Table Structure Recognition in Scanned Images Using a Clustering Method

Nam Van Nguyen[1(✉)], Hanh Vu[2], Arthur Zucker[3], Younes Belkada[3], Hai Van Do[1], Doanh Ngoc- Nguyen[1], Thanh Tuan Nguyen Le[1], and Dong Van Hoang[1]

[1] Thuyloi University, 175 Tay Son, Dong Da, Hanoi, Vietnam
nvnam@tlu.edu.vn
[2] Viettel CyberSpace Center, 41^{st} floor, Keangnam Landmark 72, Hanoi, Vietnam
[3] Sorbonne University, Polytech Sorbonne 75005, Paris, France

Abstract. Optical Character Recognition (OCR) for scanned paper invoices is very challenging due to the variability of 19 invoice layouts, different information fields, large data tables, and low scanning quality. In this case, table structure recognition is a critical task in which all rows, columns, and cells must be accurately positioned and extracted. Existing methods such as DeepDeSRT, TableNet only dealt with high-quality born-digital images (e.g., PDF) with low noise and apparent table structure. This paper proposes an efficient method called CluSTi (Clustering method for recognition of the Structure of Tables in invoice scanned Images). The contributions of CluSTi are three-fold. Firstly, it removes heavy noises in the table images using a clustering algorithm. Secondly, it extracts all text boxes using state-of-the-art text recognition. Thirdly, based on the horizontal and vertical clustering algorithm with optimized parameters, CluSTi groups the text boxes into their correct rows and columns, respectively. The method was evaluated on three datasets: i) 397 public scanned images; ii) 193 PDF document images from ICDAR 2013 competition dataset; and iii) 281 PDF document images from ICDAR 2019's numeric tables. The evaluation results showed that CluSTi achieved an F_1*-score* of 87.5%, 98.5%, and 94.5%, respectively. Our method also outperformed DeepDeSRT with an F_1*-score* of 91.44% on only 34 images from the ICDAR 2013 competition dataset. To the best of our knowledge, CluSTi is the first method to tackle the table structure recognition problem on scanned images.

Keywords: Table structure recognition · Object recognition · Clustering method

1 Introduction

Our paper aimed to recognize the table's structure from scanned images of invoices. Data tables are the main content of the documents, especially invoices.

Direct application of OCR techniques to the whole data table has been impossible since the recognized texts do not follow precisely the original table structure, which led to the recognition results on large images for most OCR techniques were not highly accurate, especially for tables with many data items [11]. Therefore, the table structure in scanned images has to be recognized so that each table cell can be correctly located and individually processed using OCR techniques. However, this has never been a trivial task because of the different shapes, sizes, and colors of the cell separators. In addition, canned invoice images are typically noisy, which can make these separations become blurred or even lost. Besides, cells must be aligned to rows and columns, this alignment nevertheless can easily be biased in specific images due to the noise. Most existing table structure recognition methods dealt with relatively clean table images, such as PDF document images [3,10,15–17,22], the recognition results however were not highly accurate due to the table complexity. Hence, those methods would not efficient to apply to noisy scanned invoice images.

In this paper, we proposed CluSTi, an efficient approach for table structure recognition in scanned invoice images, which is mainly based on clustering algorithms. CluSTi considers an invoice table as a set of text boxes, which are sorted by certain vertical and horizontal orders. CluSTi firstly uses Character Region Awareness for Text Detection (CRAFT), a semantic segmentation method, for text boxes detection in an image [2]. Given the coordinates of the text boxes, CluSTi then recognizes the correct cell row and column using Density-Based Spatial Clustering of Applications with Noise (DBSCAN) clustering algorithm [5]. Our method was evaluated with 397 public scanned table images, 193 document images from ICDAR 2013 competition, and 281 document images from ICDAR 2019 competition. The achieved $\mathbf{F_1}$**-score** were 87.5%, 98.5%, and 94.5%, respectively, which outperformed the accuracy of existing methods.

The rest of the paper is organized as follows. Section 2 presents in detail about our CluSTi method. In Sect. 3, we evaluated the results of our method using three public datasets. Finally, Sect. 4 concludes on our work and the perspectives for the future.

2 Methods

Existing table structure recognition methods can be divided into two groups: *top-down* and *bottom-up* methods. Herein, we proposed an efficient *bottom-up* approach for table structure recognition named CluSTi.

