A Rule-Based Method for Table Detection in Website Images

JIHU KIM AND HYOSEOK HWANG, (Member, IEEE)
Department of Software, Gachon University, Seongnam 13120, South Korea
Corresponding author: Hyoseok Hwang (hshwang@gachon.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIT) under Grant 2019R1F1A1057984, and in part by the Gachon University Research Fund of 2019 under Grant GCU-2019-0779.

ABSTRACT Table detection is an essential part of a document analysis because tables are among the most efficient methods for systematically summarizing information. Therefore, numerous studies on detecting tables not only from documents but also from websites have been conducted. Although, the number of websites has been growing explosively recently, most of these studies suffer from detecting tables which are image types rather than tagging due to the variability of size, contents, color, and shapes. In this paper, we propose an efficient yet robust method for detecting tables in image formats, which can apply to both documents and websites. Instead of employing recently developed deep learning methods, which require extensive training for diversity, we apply a rule-based detection method by using key features of many tables, namely, the grid format of the text provided in the tables. The proposed method consists of two stages: a feature extraction stage and a grid pattern recognition stage. In the first stage, we extract the features of the contents in the tables. We then remove the features of non-text objects and texts not included in tables. In the second stage, we build tree structures from the features and apply a novel algorithm for determining the grid pattern. When we applied our method to a website dataset, the experimental results showed a precision, recall, and F1-measure of 84.5%, 72%, and 0.778, which are improvements of 3.6%, 24.16%, and 0.276 over a previous method, respectively, while also achieving the fastest processing time. In addition, the proposed rule-based method allows the structure of the contents in the table to be easily restored.

INDEX TERMS Table detection, Web information extraction, document analysis.

I. INTRODUCTION

Tables are widely used to represent relevant and structured information to human readers because they make such information easy to search, compare, and understand [1]. Therefore, table detection from documents and website is considered an essential part of a document analysis through the extraction of summarized and well-organized information. However, table detection is a challenging task due to the high degree of intra-class variability and inter-class similarity between tables, as shown in 1(b) [2]. Intra-class variability indicates the differences in layout among tables, namely, an irregular use of the ruling lines or diversity in their contents [3]. By contrast, inter-class similarity indicates the similarity of a table with other objects presented in the same document, e.g., figures, flow charts, or code listings [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Jingchang Huang.

During the past few decades, numerous studies have been conducted on table detection from documents. Such detection focuses on detecting the location of the table region. Most previous methods on table detection from documents have been based on rule-based approaches, focusing on text blocks, lines, and gaps between objects [5]–[7]. Because rule-based methods are specialized to each document source, statistical machine learning methods have been proposed for generalization [8]–[10]. With the recent and rapid development of deep learning techniques, approaches employing convolutional neural networks (CNNs) have been proposed and have shown a superior detection accuracy [1], [2], [11].

For websites, however, table detection methods can be divided into two categories. First, several studies have focused on the classification of genuine and non-genuine tables. Using markup language, it can be easy to detect tables on websites by analyzing their HTML tags (e.g.,\textless table\textgreater,\textless div\textgreater). However, contents other than actual tables may use
these tags for the website layout or arrangement. When table tags are applied to content on a website that is not part of an actual table, the layout is called a non-genuine table, and it is essential to detect such cases. Heuristic rule-based methods for filtering out non-genuine tables have been proposed [12], [13]. The other category of table detection aims to solve the same problem found in physical documents [14]. However, there is a difficulty in applying table detection methods designed for documents directly to a website. Compared to document images, tables with website images incur severe problems in terms of detection owing to the larger intra-class diversity (see Figure 1). For this reason, approaches based on deep learning suffer when applied to table detection on websites because a large amount of labeled training data is required in proportion to the diversity of the tables. In addition, images from websites tend to be larger in size than document images, which might cause longer computational times when applying a convolution on the entire image. This makes deep-learning based approaches unsuitable for crawling data that require processing large amounts of website images within a short time.

To solve these problems, we propose a novel rule-based method to detect tables of not only website images but also document images, which are particularly robust in a diversity of tables. Our method is motivated by the observation that most of the table content follows a grid layout. The proposed method consists of two stages: a feature extraction stage and a grid pattern recognition stage. In the first stage, we extract the features from the contents of the tables using an image processing method applying constraint rules for valid features. In the second stage, we identify features arranged a grid pattern using a tree structure and grid pattern rules. By virtue of a rule-based approach, the proposed method not only detects tables from images, but also easily reconstructs the contents of the table as a structural form of the documents, such as json, xml, or CSV formats.

