A Form Document Image Parser *

Ebru Dincel *+ , **Volkan Atalay** * , **Pınar Duygulu** *

* Dep.of Computer Eng., Middle East Technical University, Ankara, TR-06531 Turkey
+ TÜBİTAK-BİLTEN, Middle East Technical University, Ankara, TR-06531 Turkey

Abstract. Forms are used extensively as input to many information management systems. An automatic way of processing a form should be preferred over the traditional methods due to high volume and numerous kinds of form documents. A system that minimizes operator error, cost and user interaction has to be developed. This paper proposes a system for automatically parsing form documents that have textual and nontextual components. The parser involves two parts. In the first part, the form document goes through scan and preprocessing steps; then line extraction, vertex identification and cell representation are carried out for geometric layout extraction. In the next part, content labelling among textual components, preprinted and user filled-in, is performed based on an unsupervised clustering. The presented system is shown to be robust by the performed experiments.

1 Introduction

A general document processing system aims at dividing a document image into categorized regions such as titles, paragraphs, captions, text and etc. In this study, our main concern is about forms. Forms are structured documents which are generally composed of horizontal and vertical lines, preprinted data (machine printed characters, symbols and pictures) and user filled-in data which are located at predefined locations. A form processing system aims at extracting meaningful data from a form document for office automation [1, 2, 3]. In order to extract the user filled-in data, the form structure should be known or extracted in advance. A model form can be used to specify predefined locations of individual data items which are considered to be meaningful. This model form may be stored in a database in order to be registered eventually with a filled-in instance form. When an instance form is presented to the system, its type can be identified by matching with one of the model forms in the database and meaningful data can then be automatically extracted.

In this study, the aim is to develop an automatic form document image parser as a part of a form processing system. This will reduce the cost and error associated with manual specification of the model form to the system. The input, which is a form, will be decomposed into its cells, which are the smallest indivisible blocks. Output is, then, the labelled cells. Furthermore, an automatic parser, in a way, has the ability to learn the steps to analyze a form document. When another instance of the model form needs to be handled, the procedure is just repeated, bypassing the ones that are intrinsic to its structure. The automatic parser aims at minimizing user interaction by benefitting from the structure information that the form carries.

* This work is partially supported by TÜBİTAK-BİLTEN (Scientific and Technical Research Council of Turkey - Institute of Information Technologies and Electronics) under the project 96-14-030.

In traditional models, specification of the form is either done manually or by dropout method. In the manual method, the position and the label of each cell is specified by a graphical user interface [4]. In the dropout method, a blank model form is subtracted from an instance form so that only filled-in data remains. For the subtraction, blank form and the instance form are to be registered. For this purpose, landmarks or symbols such as barcodes should be explicitly specified by the user. A more challenging problem is to design an automatic parser for forms. Although there is no explicit work on automatic parsing, the general approach is to benefit from line and vertex information [5]. For extraction of horizontal and vertical lines, runlength analysis approach has been proposed [3]. The lines can also be detected by projection profiles, later verified and merged in case of poor quality [6]. The resulting horizontal and vertical line segment candidates can be used to extract corner types [7, 8, 9, 10]. Shimutsi and Aseno propose a parser where the ultimate aim is text extraction via line approximation of contours, and rectangle detection [11]. On the other hand, several studies have been done on labelling the detected cells as text or nontext. Labelling is based on component size threshold [12, 13] or some predefined rules. As to our knowledge, the problem of distinguishing among textual classes which is one of the main subjects of this work has not been addressed very much in the literature.

The proposed system involves two parts: First part starts from the form document and goes till the cell extraction which is based on the vertices. Second part is the content labelling, which gives information about data type that each cell contains such as textual (typed, printed, written) or nontextual (graphics, signature, picture). We are not interested at nontextual components at this point, but we rather try to distinguish different classes of textual elements. In Figure 1, the general outline of the parser for a model form is presented. Details of the parser components are given in the next section.