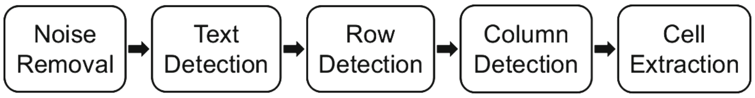


Fig. 1. Block diagram of CluSTi recognition process.

Basically, we applied and optimized a clustering technique at every steps of our recognition process. As shown in Fig. 1, CluSTi includes the following steps: i) Firstly, in the Noise Removal, we applied the DBSCAN clustering technique to clean the table images; ii) In the Text Detection, textual table elements are extracted from the images based on an object recognition deep learning model; iii) In the Row Detection, textual elements are horizontally regrouped using the above clustering technique of which the parameters was optimized; iv) Similarly, in the Column Detection, textual elements are vertically clustered with optimized parameters; v) Finally, in the Cell Reconstruction, the whole table structure are reconstructed cell by cell. The whole CluSTi process is demonstrated in Fig. 2. After Row detection step, all of the text boxes in the same rows are marked with the number on the top of the boxes, denoting the sequence number (i.e., 0, 1, 2, etc.) of their correct rows. Next, after Column detection step, all of the text boxes are labelled with the number on the top of the boxes, denoting the correct sequence number of their corresponding columns. The empty cells are also filled with blank text boxes. Then, after Cell reconstruction step, all of text boxes with the same sequence number of row and column are merged together to form the entire cells.

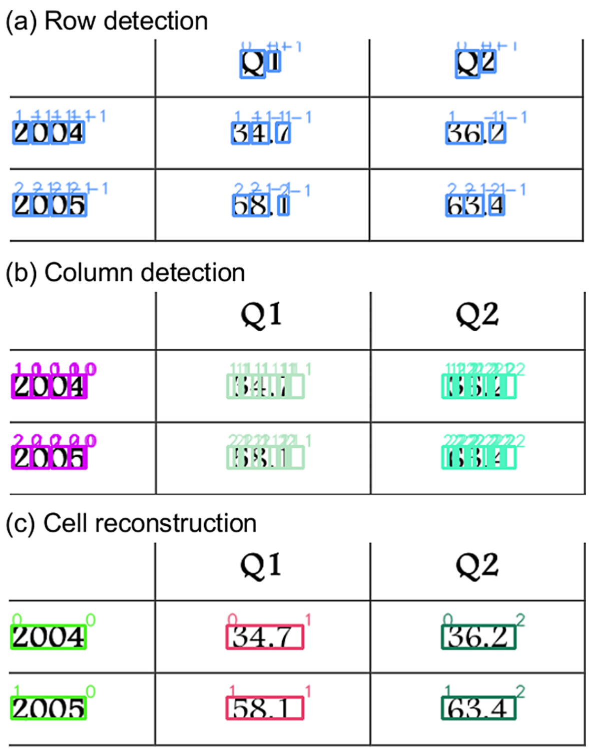


Fig. 2. Overall description of CluSTi recognition process. (a) Row detection. (b) Column detection. (c) Cell reconstruction.

2.1 Noise Removal

In scanned images, noise is defined as the image's elements which can bias the text recognition [6]. From our available scanned images, we observed that the characters are the clusters containing a high number of neighboring pixels. In contrast, noise is the disjointed clusters of several pixels. Therefore, to remove noise from a scanned image, we relied on DBSCAN, which can segment the low and high density clusters. DBSCAN, which was introduced by Ester *et al.* (1996), groups the data points and their closest neighbors into clusters, and marks the lonely points into low-density region as the outliers [5]. The input parameters of DBSCAN include ϵ and *min_samples*. ϵ (eps) corresponds to the upper limit of the distance between two neighbors in a cluster, and *min_samples* corresponds the minimum number of points in a cluster. DBSCAN starts by choosing a random point, then it checks a nearest point (i.e., neighbor) in a circle of radius ϵ. The neighbors found are added into the group and the process continues with these new members of the group. If there is no more neighbors are found, and the number of group's members is greater or equal to the *min_samples*, then the group becomes a cluster. Otherwise, the group's points are marked as the outliers. DBSCAN is therefore suitable for clustering the texts since the characters are written one after the other. Moreover, this method can also be used to remove noise (or outliers) in the text images [5,24].

