Ruling-Based Table Analysis for Noisy Handwritten Documents

Jin Chen
CSE Department
19 Memorial Drive West
Bethlehem, PA 18015, USA
jic207@cse.lehigh.edu

Daniel Lopresti
CSE Department
19 Memorial Drive West
Bethlehem, PA 18015, USA
lopresti@cse.lehigh.edu

ABSTRACT
Table analysis can be a valuable step in document image analysis. In the case of noisy handwritten documents, various artifacts complicate the task of locating tables on a page and segmenting them into cells. Our ruling-based approach first detects line segments to ensure high recall of table rulings, and then computes the intersections of horizontal and vertical rulings as key points. We then employ an optimization procedure to select the most probable subset of these key points which constitute the table structure. Finally, we decompose a table into a 2-D arrangement of cells using the key points. Experimental evaluation involving 61 handwritten pages from 17 table classes show a table cell precision of 89% and a recall of 88%.

1. INTRODUCTION
Tables are commonly expressed in various means to convey information, including ASCII files, HTML webpages, PDF files, and paper [21, 6]. As one way of organizing relational data, tables have physical and logical structure [9, 20, 8]. Physical structure describes the locations of table components, e.g., headers, rows, columns, cells, rulings. Logical structure defines the way of connecting table components to each other to form a set of relational n-tuples [21]. Note that table components may present both physical and logical structure from different perspectives. For instance, a table cell can be defined in logical structure by \((Row[i], Column[j])\), and it can also be defined in the physical structure as a rectangular region in the document image. In this paper, we constrain our discussions to the paper medium only.

The target handwritten documents present characteristics that differ significantly from those collected in controlled environment such like research labs. To facilitate our discussion, it is useful to define substrate as a layer of all pre-printed information on a document page before people write on it. Our Arabic handwritten dataset contains multiple components: machine-printed text substrate, handwriting, pre-printed rulings, signatures, logos, etc. In addition, the paper condition and scanner settings are different from those in controlled environment. For example, paper may be ripped, folded, and punched, while for scanning, we often observe skewed pages and low image quality with plenty of “salt-and-pepper” noise, etc. Our evaluation dataset is a collection of Arabic handwritten documents containing tables and forms. Figure 1 is a sample document that illustrates several kinds of challenges present in the dataset.

Hu et al. summarize the problem of table analysis as two sub-problems: detection and recognition [11]. Table detection focuses on finding table regions. Laurentini and Viada use horizontal and vertical rulings as initial evidence for tables in machine-printed documents, and then employ several tests to exclude non-tabular areas [12]. Some other work does not rely on the presence of ruling lines. Hu et al. introduce a table detection method that does not require ruling lines and can work on machine-printed document images and ASCII files [11]. After segmenting text into lines, they detect the inside-space between adjacent text blob and then employ a dynamic programming (DP) technique to solve the problem of optimally decomposing the entire page into text lines and table rows. Shamalian et al. propose a method that uses pre-defined table model as complementary input [17]. Making use of the ink template of tables, they search for a best match of ink templates and the segmented text lines. Shafait and Smith extend table detection to multi-column documents [16]. However, existing techniques cannot be used directly on our evaluation documents because they usually assume clean/simple input and/or well-segmented text lines. For complicated handwritten documents at hand, however, it is not a trivial task to make these existing techniques work on them.

Table recognition usually assumes identified table regions and the goal is to find the physical structure and the logical structure of the table model [9, 20, 8]. There has been plenty of work dealing with machine-printed table recognition [12, 10, 2]. Gatos et al. [7] make use of the complete table rulings to recognize the table structure while Hirayama [10] relies on dynamic programming to align table columns. Richarz et al. propose a method of tabular structure recognition for their semi-supervised transcription system in handwritten historical weather reports [15]. Making use of the pre-printed table substrate, they use Hough transform to detect the horizontal and vertical rulings that constitute the tabular structure. Clawson et al. present a projection-profile based method to detect and extract handwritten tabular fields from historical census forms [5]. We notice that these existing techniques are evaluated on datasets where rulings are usually salient and well displaced, meaning no other lines will distract table analysis. This is, however, not the case for our dataset where we need to handle severely broken lines and/or false-alarms.