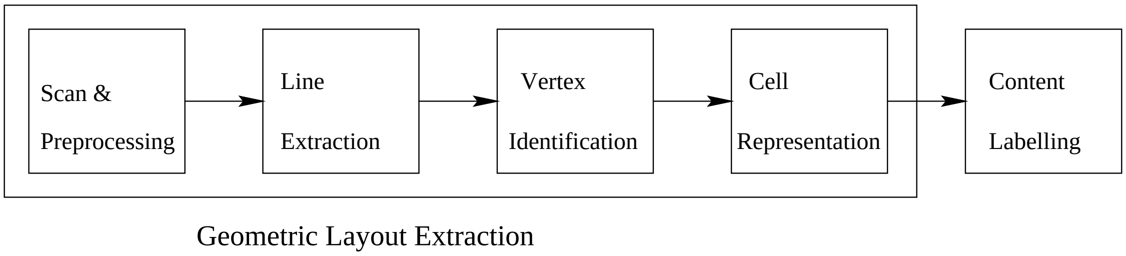


Figure 1. Outline of the parser.

2 Components of the Parser

2.1 Geometric Layout Extraction

Components of the parser are detailed in this section and the result of each component is demonstrated on a sample form. Original sample form is shown in Figure 2.

Scan and Preprocessing: The first step is scanning where the form is converted into a digital image and binarized by an adaptive thresholding with varying windowsize

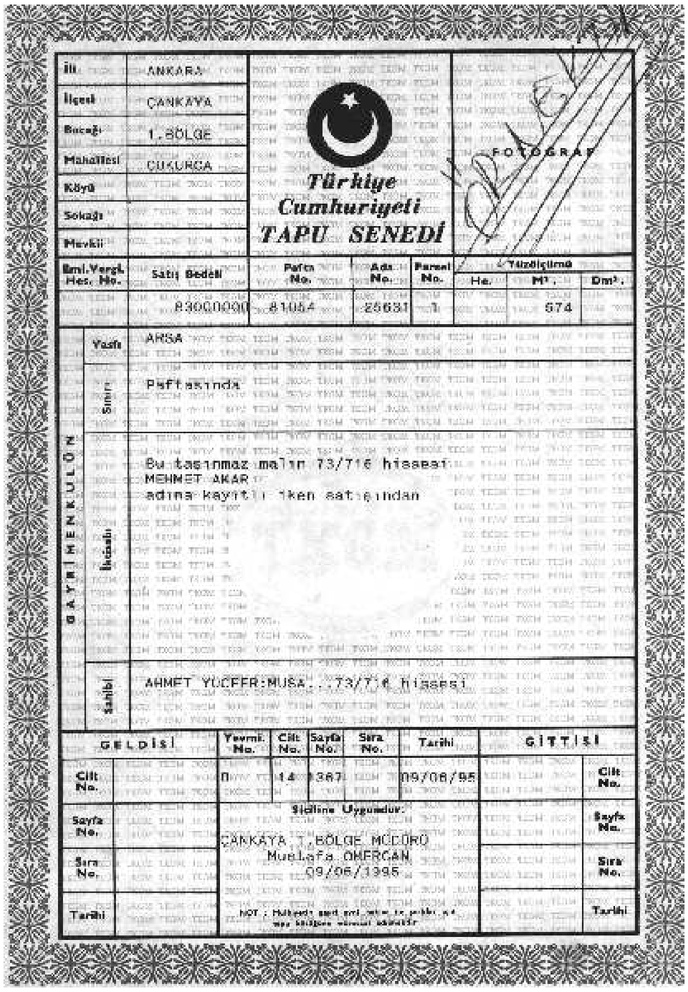
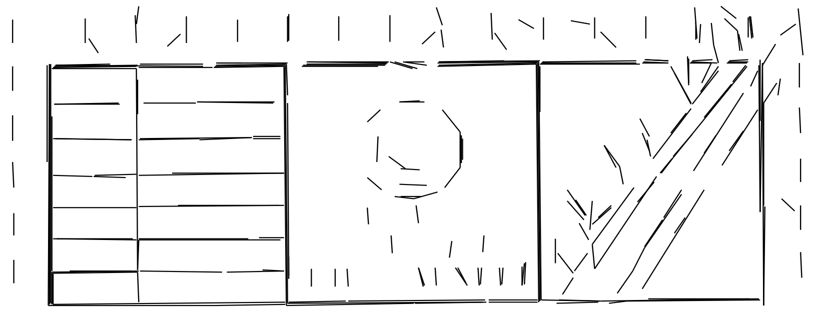


Figure 2. Original sample form.