The rest of the paper is described as follows. Previous related studies are introduced in Section II. Section III describes the process of the proposed method in detail. Section IV details the performance measures used to evaluate the different methods as well as the results of the comparative experiments. Finally, some concluding remarks are given in Section V.

II. RELATED STUDIES
A. TABLE DETECTION IN DOCUMENTS

The early table detection approaches were rule-based, focusing on text block, lines, and gaps between objects. Chandran and Kasturi [5], [15] presented a method for locating tables by extracting all horizontal and vertical lines of a document image. The dimensions of the table are estimated based on these lines. For missing demarcation lines, they propose using white stream recognition in both the vertical and horizontal directions. Intonori [6] proposed a rule-based approach using a text block arrangement and ruled line position of a document image. The cell region is obtained with the bounding box of the text blocks and rectangle enclosed by ruled lines. These cell regions are expanded to generate virtually ruled lines and align the cell boundaries. Shafait and Smith [16] proposed an approach using a layout analysis of Tesseract [17] with tab-stops used as an indication of where a text block starts and ends. The tab-stop lines form the layout of a document, and layout analysis modules locate the tables using these layouts. The detection accuracy of their algorithm for the UNLV dataset achieved a precision of 86% and a recall of 79%. Mandal et al. [7] focused on distinct columns in which the gaps between fields are quite larger than the gaps between words in the text lines. Word blobs are formatted when considering gaps between words in the text lines. Text lines that contain multiple blobs are selected as candidate lines, which are rows of the table. The authors use the University of Washington’s document image database and their collection of documents for a performance evaluation, which demonstrated a 97.21% table row detection.

Limitations to rule-based methods still exist, and rule-based approaches tend to show their weaknesses in becoming specialized to each document. For generalization, statistical machine learning algorithms have been proposed. Cesarini et al. [8] proposed a supervised learning method for optimal threshold selection that uses a table location algorithm. The method estimates the presence of a table by searching for parallel lines in the MXY-tree of the page. Experimental results show that their algorithm achieves a table location index (TLI) score of 0.709 on TPAMI and a TLI of 0.481 on the UWASH database, whereas the other
optical character recognition (OCR) engines achieve a TLI of 0.508 and TLL, respectively. Silva [18] proposed a table finding approach for text based on Hidden Markov Models used for sequence labeling. Kasar et al. [9] presented a table region detection method using the intersection of lines and feature engineering with a support vector machine (SVM). They achieved an average precision of 83.9% and a recall of 84.1% on the MAURDOR campaign dataset. Fan and Kim [10] adopt naïve Bayes, logistic regression, and SVM classifiers to detect a table region. The authors validated the algorithm by applying it to the ACL Anthology and ICDAR 2013 Table competition datasets [19]. For the ACL Anthology dataset, they achieved a precision of 0.7385, recall of 0.8780, and F1-measure of 0.8022, and for the ICDAR 2013 dataset, they achieved a precision of 0.7407, recall of 0.5217, and F1-measure of 0.6122.

In recent years, by increasing the performance of deep learning in computer vision, several studies have applied deep learning methods to a document image analysis. Hao et al. [1] first used convolution neural networks (CNNs) for table detection in PDF documents. The authors used a loose rule for proposing a table-like area, trained a convolution neural network for judgment of a table-like area, and applied metadata of a PDF document to improve the performance. They tested their model on the ICDAR 2013 dataset and achieved a precision of 0.9724, recall of 0.9215, and F1-measure of 0.9463. Gilani et al. [11] proposed a deep learning based method to employ a Faster R-CNN [20] without the use of extensive pre- or post-processing. The major advantage of their study is an invariance to changes in the table structure and layout. They evaluated their model on the UNLV dataset and recorded a precision of 82.3, recall of 90.67, and F1-measure of 86.29. Schreiber et al. [2] proposed a table detection system applying a completely data-driven method that does not require metadata of the document or heuristic rules for detection. They evaluated the algorithm on the ICDAR2013 dataset and achieved scores of 0.9740, 0.9615, and 0.9677 in precision, recall, and F1-measure, respectively. Minghao Li et al. [21] built an image-based table detection and recognition dataset with weak supervision form Word and Latex documents. This dataset contains 417K labeled tables using deep neural networks for generalization on real-world applications. In addition, the authors built a table detection method using a Faster R-CNN model with ResNeXt [22] as the backbone network architecture. Their model achieved an F1-measure of 0.9625 on the ICDAR 2013 dataset and 0.8918 on the DeepFigures dataset [23].