After applying DBSCAN with ϵ equals to 1 (i.e., pixel) and *min_samples* equals to the number of pixels of the smallest character in that specific language (e.g., Japanese), most of the noise is marked as the outliers and removed, as presented in Fig. 3.

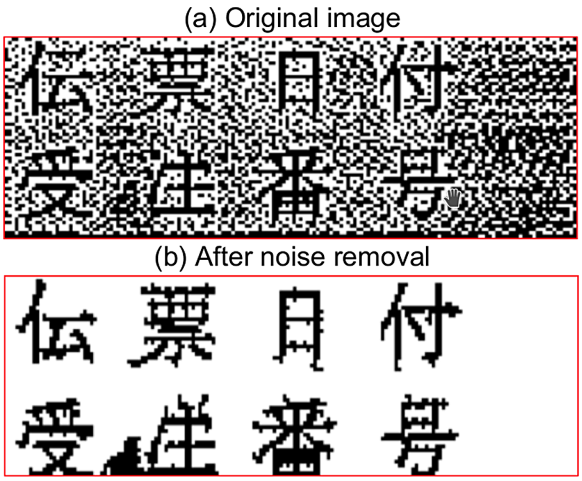


Fig. 3. An example image after applying DBSCAN for noise removal.

2.2 Text Detection

Recently, many deep learning scene text detectors have been proposed, and developed their applications in various fields [2,4,7–9,13,25]. Efficient methods have usually been inherited from object detection and semantic segmentation models such as Faster Region-based Convolutional Neural Network (Faster R-CNN) [18], Single Shot Multibox Detector (SSD) [12] and FCN [14].

CRAFT has been the best among current text detection methods thanks to its convolutional neural networks yielding the region score and affinity score [2]. Specifically, CRAFT detects character regions and links them to a text instance. This method is thus efficient for detecting any character including tiny, extremely long, curved, rotated and arbitrarily shaped characters. By applying CRAFT on the noiseless table images, we aimed to detect the texts as much as possible. Hence, CRAFT is configured to detect only character regions without linkage between them. We also fine-tuned the CRAFT's magnifier and bounding box parameters so that the small characters can be recognized, and there is limited white space in the character bounding regions.

2.3 Row Detection

In the previous step, we bounded every textual elements in the image with distinct rectangle boxes. Based on the coordinates of these character bounding regions, we then grouped them into their corresponding rows, and determined the number of rows in the table images using the following horizontal clustering algorithm.

Horizontal Clustering. The horizontal clustering technique is described in **Algorithm 1**. Firstly, the coordinates of the centroids (i.e., (x_c, y_c)) of every detected text boxes are calculated. Then, they are normalized according to the x-axis. Finally, the normalized centroids (i.e., (x_n, y_n)) are clustered using DBSCAN with optimized parameters. The output of the horizontal clustering is the correct number of rows, as well as the text boxes belonging to each row of the table images.

Fine-Tuning Horizontal Clustering. Given the *min_samples* parameter, the accuracy of horizontal clustering heavily depends on its ϵ parameter, which is the maximum distance between two neighboring centroids processed by the **Algorithm 1**. We assumed that the height (i.e., H) of table rows are equivalent. Thus, the ϵ parameter can be approximated to any value around the median height of the character bounding boxes, which is calculated as in the **Algorithm 2**.

However, there are cases where rows may include multiple lines. We therefore proposed a probing algorithm to find an appropriate ϵ parameter around the median height, which is calculated in **Algorithm 2**. We represented $f_r(\epsilon)$ as the function of the number of rows, r, found by the horizontal clustering where ϵ parameter ranging from $(0.1 * median\_height)$ to $(1 * median\_height)$. $f_r(\epsilon)$ is

