scientific data

OPEN

DATA DESCRIPTOR

A large-scale dataset for end-to-end table recognition in the wild

Fan Yang[1], Lei Hu[1], Xinwu Liu[2], Shuangping Huang[1,3] ✉ & Zhenghui Gu[4]

Table recognition (TR) is one of the research hotspots in pattern recognition, which aims to extract information from tables in an image. Common table recognition tasks include table detection (TD), table structure recognition (TSR) and table content recognition (TCR). TD is to locate tables in the image, TCR recognizes text content, and TSR recognizes spatial & ontology (logical) structure. Currently, the end-to-end TR in real scenarios, accomplishing the three sub-tasks simultaneously, is yet an unexplored research area. One major factor that inhibits researchers is the lack of a benchmark dataset. To this end, we propose a new large-scale dataset named Table Recognition Set (*TabRecSet*) with diverse table forms sourcing from multiple scenarios in the wild, providing complete annotation dedicated to end-to-end TR research. It is the largest and first bi-lingual dataset for end-to-end TR, with 38.1 K tables in which 20.4 K are in English and 17.7 K are in Chinese. The samples have diverse forms, such as the border-complete and -incomplete table, regular and irregular table (rotated, distorted, etc.). The scenarios are multiple in the wild, varying from scanned to camera-taken images, documents to Excel tables, educational test papers to financial invoices. The annotations are complete, consisting of the table body spatial annotation, cell spatial & logical annotation and text content for TD, TSR and TCR, respectively. The spatial annotation utilizes the polygon instead of the bounding box or quadrilateral adopted by most datasets. The polygon spatial annotation is more suitable for irregular tables that are common in wild scenarios. Additionally, we propose a visualized and interactive annotation tool named *TableMe* to improve the efficiency and quality of table annotation.

Background & Summary

Tables are commonly presented in images to organize and present information. To efficiently utilize information from table images, computer vision based pattern recognition techniques are used in table recognition (TR). It consists of three main tasks, table detection (TD), table structure recognition (TSR) and table content recognition (TCR), in relation to the localization of tables, the recognition of their internal structures, and the extraction of their text contents correspondingly.

Currently, the end-to-end TR task in real scenarios, with the purpose of fulfilling all three sub-tasks simultaneously, is yet unexplored. One major factor that inhibits researchers is the lack of a well-rounded benchmark dataset. For instance, as shown in Table 1, early (before 2018) TR datasets, such as UNLV[1], ICDAR13[2] and ICDAR17[3], only contain a few samples (less than 2.5k). Later, large-scale TR datasets[4–11] were proposed since 2019, but the annotations are generated by programs instead of human involved and only scanned regular tables are included, hindering the diversity of the datasets due to the monotonous backgrounds and spatial features (e.g. without rotation, distortion, etc.). In fact, to enrich the diversity, it is necessary to collect data in various real scenarios. For example, Gao *et al.*[12] proposed a dataset named ICDAR19 for TD and TSR tasks. It is the first real dataset in the historical document scenario, yet its volume is small (2.4k images). Until recently, Long *et al.*[13] proposed a large-scale (14.5k) practical dataset WTW that covers multiple scenarios in the wild. Although WTW is the largest and multi-scenario, it is only suitable for the TSR task as it lacks table location and content annotations. Furthermore, quadrilateral box is used in the cell location annotation, which is imprecise to distorted

[1]School of Electronic and Information Engineering, South China University of Technology, Guangzhou, 510641, China. [2]Zhuzhou CRRC Times Electric Co., Ltd, Zhuzhou, 412001, China. [3]Pazhou Lab, Guangzhou, 510335, China. [4]College of Automation Science and Engineering, South China University of Technology, Guangzhou, 510641, China. ✉e-mail: eehsp@scut.edu.cn

Dataset	#Images	#Tables	Task				Multiple Wild Scenarios	Spatial Annotation Flexibility	Border-incomplete Diversity	Bi-lingual	Year
			TD	TSR	TCR	End-to-End TR					
UNLV[1]	427	558	✓	✓	✓	✗	✗	✗	✗	✗	2010
ICDAR13[2]	128	156	✓	✓	✓	✗	✗	✗	✗	✗	2013
ICDAR17[3]	2417	1020	✓	✗	✗	✗	✗	✗	✗	✗	2017
DeepFigures[4]	1.67 M	1.4 M	✓	✗	✗	✗	✗	✗	✗	✗	2018
PubLayNet[5]	362 K	113 K	✓	✗	✗	✗	✗	✗	✗	✗	2019
SciTSR[6]	15 K	15 K	✗	✓	✓	✗	✗	✗	✗	✗	2019
Table 2Latex[7]	465 K	465 K	✗	✓	✓	✗	✗	NA	✗	✗	2019
ICDAR19[12]	2,439	3.6 K	✓	✓	✗	✗	✗	✗	✗	✗	2019
TableBank[8]	278 K	417 K	✓	✗	✗	✗	✗	✗	✗	✗	2020
	145 K	145 K	✗	✓	✗	✗	✗	NA	✗	✗	
PubTabNet[9]	568 K	568 K	✗	✓	✓	✗	✗	✗	✗	✗	2020
TableX[10]	1 M +	1 M +	✗	✓	✓	✗	✗	NA	✗	✗	2021
PubTables-1M[11]	1 M +	1 M +	✓	✓	✗	✗	✗	✗	✗	✗	2021
WTW[13]	14.5 K	14.5 K	✗	✓	✗	✗	✓	✗	✗	✓	2021
TabRecSet (Ours)	32.07 K	38.17 K	✓	✓	✓	✓	✓	✓	✓	✓	

Table 1. A statistical summary and comparison between our *TabRecSet* dataset and the existing datasets. **End-to-End TR**: Extract the table body position, structure, and content simultaneously from a complete image. **Spatial annotation flexibility**: The dataset uses the polygon instead of the Bounding box (Bbox) or quadrilateral to annotate the table or cell position. **Border-incomplete Diversity**: The dataset has multiple types of border-incomplete tables. **NA**: This item for the dataset is not applicable.

Field Name	Description	Field Name	Description	Field Nme	Description
version	the version of *LabelMe*	lineColor	color of the lines in annotation objects	imageData	encoded image data
flags	flags of the image	fillColor	color of the regions of the annotation objects	imageHeight	height of the image
shapes	the annotation objects for the image	imagePath	file path of the image	imageWidth	width of the image

Table 2. Fields of LabelMe format framework.

(caused by folds and bends in a paper) table images. Overall, we conclude three main drawbacks of these datasets as follows: 1. Only provide annotations for sub-tasks (TD, TSR and TCR), which are not complete for the end-to-end TR task. 2. Either the scales are small, or scenario diversities are limited. 3. Only have the Bounding box (Bbox) or quadrilateral as the spatial annotation that cannot flexibly adapt to the shape changes Table 2.

We propose a dataset named Table Recognition Set (*TabRecSet*) with samples exhibited in Fig. 1. **To the best of our knowledge, it is the largest and most well-rounded real dataset, collecting data from various wild scenarios with diverse table styles and complete & flexible annotation against for the end-to-end TR task with the purpose of filling the gap in this research area. Large Scale:** The data volume (including more than 38,100 real table images) is 2.6 times larger than the largest known dataset WTW. **Wild Scenario:** Data are collected via scanners or cameras in various wild scenarios including documents, Excel tables, exam papers, financial invoices, etc. **Robust Diversity:** It contains different table forms, such as the regular and irregular table (rotated, distorted, etc.), border-complete (all-line) and -incomplete table. The distortions of irregular tables may severely break the spatial alignment of rows, columns and cells, increasing the difficulty of the TSR task. The recognition of border-incomplete tables such as the three-[14] and no-line (without borders) tables are also more difficult and challenging. **Completeness:** In order to provide a complete annotation for the TR task, the annotation of every table sample in *TabRecSet* contains its body and cell location as well as its structure and content. **Flexibility:** In order to provide accurate and precise annotations to distorted tables, *TabRecSet* uses polygons instead of bounding boxes to annotate the outside and inside table borders. **Bi-lingual:** *TabRecSet* contains Chinese and English tables with a proportion of 46.5% and 53.5% independently Table 3.

In addition, since the process of dataset building is quite time-consuming, we developed a visualized and interactive annotation tool named *TableMe* to speed up the annotation process and ensure data quality. We also designed several automatic techniques, such as the auto annotating of table structures and the automatic generation of three- and no-line tables, to benefit the end-to-end TR task in wild scenarios.

Methods

In this section, we elaborate on all the details in the *TabRecSet* creation procedure, which ensures the quality, reproducibility, and creation efficiency of the dataset. As Fig. 2 illustrates, this procedure mainly consists of four steps. The first three steps, data collection, data cleaning, and data annotation, following a normal and standard procedure of building most datasets, output all border-complete table samples. Particularly, to increase

Fig. 1 Some representative samples in *TabRecSet*. The scenarios include the document images, ingredients form of foods, Excel tables and invoice tables. Because of the page distortions or camera views, most tables are irregular, i.e., with rotations, inclinations, concave/convex/wrinkle distortions, etc. Some special table forms are exhibited, e.g., the nested table, under- and over-exposed table, border-incomplete table, table with hand-written contents and hand-drawn table.

