A Heuristic Algorithm For Hierarchical Representation Of Form Documents

Pınar Duygulu
Dep.of Computer Eng.
Middle East Tech. University
Ankara, TR-06531 Turkey
duygulu@ceng.metu.edu.tr

Volkan Atalay
Dep.of Computer Eng.
Middle East Tech. University
Ankara, TR-06531 Turkey
volkan@ceng.metu.edu.tr

Ebru Dincel
TÜBİTAK-BİLTEN
Middle East Tech. University
Ankara, TR-06531 Turkey
dincel@ceng.metu.edu.tr

Abstract

In this paper, our aim is to develop a logical representation for form documents. We propose a hierarchical structure to represent the logical layout of a form by using lines. The approach is top-down and no domain knowledge such as the preprinted data or filled-in data is used. Logically same but physically different forms are associated to the same hierarchical tree. This representation can handle geometrical modifications and slight variations. [1]

1. Introduction

A form processing system aims to extract meaningful data from a form document for office automation [1, 2, 3]. A form is a structured document which is 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. In order to extract the user filled-in data, the form structure should be known in advance. A physical information for form structure may not be appropriate, since the forms or their images risk to be modified geometrically (enlarged/shrinked, translated/rotated, etc.) or to be distorted due to printing or digitization. The geometrical structure should then be mapped to a logical structure by considering the logical relations.

Several approaches are proposed for form recognition and identification [4, 5, 6, 7, 8]. Physical features such as length, width and position of horizontal and vertical lines [4] or line crossings [5] can be used to identify the forms. Although such approaches may solve the problems related to skew, scaling and translation, variations on the physical structure of the logically same form cannot be handled. Another approach at a more logical level is to use blocks and their relationships [6, 7, 8]. The work of Watanabe et. al.[8] is the most similar one to our approach. However, they use a bottom-up approach to form blocks and in addition to the horizontal and vertical lines, preprinted data is also used.

In this study, our aim is to develop a logical representation for form documents. A heuristic algorithm is presented to transform geometric structure of a form into a logical structure by using horizontal and vertical lines which exist on the form. The approach is top-down and no domain knowledge such as the preprinted data or filled-in data is used. The logical structure is represented by a hierarchical tree. The hierarchy of the tree corresponds to the hierarchy of the blocks in the form document. The proposed representation is close to human point of view for the form structure. Logically same forms have the same hierarchical tree structure. Also, geometrical modifications and slight variations on a form are handled by the proposed representation.

2. Definitions and Approach

A *block* is a rectangular area on the form which is surrounded by the longest horizontal and vertical lines at any given instant. For example, the biggest block of a form is the form itself. 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 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, that 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. A block frame is defined to be *ambiguous*, if it includes only the lines of length equal to the borders of the block frame.

The main aim is to partition the form into blocks which can further be divided until cells are reached. The partitioning results in a tree where the root is the form itself and the leaf nodes correspond to the cells. The heuristic behind the approach is that the blocks which contain similar information are grouped together and these group of blocks are separated by lines which are relatively longer than the others. Such lines by definition are called as baselines. Thus,

[1] 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.

the information about how to partition a block is given by the baselines. However, not all baselines provide this information, but the orthogonal lines give clues about which baselines are separators

In order to achieve a sequence of block partitioning, switching of horizontal and vertical divisions is proposed. This exploits all of the information inherited by the logical structure of the form via horizontal and vertical lines.

3. Heuristic Algorithm

The heuristic algoritm to implement the above approach is given as follows.

- Phase I : Initialization.
  1. Initialize the current block frame to the form frame.
  2. Goto Phase II.
- Phase II : Block finding.
  1. Find the baselines on the current block frame.
  2. Initialize the current baseline to the first line of the current block.
  3. Define the block by searching the orthogonal lines for the current baseline.
  4. Continue with 3 until there are no other remaining baselines in the current block.
  5. Goto Phase III.
- Phase III : Hierarchical tree construction.
  For each defined block
  1. Insert a node corresponding to the defined block into the hierarchical tree.
  2. If the block is a cell, then stop,
  3. Else assign the block to the current block and Goto Phase II.