In this work, we address the problem of table detection and decomposition for noisy handwritten documents. These documents differ from most in the literature and they are not collected in a controlled environment but from the field. As a result, various types
of noise and artifacts are present and the low image quality makes ruling lines hard to detect. After separating clutter noise from the image, our approach is to ensure high recall of table rulings and then compute the “key points” that intersect horizontal and vertical rulings. Then we employ an optimization procedure to select the most probable subset of rulings that constitute the table. Finally, given the selected key points, we decompose a table image into a 2-D arrangement of table cell images.

2. WORKFLOW OVERVIEW

The high-level workflow is shown in Figure 2. In our implementation, an image is modeled as multi-layer structure where each layer consists of one document component. In addition, we observe that the same document substrate is used multiple times in the dataset. Consider this fact in practice, we can build a table model that summarizes the logical and physical structure and use it to register an unknown input. Hence, table model serves as an input to help detect and analyze tables.

2.1 Clutter Detection

Clutter noise, which refers to the black margin near the image border, is introduced by imperfect scanning. Agrawal and Doerman present a distance transform based approach to detect and remove clutter noise [1]. They detect it using a 2-class SVM classifier in which a number of connected-component based features are constructed.

Our clutter detection, however, is based on the fact that clutter noise is usually much larger than the other components. Thus, at lower resolution, we may only be able to see the clutter noise. To do this, we scale the image down to $1/4$ and extract connected-components from the scaled image. As expected, large components are mostly clutter noise in the original image, so we mark these clutter pixels so that they will not be considered in the following processing operations in another layer. We show the two layers of image components in Figure 3(a) and Figure 3(b).

2.2 Ruling Detection

Although many of the ruling lines in our evaluation dataset are pre-printed on a document substrate, they differ from the ones exhibit in our previous work [3] in that they may not present consistent spacing. In this work, we still use a probabilistic variant of Hough Transform [13] to extract salient line segments. Since many rulings are broken, small gaps (up to 20 pixels) are allowed during ruling detection. Next, we make use of the fact that most correct rulings are parallel or orthogonal in order to exclude some of the false-alarms detected in the text area. Then, we employ the Adaptive Basic Sequential Algorithmic Scheme (Adaptive BSAS [19]) to group clustered line segments based on their $\rho$ values, and compute their parameters (slope, intercept, etc.) using standard line fitting. Due to the low image quality, we adjust parameters in the Hough Transform to obtain high recall of line segments, which, of course, may result in false-alarms in packed text regions. Figure 3(c) shows the results of clustered horizontal and vertical rulings, marked by red and green, respectively.

2.3 Text Detection

Spatial displacement of text can be valuable information to exploit for tabular structure detection and recognition [11, 3]. In our current work, text blob are detected by separating the layers of rulings and clutter noise and then detecting connected-components. Next, we transform the text blob based on the skew angle and then exclude those having unexpected aspect ratios ($\alpha < 0.1$ or $\alpha > 10.0$). This effectively excludes most of the line segments left in the current image layer, as shown in Figure 3(d).

3. RULING SELECTION

Our ruling selection algorithm is designed to work on a set of rulings that are either parallel or orthogonal, which should be satisfied by the filtering in Section 2.2. We now adapt our previous idea of key points in this work [4]. Key points are defined to be a white space region within a local $2 \times 2$ array of text blob in which any horizontal or vertical cuts will not affect table cells. For tables that are ruled or unruled, key points usually refer to the white space between text blob such as words. For ruled tables that have row/column spans, however, we need to adapt key points to be the intersections of table rulings. Therefore, the problem of ruling selection is converted into key point selection. In addition, this con-
version also reduces the search space because not every ruling intersects another.

As a result, several rulings in the text area isolated from the table header are excluded at this stage, as shown in Figure 4(a). Note that we have allowed relatively large gaps (up to 100 pixels) between rulings due to degraded images in order to obtain high recall of key points in the table structure.

Then we employ the Adaptive BSAS clustering again to group key points into horizontal and vertical clusters. Each key point is indexed by the horizontal and vertical clusters, e.g., \((H_1,...,p, V_1,...,q)\), where \(p, q\) are the numbers of horizontal and vertical key point clusters, respectively.