Algorithm 1. *Horizontal Clustering Algorithm*

```
Data: $N \leftarrow$ the total number of character bounding boxes
$\{(x^i_{min}, y^i_{min}), (x^i_{max}, y^i_{max})\} \leftarrow$ the coordinates of the upper-left and the lower-right corners of the $i^{th}, \forall i \in [1; N]$ character bounding boxes
$min\_samples \leftarrow$ the number of pixels of the smallest character in a specific language
$\epsilon \leftarrow$ to be fine-tuned
Result: Number of rows
$i \leftarrow 1$
while $i \leq N$ do
    $x^i_c = (x^i_{min} + x^i_{max})/2$
    $y^i_c = (y^i_{min} + y^i_{max})/2$
    $x^i_n = 0$
    $y^i_n = y^i_c$
    $i \leftarrow i + 1$
end
$num\_clusters$ = DBSCAN($(x^i_n, y^i_n)$, $\epsilon$, $min\_samples$)
```

Algorithm 2. *Median Height Calculation Algorithm*

```
Data: $N \leftarrow$ the total number of character bounding boxes
$(y^i_{min}, y^i_{max}) \leftarrow$ the upper-left and lower-right $y$-coordinate of the $i^{th}, \forall i \in [1; N]$ bounding box
Results: $\epsilon$ as the median height
$H_i = |y^i_{max} - y^i_{min}|$;
$Sort(H_i, \forall i \in [1; N])$;
```

$$\epsilon = \begin{cases} H_{(\frac{N}{2})} & \text{if } N \text{ is odd} \\ \left(H_{(\frac{N}{2})} + H_{(\frac{N}{2}+1)}\right)/2 & \text{if } N \text{ is even} \end{cases} \tag{1}$$

then calculated as in the **Algorithm 3**. Next, the density distribution of $f_r(\epsilon)$ for each table image was plotted. Then, we applied a peak detection algorithm [21] on the density distribution to find the best ϵ parameter for the horizontal clustering.

2.4 Column Detection

In the column detection process, we based on the following vertical clustering algorithm to calculate the number of columns, as well as to group the detected text boxes into their corresponding columns.

Vertical Clustering. **Algorithm 4** describes our vertical clustering algorithm.

Algorithm 3. *Probing Algorithm for Horizontal Clustering*

```
**Data**: median height from **Algorithm 2**
**Result**: $f_r(\epsilon)$
$\epsilon \leftarrow$ median height
$k \leftarrow 1$
while $k \geq 0.1$ do
    $\epsilon = k * \epsilon$
    $num\_clusters$ = HorizontalClustering($min\_samples, \epsilon$)  $k \leftarrow k - 0.01$
end
```

Algorithm 4. *Vertical Clustering Algorithm*

```
Data: $N \leftarrow$ the total number of character bounding boxes
$\{(x^i_{min}, y^i_{min}), (x^i_{max}, y^i_{max})\} \leftarrow$ the coordinates of the upper-left and the lower-right corners of the $i^{th}, \forall i \in [1; N]$ character bounding boxes
$min\_samples \leftarrow$ the number of rows found in the previous step
$\epsilon \leftarrow$ to be fine-tuned
Result: Number of columns  $i \leftarrow 1$
while $i \leq N$ do
    $x^i_c = (x^i_{min} + x^i_{max})/2$
    $y^i_c = (y^i_{min} + y^i_{max})/2$
    $x^i_n = x^i_c$
    $y^i_n = 0$
    $i \leftarrow i + 1$
end
$num\_clusters$ = DBSCAN($(x^i_n, y^i_n)$, $\epsilon$, $min\_samples$)
```

Fine-Tuning Vertical Clustering. Since the *min_samples* parameter is fixed to the number of rows found from the previous step, we tried to find the best ϵ parameter for our vertical clustering algorithm. We noticed that the width of columns in the table images are not equivalent as in the case of row's height. We therefore proposed another technique to probe for the converged value of ϵ parameter. We represented $f_c(\epsilon)$ as the function of the number of clusters, c, found by the above vertical clustering with regard to ϵ. $f_c(\epsilon)$ is then calculated as in the **Algorithm 5**. The curvature of a continuous $f_c(\epsilon)$ can be defined as follows [20]:

$$K_{f_c}(\epsilon) = \frac{f''_c(\epsilon)}{(1 + f'_c(\epsilon)^2)^{\frac{3}{2}}} \tag{2}$$

Furthermore, there exist a critical point in this curve called knee, where the curvature is a local maximum [20]. We observed that this point gives the best accuracy for column detection in scanned images. An example of knee point is shown in Fig. 4, where the point (40,10) is the knee point of the curve.