The algorithm consists of three phases : initialization, block finding and hierarchical tree construction. In fact, the whole algorithm is recursive in the sense that the last phase may refer to the second phase and itself during the process. An initial partitioning of the block is done at the second phase. The partitioned blocks are stored to be inserted into the tree and then to be further divided into smaller blocks, if possible. The partitioning is performed at the second phase whose third step is used to define a block. A block is always defined between the current baseline and the ending baseline. Defining a block by using baselines and orthogonal lines is not a very straightforward process. We will start with a basic algorithm and then refine it by considering different cases. a basic algorithm for defining a block is given as follows.

A. If the orthogonal lines are only the borders of the frame, then
   1. Define the next baseline as the ending baseline.
   2. Define a block as between the current baseline and the ending baseline.
   3. Assign the ending baseline to the current baseline.

B. Else
   1. Find the maximum orthogonal line that starts at the current baseline.
   2. Define the baseline where the maximum orthogonal line ends as the ending baseline.
   3. Search for the orthogonal lines that also start at current baseline and which are shorter than the maximum orthogonal line.
   4. If there are orthogonal lines that satisfy the above condition, then
      - Find the shortest one among them.
      - Change the ending baseline as the baseline where this orthogonal line ends.
   5. Define a block as between the current baseline and the ending baseline.
   6. Skip the other baselines between the current baseline and the ending baseline. Assign the ending baseline to the current baseline.

The above algorithm works fine for simple forms. However, when we consider more complex forms some additional conditions should be checked. With the current algorithm, although initially the maximum orthogonal line is searched, in fact blocks are defined by the shortest orthogonal line that starts at the current baseline. However, the maximum orthogonal line becomes significant when there are other lines starting from the baselines which succeeds the current one and shares at least two baselines with the shortest orthogonal line. Details of this part can be found in [10]. We put one more constraint into the algorithm to handle an exceptional case which occurs when the orthogonal lines for the top baseline of the form have the same length as the borders and there are other orthogonal lines that start at the succeeding baselines. If the basic algorithm were applied, the whole block would be taken as a single one. To avoid this, a block is defined between the current baseline and the next baseline.

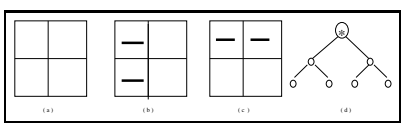
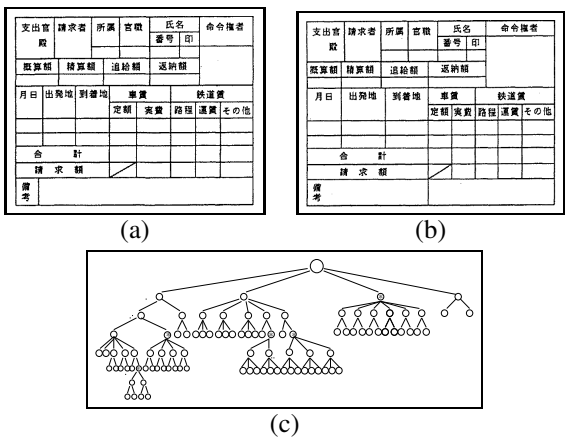
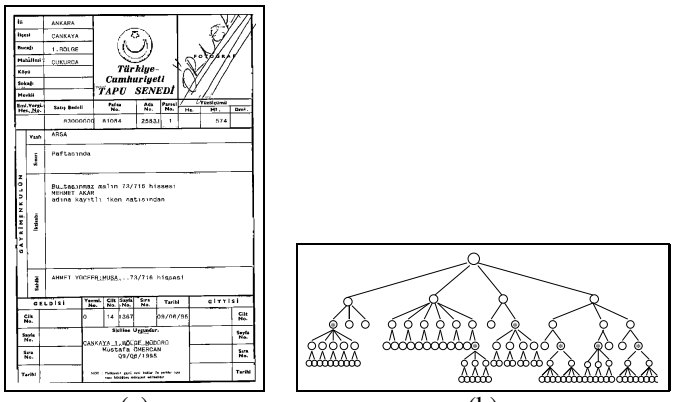


Figure 1. (a)Block without preprinted data, (b),(c) alternatives for preprinted data, (d) corresponding hierarchical tree.