B. TABLE DETECTION IN WEBSITES
Extensive studies on table detection from website images have focused on distinguishing between genuine and non-genuine tables. Chen et al. [12] introduced a table mining method targeting Chinese Yahoo websites. Table filtering, recognition, and interpretation modules are aspects of this method for identifying genuine tables. The table filtering module is based on heuristic rules. The table recognition module creates a cell similarity using string similarity, called entity similarity, and number category similarity. The table interpretation module focuses on the column-wise or row-wise attribute-value relationship. Wang and Hu [14], [24] proposed a machine learning approach to detect and separate a genuine table. In their approach they select various features reflecting the layouts and characteristics of the content. In addition, they use decision trees and support vector machines as classifiers. The experimental results showed that they achieved an F1-measure of 95.89% for a non-genuine or genuine table classification for a dataset with 1.4K HTML pages collected from approximately 200 websites using the Google search engine. Kim and Lee [13] presented a table detection method consisting of two phases, preprocessing and relation extraction. During the preprocessing step, a part of a genuine or non-genuine table is detected based on predefined rules. Relation extraction deals with the <TABLE> tags that are undetected during the preprocessing step. For the evaluation, they constructed databases using 1,393 HTML files for training and evaluation, achieving a validation accuracy of 95.88% in terms of the F1-measure for genuine table detection.

Studies have also been conducted on accurately dividing the types of tables found on websites for better detection and applying the results to real-world applications. Crestan and Pantel [25] expanded the table types to include a more fine-grained class that considers the layout and structure of the tables. The authors divide the tables into 10 types and consider 20 different features. A neural network is then used to classify the tables from websites. Eberius et al. [26] proposed dividing a genuine table into three main layout types. The authors divided the features into global and local levels to consider the structure and content. The classification process is parted detection of genuine tables and identification of the layout type of these tables with two different approaches, a single-layer, and a double-layer approach. They extracted a training and test dataset from the WARC files of the July 2014 version of Common Crawl, and achieved a precision of 95.29, recall of 96.16, and F1-measure of 95.72 for the classification of genuine tables, respectively.

Krüpl et al. [27], [28] focused on tables that can be visually rendered similarly by a Web browser but using a different source code such as div tags or CSS instead of a table. They presented methods for locating tables based on the rendered images of words on Websites rather than by analyzing the HTML elements or DOM tree. In addition, they proposed a bottom-up approach to distinguish genuine tables from a visual rendition of a Website. To the best of our knowledge, there have been few studies on table detection and analysis that focus on website images.

III. PROPOSED METHOD
In this chapter, we describe the proposed method for table detection on various images taken from different websites. Our method does not use metadata of the markup language but only visual cues of images from websites. The detection
process consists of two major stages: feature extraction and grid pattern recognition, as shown in Figure 2. During the first stage, we extract features that are capable of representing the contents of the table. To do so, we extract the position and size of each contour from the text and lines in the image. Before proceeding to the second stage, several false-positive features that are detected outside of the tables or are parts of the textured objects are removed. During the second stage, we build tree structures for identifying the grid pattern from the feature set. By using tree structures of features and their gaps, we determine the candidate feature sets composed of features in a grid pattern, as described in detail below.

A. FEATURE EXTRACTION

In the feature extraction stage, we extract the positions and areas of the bounding boxes of those features that can represent the contents of the tables. This stage consists of two steps: a contour extraction step and a filtering step. Here, we set those features that have the position and area from text words because almost all contents that constitute the table are text. Therefore, we first extract those contours from textured regions, then remove candidates that are not from text but are images or outside parts of the tables.