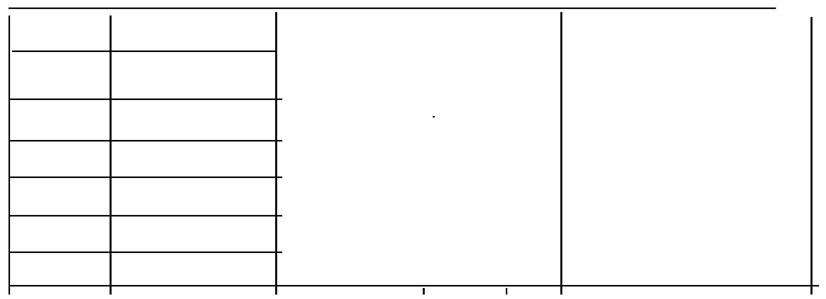
and slidesize parameters. Then, in general, follows the preprocessing where the imperfections, if needed, are recovered. These may include noise reduction, skew detection, offset and over/underthreshold correction. These routines are standard in many document processing systems.

Line Extraction: At this step, horizontal and vertical lines which are part of the geometric structure of a form are extracted. We assume that these lines are not very thick so that they can be captured by a simple edge detection algorithm. For the detection of edge points, a gradient operator is followed by hysteresis thresholding [14]. Then, the extracted edge points are linked to form groups. Finally, polygonal approximation is applied to these groups. Figure 3(a) shows the results of this approximation on the upper block of the sample form shown in Figure 2. However, a refinement may still be necessary; for example the broken lines should be united or some multiple lines representing the same logical line should be reduced to one. A result of this step is shown in Figure 3(b) where the found lines of Figure 3(a) are redrawn.

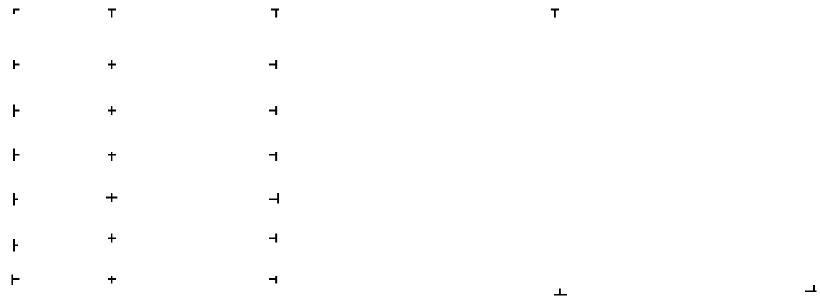
Vertex Identification: A vertex is the intersection of one or more horizontal and ver-



(a) Approximated edges.



(b) Extracted lines.



(c) Identified vertices.

Figure 3. Parser results on the upper block of the sample form shown in Figure 2.

tical lines. On a form document, it is crucial to extract and identify the vertices in order to define the cells. Initially, the lines are grouped as horizontal and vertical and their intersections are collected under nine groups, as shown in Figure 4. However, a refinement on the vertices may still be necessary depending on the image quality. In Figure 3(c) the identified vertices of Figure 3(a) are shown.

Cell Representation: By visiting the identified vertices with right, down, left and up relations, cells are extracted. As the vertices are traversed from left to right and top to bottom, four vertices identifying a cell are kept for further processing.

└	┘	┌	┐	┼	├	┴	┤	┬
LRight	LLeft	LDown	LUp	Cross	TRight	TUp	TLeft	TDown

Figure 4. Vertex types.