In this case, since $f_c(\epsilon)$ is a discrete function, the knee point can be detected using Kneedle algorithm [20]. Kneedle alculates the distance from all discrete points to the straight segment formed by the first and the last point of the curve. Local maxima (or knees) are considered as the points of the curve which

Algorithm 5. *Probing Algorithm for Vertical Clustering*

```
Data: $\epsilon_{min}, \epsilon_{max}$
Result: $f_c(\epsilon)$
$\epsilon \leftarrow \epsilon_{min}$
while $\epsilon \leq \epsilon_{max}$ do
    $num\_clusters$ = VerticalClustering($min\_samples, \epsilon$)  $\epsilon \leftarrow \epsilon + 1$
end
```

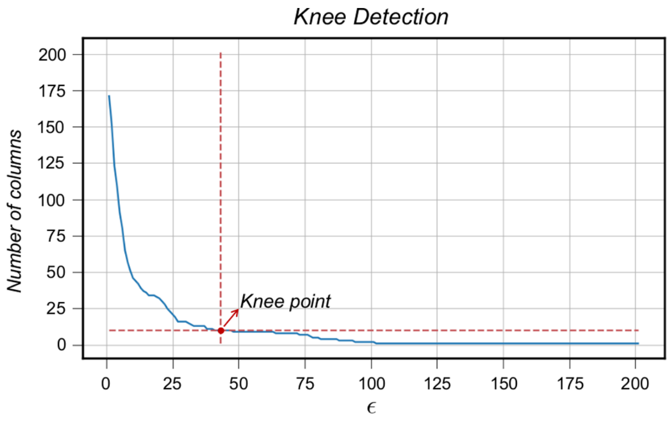


Fig. 4. The curve representing the dependence of the number of columns on ϵ parameter for DBSCAN and its corresponding knee point.

are the most distant to this straight segment. Knees can be detected faster or slower depending on a predefined sensitivity parameter S [20]. This is a measure of how the required number of knee points, for the best result, is was set to 10.

Column Detection in Low Resolution Table Image. Knee detection is applicable in most cases to determine the best ϵ parameter for the vertical clustering. However, in low resolution table images where the distance between columns in terms of pixel count is relatively small, this algorithm is not efficient. Supposing that there exist vertical lines separating table columns in such table image, we proposed another technique to recognize these lines, and then the columns can be detected.

Particularly, after row detection, we extracted text boxes for every rows, then after applying a binary filter, we determined the pixel count of each row as the summation of pixel counts of its text boxes. We then chose the row with the lowest pixel count since it is the least noisy. The morphological closing and Gaussian blur filtering [23] are then applied to this row so that the vertical lines are exposed. Next, the pixels in the resulting row are vertically clustered with ϵ set to one pixel. Finally, the vertical lines separate the columns correspond to the clusters with the smallest width, and with the same height to the table row.

2.5 Cell Reconstruction

Cells can be reconstructed by determining their actual width, height and coordinates. Specifically, after row detection, the height of a row and the y-axis coordinate of the row's center are approximated as the median height and the median y-axis coordinate of all the text boxes and their centroids, respectively. Similarly, after column detection, the width of the columns and their center's x-axis coordinate are also computed.

However, in the table images, there may be empty cells which can not be detected by the text recognition technique. Therefore, we rebuilt an anchor row which are fully filled. The anchor row is built by normalizing coordinates of all detected text boxes so that the y-axis coordinate of their centroids is the same (say *zero*), and then merging together to the normalized text boxes of the same column. In the merging process, the width of the cells is updated as the difference between the maximal and minimal x-axis coordinate of the text boxes in the same column. The empty cells of all rows are then filled by moving the anchor row along the y-axis to every rows in the table image. The final result of text boxes detection can be seen as in Fig. 5.