Field Name	label	points	group_id	shape_type	flags
Description	class of the annotation object	coordinates of the annotation	instance id of the annotation object	type of the annotation object (polygon, circle, line, point)	flags of the annotation object

Table 3. The sub-fields of the *"shapes"* field.

the number of samples and the variety of table styles, they are also used to generate border-incomplete tables (e.g., three- or no-line tables) in the fourth step. The generation process is automatic and image-based, which basically replaces border pixels from border-complete table images with its background pixels.

Data collection. The general purpose of the data collection step is to build a raw data pool by searching and downloading enough table-related images through the Internet. Firstly, we randomly pick camera-taken table image samples from open source datasets such as WTW or *Tal ocr_table*[15] as search seeds. Then they are input into search engines (e.g., Google or Baidu) with the Usage Rights filter enabled and return plenty of similar images that comply with the Creative Commons licenses. For the search engine that does not have the Usage Rights filter, we manually open the image's original source to check whether the image complies with the licenses. After that, the search results are downloaded via a web page-based image downloader called *ImageAssistant*[16]. The whole process stops when the total size of the downloaded images exceeds a specified threshold. The advantage of using search engines to collect table images is that the search result covers a wide variety of data collection scenarios, including reports, exam papers, documents, invoices, books, etc. In addition, to further increase data diversity, we use raw images rather than keywords as the search seeds because the image content does not play an important role in the way of search by image, and the search engine searches for matched images based on the pixel-level similarity of input images. In this way, the search engine will return a large number of table images with a diverse range of formats and styles, covering many special cases such as irregular, distorted, and incomplete tables, while in the way of search by keyword, the extent and degree of the distortion or the incompleteness are difficult to

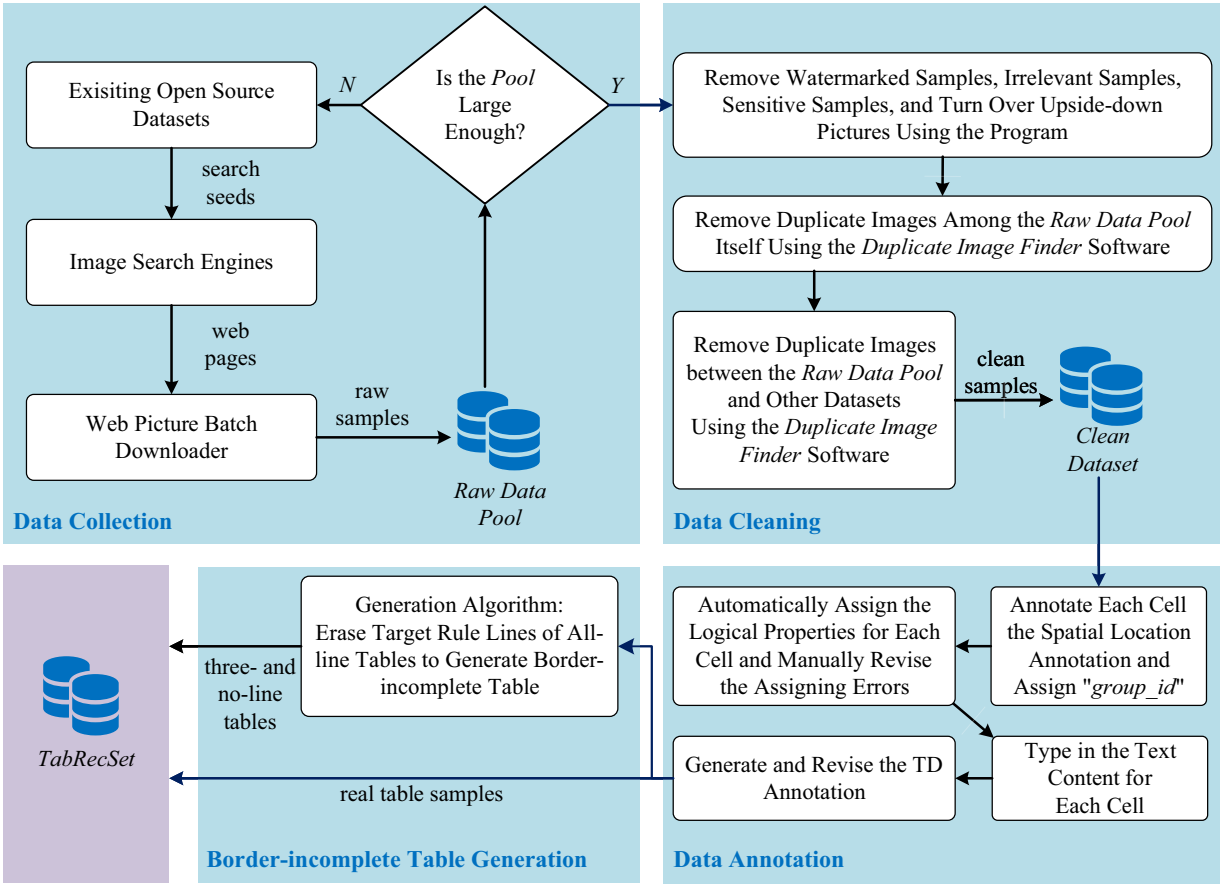

Fig. 2 The creation flow chart of TabRecSet. The data collection aims to collect raw image samples and outputs a *Raw Data Pool*, which stores candidate data samples. The data cleaning step generates clean samples from *Raw Data Pool* and gathers them into a *Clean Dataset*. In the data annotation step, we use *TableMe* to annotate the clean sample and save the annotation in the *TabRecSet* annotation format. This step is aided by several auto-annotation algorithms to improve efficiency. The border-incomplete table generation step aims to enlarge the scale *TabRecSet* by our proposed three-line table generating algorithm.

describe. As shown in Fig. 2, we repeat the search and download process until the data size of the *Raw Data Pool* exceeds the expected dataset scale by a pre-specified margin based on the filtering rate in the cleaning step.

Data cleaning. The data cleaning step in Fig. 2 is a human-involved process of fixing or removing incorrect, incomplete, incorrectly formatted, duplicate or irrelevant data within the *Raw Data Pool*.

Since the raw data is collected through the Internet in the way of search by image, some images that do not include actual table instances may be returned by the search engine. These incorrect and table irrelevant data are removed in the first place. Meanwhile, watermarked images are removed because of copyright protection. In addition, for privacy considerations, we also remove sensitive information, such as location, ID, phone number, etc., from those images. In terms of the data format, in order to keep the variety of table format and styles, we only fix sideways or upside down images. After that, we detect and remove duplicate images (keep the image with the highest image resolution and remove the rest) via *Duplicate Image Finder*[17] software. It helps the user identify duplicate images, even if they are resized, edited, flipped, color-corrected, etc., by grouping them together. Moreover, samples duplicated from other datasets without a derivative license, e.g. the *Tal ocr_table* dataset, are also entirely removed.

The data cleaning step filters out approximately 30% (based on our experience) of "dirty" data from *Raw Data Pool* to obtain a clean dataset with high-quality data for the annotating process in the following step.

Data annotation. We first introduce the annotation format of *TabRecSet* ([subsubsec: data annotation format]Annotation Format) and our developed annotation tool *TableMe* ([subsubsec: Data Annotation Tool] Annotation Tool). Then, we propose a TSR auto-annotating algorithm ([subsubsec: TSR Annotation Generating Algorithm]TSR Auto-annotating Algorithm) to automatically generate logical structure annotation based on the spatial structure annotation. Finally, in the [subsubsec: data annotation step]Annotation Step subsection, we describe the data annotation step for the *Clean Dataset*, which is mainly performed on our tool, including cell polygons drawing, algorithm-assisted logical locations generation, typing in text contents and table body polygons generation.