2.2 Content Labelling

The proposed system is an automatic one, so it should know where to extract data based on the information of the cell. Since our forms do not contain landmarks or signs to help us in distinguishing between these classes, some features needed to be derived based on the content of the cell.
After the extraction of a cell, it can be labelled as textual or nontextual with respect to its content. For the time being, we are interested in textual cells. Among textual cells, a further classification can be made: user filled-in and preprinted. Printing and writing styles vary much from one type of form to the other. Thus, we have only general information about preprinted and filled-in texts. It's generally assumed that preprinted writing is typographically printed, mostly machine printed, proportionally spaced (that is different letters occupy different horizontal space), and has regularly alligned edges and a smaller size. On the other hand, user filled-in writing can be machine printed, handprinted or handwritten, monospaced (that is fixed pitched), and has irregularly aligned edges and a larger size. Thus, a method is necessary to distinguish between two classes based on some properties such as mono or proportionalspacedness, spatial frequencies, the size and the regularity of the components. Related to these properties the following four features are selected:
1)Average gray value of text in the cell: preprinted data is expected to have different gray level than that of the other class.
2)Area per character: size of the characters vary between two classes.
3)Horizontal size variance: irregular and proportional spaced writing will have different variances.
4)Average horizontal size: preprinted data is generally preferred with smaller fonts to enhance the visual appearance of the form.
In order to compute these features, the individual characters should be extracted. This process requires a connected component analysis. The resulting connected components, each representing a character, are enclosed in a bounding rectangle. In order to label each textual cell as either preprinted or filled-in, an unsupervised clustering approach is preferred. For this purpose, Self Organizing Map [15] is used as an unsupervised classifier. For a cell, the input to the map is a normalized vector of four features as described above. In this sense, each vector represents a cell. The training step organizes the nodes in the map into local neighborhoods that act as feature classifiers on the input data. As the input vector is presented, the node closest to the input is selected as "winner". The winning node modifies both itself and the nodes in the neighborhood in order to create regions that will respond to a spread of values around the input. Thus, the output is stored in such a way that spatial and topological relationships are represented in a meaningful manner. In our case, the input vectors are clustered around

two centers representing two classes. Different SOM sizes are experimented. Since, we have only two classes, a bigger map size spreads out the vectors nonuniformly. It is found out that a map size of 2x3 gives the best results. Figure 5 presents the output of SOM when the cells of the form in Figure 2 is presented as input. '+' label indicates the winning node for an input vector corresponding to a cell containing preprinted data. Similarly, a winning node for an input vector corresponding to a cell containing filled-in data is labelled by '*'. Preprinted data is clustered successfully at one side, but there are some misplaced vectors of the filled-in class. This happens when the number of characters are not adequate to test the regularity of the components.
At the end of this component, labels are assigned to the cells. When an identical form is presented as input, filled-in data can be directly extracted skipping content labelling component.

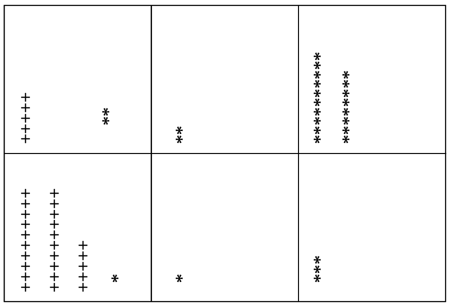


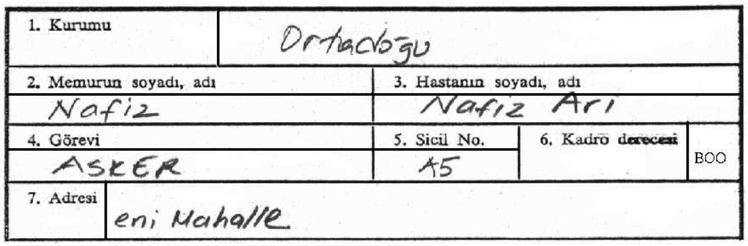
Figure 5. Output of SOM for the cells of the form shown in Figure 2, '+' corresponds to preprinted and '*' indicates filled-in data