Next, we formulate key points selection as an optimization problem:

\[
\text{argmin}_{W} f(W, M)
\] (1)

where \(W\) is a configuration of selected key points for scoring, \(\Omega\) is the set of all configurations, and \(M\) is the table model containing the number of rows \(r\) and columns \(c\), and cell dimension information. We decide the key points for table structure by selecting horizontal and vertical key points separately. Taking the horizontal case as an example, the formulation is specified as follows:

\[
\text{argmin}_{W_h} f(W_h, M_h)
\] (2)

with the constraints that \(||W_h|| = r + 1\) and \(||\Omega_h|| = (p+1)r\). The vertical key point selection is similar. The cost function \(f(\cdot)\) computes a real value indicating how close the current key point cluster configuration matches with the table model. In our implementation, we compute as follows:

\[
f(\cdot) = ||W_h - M_h|| + C(W_h)
= \sum_{i=1}^{p} ||W_{hi} - M_{hi}|| + C(W_h)
\] (3)

where \(W_{hi}\) and \(M_{hi}\) are heights for table row \(i\), respectively. In words, the first term in this equation computes the accumulated differences of row heights between selected and the table model. The second term \(C(W_h)\) is the cost for text displacement against the horizontal rulings. Currently, we only consider the text displacement in the table header, i.e., the accumulated cost of bounding boxes of text blob adjacent to the second horizontal ruling. Of course, more sophisticated cost functions are possible for future improvement.

Since \(p, q\) are small numbers, it is feasible to enumerate the configuration space to find the global optimum that constitutes the table structure. Figure 4(b) shows the result of selecting horizontal key point clusters, marked by circles. Then we run the selection algorithm again using the corresponding vertical key point clusters, and obtain the correct result. Note that for tables with opened sides (left and right in Figure 4(a)) we need to add imaginary key points accordingly to comply with the constraints in Equation 2. Finally, we scan through the obtained key point grid to decompose the tabular structure into a 2-D arrangement of table cells, using one \(2 \times 2\) tuple at a time. To handle row or column spanning, we simply skip corresponding adjacent key points vertically/horizontally.

4. EXPERIMENTAL SETUP

4.1 Data Preparation

Our evaluation involves a noisy Arabic handwritten document dataset that contains field data [14]. Since this is not prepared in controlled environment, it presents various kinds of noise, artifacts, and complicated document layouts. A sample document substrate presenting some of the challenges is shown in Figure 1. So far, we have annotated and evaluated 61 Arabic documents from 17 document substrates with the corresponding table models (158 images in total). Table 1 shows some statistics in this 158-image dataset based on manual investigation.

The table model specifies the number of rows and columns, row or column spanning, and approximate cell dimensions. Figure 5 shows an example of such a table model file. Note that since the
same table substrate appears many times in the dataset, the cell dimensions (heights, widths) are only the average values.

4.2 Evaluation
We evaluate the system by computing precision and recall on table cell images:

\[
\text{precision} = \frac{\text{number of correctly detected table cells}}{\text{total number of detected table cells}} \\
\text{recall} = \frac{\text{number of correctly detected table cells}}{\text{total number of ground-truthed table cells}}
\]

Currently, we consider a correct table cell detection as long as the text information defined by the \(2 \times 2\) tuple of key points corresponds to the ground-truth.

4.3 Comparison
In this work, we compare our algorithm with a cross matrix based method by Shi et al. [18]. The idea is to first compute a 2-D matrix of rulings which have intersections with their orthogonal rulings, for both the table model and input image (Model[:][:] and Scene[:][:]). Next, considering relatively small number of vertical rulings, they enumerate all possible combinations of vertical rulings. In each enumeration, they design a dynamic programming (DP) algorithm to select the optimal horizontal rulings. In this way, they transform the problem of selecting optimal subsets of rulings to optimally aligning two sets of rulings, which is similar to the edit distance computation between two strings. Finally, the horizontal rulings are computed by back tracking the score matrix in the DP framework. In essence, both methods use similar logic to select horizontal/vertical rulings as the tabular structure, but our method makes use of cell information in the table model, which is expected to be more reliable than the cross matrix based approach.