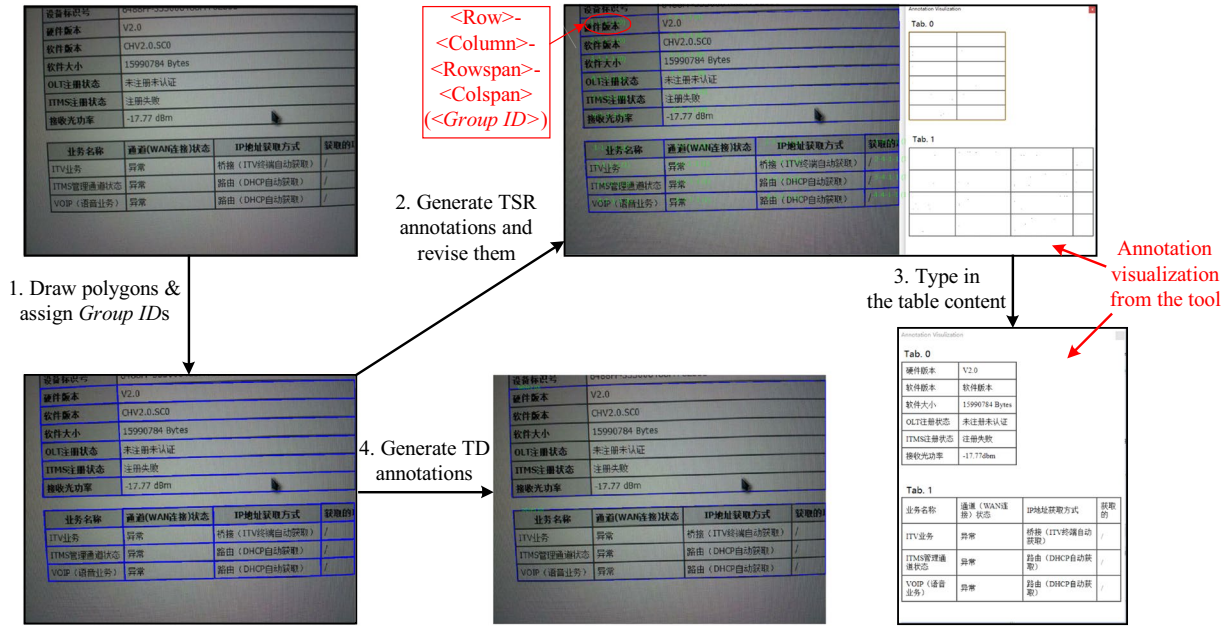

Fig. 3 An intuitive illustration of the data annotation step showed in Fig. 2. Please zoom in for details.

Annotation format. The complete annotation for the end-to-end TR task is complex as it includes table body position for the TD task, the cell spatial & logical location for the TSR task and the cell text content for the TCR task, covering multiple heterogeneous information from spatial and logical to text data. It is necessary to utilize proper annotation formats to organize the information coherently and concisely. We choose the *LabelMe*[18] annotation format as the framework for our annotation formats since this framework supports compactly organizing the heterogeneous table information. This annotation format framework is defined by the fields and sub-fields listed in Tables 2,3.

We form two annotation formats: a table-wise annotation format in which the annotation object is a table and a cell-wise one with a cell as the annotation object. The table-wise annotation format is for the TD task containing the spatial location information for the table. The cell-wise format contains the spatial & logical locations and text content information for a cell annotation, and the collection of these cell annotations completely describes TSR and TCR annotations for the whole table[19].

Concretely, the table-wise annotation format utilizes the *"points"* field to store the vertexes coordinates of the table body polygon and the same *"group_id"* (an integer) to distinguish different table instances. The cell-wise format uses the *"label"* field to store a text string that encodes the cell's logical locations and text content, the *"points"* field to store the vertexes coordinates of the polygon along the border of the cell, and the *"group_id"* field as in the table-wise format to mark the cell to which table instance it belongs. The text string in the *"label"* field is in the form of "*<Row>-<Column>-<Rowspan>-<Colspan>-<Text content>*" in which the <Row> (<Column>) means the row (column) number of the cell and the <Rowspan> (<Colspan>) means how many rows (columns) the cell spans. The row number, column number, rowspan and colspan are also called logical properties for short. Figure 4 gives examples of three cell annotation instances in the cell-wise annotation format. The #1 instance indicates the cell annotation object located by polygon [[19,202],[92,204],[391,212],[391,227],[168,221],[18,217]], in the tenth row and the first column of Table 1 (group id = 1), spanning three columns, with the text content "预计费用总额". The #2 instance indicates the cell annotation object located by polygon [[12,308],[8,402],[121,403], [123,341],[125,310]], in the fourth row and the first column of the table 0, spanning five rows, with the text content "担保公司". The #3 instance indicates the cell annotation object located by polygon [[362,298],[363,313],[462,315], [462,299]], in the third row and the fourth column of the table 0, with the **hand-written "1600" as the text content**.

Annotation tool. *TableMe* originate from the famous annotation tool *LabelMe*, which is powerful in providing the annotation for the image segmentation task, and *TableMe* completely inherits this feature leading to a great capacity for the table or cell's spatial annotating. Besides, it possesses annotating functions for the table structure & content and supports assigning logical properties and *"group_id"* for a group of selected cell annotations, enhancing the structure annotating speed significantly. Most amazingly, it can intuitively visualize the logical structure and content of the table, which helps us to transcribe the text content to the annotation straightforwardly and efficiently (as shown in Fig. 3).

As illustrated in Fig. 5, *TableMe* is mainly composed of three parts: an image panel (upper-left), a setting panel for polygon properties (structure logical & *"group_id"*) (lower-right) and an annotation visualization region (lower-left). The image panel is a feature originated from the *LabelMe*, which not only inherits the convenient polygons drawing functions of *LabelMe* for the table/cell position annotating but also supports selecting these polygons in a group for the setting of polygon properties in the setting panel.

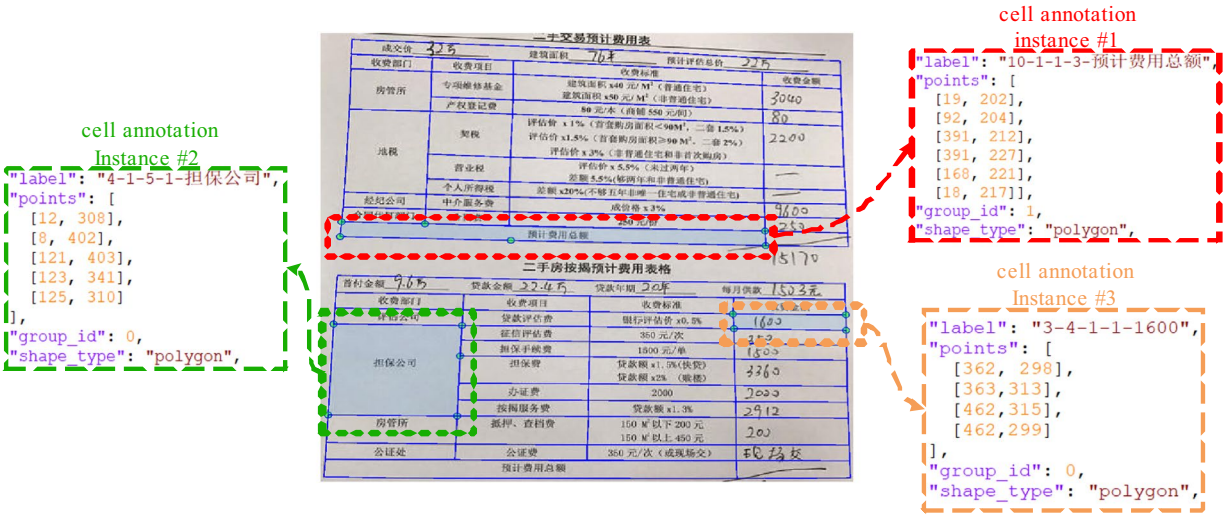

Fig. 4 Three annotation instances in the cell-wise annotation format.