To extract the contours from an image, we exploit a morphological operation. Images of websites are mostly colored and often contain text, pictures, and complex backgrounds. Because our method uses textures rather than colors, we convert color images into gray-scale images. The edges of the grey-scale images are extracted using a morphological gradient, which indicates the difference between the dilated and eroded portions of the image using a structuring element [29]. A gradient image $G$ from a grey-scale image $I$, can be obtained as follows:

$$G = D(I, K_1) - E(I, K_1), \tag{1}$$

where $K_1$ is the circular shaped structuring element of 3 pixels radius, $D$ is a dilation operation, and $E$ is an erosion operation, which is defined as follows:

$$D(I, K) = \{ z \mid (\hat{K}z) \cap I \subseteq I \}, \tag{2}$$

$$E(I, K) = \{ z \mid (K)_z \subseteq I \}. \tag{3}$$

where $\hat{K} = \{ w \mid w = -k, \text{ for } k \in K \}$ indicates a reflection of $K$. The function $(K) = \{ c \mid c = k + z, \text{ for } k \in K \}$, \tag{4}$$

where the structuring element $K$ is translated by point $z = (z_1, z_2)$, which is an element of set $\mathbb{Z}^2$. The edge from the figure has a smaller value than the edge from the text in the gradient image $G$ because the edges of the text have a high frequency [30]. Based on this rationale, we convert the gradient image $G$ into a binary image $B$ with a threshold for removing edges from non-text objects. To make a single feature that represents a cell of tables, we cluster the object boundaries from the edges distributed within a close range. The closing operation tends to fuse narrow breaks, and long thin gulfs eliminate small holes and fill in the gaps. The closing image $C$ from binary image $B$ is obtained as follows:

$$C = E(D(B, K_2), K_2), \tag{5}$$

where $K_2$ is a structuring element for a closing operation. At this time, we use a horizontally extended rectangular shape for $K_2$ because a text is generally written in the horizontal direction. Using a closing operation, the edges from the text are merged so that they are in a single contour. Often, the contours in the box of the table cell and those in the content overlap each other. Because we focus on the contents of the table, we select contours that do not contain any other contours. For selecting a contour, we use the topological structural analysis algorithm proposed by Suzuki to determine a topological structure having an inclusion relation [31]. We define a feature with the position and dimension of selected contours. Hence, the feature $f$ has four values, namely, the $x$ and $y$ coordinates of the center position and the width and height of the bounding box of the contour, which in our study are represented as $f_x, f_y, f_w,$ and $f_h$, respectively. The center positions and their bounding boxes are represented in Figure 3(a).

The selected contour still includes false-positive contours because noise and high-contrast images also have high-frequency components. During the second step, we remove these features from a non-text object by applying rules for the valid features below:

- A valid feature from the text is not duplicated with other features.

FIGURE 2. Overview of our method. The method consists of two stages: feature extraction and grid pattern recognition.
processing. The operation that extracts \( \bar{\alpha} \) considering the positional error that occurs during the image feature along the axis direction as follows:

\[
\text{FIGURE 3. Example of the filtering step for removing false-positive features. The green rectangles of figures in the top row represent the results of each rule. Black points of figures satisfy each condition. (a) Removing duplicated features (represented as yellow points). (b) Removing features outside of grid (represented as green points). (c) Removing vacant features (represented as blue points).}
\]

- A valid feature has at least one or more features in the same level of row or column.
- The inside of a valid feature area is not empty.

Before describing these conditions, we define an operation that determines the distance from a feature to the nearest feature along the axis direction as follows:

\[
D_\alpha(a, A) = \min_{a, b \in A, a \neq b} ||a_\alpha - b_\alpha||_2, \quad \alpha \in (x, y),
\]

where \( \alpha \) represents basis \( x \) or \( y \), and \( a_\alpha \) indicates the \( \alpha \)-coordinate value of the feature \( a \).

We first remove features extracted from the textured image region. Unlike text areas, the features from textured images tend to be duplicated within a narrow area. Thus, we extract \( \hat{F} \) from the feature set \( F \) by the following:

\[
\hat{F} = \{ f \mid P(f, F) = 1, f \in F \},
\]

where

\[
P(a, A) = \begin{cases} 
1, & D_x(a, A) > \theta \quad \text{and} \quad D_y(a, A) > \theta \\
0, & \text{otherwise}
\end{cases}
\]

Here, the function \( P \) extracts features of which the minimum distance to other features is greater than \( \theta \). The results of an exclusion are shown in Figure 3(a).