3 Sample Parser Results

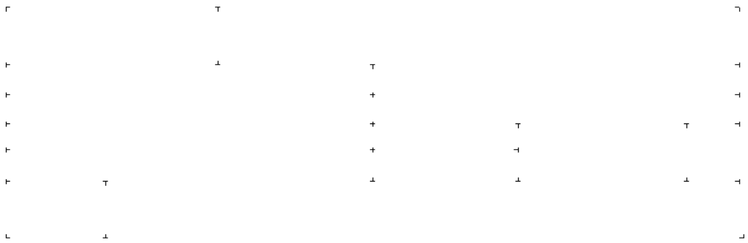
In this section, two forms along with the parser results are given. The first form shown in Figure 6(a) is handwritten. In Figure 6(b), the identified vertices are shown and in (c) are content labelling results. The second form is machine typed. Figure 7 presents the original form and only the content labelling results.

4 Discussion and Conclusion

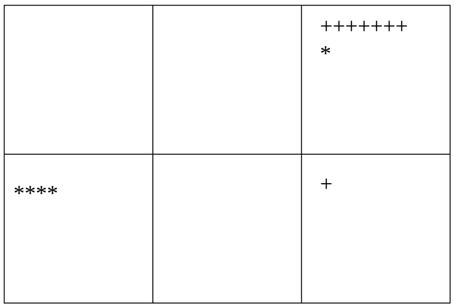
A system for automatically parsing form documents is presented. The parser involves two parts. In the first part, the form document goes through scan and preprocessing steps; then line extraction, vertex identification and cell representation are carried out for geometric layout extraction. In the next part, content labelling among textual components, preprinted and user filled-in, is performed based on an unsupervised classification. In the literature, a few number of systems for automatic parsing of form documents are encountered. However, these systems generally use prior information. In addition, existing systems are not generic enough and they are tested mostly with



(a) A handwritten form.



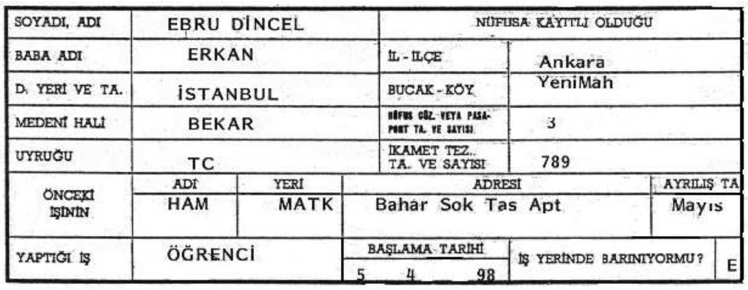
(b) Identified vertices.



(c) Corresponding SOM Output, '+':preprinted, '*':filled-in.

Figure 6. Parser results for a handwritten form.

simple forms. Thus, this work may be regarded as a further extension to the existing systems. The most significant point of this study is that it minimizes user interaction during the extraction of geometric layout and only uses the intrinsic properties of the form. Although relatively good performance(approximately 70% classification rate) is obtained on sample forms, some problems particularly during the first part may be encountered. Due to the bad quality of images, sometimes lines cannot be extracted successfully. This problem can be overcome by adjusting parameters adaptively for the same type of forms. Content labelling is quite successful. But, there may occur some classification problems. However, these can be solved by the use of neighboring information. The sequential processing of existing parsing systems may fail for certain cases. From our experience, the solution comes out to integrate a distributed environment



SOYADI, ADI	EBRU DİNCEL	NÜFUSA KAYITLI OLDUĞU	
BABA ADI	ERKAN	İL - İLÇE	Ankara
D. YERİ VE TA.	İSTANBUL	BUCAK - KÖY	YeniMah
MEDENİ HALİ	BEKAR	NÜFUS CÜZ. VEYA PASAPORT TA. VE SAYISI	3
UYRUĞU	TC	İKAMET TEZ. TA. VE SAYISI	789

ÖNCEKİ İŞİNİN	ADI	YERİ	ADRESİ	AYRILIŞ TA
	HAM	MATK	Bahar Sok Tas Apt	Mayıs

YAPTIĞI İŞ	ÖĞRENCİ	BAŞLAMA TARİHİ 5 4 98	İŞ YERİNDE BARINIYORMU?	E

(a) A machine typed form.