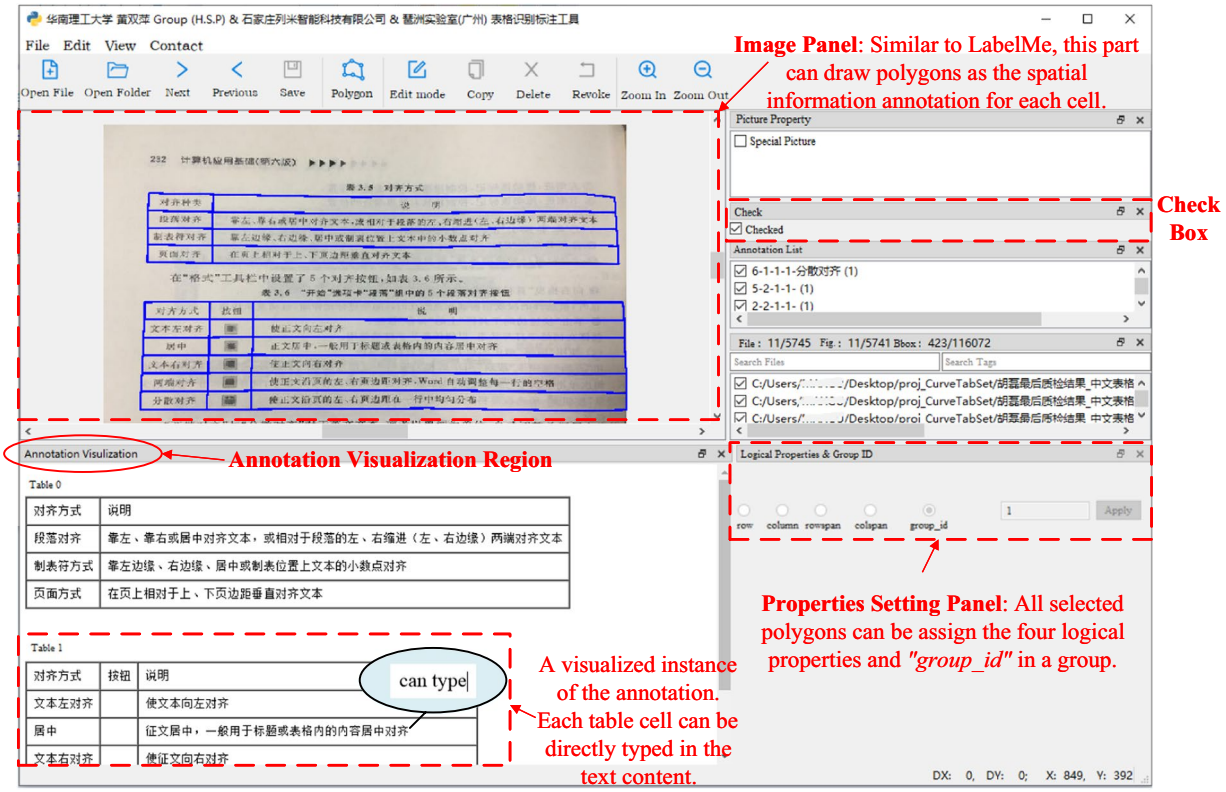

Fig. 5 The main interface of *TableMe*. Please zoom in for details.

The properties setting panel includes five options, i.e., the row, column, rowspan, colspan and "*group_id*", and two widgets, i.e., an editable text for the option value input and an "Apply" button. For example, for the setting of the row property, the annotator can first select the polygons on the same row in the image panel, choose the "row" option, input the row number, and click the "Apply" button to confirm. The other properties can be set with totally the same operations. With this feature, *TableMe* enables an intuitive annotating way for the table structure, which is high-efficiency and has less error tendency when people annotate.

The annotation visualization region supports two functions: visualizing the table structure and content annotation in a digital table form and annotating the table content directly in the digital table. After annotating the table/cell position in the image panel and table structure in the properties setting panel, this region will immediately show digitalized tables with the same logical structure as the tables in the image. Amazingly, as these digital tables are interactable, it allows users to annotate the text content of a cell simply by clicking that digital cell in

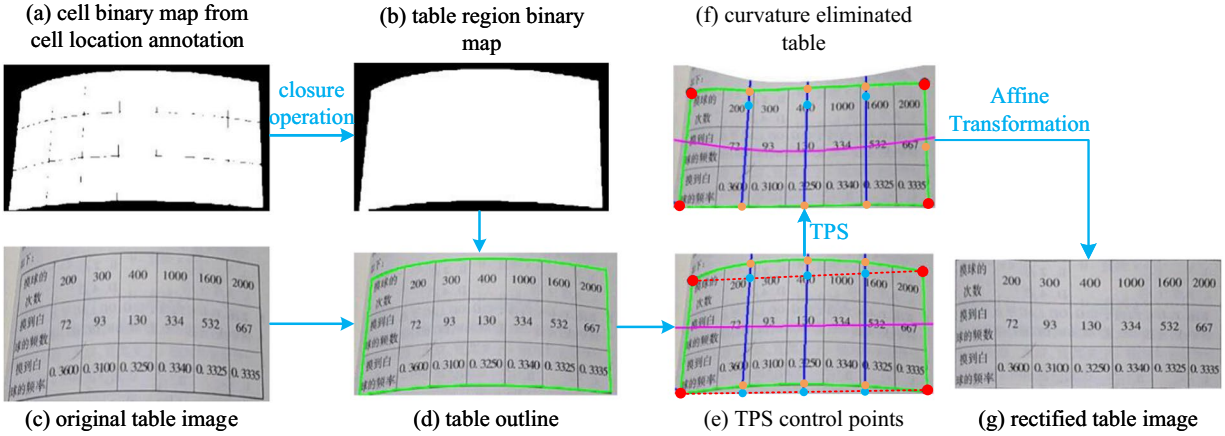

Fig. 6 The table image rectification algorithm. The blue, red, orange points, blue lines and two red dashed lines in Fig. (e,f) are the TPS target points, TPS source points, corner points of the table outline, normal lines and corner lines, respectively. A purple line is drawn horizontally in the middle of Fig. (e) and is distorted in Fig. (f), visualizing the extent and direction of the TPS transformation.

this region and typing in the text string directly. This feature, on the one hand, increases the speed of annotating the text content and, on the other hand, provides an effective way for users to check whether the manually or automatically generated structure annotations are correct or not.

In the multiple tables case, the annotator can set the polygons in the same table with the same *"group_id"* and ensure polygons in different tables have different *"group_id"* by operating the widgets on the properties setting panel. When finished setting the *"group_id"* for each polygon, as shown in the figure, the annotation of multiple tables will be visualized in the annotation visualization region and distinguished from each other via table numbers that are equal to the *"group_id"*s.

Besides above mentioned three parts, there is a check box named "Checked" on the right side of the interface. When an image is annotated, we can check this box to mark the image as annotated, and the value of the *"flags"* field in 4 will be set to "true". The annotation is saved as a *LabelMe* JSON file in the *TabRecSet* annotation format with the same filename of the image but a different file extension. After finishing annotating all images in *Clean Dataset*, a folder has paired JSON files and JPG files can be obtained, which contains the complete TR annotation for each image.

In conclusion, in terms of functionality, *TableMe* is dedicated to annotating the end-to-end TR task in the wild scenario as it supports annotating table position, structure and content in multiple tables and irregular table cases. In terms of efficiency, it highly improves the speed of table annotating, annotation checking and revising, especially when the image contains many tables and the table has a large cell number, as it avoids trivially treating each table cell one by one.

TSR Auto-annotating algorithm. The TSR annotation generating algorithm consists of two parts: 1) A table image rectification process to eliminate the irregularity of tables (distortions, rotations, etc.) and 2) a logical property computing process to compute the logical properties for each cell. Without the table irregularities, the logical property computing process can get the most out of it, thus making this algorithm has a high computing accuracy, essentially preventing us from manually annotating these properties and improving efficiency.

The rectification process uses the cell location annotation to remove the distortions by Thin-Plate Spline (TPS)[20] transformation and remove the rotations and inclinations of the table by Affine transformation[21]. Fig. 6 illustrates the main steps of the algorithm. Firstly, we generate a binary map of each cell based on the spatial annotations of a table and apply the morphological closure operation, a classical image processing algorithm, to fill the gap between each cell to obtain a table-region based binary map. Secondly, find the table outline by tracing the border of the binary map via the *findContour* API in OpenCV. Then we find the corner points of the outline (red points in Fig. 6e) and link the corner points to obtain the corner lines (red dashed lines). The TPS control points (blue points) are the equal-division points of the corner lines. Make lines pass through target points and are perpendicular to the corner lines, and we obtain the normal lines (blue lines) whose intersections with the table outline are exactly the source points (orange points). The next step uses the TPS transformation to minimize the distances between target points and source points with the smallest bending energy and establishes a coordinate map between the original table image and the transformed table image of which the curvature distortion is eliminated. With the four corner points, the algorithm computes the Affine parameters to transform the image, which can remove the rotation and inclination of the table, and finally, we obtain the rectified table image.

The logical property computing process computes the logical properties by analysing the spatial relationships among cells on the rectified table and assigns the logical properties to the cell on the original table according to the coordinate map from the rectification process. Figure 7 shows key steps to analyse the spatial relationship and compute the logical properties:

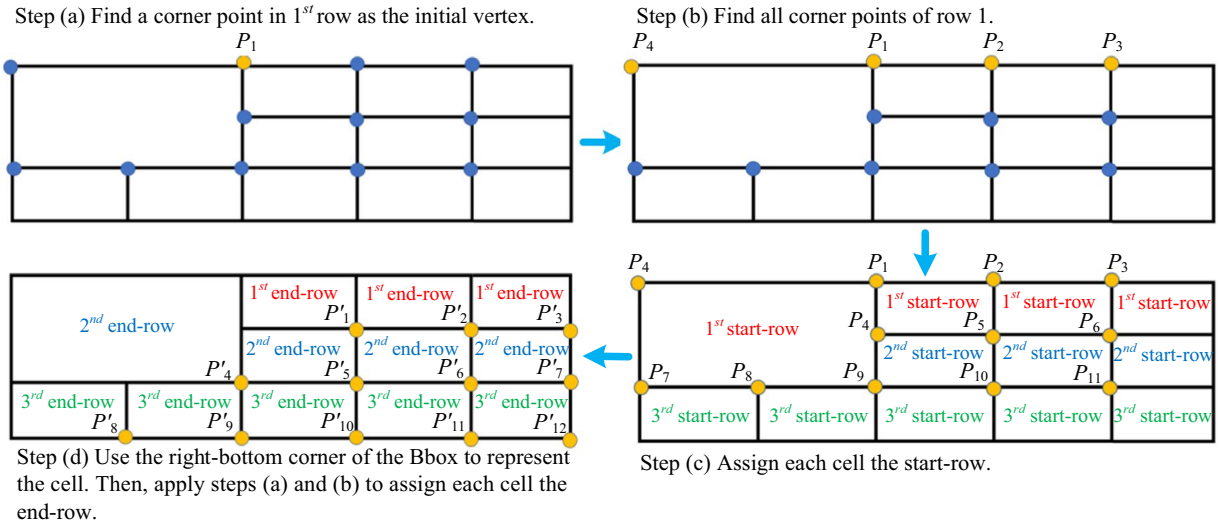

Fig. 7 The key steps of our logical property computing algorithm on the rectified table image. We use a regular table on a plain white background to represent the rectified table image.

