniques such as Hough transform, vectorization and contour detection have been proposed. Jiun-lin Chen and His-jian Lee [3] proposed a Hough transform algorithm for extraction using strip projection. The efficiency of the algorithm is improved compared with traditional Hough transform. This approach fits some situations very well, but it still faces the problem of how to differentiate character strings and pictures from lines in some

cases. And because of the property of Hough transform, it is time consuming and occupies the resource of computer system. X. N. Chen and D.C. Tseng [4] presented a method according to the lengths and aspect ratios of components. In their method, a parallel line for each straight line is generated. Through the use of the geometric property on a pair of parallel lines, the parameter sets of those lines possibly on image can be obtained immediately. It is more efficient than Hough transform in detecting the horizontal and vertical lines, but it cannot handle lines touching characters or other data. Jinhui Liu and A.K.Jain [12] develop a prototype form retrieval system based on the proposed similarity measure. They use a block adjacency graph to recognize the frame lines and characters. In the work of Yefei Zheng [18], form frame line is detected by directional single-connected chain (vectorization). Directional single-connected chain bears appropriate size and can be easily stored and processed, in addition to the capability to solve most types of character-line crossing problems. But most form lines are horizontal and vertical lines. Algorithms that can detect lines in arbitrary directions and characters are not necessary in form-processing system.

Form structure extraction is just one step in form document processing system; so the efficiency is very important in the practical system. Based on the need of the hierarchical tree structure, only horizontal and vertical lines will be extracted. Hough transform, vectorization and contour detection are all time consuming methods. A more direct and efficient line extraction method is described in the thesis. The algorithm operates on a sequence of pixels, i.e. the rows of the bitmap. Compared with hough transform, the pixel tracing algorithm required less computation. Because of its easy implementation, it can be widely employed with some modifications. The most important property is that it can

extract the line information directly. Without analyzing useless information of the form (tracing the contour of the characters and transforming the image to Hough space), the efficiency of the system is highly improved.

CHAPTER 3

LINE EXTRACTION AND PREPROCESSING OF THE LAYOUT

3.1 Preprocessing

This thesis focuses on form layout extraction and layout representation procedures. The algorithm developed here is only suitable for table documents. The sample form (Figure 3) picked here is 662x1024 with 677888 bytes, which is a very typical size for a table form document. A typical table form has three elements: form layout lines, printed data and user filled-in data. A form document system's task is to extract user filled-in data and associate it with the corresponding preprinted field. For this reason, the structure of the form should be known in advance. Printed data include characters, symbols, checkbox, hanging lines and pictures. The user filled-in data might have crosses with the layout lines which can be seen frequently in user filled forms. The sample image includes all these elements (Figure 2).

Recital Parade

(a) picture

(b)checkbox

APPROPRIATE DEPARTMENT

(c) hanging lines

(d) printed data

UNLV

University of

(e) user filled-in data

(f) the crosses of filled-in data and layout lines

Figure. 2 Typical elements in the sample table form

The sample image is not very noisy and there is no distortion in the form. If the form is moderate skew, some skew detection and correction algorithms [1, 2, 11, 16] can be used to correct the skew. Figure 3 shows a typical table form after the skew correction procedure; all the frame layout lines are horizontal and vertical.

Reginal Nevada		OCCUPATIONAL MEDICINE REFERRAL	
REFERRED TO:	University of Nevada, Las Vegas	REFERRING PHYSICIAN:	03/18/93
EMPLOYEE:	Dr. Chen	WORK ADDRESS:	University of Nevada, Las Vegas
SOCIAL SECURITY NUMBER:	578 674 368	EMPLOYEE:	UNLV
DEPARTMENT:	Medical Care	DATE OF REFERRAL:	04/26/93
REASON: Eyes and hands were burned by chemical Liquid.			
PLEASE RETURN ONE COPY OF THIS REPORT AND BILLING TO THE BILLING COMPANY ADDRESS: <div style="display: flex; justify-content: space-between;"> <div> ☐ CDS Corp. First 1250 East Tropicana, Suite 300 Las Vegas, NV 89119 (702) 740-8020 </div> <div> ☐ Occupational Medicine Post Office Box 510 Mercury, NV 89023 (702) 478-6224 </div> </div>			

Figure.3 A sample form after skew correction

3.2 Line Extraction Algorithm

After the skew correction, there are some short line segments attached to the original lines. So the first step is to erase these attached line segments or unite these segments to a long line segment by setting up the length threshold of lines. Figure 4 shows two lines on which lines segments attach. In the first line, because the attached line segments are very

short, they will be eliminated. On the second lines, the attached line segments are long and the spaces between them are short. The segments will be connected to a long line segment. The detail values of the thresholds can be adjusted to fit respective forms.