The second condition is for removing features from texts not included in the table. Features from the text in the table must have different features in the same level of rows or columns. This is because all tables must consist of two or more horizontal or vertical cells. We define the same level of rows and columns as having an error of \( \kappa \) pixels or less, considering the positional error that occurs during the image processing. The operation that extracts \( \hat{F} \) from \( F \) can be formalized as follows:

\[
\hat{F} = \{ f \mid Q(f, \hat{F}) = 1, f \in \hat{F} \},
\]

where

\[
Q(a, A) = \begin{cases} 
1, & D_x(a, A) < \kappa \quad \text{or} \quad D_y(a, A) < \kappa \\
0, & \text{otherwise}
\end{cases}
\]

When a feature has another feature in the same row, the minimum distance between these two features is in the row is less than \( \kappa \). This satisfies \( D_x(a, A) < \kappa \). The same is true for two features in the same column, which is established by satisfying \( D_y(a, A) < \kappa \). Therefore, only features that have another feature in the same level row or column satisfy this condition. Note that both features in the same level of rows and columns are duplicated features, and have already been removed through the previous condition. Figure 3(b) shows an example of this operation.

The final condition is to remove features from the grid array of the extracted images that form a grid pattern, picture, or graph. In a binary image, it is possible to determine whether the area of a feature point is filled or empty by adding the pixel values of the area around the feature. The final feature set \( \hat{F} \), which excludes features from a vacant region, is defined as follows:

\[
\hat{F} = \{ f \mid R(f, B) > \lambda, \ f \in \hat{F} \},
\]

where

\[
R(f, I) = \sum_{i=-\mu}^{\mu} \sum_{j=-\mu}^{\mu} I(f_x + i, f_y + j),
\]

in which \( B \) is a binary image, \( \lambda \) is a threshold for determining a text region, \( \mu \) is the range of the summation window, and \( f_{\in(x,y)} \) is a value with \( x, y \) coordinates as its basis. The result of this operation is shown in Figure 3(c). With the position of the filtered feature \( \hat{F} \), the grid pattern is recognized in the next stage.

B. GRID PATTERN RECOGNITION

As mentioned previously, contents or cells of a table are arranged side by side or up and down. Therefore, a cell has neighbor cells located horizontally and vertically in rows and columns. Each of the cells has its content at the center which belongs to the feature set \( \hat{F} \). Therefore, we can simplify the table detection problem as finding a set of features located in a grid from the extracted feature set \( \hat{F} \). To solve this problem, we propose an algorithm using a tree data structure that has features and their gaps as nodes. We first construct two trees, called feature trees, that have nodes in the same rows or columns as the child node. With the gaps between a parent and its child node, two additional trees are created, which we call gap trees. We then extract a candidate set from the given trees, which satisfy the conditions for establishing the content of the table. Finally, we extract the table regions using the connectivity of the two candidate sets. Now, the second stage is applied in three steps: tree construction, candidates set extraction, and table region extraction.

In the first step, we construct feature trees \( T_{row} \) and \( T_{col} \) using the positions of the features. The children of the root node of \( T_{row} \) are the leftmost features from different rows. We denote the subtree in which a child node of the root node of \( T_{row} \) becomes the root node as \( T_{row}' \). Here, index \( i \) is assigned by sorting in a row value. Each subtree is a unary
tree composed of features located in the same row or column of its root node. Before being assigned as nodes, features in the subtree are sorted in the column direction. In this way, all features become a node $T_{row}^i$, which indicates the depth $j$ node of the $i$-th subtree. The other feature tree $T_{col}$ is created in the same way. In this case, however, the topmost features from different columns become children nodes of the root node. Therefore, features become the node in both $T_{row}$ and $T_{col}$. An example of a feature tree $T_{row}$ from feature set is shown in Figure 4(a).

With $T_{row}$ and $T_{col}$, we also build gap trees $G_{row}$ and $G_{col}$. The nodes of the gap trees represent distances between nodes of the feature trees. The gap tree $G_{row}$ have horizontal gaps between nodes of $T_{row}$ as their nodes and each node is obtained by

$$G_{row}^{i,j} = \left| \left( T_{row}^{i,j} \right)_x - \left( T_{row}^{i,j+1} \right)_x \right|, \quad (13)$$

and each node of a gap tree $G_{col}$ is assigned using $T_{row}$ as follows:

$$G_{col}^{i,j} = \left| \left( T_{col}^{i,j} \right)_y - \left( T_{col}^{i,j+1} \right)_y \right|. \quad (14)$$

Here, it is worth mentioning that a feature that can be identified from the index of the gap trees. In other words, nodes of the feature tree $T_{col}^{i,j}$ and $T_{col}^{i,j+1}$ are associated with the node of the gap tree $G_{col}^{i,j}$. Figure 4(b) shows an example of constructing a gap tree from a feature tree.