Step (a), we choose the left-top corner point of a Bbox to represent a cell. The algorithm first finds an initial point P_1 of which the y-coordinate is the smallest.

Step (b), find all the points satisfying the restriction $|P_1^y - P_2^y| < min\{H_{Bbox}\} \times 0.4$ which ensures they are in the same row. The P^y is the y-coordinate, $min\{H_{Bbox}\}$ is the height of the shortest Bbox in the table, and the parameter 0.4 is an empirical constant. Then, we can obtain all cells with the 1^{st} start-row (or the 1^{st} row in short) and remove them from the table for the next step.

Step (c), repeat Step (a)-(b) until there is no point left on the table, and all cells are assigned with the start-row property.

Step (d), to obtain the end-row property, we can use the right-bottom corner point to represent each cell and do steps (a)-(c).

Step (e), the rowspan of each cell is computed via the formula: rowspan = end-row - start-row + 1.

Step (f), symmetrically, the colspan of each cell is computed via the formula: colspan (= end-column - start-column + 1) after obtaining the start- and end-column of the cells by following (a–d) steps on the x-coordinate.

Annotation step. According to the data annotation step shown in Fig. 2 and the intuitive illustration shown in Fig. 3, we first draw polygons along the cell borders in the image panel to annotate each cell the spatial location and assign *"group_id"*s to these polygons for distinguishing table instances using the properties setting panel.

For the efficiency consideration, we apply the TSR auto-annotating algorithm (see the [subsubsec: TSR Annotation Generating Algorithm]TSR Annotation Generating Algorithm subsection) to generate the logical properties for each cell automatically, then detect the occasional generating errors in the annotation visualization region and manually revise the errors via the properties setting panel. This algorithm has approximately 80% accuracy, so we only needed to fix the remained 20% of the annotations, which significantly improved our annotation efficiency.

As shown in Fig. 3, for text content transcription, we can directly type in the text for each cell in the digital table shown in the annotation visualization region, and the tool will store the text in the <*Text content*> part of the encoded string in the *"label"* field. Note that for an indistinguishable blurred character, we replace its actual annotation with the # symbol to indicate its existence. For a cell with multiple text lines, we use the\n escape symbol to separate each line and concatenate the text lines to a single text string.

Finally, we auto-generate the TD annotation for the whole table based on the spatial cell annotations via an image processing program. The program process is as follows: (1) convert spatial annotations (i.e. polygon list) to binary maps, then concatenate them together to obtain a single segmentation map[22]; (2) use the morphological closure operation on the segmentation map to fill the gaps between each cell's binary region; (3) convert the segmentation map to the polygon along the map contour via the *findContour* API in OpenCV. After the generation, we refine the generated annotation manually in the image panel to ensure the rightness of the annotation.

Border-incomplete table generation. As shown in Fig. 2, this border-incomplete table generation step aims to produce the three-line and no-line table by erasing the target rule lines of the annotated all-line table.

A table rule line is composed of the edges of the cells on the same row or column, and thus we can erase a table rule line by erasing the cell edges. Since the cell-wise polygons along the cell borders are annotated, we apply an image processing program using these polygon annotations to find the cell edges in the image and remove these edges. Figure 8 shows how the program process erases a single cell edge. Given a table image and the cell border from the annotation, the first step of erasing a cell edge is to obtain all pixel points on the edge (red line) by extracting the sorted points on the cell border bounded by two adjacent corner points (red points).

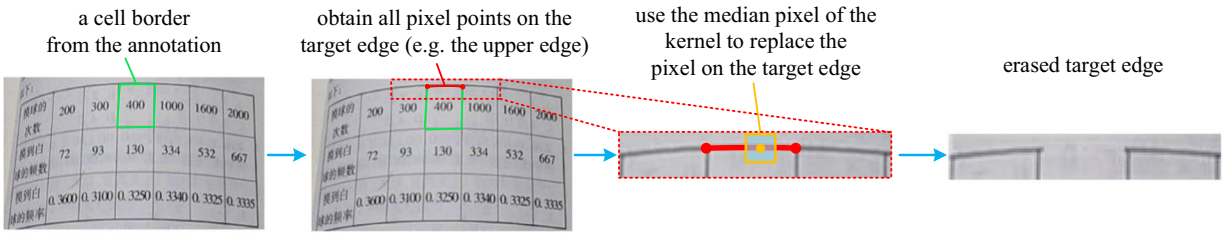

Fig. 8 The key steps of erasing a target cell edge. Red points: Two corner points of the cell border. Red line: A target edge of a cell. Orange square: A kernel centered on a pixel (the orange point) on the target edge.

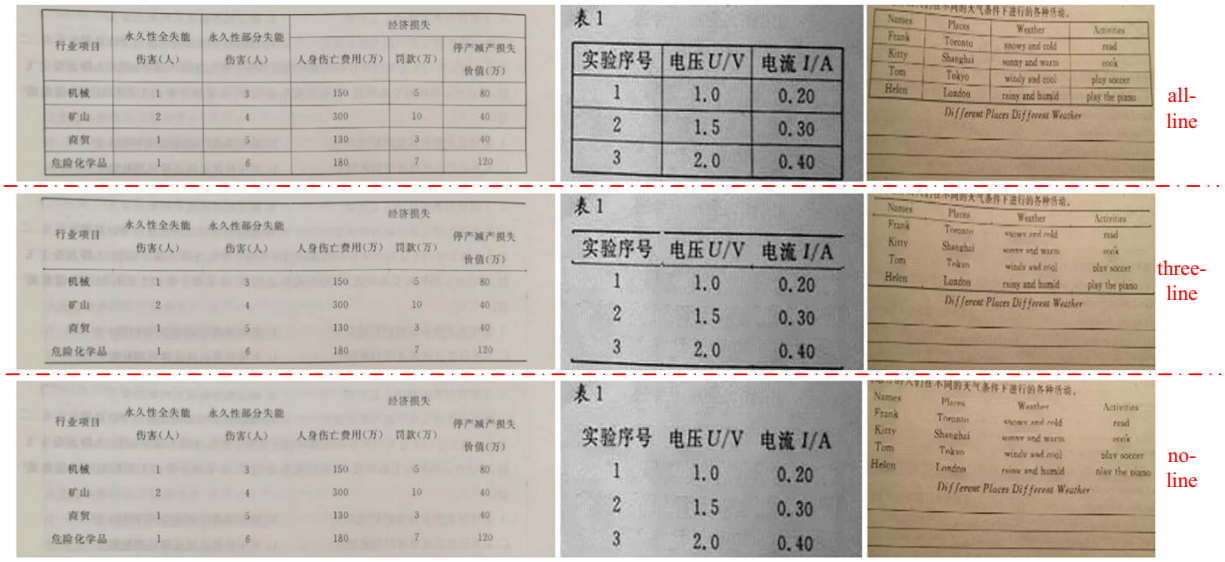

Fig. 9 Generated three-line table examples.

For example, the upper edge is a sorted point list starting with the upper-left corner point and ending with the upper-right corner point. The second step is to replace each pixel on the edge with the median pixel of a kernel (orange square), which is centered on the pixel (orange point). Because the median pixel of the kernel mainly refers to the background color, this step actually replaces the border pixel with the background color. When all pixels on the edge are replaced, this edge can be regarded as removed from the image.

To generate a three-line table, we should erase the horizontal (row) rule lines, ranging from the third to the last but one, and all vertical (column) rule lines. Concretely, we erase the left and right edges of the cells on the 1^{st} row and erase all cell edges on other rows except the upper edge on the 2^{nd} row and the bottom edge on the last row. As for the no-line table generation, we should erase all target rule lines, and thus we simply erase all edges of every cell in the table. Figure 9 illustrates the generating performance.