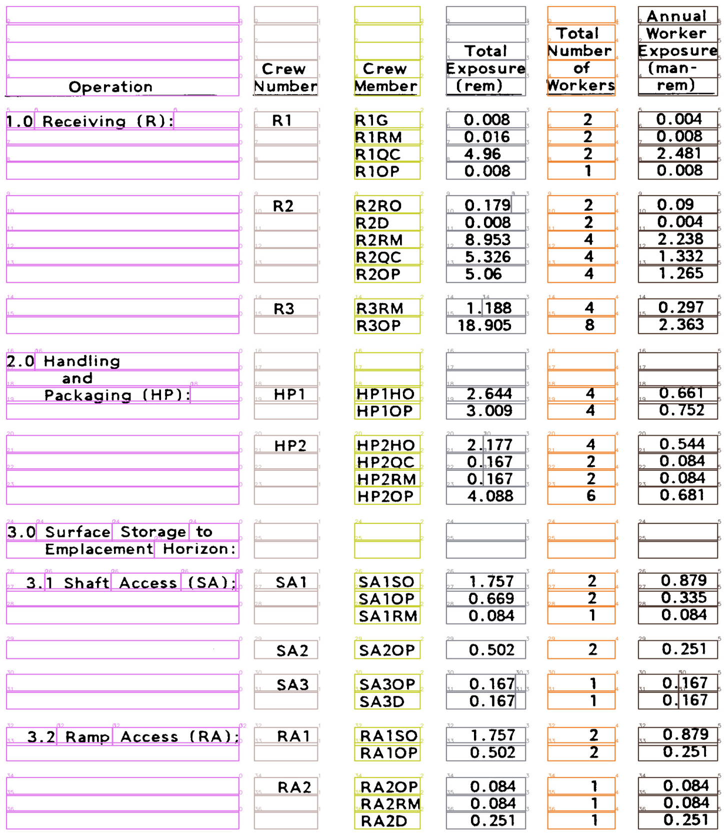
3 Evaluation Results

In this section, we evaluated the accuracy of table structure recognition using our proposed CluSTi method on three different datasets. We also compared the performance of CluSTi to DeepDeSRT [22], which is known as the best recent method for table structure recognition on the ICDAR 2013 and ICDAR 2019 competition's datasets.

3.1 Performance Evaluation of CluSTi

CluSTi is firstly evaluated on 397 table images, which were selected from a public dataset of 403 scanned images [1]. Six images were removed because of their lack of table. An example of table scanned image is presented in Fig. 5. The table is composed of 6 columns and 37 rows. However, the first two columns are typically sparse while the other columns are fully filled. Moreover, since the column's names consist of multiple lines in this table, the height of the first row is greater than the remaining rows. In this case, the table structure can not be recognized using DeepDeSRT due to the fact that there exist many empty cells. In contrast, CluSTi can detect the structure with an accuracy of 100%. Corresponding table cells are represented by color rectangle bounding boxes in Fig. 5.

CluSTi's performance on the 397 scanned table dataset is shown in Table 1. The overall $\mathbf{F_1}$**-score** [19], which is the harmonic mean of precision and recall of CluSTi on this dataset, is 87.5%. The accuracy of the row detection (i.e., 92.9%) is higher compared to the column detection (i.e., 82.0%) since the height of rows is mostly uniform while the width of columns is different.



Operation	Crew Number	Crew Member	Total Exposure (rem)	Total Number of Workers	Annual Worker Exposure (man-rem)
1.0 Receiving (R):	R1	R1G	0.008	2	0.004
		R1RM	0.016	2	0.008
		R1QC	4.96	2	2.481
		R1OP	0.008	1	0.008
	R2	R2RO	0.179	2	0.09
		R2D	0.008	2	0.004
		R2RM	8.953	4	2.238
		R2QC	5.326	4	1.332
		R2OP	5.06	4	1.265
	R3	R3RM	1.188	4	0.297
		R3OP	18.905	8	2.363
2.0 Handling and Packaging (HP):	HP1	HP1HO	2.644	4	0.661
		HP1OP	3.009	4	0.752
	HP2	HP2HO	2.177	4	0.544
		HP2QC	0.167	2	0.084
		HP2RM	0.167	2	0.084
		HP2OP	4.088	6	0.681
3.0 Surface Storage to Emplacement Horizon:					
3.1 Shaft Access (SA):	SA1	SA1SO	1.757	2	0.879
		SA1OP	0.669	2	0.335
		SA1RM	0.084	1	0.084
	SA2	SA2OP	0.502	2	0.251
	SA3	SA3OP	0.167	1	0.167
		SA3D	0.167	1	0.167
3.2 Ramp Access (RA):	RA1	RA1SO	1.757	2	0.879
		RA1OP	0.502	2	0.251
	RA2	RA2OP	0.084	1	0.084
		RA2RM	0.084	1	0.084
		RA2D	0.251	1	0.251