In the candidate extraction step, we create candidate sets in the row and column directions using feature trees and gap trees. Elements of the candidate sets are selected nodes of the feature trees, which follow the conditions to be a table of contents. We empirically determine the characteristics of the grid patterns in a table of website images: In the same direction, the spacing of content is constant or the intervals repeat in the orthogonal direction. Motivated by this, we establish the conditions for including a node of the feature tree to a candidate set as follows:

- In a subtree of a gap tree, if consecutive nodes have similar values, we include nodes in the feature tree associated with these nodes to a candidate set;
- In a gap tree, if a node has a value similar to that of a node of another subtree, we include nodes in the feature tree associated with the node to a candidate set.

The first condition is to identify features with a similar spacing in succession. By comparing successive gaps of features, the gap tree is utilized. In a subtree of a gap tree, the values of two consecutive nodes indicate the spacing of three nodes in the feature trees, e.g., we can identify three consecutive features located at the same interval when the values of two consecutive nodes are equal in the gap tree. Here, because it is difficult to match the gaps accurately, we set a margin proportional to the sum of two gaps and consider the gaps to be similar when the difference is less than the margin. All nodes of the candidate set $C_{row}$ are extracted from the feature tree $T_{row}$ using the gap tree $G_{row}$, and those of $C_{col}$ are extracted from the feature tree $T_{col}$ using the gap tree $G_{col}$. The algorithm of the created $C_{row}$ is summarized in Algorithm 1.

Other nodes from feature trees can be also included in $C_{row}$ and $C_{col}$ when they satisfy the second condition. Some tables have a different spacing in the row or column direction, particularly when various types of attributes are represented in the table. The second condition is solving this problem. When the intervals between features in the same row or column are irregular and repeated in the orthogonal direction, we consider those features to be in the grid pattern.
Algorithm 3 Method for Extracting Table Region

**Result:** Set of table feature groups, \( \text{Tables} \)

**Data:** Candidate sets, \( C_{\text{row}}, C_{\text{col}} \)

\[
\text{JobStack}.\text{clear}()
\]

\[
C \leftarrow C_{\text{row}} \cup C_{\text{col}}
\]

\[
i \leftarrow 1
\]

**while** \( C \neq \emptyset \) **do**

\[
T \leftarrow \emptyset
\]

\[
c \leftarrow \text{GetAnElement}(C)
\]

\[
\text{JobStack}.\text{push}(c)
\]

**while** \( \text{JobStack} \neq \emptyset \) **do**

\[
p \leftarrow \text{JobStack}.\text{pop}()
\]

\[
R \leftarrow \{\text{ParentOf}(p)\} \cup \{\text{ChildOf}(p)\}
\]

\[
\text{if} \ 2q \ s.t. \ (p)_{x,y} = (q)_{x',y'} \land q \in C \text{ then}
\]

\[
R \leftarrow R \cup \{q\}
\]

\[
R \leftarrow R - T
\]

\[
T \leftarrow T \cup R
\]

**while** \( R \neq \emptyset \) **do**

\[
r \leftarrow \text{GetAnElement}(R)
\]

\[
\text{JobStack}.\text{push}(r)
\]

\[
C \leftarrow C - T
\]

\[
d_x \leftarrow \text{MaxXDistanceOfElementIn}(t)
\]

\[
d_y \leftarrow \text{MaxYDistanceOfElementIn}(t)
\]

**if** \( d_x > \theta \land d_y > \theta \) **then**

\[
B_i \leftarrow \{T\}
\]

\[
i \leftarrow i + 1
\]

\[
\]

this condition, a node of one subtree of a gap tree should have a node with the same value in another subtree. In addition, to confirm that these two nodes repeat in an orthogonal direction, the positions of the feature nodes associated with the two gap nodes are compared. A pseudo-code for including additional nodes to \( C_{\text{row}} \) can be found in Algorithm 2.

The candidate set contains features from the table shown in the images, and also includes features that satisfy such conditions but are extracted from non-table regions. For instance, consecutive features with regular gaps in only one row or column are included in the candidate set because they satisfy the first condition. In addition, even if a gap has the same value as a gap far away in the orthogonal direction, the second condition is satisfied and thus becomes an element of the set. Such cases violate the definition of the grid pattern, which should have multiple rows and columns and must be connected to each other. Therefore, in the final step, we extract table regions from images using candidate sets \( C_{\text{row}} \) and \( C_{\text{col}} \).