Note that the border-incomplete table shares the same cell location annotation with the original all-line table. The polygon annotation is originally drawn along the cell border in the all-line table, while these borders may be erased in the border-incomplete table, so the spatial annotation for a cell is a loose polygon relative to the text content in the border-incomplete table case. Though existing datasets[6,9] use a compact Bbox for the text content as the cell location annotation, we insist on our loose annotation because of the existence of the cell borders even though they are invisible (erased). The insight is that a human can somehow infer where are the invisible borders in the image by visual cues or semantic meanings and this insight means the existence and uniqueness of invisible borders in the border-incomplete table. We believe that providing polygon annotation for the invisible borders can facilitate the emergence and development of the cell invisible border recovery task.

Summary of tools. Table 4 is a complete summary of the tools we used during the dataset creation procedure. In the table, we describe the primary uses of these tools and their advantages compared to the alternatives. The last column of the table lists the tool versions, which sometimes matter during the creation procedure.

Data Records

Directory structure of *TabRecSet*. *TabRecSet* is publicly available in figshare[23]. Its directory structure is shown in Fig. 10. The image folder contains original and generated table images in JPG format in which tables with different languages and border-incomplete types are separated into corresponding sub-folders. Each image is one-to-one mapped to a unique TD annotation (JSON files in the TD_annotation folder) and TSR/TCR

Name	Description	Advantages	Version
TableMe	It is our proposed tool for table-specific annotating, which supports annotation for multiple task types, such as table detection, table segmentation, table structure, and table content recognition tasks.	1. It uses interactive visualization to execute the annotation process effectively. 2. Compared to alternatives, it supports the table segmentation task.	1.0.0
Duplicate Image Finder[17]	*Duplicate Image Finder* "looks" at your images to find look-alike images in a folder. It can identify similar and duplicate images even if they are edited, rotated or flipped.	1. It can identify rotated at 90°, 180°, 270°, flipped horizontally and/or vertically duplicate images. 2. It can show all the duplicate images in groups and mark the smaller resolution and/or smaller file size (lower quality ones) images to be deleted.	4.8.0
Image-Assistant[16]	*ImageAssistant* is an extension software running in Chrome and its derivative browsers to analyze and extract pictures in web pages and provide multiple filtering methods to assist users in selecting and downloading.	Different from browser extensions that provide similar functions in the past, this extension combines multiple data extraction methods to ensure that the images that have appeared can be extracted as comprehensively as possible from various complex structure pages.	1.66.6

Table 4. Summary of the tools we used during the dataset creation procedure.

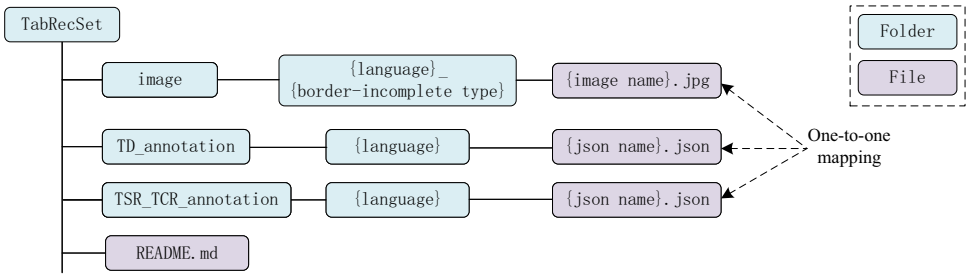

Fig. 10 Structure of the data included in *TabRecSet* dataset.

annotation (JSON files in the TSR_TCR_annotation folder) according to its filename, and each JSON file is also divided into different sub-folders by their language. Note that an original all-line table shares the same filename and annotations with its generated border-incomplete tables since the annotations for the all-line table are also valid for the generated images. In the README.md file, we summarise the meta information, such as the dataset license, download links, description of the file format and a link to the source code repository, etc.

Statistics of *TabRecSet*. To quantitatively verify the data quality and challenge of *TabRecSet* for each sub-task, we analyse its overall size and several instance-wise (e.g., table-wise, cell-wise) statistical characteristics and draw their distributions in Fig. 11 and Table 5. The instance-wise statistical characteristics include the table number of each cell (Fig. 11a), cell number of each table (Fig. 11b), rowspan/colspan of each cell (Fig. 11c,d), vertex number of each cell (Fig. 11e), content length of each cell (Fig. 11f), word frequency (Fig. 11g,h) and character frequency (Table 5).

In the aspect of overall size, *TabRecSet* contains 32,072 images and 38,177 tables in total among which 16,530 images (17,762 tables) are in Chinese, 15,542 images (20,415 tables) are in English and 21,228 images (25,279 tables) are generated (three-line and no-line). The generated table subset (border-incomplete folder in Fig. 10) contains 5,113 images and 6728 tables (both three- and no-line tables) in English, 5,501 images and 5,911 tables in Chinese (both three- and no-line tables).

We count the table number for every image, which is an image-wise indicator to measure the difficulty of the TD task, and draw the distribution over images in Fig. 11a. According to the statistical result, approximately 300 images in the English subset contain multiple tables, and 1,000 images in the Chinese subset contain multiple tables. The maximum number of tables in an image on English and Chinese subsets are 7 and 4, respectively. We believe that our dataset can benchmark the performance of the TD model in the case that the image contains many tables.

The difficulty of TSR for a table varies with the table size and structure complexity. We choose the cell number as the table-wise indicator and the number of spanning cells as the cell-wise indicator to reflect the two respects, respectively. Figure 11b shows the cell number distribution of which the average number is 29 for the English subset, 18 for the Chinese subset, and the maximum number is 351 for the English subset, 207 for the Chinese subset. The spanning cell refers to the cell of which the rowspan or colspan is larger than one, so we summarize the rowspan/colspan of each cell in Fig. 11c,d to indicate the structure complexity of the dataset. The spanning cell number of the English subset is more than 6,600, and the Chinese subset is more than 2,200. The maximum rowspan and colspan are 41 and 24 for the English subset; 22 and 24 for the Chinese subset. These statistics show that our dataset contains a great number of large tables and has a high overall structure complexity. TSR not only needs to recognize the logical relation among cells but also needs to locate the cell position in the image. The vertex number of a cell polygon reflects the curvature of the cell, which affects how difficult to segment the cell, so we count the distribution of the vertex number for each polygon annotation to verify the challenge of our dataset in cell locating. As shown in Fig. 11e, thousands of cell polygons have more

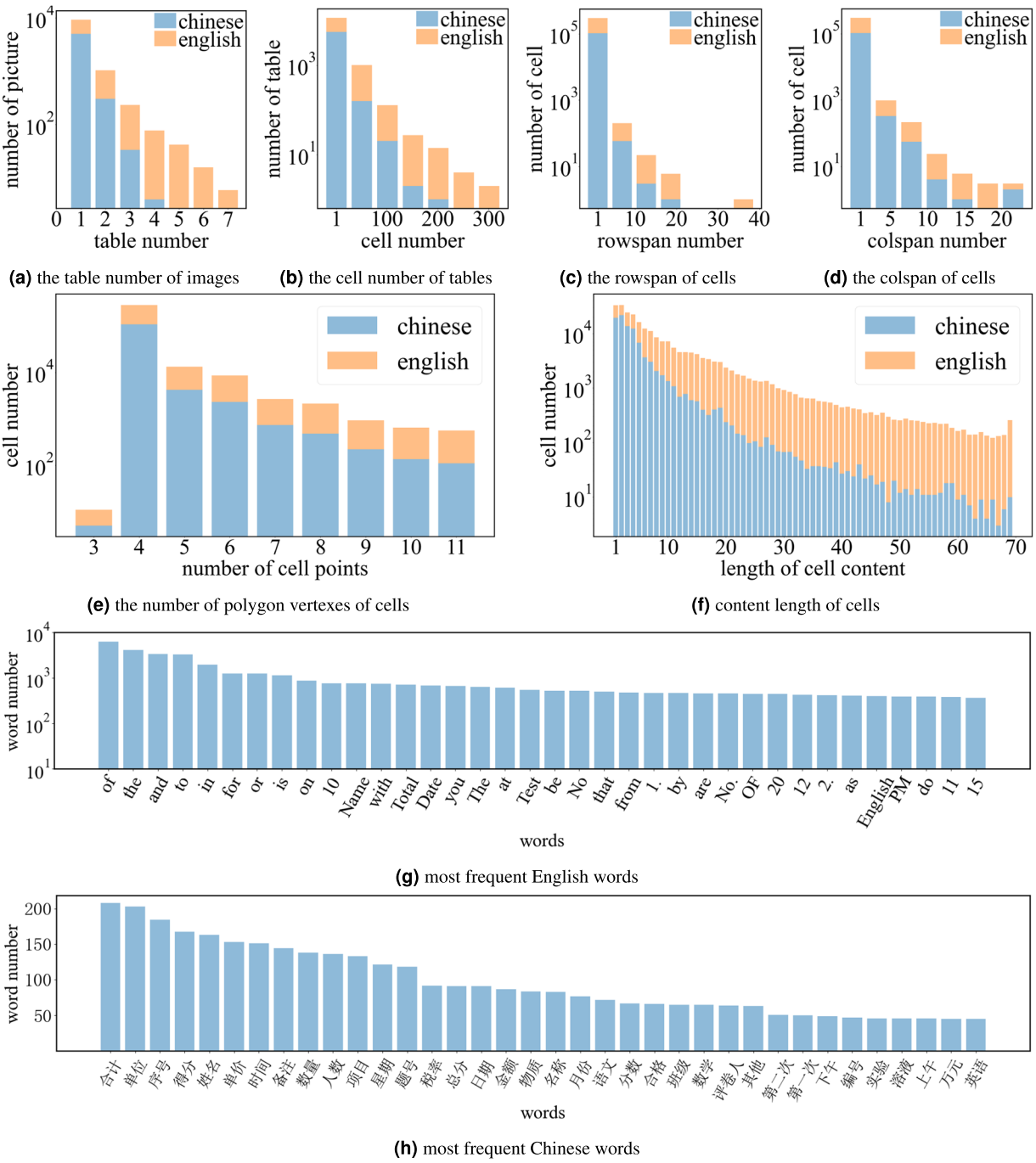

Fig. 11 Statistics data of *TabRecSet*.