Fig. 5. Cell reconstruction result. (Color figure online)

CluSTi is also evaluated on the ICDAR 2013 competition's dataset, which contains 193 document images. Note that these are PDF born-digital images, noiseless, and not scanned documents. Unsurprisingly, the overall $\mathbf{F_1}$**-score** of CluSTi achieved on ICDAR 2013 dataset is significantly higher compared to the scanned images dataset (i.e., 98.5% and 87.5%, respectively; Table 1). In fact, these document images have considerably high resolution, and the column's width is relatively equivalent. Thus, the CluSTi's $\mathbf{F_1}$**-score** corresponding to column detection in this case is 96.9%, which is much higher than 82.0% on scanned images. Similary, on the 281 ICDAR 2019 document images, CluSTi also achieved a very high $\mathbf{F_1}$**-score** accuracy of 94.5%.

Table 1. Detection accuracy (%) of CluSTi on 397 scanned images, ICDAR 2013 and ICDAR 2019 document images.

Dataset	Accuracy	Row	Column	Overall
397 Scanned Images	Precision	93.2%	83.2%	**88.3%**
	Recall	92.8%	82.4%	**87.6%**
	F_1-score	92.9%	82.0%	**87.5%**
ICDAR 2013	Precision	99.9%	97.0%	**98.5%**
	Recall	99.9%	97.2%	**98.6%**
	F_1-score	99.9%	96.9%	**98.5%**
ICDAR 2019	Precision	99.8%	92.9%	**96.4%**
	Recall	99.7%	87.6%	**93.7%**
	F_1-score	99.8%	89.3%	**94.5%**

Table 2. Comparison of detection accuracy (%) among CluSTi, DeepDeSRT [22], and TableNet [15] on ICDAR 2013 dataset

Method	Number of images	Recall	Precision	F1-score
DeepDeSRT [22]	34	87.36%	95.93%	**91.44%**
TableNet [15]	34	90.01%	93.07%	**91.51%**
CluSTi	193	98.60%	98.51%	**98.48%**

3.2 Comparison of CluSTi and Other Methods

CluSTi outperforms DeepDeSRT and TableNet with an overall **F_1-score** of 98.48% on 193 document images compared to 91.44% on 34 images (Table 2). In fact, CluSTi concentrates on the detection of characters since these are the most essential elements in table cells. Then, the table structure can be deduced and filled thanks to its horizontal and vertical clustering. In contrast, DeepDeSRT is based on Faster R-CNN, a semantic segmentation model which focuses on cell object detection [18]. That's why when cells are empty or not large enough, they are still recognized by CluSTi but not by DeepDeSRT. This approach also overcome the limitations of DeepDeSRT method, which segments table cells relying on their boundaries [22]. TableNet showed comparable results to DeepDeSRT method [22], and their model is end-to-end which means further improvements can be made with richer semantic knowledge, and additional branches for learning row-based segmentation.

4 Conclusion

This paper introduced CluSTi, an efficient approach for table structure recognition problem in scanned images, which have not been addressed in the literature. This is a *bottom-up* method, which emphasizes that the table structure is formed by relative positions of text cells, and not by inherent boundaries. Therefore, CluSTi firstly detects the character regions with an accurate scene text detector called CRAFT. Then, the detected text boxes are spatially clustered into their corresponding rows and columns using the Horizontal and Vertical clustering methods, respectively. Finally, every table cells are correctly aligned and extracted according to their detected rows and columns. CluSTi is evaluated on both scanned images and document images, and the achieved $\mathbf{F_1}$**-score** are 87.5%, 98.5%, and 94.5% on three datasets including 397 scanned images, ICDAR 2013 and ICDAR 2019, respectively. This is the highest accuracy for table structure recognition problem executed on scanned image datasets.

However, CluSTi bears certain inconveniences, especially for complicated table structures where exist spreading rows or columns. In such cases, the columns' (or rows) names may not be aligned to the texts of the other rows (or columns) in the same column (or row). These columns (or rows) need to be recognized and processed separately.