Figure 4 Attaching lines after skew correction

Here a flag is used to tag the line. If the pixel is black and the flag=0, this means this pixel is the beginning of the line segment. The flag is set to 1 and then the next pixel is checked. If it is black, then the next pixel is checked. If it is white, it is considered the end pixel of the line segment and the flag is set to 0. If one line segment's end pixel is in the range of three pixels of another line segment's beginning pixel, these two segments are connected to one segment. If not, the length of segment is checked. As long as the length is longer than the threshold length, this pixel chain is considered to be a line segment; otherwise, it will be eliminated.

Algorithm

(y, x) is the position of pixel in the image. Here the algorithm for horizontal lines is provided.

For y = 1 To the number of the pixels on Y direction Do

 Flag = 0

 For x = 1 To the number of pixels on X direction Do

 If the pixel is black and Flag = 0 Then

Define the pixel as the start point of the current line segment

Flag = 1

Else If the pixel is white and Flag = 1 Then

Define the pixel as the end point of the current line segment

Flag = 0

If the distance between the current line segment's start point and the last
line segment's end point < line broken threshold Then

Connect the two line segments to a new line segment

Else If the length of the current line segment > line segment length
threshold Then

Define the current line segment as a new line segment

Else discard the current line segment

The vertical lines can be processed by the same way as horizontal lines. Figure 5 is
the image after the processing.

Figure.5 the form after attaching line segments deletion algorithm

Then a form with multi-pixel lines is obtained. In the next step, the thinning algorithm is performed from top to bottom. Every time a horizontal line is found, all lines one pixel above this line will be checked. If they share the same start point, the upper line will be merged to the lower line. A threshold is set to eliminate the influence of thick vertical lines. Vertical lines are processed in the same way.

For y = 1 To the number of the pixels on Y direction Do

Flag = 0

For x = 1 To the number of pixels on X direction Do

If the pixel is black and Flag = 0 Then

Define the pixel as the start point of the current line segment

Flag = 1

Else If the pixel is white and Flag = 1 Then

 Define the pixel as the end point of the current line segment

 Flag = 0

 If there is at least one line segment on the position which is one pixel
 above the current position Then

 Find all the line segments whose start point is at the same x position
 as those of line segments one pixel above

 If there are no such lines Then

 Define the current line segment as a new line segment

 Else

 Merge the current line segment with the line segment one pixel
 above it to form a new line segment

 Else

 Define the current line segment as a new line segment L

Figure. 6 The form after thinning procedure

After the thinning procedure (Figure 6), horizontal and vertical lines which connect on one end cross each other and extend further out (Figure 7) and so this effect needs to be eliminated to achieve the original line connection situation.

Figure 7 Line crossing as a result of thinning

There are also some hanging lines which are not the frame lines in the form (Figure 8). Some are from the picture of the form, some are the effect of the continuous characters because of the previous processing, and some are the fill-in blank hanging lines. These lines are not needed in later hierarchy tree algorithm. But some of these lines have a common property, which is that any of these lines is crossed by only one other line or does not cross with other lines. The algorithm is developed based on this property. Some non-frame lines are eliminated and the crossing problem caused by thinning algorithm is solved in the algorithm below. The result shows in Figure 9.

Figure 8 Hanging lines

For $i = 1$ To the number of horizontal lines Do

Find all vertical lines which have a cross with the current horizontal line

If (there is no vertical line across the current horizontal line) Or (there is only one vertical line across the current horizontal line) Then

The current horizontal line is a hanging line

Else

The start point of current horizontal line = the position of the first vertical line
crossing the current horizontal line

The end point of current horizontal line = the position of the last vertical line
crossing the current horizontal line

Header section with drawing			

Figure.9 The form after deleting hanging lines and redrawing crosses

Up until now, the only factor needed to be deleted is the pictures in the form. The pictures can be drawings or large characters (Figure 10). After the previous processing, they become some vertical and horizontal lines.