than five vertexes, and hundreds of polygons have more than nine vertexes, which is an extremely distorted case, indicating a high challenge for our dataset in the cell locating task.

Table 5 exhibits the occurring frequency of commonly used characters in the English and Chinese subsets, which shows the coverage of characters. The commonly used characters include the upper and lower case English letter, digit, English and Chinese punctuation mark, and the most commonly used thirty-two Chinese characters[24] (last three rows). Note that we give the frequency of the Chinese character for the English subset and that of the English letter for the Chinese subset because we differ the English and Chinese tables not by the language of the table content but the context, so an English table may contain Chinese characters and vice-versa. Besides the character-wise data for the table content annotation, we also summarize the first thirty-five most occurred words in two subsets, as shown in Fig. 11g,h. Unsurprisingly, the most frequently occurring word in the English subset is mainly prepositions, while the one in the Chinese subset mainly depends on the domain. Table 5 and Fig. 11g,h manifest the completeness of the table content annotation, while Fig. 11f illustrates the content length of each cell, which manifests the difficulty of our dataset in the TCR task.

Character	A	B	C	D	E	F	G	H	I	J	K	L	M
Count	34,067 1,318	13,002 989	25,970 1,499	18,093 672	27,634 372	10,560 423	9,072 369	13,034 774	23,148 296	3,011 171	5,444 233	15,630 598	9,880 448
Character	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Count	22,473 645	18,390 811	19,722 492	1,196 131	21,210 256	32,312 616	25,946 373	8,149 122	5,923 304	6,378 213	2,249 171	5,151 163	951 155
Character	a	b	c	d	e	f	g	h	i	j	k	l	m
Count	125,450 1,436	25,272 3,285	50,562 1,105	49,910 542	167,771 998	23,973 233	33,306 1,826	42,310 423	105,066 773	2,024 29	12,292 491	68,205 1,014	43,042 3,023
Character	n	o	p	q	r	s	t	u	v	w	x	y	z
Count	102,827 1,122	111,618 931	38,488 2,912	2,101 142	97,500 577	100,944 6,096	110,944 1,210	60,738 5,641	13,722 169	14,070 133	5,661 868	24,943 467	2,255 144
Character	0	1	2	3	4	5	6	7	8	9	~	‘	!
Count	72,091 29,532	50,827 22,881	40,859 17,216	23,530 11,122	19,493 9,532	23,627 12,093	15,235 7,452	13,832 5,595	13,950 6,871	14,121 5,102	65 551	3,184 1	253 2
Character	@	#	$	%	^	&	*	(	)	-	_	=	+
Count	489 13	34,674 2,289	1,955 7	2,058 1,577	22 5	1,917 356	1,341 129	12,959 3,216	13,420 3,221	21,091 4,141	1,749 439	1,407 414	2,426 1,362
Character	{	}	[	]	|	\	/	<	>	,	.	?	,
Count	61 7	64 7	461 120	464 76	282 24	241 74	19,707 7,816	14,948 5,818	14,886 5,782	16,895 1,076	46,387 11,730	1,485 43	75 1,598
Character	°	!	¥	(	)	、	:	;	“	”	?	<<	>>
Count	14 543	7 20	1 82	77 2,657	104 2,696	41 1,785	8,547 2,014	0 195	55 128	20 124	14 56	0 132	1 128
Character	的	一	是	在	不	了	有	和	人	这	中	大	为
Count	5 2,818	11 1,677	1 294	3 366	5 726	1 152	3 627	2 416	1 1,892	1 64	3 924	1 758	0 297
Character	上	个	国	我	以	要	他	时	来	用	们	生	到
Count	8 830	4 639	1 299	0 170	9 520	3 297	1 161	7 1,385	1 107	1 890	0 70	0 935	3 291
Character	作	地	于	出	就	分	对	成	会	可	主	发	年
Count	0 467	2 665	0 235	2 565	2 63	0 1,863	0 285	2 619	1 256	5 254	2 288	0 378	4 1,147

Table 5. Occurring frequency of the most commonly used characters. Each counting data has two values: the upper one is the frequency of the English subset, and the lower one is the Chinese subset.

Technical Validation

Cross-check. A total of five qualified persons (including part of the authors) were involved in the dataset creation procedure. We were responsible for data collection, cleaning, annotating, and cross-checking the annotations. As shown in Fig. 3, there are four annotating steps in total. Whenever an annotator finishes an annotating step and is about to move to the next step, he will first exchange his assigned sub-dataset with another annotator and cross-check the annotations.

Proofreading. After *TabRecSet* was created, a qualified checker (one of the annotators) was designated to filter out or revise bad samples missed to be dealt with during the creation procedure via *TableMe*. There were two types of bad samples to be checked, dirty images (watermarks, sensitive information, etc.) and incorrect annotation, and the checker cancelled the *Check Box* in *TableMe* for the dirty image and revised the incorrect annotation by the tool. After checking a round, the checker filtered out the images for which the *Check Box*es were not checked by a program. Note that we regard the wrongly generated border-incomplete tables, for example, the not fully erased no-line table, as a type of dirty image, so we directly deleted these border-incomplete tables instead of fixing them. Through this round of checking and programming-based filtering procedure, the two types of bad samples were finally cleaned.

Usability validation. To validate the usability of *TabRecSet*, we train or fine-tune a few state-of-the-art methods on our training set (80% of the whole *TabRecSet*) and evaluate them on the test set (20%) and record the evaluation results in Tab. 16. There is no end-to-end TR model yet, so we validate the usability as completely as possible by covering all sub-tasks.

For the topology structure recognition and content recognition tasks, we choose EDD[9] as the baseline model, which is only the model that supports TCR so far. It predicts tables' Hyper Text Markup Language (HTML) sequences as the results. This HTML sequence contains the table's topology structure (without cell location) and text content information and can be obtained by converting our annotation[19]. We choose Tree-Edit-Distance-based Similarity (TEDS)[9] as the metric, which compares the similarity between two tables' HTML sequences. This metric supports evaluating topology structure or content recognition performance according to whether the sequence contains the text content. Tab. 16 shows that EDD fine-tuned on TabRecSet can achieve significantly higher TEDS scores on structure (72.34%→ 90.68%) and content recognition (50.93%→ 70.70%). This result illustrates that our training set can help EDD improve the performances and thus validate the usability of *TabRecSet* for these two sub-tasks. As for direct training on our dataset, the performances of EDD are limited (51.75%, 17.04%), which reveals that the TSR and TCR tasks on *TabRecSet* are challenging.

Model	Support Tasks	Pre-traininig set	Training set	Testing set	TEDS-S (%)	TEDS-All (%)	TSR(-) Acc. (%)	P-Cell (%)	AP-Table (%)
EDD[9]	TSR(−)+TCR	—	PubTabNet	*TabRecSet*	72.34	50.93	NA	NA	NA
			TabRecSet		51.75	17.04			
		PubTabNet			90.68	70.70			
TableMaster[25]	TSR	—	PubTabNet	*TabRecSet*	55.52	NA		2.974	
			TabRecSet		16.61			0.3524	
		PubTabNet			93.13			11.00	
TGRNet[19]	TSR	—	*TabRecSet*	*TabRecSet*	NA		65.66	74.82	
CDeC-Net[27]	TD		*TabRecSet*				NA		92.80

Table 6. Evaluation results of state-of-the-art methods on *TabRecSet*. **TSR(-)**: Table topology structure recognition without detecting the cell spatial locations. **TEDS**: Tree-Edit-Distance-based Similarity[9] (TEDS) metric for the topology structure or content recognition. **TEDS-S**: The TEDS result for TSR(-). **TEDS-All**: The TEDS result for both TSR and TCR. **TSR(-) Acc.**: The classification accuracy of logical properties for TSR(-). **P-Cell**: The precision[26] (P) of cell detection. **AP-Table**: Average Precision[26] (AP) of table segmentation. **NA**: Not applicable.