(a) (b) (c)

Figure 2. Multi-kinds of the same table form document represented by the same hierarchical tree.



(a) (b)

Figure 3. A non-table form document and its hierarchical tree.

As a special case, the current block may consist of only the orthogonal lines with the same length equal to the length of the borders of the block frame and there may be no other line parallel to the baselines. In such a case, there is an ambiguity in defining blocks in the sense that only the vertical and horizontal lines do not give enough information about the type of division. If we consider the example in Figure 1(a), without preprinted data, it may represent both blocks shown in Figure 1(b) and shown in Figure 1(c). This ambiguity can be handled by defining the current block as ambigious and forming new blocks between each baseline. In the tree construction part, a flag is used to define the ambiguity for the node which represents the current block. The rest of the tree is constructed in a usual manner. The hierarchical tree for Figure 1(a) is shown in Figure 1(d).

4. Experiments

An artificial form generator is implemented to test the proposed algorithm. In addition to the artificial forms, the algorithm is also tested on the examples adapted from Watanabe et. al. [8] and on several non-tabular forms. A preprocessing is applied to extract the lines, and then the vertices [9]. This step is insensitive to skew angles, rotation and broken lines. Here, we give two example form documents and their representations. The first example is adapted from Watanabe et. al. [8]. When the proposed heuristic algorithm is applied to these form documents which are logically same but physically different, the same tree is obtained (c.f. Figure 2). The second example form document which is shown in Figure 3(a) is an official one used for registration of buildings. The algorithm easily extracts the representation of such a non-tabular form document as shown in Figure 3(b).

5. Conclusion

In this study, a heuristic algorithm is proposed in order to represent the logical structure of a form document. The representation will be used for identification purposes and can be further extended for parsing the form documents.

References

[1] Y. Y. Tang, C. D.Yan, M.Cheriet, C. Y.Suen. Automatic Analysis and Understanding of Documents. In *Handbook of Pattern Recognition and Computer Vision*, pp. 625–654, 1993.

[2] D. Wang, S. N.Srihari. Analysis of Form Images. In *Proc. First Int. Conf. on Document Analysis and Recognition*, ICDAR'91, Saint-Malo, France, Sept, 1991, pp. 181–191.

[3] R. Casey, D.Ferguson, K. Mohiuddin, E.Walach. Intelligent Forms Processing System. *Machine Vision and Applications*, vol.5, no.3, pp. 143–155, 1992.

[4] J.Mao, M.Abayan, K.Mohiuddin. A Model-Based Form Processing Sub-System. In *Proc. 13th Int. Conf. on Pattern Recognition*, Vienna, Austria, August, 1996, pp. 691–695.

[5] S.Taylor, R.Fritzson, J.Pastor. Extraction of Data From Preprinted Forms. *Machine Vision and Applications*, vol.5, pp. 211–222, 1992.

[6] S.Shimotsuji, M.Asano. Form Identification based on Cell Structure. In *Proc. 13th Int. Conf. on Pattern Recognition*, Vienna, Austria, August, 1996, pp. 793–797.

[7] Y.Hirayama. Analyzing Form Images by Using Line-Shared-Adjacent Cell Relations. In *Proc. 13th Int. Conf. on Pattern Recognition*, Vienna, Austria, August, 1996, pp. 768–772.

[8] T.Watanabe, Q.Luo, N.Sugie. Layout Recognition of Multi-Kinds of Table-Form Documents. *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.17, no.4, pp.432–445, April 1995.

[9] E.Dincel,P.Duygulu,V.Atalay. A Form Document Image Parser. *The Seventh Turkish Symposium on Artificial Intelligence and Neural Networks*, Ankara, Turkey, June, 1998.

[10] P.Duygulu,V.Atalay,E.Dincel. Logical Structure Representation Of Form Documents Based on Line Information. Technical Report, TR-97-6, Dept. of Comp. Eng., METU