To achieve this goal, we present an iterative method to expand the table region from a feature. We first randomly choose a node from \( C \), which is a union of \( C_{\text{row}} \) and \( C_{\text{col}} \). If a child or parent of the node also exists in \( C \), it is added to the table set \( B_i \), where \( i \) is the index of the table. In the union set \( C \), the same feature can be included as a different node, e.g., \( T_{a,b} \) and \( T_{b,a} \), because during the first step, a feature became nodes of both feature trees. Although the two nodes indicate the same feature, they have different connected nodes. Because we need to extend the region in horizontal and vertical directions, we also include the node from the same feature to \( B_i \).

We can find other nodes connected with the added nodes until no more additional nodes exist. If a table set is expanded into a grid shape, we can confirm it as a table set and add index \( i \).

We remove all used elements of \( B_i \) from \( C \). The same process is then conducted from another node because there can be two or more tables in an image. We describe a method for extracting a table region in Algorithm 3.

For each table set \( B \), we finally determine the bounding box by defining the left-top point \( B_{LT} \) and the right-bottom point \( B_{RB} \). The values of \( B_{LT} \) and \( B_{RB} \) are given by the following:

\[
B_{LT} = \left( \min_{f \in B} \left( f_x - \frac{f_w}{2}, \min_{f \in B} \left( f_y - \frac{f_h}{2} \right) \right) \right), \tag{15}
\]

\[
B_{RB} = \left( \max_{f \in B} \left( f_x + \frac{f_w}{2}, \max_{f \in B} \left( f_y + \frac{f_h}{2} \right) \right) \right). \tag{16}
\]

When a table has margin between contents and boundary, e.g., table surrounded by border lines, we expand extracted bounding box. In the binary image, if any pixels at the bounding box has non-zero value, this means that lines are crossing the bounding box. In this case, we adjust positions of \( B_{LT} \) or \( B_{RB} \) outward until the sum of pixels at the bounding box is zero.

### IV. EXPERIMENTAL RESULTS

In order to verify the performance of the proposed method, we conducted experiments on two types of dataset from
documents and websites. We applied the proposed method on ICDAR 2013 dataset to detect tables on document images. However, there is no available website dataset labeled with the location of the table. Therefore we created a dataset named as Table on Website (TOW) by crawling extensive amount of images with tables from websites and labeling their positions. We also compared our results to the existing methods based on CNNs.

### A. DATASET

For a performance comparison of the document and website images, we used two different datasets: The ICDAR 2013 Table Competition dataset and the TOW image dataset. The ICDAR 2013 Table Competition dataset consists of 128 PDF documents including the location and structural information of 156 tables. This dataset considers the documents to have been collected as generically as possible rather than concentrating on one specific sub-class. Documents from both the European Union and the US Government are used, and 75 tables are included. Because we selected an image-based table detection method, a document page which has at least one table is converted to a 200 DPI image. However, we still use the coordinates of the table bounding box as the ground truth for the table detection.

We created the TOW dataset by crawling images from various websites such as e-commerce sites and homepages describing product specifications. The TOW dataset contains 2,340 images that include 2,850 tables in total. Because we crawled from various sources, the images of TOW include various texts: English, Chinese, Korean, and Japanese. The dataset is composed of images with tables only and contains no metadata. We labeled the ground truth bounding box of the table with two points: the top-left point and the bottom-right point. Intuitively, the ground truth bounding box is matched with the boundary of the table. However, when the boundary of the table is ambiguous, e.g., lines do not exist in the table but only text, the ground truth is determined as the minimum area that includes all contents of the table. Because the proposed rule-based method does not require learning, all data are used as a test set.

Both datasets are composed of images including tables, but there are several critical differences. As shown in Table 1, the important difference between the two datasets are the sizes of the images. The variations in the sizes of the images from websites are much larger than those of images from documents. In addition, there are numerous types of tables for website images, whereas document images have formal

### B. EXPERIMENTAL PROCEDURE

We compared our proposed method with previous table detection methods: TableTrainNet [32] and TableBank [21]. Both methods are based on a convolutional neural network (CNN) and were originally proposed to detect tables in document images. TableTrainNet was implemented using the Faster R-CNN algorithm along with the Inception-v2 network [33] by referring to Gilani et al. [11]. The authors used a document image dataset composed of the ICDAR 2017 POD competition dataset [34], UNLV dataset, and Marmot dataset for network training in which the model was pre-trained on the COCO dataset [35]. We used the weight provided by the authors and set the threshold for the inference bounding boxes to 0.8, which is the same value that the authors used. The table detection method applied by TableBank is based on the Faster R-CNN with the ResNeXt [22] backbone network. This method was trained using the TableBank dataset collected from Word and Latex documents. The authors initialized the weights of their model by using a pre-trained weight on the ImageNet dataset [36]. During the test, we used 0.9 as the threshold of the inference bounding boxes, as applied by the authors.