For the TSR task (spatial & topological structure), we choose TableMaster[25] and TGRNet[19] as baselines. Both TableMaster and TGRNet output Bboxes as cell detection results, so we use Bbox precision[26] to evaluate the performance of spatial structure recognition. As for the topology structure, there is a difference between the output of the two models. TableMaster output the tables' HTML sequence as the topology structure prediction, while the TGRNet model formulates the TSR task as a classification problem for each graph node and outputs the Table Graph as the prediction result. In other words, TableMaster and TGRNet represent two different categories of methods to deal with this task, i.e., sequence-based and graph-based methods. The metrics for these two kinds of models are different. The sequence-based model uses TEDS, and the graph-based model uses classification accuracy of the logical properties. As illustrated in Tab. 16, TGRNet can achieve moderately high performance on TSR (74.82% & 65.66%). TableMaster with fine-tuning can achieve much higher TSR performance than without fine-tuning (55.52%→ 93.13%, 2.974%→ 11.00%). These experiment results are strong evidence of usability for the TSR sub-task. TableMaster without pre-training has a TEDS score of 16.61%, which reveals the challenge of our dataset for sequence-based methods on topology structure recognition. Note that the performances of TableMaster for spatial structure recognition are low (2.974%, 0.3524% and 11.00%) because TableMaster belongs to the regression-based method, which cannot precisely predict the cell location when the table has a large distortion.

CDeC-Net[27] is used to verify the usability of our dataset for the TD task. Table 6 illustrates that the Average Precision[26] (AP) is high enough (92.8%) to prove the usability for table detection and segmentation.

Usage Notes

The data is organized as shown in Fig. 10. We provide a Python script to load the samples from *TabRecSet* and organize them in a proper data structure. For deep learning research, it is suggested to combine and mix different types or scenarios of tables at first, according to the task needs, and divide the mixed datasets into training, validation, and testing sets for model training, validating, and testing.

Code availability

A link to the dataset, along with Python codes that are used to create the dataset, statistical analysis and plots, is released and publicly available at https://github.com/MaxKinny/TabRecSet.

Received: 11 October 2022; Accepted: 24 January 2023;
Published online: 23 February 2023

References

1. Shahab, A., Shafait, F., Kieninger, T. & Dengel, A. An open approach towards the benchmarking of table structure recognition systems. In *International Workshop on Document Analysis Systems*, 113–120, https://doi.org/10.1145/1815330.1815345 (2010).
2. Göbel, M., Hassan, T., Oro, E. & Orsi, G. ICDAR 2013 table competition. In *International Conference on Document Analysis and Recognition (ICDAR)*, 1449–1453, https://doi.org/10.1109/ICDAR31910.2013 (2013).
3. Gao, L., Yi, X., Jiang, Z., Hao, L. & Tang, Z. ICDAR 2017 competition on page object detection. In *International Conference on Document Analysis and Recognition (ICDAR)*, 1417–1422, https://doi.org/10.1109/ICDAR.2017.231 (2017).
4. Siegel, N., Lourie, N., Power, R. & Ammar, W. Extracting scientific figures with distantly supervised neural networks. In *Joint Conference on Digital Libraries (JCDL)*, 223–232, https://doi.org/10.1145/3197026.3197040 (2018).
5. Zhong, X., Tang, J. & Jimeno-Yepes, A. PubLayNet: largest dataset ever for document layout analysis. In *International Conference on Document Analysis and Recognition (ICDAR)*, 1015–1022, https://doi.org/10.1109/ICDAR.2019.00166 (2019).
6. Chi, Z. *et al*. Complicated table structure recognition. Preprint at https://doi.org/10.48550/arXiv.1908.04729 (2019).
7. Deng, Y., Rosenberg, D. S. & Mann, G. Challenges in end-to-end neural scientific table recognition. In *International Conference on Document Analysis and Recognition (ICDAR)*, 894–901, https://doi.org/10.1109/ICDAR.2019.00148 (2019).
8. Li, M. *et al*. TableBank: table benchmark for image-based table detection and recognition. In *Language Resources and Evaluation Conference (LREC)*, 1918–1925 (2020).
9. Zhong, X., ShafieiBavani, E. & Jimeno-Yepes, A. Image-based table recognition: data, model, and evaluation. In *Eur. Conf. Comput. Vision (ECCV)*, 564–580, https://doi.org/10.1007/978-3-030-58589-1_34 (2020).
10. Desai, H., Kayal, P. & Singh, M. TabLeX: a benchmark dataset for structure and content information extraction from scientific tables. In *International Conference on Document Analysis and Recognition (ICDAR)*, 554–569, https://doi.org/10.1007/978-3-030-86331-9_36 (2021).

11. Smock, B., Pesala, R. & Abraham, R. PubTables-1M: towards comprehensive table extraction from unstructured documents. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4634–4642, https://doi.org/10.1109/CVPR52688.2022.00459 (2022).
12. Gao, L. *et al*. ICDAR 2019 competition on table detection and recognition (cTDaR). In *International Conference on Document Analysis and Recognition (ICDAR)*, 1510–1515, https://doi.org/10.1109/ICDAR.2019.00243 (2019).
13. Long, R. *et al*. Parsing table structures in the wild. In *IEEE Int. Conf. Comput. Vision (ICCV)*, 924–932, https://doi.org/10.1109/ICCV48922.2021.00098 (2021).
14. Journal of Refrigeration Editorial Board. Tables requirements. *Website of Journal of Refrigeration* http://www.zhilengxuebao.com/zlxben/ch/common_item.aspx?parent_id=20180124025943634 menu_id=20180124035950729 (2022).
15. Gao, L. *et al*. A survey on table recognition technology. *Journal of Image and Graphics* **27**, 1898–1917, https://doi.org/10.11834/jig.220152 (2022).
16. Bug, S. ImageAssistant. *Website of Pullywood* https://www.pullywood.com/ImageAssistant (2022).
17. MindGems Team. Duplicate image finder. *Website of MindGems* https://www.mindgems.com/products/VS-Duplicate-Image-Finder/VSDIF-About.htm (2022).
18. Russell, B. C., Torralba, A., Murphy, K. P. & Freeman, W. T. LabelMe: a database and web-based tool for image annotation. *International Journal of Computer Vision* **77**, 157–173, https://doi.org/10.1007/s11263-007-0090-8 (2008).
19. Xue, W., Yu, B., Wang, W., Tao, D. & Li, Q. TGRNet: a table graph reconstruction network for table structure recognition. In *IEEE Int. Conf. Comput. Vision (ICCV)*, https://doi.org/10.1109/ICCV48922.2021.00133 (2021).
20. Keller, W. & Borkowski, A. Thin plate spline interpolation. *J Geod* **93**, 1251–1269, https://doi.org/10.1007/s00190-019-01240-2 (2019).
21. Weisstein, E. W. Affine transformation. *MathWorld-A Wolfram Web Resource* https://mathworld.wolfram.com/AffineTransformation.html (2022).
22. Minaee, S. *et al*. Image segmentation using deep learning: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **44**, https://doi.org/10.1109/TPAMI.2021.3059968 (2020).
23. Yang, F. & Hu, L. TabRecSet: a large scale dataset for end-to-end table recognition in the wild. *Figshare* https://doi.org/10.6084/m9.figshare.20647788 (2022).
24. Lou, J. & Wang, G. *List of Commonly Used Modern Chinese Characters* (Beijing Education Press, 1987).
25. Ye, J. *et al*. Pingan-vcgroup's solution for ICDAR 2021 competition on scientific literature parsing task B: table recognition to HTML. Preprint at https://arxiv.org/abs/2105.01848 (2021).
26. Liu, L. *et al*. Deep learning for generic object detection: a survey. *Int. J. Comput. Vis.* **128**, 261–318, https://doi.org/10.1007/s11263-019-01247-4 (2020).
27. Agarwal, M., Mondal, A. & Jawahar, C. V. CDeC-Net: composite deformable cascade network for table detection in document images. In *International Conference on Pattern Recognition (ICPR)*, 9491–9498, https://doi.org/10.1109/ICPR48806.2021.9411922 (2020).

Acknowledgements

The research is partially supported by National Nature Science Foundation of China (No. 62176093, 61673182), Key Realm R & D Program of Guangzhou (No. 202206030001), Guangdong Basic and Applied Basic Research Foundation (No. 2021A151501h2282).

Author contributions

Fan Yang originated the concept of this study, designed the study, wrote the codes, annotated the data, and wrote the manuscript. Lei Hu helped with the study designing, data annotating and coding. Xinwu Liu reviewed and revised the manuscript. Shuangping Huang reviewed and revised the manuscript and supervised the study. Zhenghui Gu reviewed and revised the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to S.H.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.