+++		++ ***** ****
+++++ +++++ +++++ **	*	**

(b) Corresponding SOM Output, '+':preprinted, '*':filled-in.

Figure 7. Content labelling for a machine typed form.

where each process acts as an interacting agent.

Acknowledgement
We would like to thank Dr. Charles Bigelow for his valuable comments and contributions on typographic information and Adem Mulayim for his line extraction code.

References

1. Y.Tang, C.Yan, M.Cheriet, C.Suen, "Automatic Analysis and Understanding of Documents", *Handbook of Pattern Recognition and Computer Vision*, pp. 625-654, 1993.

2. D.Wang, S.Srihari, "Analysis of Form Images", *Proc.* 1^{st} *Int. Conf. on Document Analysis and Recognition(ICDAR)*, vol. 1, pp. 181-191, 1991.

3. R.Casey, D.Ferguson, K.Mohiuddin, E.Walach, "Intelligent Form Processing System", *Machine Vision and Applications*, vol. 5, pp. 143-155, 1992.

4. J. Shamilian, H. Baird, T. Wood, "A Retargetable Table Reader", *Proc.* 4^{th} *Int. Conf. on Document Analysis and Recognition(ICDAR)*, vol. 1, pp. 158-163, 1997.

5. P.Duygulu, V.Atalay, E.Dincel, "A Heuristic Algorithm For Hierarchical Representation of Form Documents", *Proc.* 14^{th} *Int. Conf. on Pattern Recognition(ICPR)*, 1998.

6. J. Chen, H. Lee, "A Novel Form Structure Extraction Method Using Strip Projection", *Proc. 3^{rd} Int. Conf. on Pattern Recognition(ICPR)*, vol. 2, pp. 823-827, 1996.
7. S. Lam, L. Javanbakht, S. Srihari, "Anatomy of a Form Reader", *Proc. 2^{nd} Int. Conf. on Document Analysis and Recognition(ICDAR)*, vol. 2, pp. 506-509, 1993.
8. J. Mao, M. Abayan, K. Mohiuddin, "A Model-Based Form Processing Sub-System", *Proc. 3^{rd} Int. Conf. on Pattern Recognition(ICPR)*, vol. 2, pp. 691-695, 1996.
9. S.Taylor, R.Fritzson, J.Pastor, "Extraction of Data From Preprinted Forms", *Machine Vision and Applications*, vol. 5, pp. 211-222, 1992.
10. D. Dori, D. Doermann, R. Haralick, I. Phillips, M. Buchmann, D. Ross, "The Representation of Document Structure", *Handbook on Optical Character on Recognition and Document Analysis*, Chapter XX, 1996.
11. S. Shimotsuji, M. Asano, "Form Identification based on Cell Structure", *Proc. 3^{rd} Int. Conf. on Pattern Recognition(ICPR)*, vol. 2 , pp. 793-797, 1996.
12. S. Wang, T. Yagasaki, "Block Selection: A Method for Segmenting Page Image of Various Editing Styles", *Proc. 3^{rd} Int. Conf. on Document Analysis and Recognition(ICDAR)*, vol. 1, pp. 128-133, 1995.
13. Y. Tang, C. Suen, "Document Structures: A Survey", *Pattern Recognition and Artificial Intelligence(PRAI)*, vol.2, pp. 1081-1111.
14. J.Canny, "A Computational Approach to Edge Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, pp. 679-698, 1986.
15. T.Kohonen, *Self-Organizing Maps*, Springer-Verlag, New York, 1995.