We restate the DP framework here for reference. Denote horizontal model rulings as \(M_i, i \in [1, n]\), where \(n\) is the number of horizontal rulings in the table model. Likewise \(S_j, j \in [1, N]\) where \(N\) is the number detected in the image. Now the matching score of a model ruling and a scene ruling is defined as the Hamming distance between two rows in the cross matrices.

\[
C(M_i, S_j) = \sum_{k=1}^{K} ||\text{Model}[i][k] - \text{Scene}[j][k]||
\]

where \(k \in [1, \ldots, K]\) denote the index to the current vertical ruling configuration.

The definition for the score matrix \(H\) resembles the computation of edit distance, as follows:

\[
H[i][j] = \min \begin{cases} 
H[i - 1][j - 1] + C_{\text{sub}}(M_i, S_j) \\
H[i - 1][j] + C_{\text{del}}(M_i) \\
H[i][j - 1] + C_{\text{ins}}(S_j)
\end{cases}
\]

where \(i \in [1, n], j \in [1, N]\), and \(C_{\text{del}}(\cdot), C_{\text{ins}}(\cdot)\) are the cost for deletion and insertion in computing the edit distance between two strings. In the task of ruling selection, we need to forbid deletions by assigning it a high cost, otherwise deletion may reduce the number of rulings selected after the alignment. Thus, we set \(C_{\text{del}}(\cdot) = 100\) and \(C_{\text{ins}}(j) = C(0, S_j)\). After computing the score matrix, we back track from \(H[n][N]\) to obtain the alignment of model and scene rulings, in which the positions of substitution are recorded as the indices of selected rulings.

5. EXPERIMENTAL RESULTS
We evaluated our algorithm using 61 document from 17 table classes involving 3,627 table cells and found out that the precision is 88.60% and the recall is 87.90%. On the other hand, the cross matrix based method obtained precision and recall of 85.90% and 84.20%, respectively. We should clarify that both methods did not undergo sophisticated parameter tuning. Note that the detected key points do not have to form a complete 2-D grid since we can compute the rest based on the table model.

Our method did outperform the cross matrix based method, however, there are several observations that we would like to point out which may be useful for future improvements. First, spacing between adjacent rulings is characteristic information to exploit for selecting the correct rulings. Second, the cross matrix method assumes less input information but relies on high recall of line segments, which is challenging in such a degraded dataset, i.e., if the expected intersection in one ruling is missing, there will be a cost in the cross matrix. Lastly, the spatial displacement of text blob can be useful, especially the text in table headers, because they are usually well spaced by their adjacent rulings, while those in other part of the table or page may lie on rulings and/or in different directions.

An example of failure cases is shown in Figure 6. First, there are a number of false-alarms detected due to the fact that the text area is packed, hence false-alarm rulings are detected. On the other hand, this table model is not precise enough for this specific input document, so the optimization procedure gives us an incorrect solution. However, this example does provide us with some hints on future improvements. For example, it is reasonable to incorporate number of key points in selecting the subsets of key points.

6. CONCLUSION
Tables analysis can be a valuable step in document image analysis and over the years, there has been much research on table detection and recognition on carefully prepared datasets. In this paper, we have shown the complexity of processing handwritten documents that are from real life. Our table analysis algorithm is based on detecting and analyzing salient ruling lines in the documents, which may contain false-alarms including paper folds, pre-printed rulings for handwriting, etc. Our approach is to aim for high-recall of the line segments and then employ Adaptive BSAS clustering
and orthogonal analysis to obtain horizontal and vertical rulings. We then adapt the idea of “key points” and employed an optimization procedure to select the most probably subset that constitutes the table structure. Finally, we use the key points and the table model to decompose the table image into a 2-D arrangement of table cell images. Experimental results involving 61 pages from 17 table classes show a table cell precision of 88.60% and a recall of 87.90%. As for the future work, we would like to enhance the robustness of our algorithm and design more powerful cost functions for the optimization procedure.

Acknowledgement

The authors acknowledge stimulating discussions with George Nagy. This work is supported by a DARPA IPTO grant administered by Raytheon BBN Technologies.

7. REFERENCES