Figure 10 A picture in form document

The difference between these lines and frame lines is that these lines do not have connections with the outermost frames lines by use of the pathfinder. Any lines belonging to the layout lines must have a connection with the outermost frame or connect the outermost frame by other lines. Based on this property, a connectivity algorithm is developed. Firstly, the outermost horizontal line is picked and all the vertical lines that have a connection with it are found. (The top of vertical lines meet the horizontal line.) Then all the horizontal lines which have a connection with these vertical lines are searched. By this way, all the lines that connect the outermost frame lines or connect outermost frame lines through other layout lines are found. The pictures in the form are successfully eliminated by this algorithm.

i =1

j =1

Lhn = [the first horizontal line]

Lvn = [the first vertical line]

While i <= the number of lines in queue Lhn

and j <= the number of lines in queue Lvn Do

If j <= the number of lines in queue Lvn Then

Find all horizontal lines whose start points are on the current vertical line

If $i \leq$ the number of lines in queue Lhn Then

Find all vertical lines whose start points are on the current horizontal line

If $i \leq$ the number of lines in queue Lhn

$i = i + 1;$

If $j \leq$ the number of lines in queue Lvn Then

$j = j + 1;$

If there are horizontal lines whose start points are on the current vertical line Then

Add these horizontal lines to the queue of Lhn

If there are vertical lines whose start points are on the current horizontal line Then

Add these vertical lines to the queue of Lvn

Figure. 11 The final form after processing

After the above algorithms, the printed data and user filled-in data are successfully eliminated. Printed data includes characters, symbols, checkbox and pictures (Figure 11).

3.3 A Special Case

Based on the hierarchical tree algorithm, the tree can be constructed according to the lines of information. There are some lines that need to be eliminated. In the form Figure. 14, on the left corner of the form there is a box. In the hierarchical tree algorithm, which will be stated in the next chapter, the box cannot be correctly expressed in the tree structure. So here, the two non-frame border lines of the box must be eliminated.

In the four borders of the box (Figure.12), two of them are different from other lines of the form (the two borders with arrows). In a form, except the outermost four lines of frame, no other lines have just one end-to-end joint. Layout lines will not connect another layout lines' ends by its ends. Based on this property, the algorithm is developed to eliminate the two border lines of the box.

Figure.12 A special case with a box structure

After the processing, the two borders of the box are eliminated. The result is shown in Figure .13.

Figure.13 the form document after the box deletion processing

CHAPTER 4

HIERARCHICAL REPRESENTATION OF FORM DOCUMENTS

4.1 Definitions and Approach

After the extraction procedures, the layout of the form consists of only horizontal and vertical lines. The user filled-in data, preprinted data and non-frame lines are already filtered by the previous work. A hierarchical tree is used to represent the logical layout of a form by using these extracted lines. First some definitions are given before the algorithm is presented.

A block is defined as a rectangular area which is surrounded by the longest horizontal and vertical lines. A block frame is the horizontal and vertical lines that surround a block and the lines which have the same length as the borders of the block frame are defined as the baselines. Orthogonal lines are the lines which are orthogonal to the baselines, which start at any baseline and end at another baseline. For the overlapping orthogonal lines, the one with the maximum length is taken as the orthogonal line.

The procedure of building the hierarchical tree is to partition a form into blocks which can further be divided until the cell is reached. So the root of the tree is the form, which is the biggest block. Every leaf represents a cell. The blocks in the same hierarchy and under the same root node share the same baseline as their boundaries. To achieve such a partition, a switching of horizontal and vertical divisions is performed. By this way,

finally the cell will be reached. The approach is top-down and completed by an algorithm to extract a hierarchical representation of a given form document.

4.2 The Heuristic Algorithm

The algorithm consists of three parts: initialization, block finding and tree construction. The index order of lines extracted from the form is from top to down and from left to right.

4.2.1 Definitions

Here the definition of the block' boundary is provided. The four boundaries consist of the current baseline, the ending baseline and the parent block's boundaries. Because the direction is switched, the boundaries also depend on the block's direction.