References

1. Table-detection-dataset. https://github.com/sgrpanchal31/table-detection-dataset
2. Baek, Y., Lee, B., Han, D., Yun, S., Lee, H.: Character region awareness for text detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 9365–9374 (2019)
3. Clinchant, S., Déjean, H., Meunier, J.L., Lang, E.M., Kleber, F.: Comparing machine learning approaches for table recognition in historical register books. In: 2018 13th IAPR International Workshop on Document Analysis Systems (DAS), pp. 133–138. IEEE (2018)
4. Deng, D., Liu, H., Li, X., Cai, D.: Pixellink: detecting scene text via instance segmentation. In: 32nd AAAI Conference on Artificial Intelligence (2018)
5. Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD, pp. 226–231 (1996)
6. Farahmand, A., Sarrafzadeh, H., Shanbehzadeh, J.: Document image noises and removal methods. In: Proceedings of the International MultiConference of Engineers and Computer Scientists, pp. 436–440. Newswood Ltd. (2013)
7. Hartigan, J.A., Wong, M.A.: Algorithm as 136: a k-means clustering algorithm. J. Roy. Stat. Soc.: Ser. C (Appl. Stat.) **28**(1), 100–108 (1979)
8. He, T., Tian, Z., Huang, W., Shen, C., Qiao, Y., Sun, C.: An end-to-end textspotter with explicit alignment and attention. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5020–5029 (2018)
9. He, W., Zhang, X.Y., Yin, F., Liu, C.L.: Deep direct regression for multi-oriented scene text detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 745–753 (2017)
10. Hu, J., Kashi, R.S., Lopresti, D.P., Wilfong, G.: Table structure recognition and its evaluation. In: Document Recognition and Retrieval VIII, vol. 4307, pp. 44–55. International Society for Optics and Photonics (2000)
11. Kboubi, F., Chabi, A.H., Ahmed, M.B.: Table recognition evaluation and combination methods. In: 8th International Conference on Document Analysis and Recognition (ICDAR 2005), pp. 1237–1241. IEEE (2005)

12. Liu, W., et al.: SSD: single shot multibox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2
13. Liu, X., Liang, D., Yan, S., Chen, D., Qiao, Y., Yan, J.: FOTS: fast oriented text spotting with a unified network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5676–5685 (2018)
14. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440 (2015)
15. Paliwal, S.S., Vishwanath, D., Rahul, R., Sharma, M., Vig, L.: Tablenet: deep learning model for end-to-end table detection and tabular data extraction from scanned document images. In: 2019 International Conference on Document Analysis and Recognition (ICDAR), pp. 128–133. IEEE (2019)
16. Qasim, S.R., Mahmood, H., Shafait, F.: Rethinking table recognition using graph neural networks. In: 2019 International Conference on Document Analysis and Recognition (ICDAR), pp. 142–147. IEEE (2019)
17. Rashid, S.F., Akmal, A., Adnan, M., Aslam, A.A., Dengel, A.: Table recognition in heterogeneous documents using machine learning. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 1, pp. 777–782. IEEE (2017)
18. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, pp. 91–99 (2015)
19. Sasaki, Y., et al.: The truth of the f-measure. Teach Tutor mater **1**(5), 1–5 (2007)
20. Satopaa, V., Albrecht, J., Irwin, D., Raghavan, B.: Finding a "kneedle" in a haystack: detecting knee points in system behavior. In: 2011 31st International Conference on Distributed Computing Systems Workshops, pp. 166–171. IEEE (2011)
21. Scholkmann, F., Boss, J., Wolf, M.: An efficient algorithm for automatic peak detection in noisy periodic and quasi-periodic signals. Algorithms **5**(4), 588–603 (2012)
22. Schreiber, S., Agne, S., Wolf, I., Dengel, A., Ahmed, S.: Deepdesrt: deep learning for detection and structure recognition of tables in document images. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 1, pp. 1162–1167. IEEE (2017)
23. Soille, P.: Morphological Image Analysis: Principles and Applications. Springer Science & Business Media, Heidelberg (2013)
24. Sudana, O., Putra, D., Sudarma, M., Hartati, R.S., Wirdiani, A.: Image clustering of complex balinese character with dbscan algorithm. J. Eng. Technol. **6**(1), 548–558 (2018)
25. Xu, R., Wunsch, D.: Survey of clustering algorithms. IEEE Trans. Neural Netw. **16**(3), 645–678 (2005)