A cell is defined as the smallest block which only consists of a block frame. A block frame is the horizontal and vertical lines that surround a block. To judge whether the block is a cell, the extracted lines information can be used. If there are no lines that cross through the current block, the block is a cell.

4.2.2 Algorithm

4.2.2.1. Initialization

The first part is to initialize the root block to the frame form.

The current block = Form

the direction of the current block = horizontal

Create a queue of BlockQ

/* add the first block to the queue */

BlockQ = BlockFinding (the current block);

TreeConstruction(BlockQ, the current block)

4.2.2.2. Block Finding

The second step is block finding, which includes building baseline queue, finding the ending baseline and block finding. The algorithm is given below:

Firstly, all the baselines for the current block need to be found because the length of lines in the current block is compared. Some lines will be cut to be limited in the current block size.

Blocks are defined by the current baseline and the ending baseline, so the determination of the ending baseline plays the key role in this algorithm. The information of all the orthogonal lines of the baselines is used to determine the ending baseline.

When there is no orthogonal line at the current baseline, the next baseline is defined as the ending baseline. Otherwise, the shortest basic orthogonal line at the current baseline defines the ending baseline. If there are baselines shared among several orthogonal lines, then all the orthogonal lines information needs to be combined to determine the ending baseline.

Findendingbaseline (extracted baselines for current block, the current baseline, the ending baseline)

If there is no orthogonal line starting at current baseline Then

Line = 0

Else

Line = 1

For i = 1 To the number of other orthogonal lines for the succeeding baselines Do

If (the end of the shortest orthogonal line starting at the current baseline

> the start of the current orthogonal line for the succeeding baselines)

And (the end of the shortest orthogonal line starting at the current baseline

< the end of the current orthogonal line for the succeeding baselines)

Then

Line = 2

Else

linetemp = the current orthogonal line for the succeeding baselines

Line = 3

Case Line

0: the ending baseline = the next baseline of current baseline

1: the ending baseline = the baseline which is on the end of the shortest

orthogonal line starting at the current baseline

2: the ending baseline = the baseline which is on the end of the longest orthogonal

line starting at the current baseline

3: the ending baseline = the baseline which is on the start of linetemp

Return the ending baseline

There is an ambiguity case in the algorithm when the current block consists of only the orthogonal lines with the same length equal to the length of the borders of the block frame. In such a case, only vertical lines and horizontal lines do not provide enough information about the division of the block. In Figure 14, the baseline is horizontal. There are two different ways to express the block in tree structures.

Figure. 14 An ambiguity case

The case can be handled by making a definition. In the case above, the second one as the representation is chosen.

After the ending baseline is obtained, the block can be defined in the following algorithm and the block is inserted in the queue.

Blockfinding(BlockQ, current block)

Construct a baseline queue of current block($b_1, b_2 \dots b_n$)

$i = 1$

While ($i < \text{the number of extracted baselines for the current block}$) Do

current baseline = b_i

Findendingbaseline(extracted baselines for the current block, the current baseline,
the ending baseline)

Define a block(the current baseline, the ending baseline, current block)

Add the new block to a queue Block Q

Add the new block's physical information to a queue of Blocks' physical
information

current baseline = the ending baseline

While (i < the serial number of the ending baseline in the queue) Do

i = i + 1

4.2.2.3. Tree Construction

The third part is tree construction. After finding the blocks, these blocks will be inserted to the tree. This procedure performs the insertion until the cells are reached. As long as the block is not a cell and the procedure is going on, the direction of the block is switched (turn around 90 degrees), and a new block finding procedure is pursued.

TreeConstruction(BlockQ, the current block)

While the queue of blockQ is not empty

DeQueue BlockQ to get a dequeued block

Insert the dequeued block to a Tree under the Parent block node

If the dequeued block is not a cell Then

the direction of the block is switched

Blockfinding(BlockQ, the dequeued block)

TreeConstruction(BlockQ, the dequeued block)

4.2.3 Experiments

This algorithm has been tested on more than 50 different table forms. These forms were scanned with a little bit skew (<5 degree). They were also processed by a skew correction algorithm. The images were not very noisy and were binary. The algorithm is coded in Matlab running on an Intel Celeron 800 computer with 256M memories. The sizes for the form show in Figure 3 are 662x1024 with 677888 bytes. It takes approximately 70 seconds to extract the frame horizontal and vertical lines. It takes another 2 seconds to construct the hierarchical tree. The result of the sample form is:

Node 1: parent: 0

dataptr: 1

children: [2 3 6 9 12 15 16]

Node 2: parent: 1

dataptr: 2

children: []

Node 3: parent: 1

dataptr: 3

children: [4 5]

Node 4: parent: 3

dataptr: 9

children: []

Node 5: parent: 3

dataptr: 10

children: []

Node 6: parent: 1

dataptr: 4

children: [7 8]

Node7: parent: 6

dataptr: 11

children: []

Node 8: parent: 6

dataptr: 12

children: []

Node 9: parent: 1

dataptr: 5

children: [10 11]

Node 10: parent: 9

dataptr: 13

children: []

Node 11: parent: 9

dataptr: 14

children: []

Node 12: parent: 1

dataptr: 6

children: [13 14]

Node 13: parent: 12

dataptr: 15

children: []

Node 14: parent: 12

dataptr: 16

children: []

Node 15: parent: 1

dataptr: 7

children: []

Node 16: parent: 1

dataptr: 8

children: []

In the result, every node has three fields: parent, children and dataptr. Field “parent” points to current node’s parent node. Field “children” shows its children nodes. Field “dataptr” points to the node’s position in another array that contains physical information of the block. Here a tree is drawn to make the structure more clearly (Figure 15).

Figure 15 The hierarchical tree of sample form in Figure.3

A more complicated case is showed in Figure 16. The image is 800x600. The resolution is 300dpi. It takes 60s to finish the process. The image after line extraction procedure is showed in Figure 17. The hierarchical tree is presented in Figure 18.

Figure 18 The hierarchical tree of the sample in Figure 16

4.2.4 Application Areas of the Algorithm

The algorithm is developed based on table form documents, so it is suitable for form documents that have the following layout characteristics:

- 1) A form has two horizontal frame lines serving as top and bottom boundaries, two vertical frame lines serving as left and right boundaries
- 2) Most of the useful data items in a form are enclosed by rectangles formed by frame lines and most of the user filled-in data are in the reserved block. Otherwise it will influence the OCR procedure.
- 3) A form is divided to some blocks by some straight lines.

For example, a document only has a frame structure. There is no blocks division in the frame. Based on the algorithm, only frame lines can be extracted and there is no

hierarchical tree for this form. In such a case, this algorithm cannot provide valuable information for OCR procedure. The form in Figure 19 has no frame lines and cannot be handled by this algorithm.

PART I: DATA FORM (continued) **CR-2** **11**

1. Employee Name
Employee's name must state position, place, and year full-time work began in the institution.

This form **Location** **Reason**

Responsibilities

2. Employee Name
Employee's name must state position, place, and year full-time work began in the institution.

This form **Location** **Reason**

Responsibilities

3. Employee Name
Employee's name must state position, place, and year full-time work began in the institution.

This form **Location** **Reason**

Responsibilities

4. Employee Name
Employee's name must state position, place, and year full-time work began in the institution.

This form **Location** **Reason**

Responsibilities

Figure.19 A form only with frame lines

- 4) Most of the layout lines must have connections with the outermost frame or connect to the outermost frame lines by other layout lines. Figure 20 shows a sample that cannot be handled by the algorithm. The form consists of three tables and it will not be recognized by the algorithm.

Figure.20 A form whose layout lines have no connection with frame lines

5) A mixed form (Figure. 21) will cause mismatching. Here “a mixed form” means a form that contains tables and other elements which are outside the table. For example in the form Figure 21 below, two documents both contain table structures. The tables have the same logical structure if they are represented by hierarchical trees. But one of the tables not only contains table, but also other elements outside the table. Unfortunately, other information outside the table will be eliminated when the algorithm is performed. In such a case, a mismatch will happen.

Figure. 21 A form that contains a table and other elements

4.2.5 Improvement of the Algorithm

In a hierarchical tree, a geometrical structure will be mapped to a logical structure by considering the logical relation. When a form structure is represented by a logical structure, the geometrical changes in the processing procedure will not influence the logical structure. The hierarchical tree representation will not cause mismatches even when the geometrical change happened in the layout extraction procedure. But this property also causes a problem. Sometimes physically different forms might have the same logical structure. How to differentiate these cases becomes a problem. The algorithm must be improved or be used in combination with form physical information in some cases. For example, in Figure 22, there are two different forms, but they have the

same logical structure. They are presented by the same hierarchical tree. This could cause mismatch in the logical structure.

Figure 22 Two different form presented by the same structure

Another ambiguity case can be found in the sample form. The form (Figure 23) below has a different physical structure with the sample in Figure 15. But they have the same logical representation.

Figure. 23 The form has the same logical structure with sample in Figure 3

In these situations, the representatives of logical trees cannot differentiate these forms. The method must be combined with physical structure information of the form. In the algorithm of line extraction, the physical information of horizontal and vertical lines is obtained. Combining the physical information and logical tree to represent the structure of the form will be a solution to solve the ambiguity problem.

In the line extraction algorithm, a line's start point, endpoint and position in a coordinate system can be obtained, so the detailed information of every block can be obtained through the four borders. In the hierarchical tree, every node's length and width can be calculated through its borders. The ratio of the width of children node to the length of parent node can be used to differentiate these ambiguity cases. Because there are some geometrical changes when forms are processed, for example the thinning algorithm will

cause the new line not to be in the middle of the original thick lines. To adjust the small changes, a range can be set to fit small physical changes.

CHAPTER 5

CONCLUSIONS AND FUTURE WORKS

In this thesis, an algorithm is proposed to extract form layout structures from form documents. By eliminating the preprinted data, user filled-in data, check boxes and the picture in the form, the algorithm extracts the horizontal and vertical lines information only. Because lines in arbitrary directions and hanging lines are useless in the hierarchical tree representation, the extracted lines only include the layout horizontal and vertical lines. With the extracted lines information, a hierarchical structure is used to represent the logical structure of the form. The blocks which contain similar information are grouped together and separated from other groups.

Because the form structure is represented by a logical tree, forms with different physical structure may be represented by the same logical tree. To solve the ambiguity cases, physical information of the blocks can be combined with logical tree to make a further recognition of the forms. But when the small blocks are reached, the solution will not be very effective. Because the lines' geometrical information might be changed in the procedure of line extraction, the ratio of the width of children node to the length of parent node will be less and less accurate when the blocks become smaller and smaller. A small position change of the lines in a small block will cause the ratio to change a lot. In such a case, the solution will not be very effective any more. But this situation only happens in

very few cases. For example, compared with the size of the form, blocks in the form are very small.

This algorithm is developed for table form documents, but for other types of document this algorithm cannot handle well. For example, a document which only has a frame structure (Figure 24). There is no blocks division in the frame. Based on the algorithm, only frame lines can be extracted and there is no hierarchical tree representation for this form. In such a case, all the important information about the form is lost. The future work will be focused on how to recognize this type of form.

Figure 24 A document that has only frame structure

In this type of form, there are many hanging lines. User filled-in data are above these lines. Every hanging line corresponds to some specific user filled-in data. In another word, if information about these lines can be represented by a structure, it will provide valuable information to OCR procedure. In this type of form, many line segments are in the same horizontal level. This property provides a way to build a structure that is the same as the one used in this thesis. These line segments can be stretched to connect to the frame lines in horizontal direction. In the vertical direction, pseudo line segments are created to connect with horizontal frame layout lines or other horizontal lines.

Figure 25 The blocks creation of a non-block structure form

In this type of form, all the user filled-in data are above the hanging lines. Because of this property of the form, the pseudo blocks creation procedure is from up to down. In Figure. 25, pseudo vertical line segments are added from one end of hanging lines to other line segments or pseudo line segments which are right above the hanging lines. This way, the document will be divided to many pseudo blocks. User filled-in data will be in the corresponding pseudo blocks. These blocks can be represented by a hierarchical tree structure (Figure.26)

Figure 26 The hierarchical tree of the form which has a pseudo block structure

The main interest in form document systems is to extract user filled-in data and associate them with the corresponding preprinted fields. In this type of form, user filled-in data correspond to hanging lines, which represent preprinted fields. Every pseudo block contains only one hanging line. The extracted user filled-in data will be correctly associated with the corresponding preprinted fields. The future work will be focused on the implement of this method.

BIBLIOGRAPHY

- [1] Amin, A., Fischer, S., "A Document Skew Detection Method Using the Hough Transform", PAA(3), No. 3 2000, pp. 243-253.
- [2] Amin, A., Fischer, S., Parkinson, A.F., Shiu, R., "Comparative-Study of Skew Detection Algorithms", JEI(5), No. 4, October 1996, pp. 443-451.
- [3] JIUN-LIN CHEN and HIS-JIAN LEE, "An Efficient Algorithm for Form Structure Extraction Using Strip Projection", Pattern Recognition, Vol.31, No.9, 1998, pp. 1353-1368
- [4] X. N. Chen and D.C.Tseng, "Form structure extraction for table-form recognition", Proc, 8th IPPR Conf. Computer Vision, Graphics and Image Processing, Taiwan, pp. 496-503.
- [5] Pinar Duygulu, Volkan Atalay "A Hierarchical Representation of Form Documents for Identification and Retrieval", International Journal on Document Analysis and Recognition IJDAR 5, 2002 1, pp,17-27
- [6] Pinar Duygulu, Volkan Atalay, E.Dincel, "A Heuristic algorithm for Hierarchical Representation of Form Documents", Proc. 14th Int. Conf. on Pattern Recognition ICPR'98, Brisbane, Australia, August 1998
- [7] K. C. Fan, J. M. Lu, L.S. Wang and H. Y. Liao "Extraction of characters from form documents by feature point clustering", Pattern Recognition Lett. 16, 1995, 963-970.
- [8] H. Fujisawa, Y. Nakaro and K. Kurino "Segmentation method for character recognition: from segmentation to document structure analysis", Proc. IEEE 80(7),1992, 1079-1091.
- [9] J. Illingworth and J. Kittler, "A Survey of the Hough Transform", Computer Vision, Graphics, & Image Processing, Vol. 44, 1998, pp. 87-116
- [10] Y.Ishitani, "Flexible and Robust model matching based on association graph for form image understanding", Pattern Anal Appl 3:104-119, 2000
- [11] H.F. Jiang, Han, C.C., Fan, K.C., "A Fast Approach to Detect and Correct Skew Documents", ICPR96(C90.10). 9608(Academia Sinica, ROC)

- [12] Jinhui Liu, Jain, A.K., "Image-based form document retrieval", Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on , Volume: 1 , 16-20 Aug. 1998, pp: 626 -628 vol.1
- [13] J. Liu, X. Ding, Y. Wu: "Description and recognition of form and automated form data entry", Proc. Third Int. Conf. on Document Analysis and Recognition, ICDAR'95, 1995, pp. 579-582.
- [14] S. L. Taylor, R. Fritzson and J. A. Pastor, "Extraction of data from reprinted forms", Mach. Vision Applic. 5, 1992, 211-222.
- [15] T. Watanabe, Q. Luo, N. Sugie, "Layout recognition of multi-kinds of table-form documents", IEEE Trans Pattern Anal Mach Intell 17(4); 432-445, 1995.
- [16] Cao, Y., Wang, S., Li, H., "Skew detection and correction in document images based on straight-line fitting", PRL(24), No. 12, August 2003, pp. 1871-1879.
- [17] B.Yu, A.K. Jain, "A form dropout system". Proc. 13th Int, Conf. on Pattern Recognition", ICPR'96., Vienna, Austria, August, 1996, pp. 701-705
- [18] Yefeng Zheng, Changsong Liu, Xiaoqing Ding; Shiyan Pan, "Form frame line detection with directional single-connected chain", Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on, 10-13 Sept. 2001 pp: 699 -703

VITA

Graduate College
University of Nevada, Las Vegas

Kai Luo

Local Address:

6223 Meadow Haven Ln
Las Vegas, Nevada 89103

Degree:

Bachelor of Science, Electronic Engineering Department, 1998
Southeast University

Thesis Title: Recognition and Identification of Form Document Layouts

Thesis Examination Committee:

Chair: Dr. Shahram Latifi, Ph. D.
Committee Member: Dr Emma Regentova, Ph. D.
Committee Member: Dr Henry Selvaraj, Ph. D.
Graduate Faculty Representative: Dr Kazem Taghva, Ph. D.