We conducted the tests on a PC with an Intel Xeon E3-1230 CPU, NVidia GTX 1080 GPU, and 8 GB of RAM. During the experiments, we set the range of the summation window $\mu$ to 5 (pixel), the text region threshold $\lambda$ to 3, the margin for the same row or column level $\kappa$ to 10 (pixel), and the position similarity threshold $\gamma$ to 10 (pixel).

### C. EVALUATION METRICS

In this study, because table detection aims at locating a table region, we use the table evaluation measure describe by Gilani et al. [11], which is based on a bounding box region, for a performance comparison. Both regions of the ground-truth tables and tables detected by the method are represented as their bounding boxes. Here, $G$ represents a bounding box labeled as the ground truth, and $D$ represents a bounding box detected by an algorithm. To validate the performance during the experiments, four evaluation metrics are used: the processing time, precision, recall, and F1-measure. The following describes how to measure each metric in detail.

---

### TABLE 1. Comparison of TOW and ICDAR 2013 table competition datasets.

| Dataset     | Minimum | Maximum   | Average   |
|-------------|---------|-----------|-----------|
|             | (size: width x height, unit: pixel) |           |           |
| ICDAR 2013  | 1100×1700 | 1654×2339 | 1659.9×2188 |
| TOW         | 750×772  | 1000×27189| 782.7×4773.5 |

---

### FIGURE 6. Results of ICDAR 2013 table competition dataset. Red boxes are extracted table regions from (a) the ground truth, (b) TableBank, (c) TableTrainNet, and (d) the proposed method.
The precision measure was used to evaluate the overall performance of the table detection algorithm. It represents the percentage of detected table regions belonging to the actual table regions in an image. The formula for calculating the precision is as follows:

$$\text{Precision} = \frac{|D \cap G|}{|D|}$$  \hspace{1cm} (17)

The recall measure represents the percentage of ground-truth table regions detected by the algorithm. The formula for calculating the recall is as follows:

$$\text{Recall} = \frac{|D \cap G|}{|G|}$$  \hspace{1cm} (18)

The F1 measure conveys a balance between both precision and recall when evaluating the accuracy of the methodology. The formula for calculating the F1 measure is as follows:

$$\text{F1-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$  \hspace{1cm} (19)

### TABLE 2. Comparison results with previous methods on ICDAR2013 dataset.

| Dataset   | TableBank | TableTrainNet | Ours   |
|-----------|-----------|---------------|--------|
| ICDAR 2013| precision | 0.9362        | 0.9153 | 0.9148 |
|           | recall    | 0.6376        | 0.9027 | 0.5737 |
|           | F1-measure| 0.7586        | 0.9089 | 0.7051 |
|           | time (s)  | 0.7659        | 3.3495 | 0.5654 |

### D. RESULTS

1) RESULT ON ICDAR 2013

A performance comparison among TableTrainNet, TableBank, and the proposed method on the ICDAR 2013 Table competition dataset is shown in Table 2. TableBank achieved the best precision at 0.9362, which is slightly higher than that of the other methods. TableTrainNet showed the best performance in terms of the recall and F1-measure, although the detection time took 4-5 times longer than that of the other methods. The performance of the proposed method is slightly lower than that of TableBank except for the
processing time, which shows the best performance among the different methods. It is worth mentioning that our rule-based approach is applicable to the table detection from a document. Figure 9 shows the results of the table detection using various methods.

2) RESULT ON TOW

The test results on the TOW image dataset are summarized in Table 3. Our proposed method clearly outperforms the other table detection methods for website images. The precision of our method is 0.0369 higher than that of TableTrainNet, with both methods measured at over 0.8; however, the precision of TableBank decreases to 0.5285. Our method achieves a recall of 0.7206, which is an observable difference of 0.2416 from that of TableBank, which achieves the second-best performance in this category. From the best precision and recall, our method also records the best F1-measure of 0.778, which is also higher than that of TableBank by 0.2755. The average processing time is comparable to that of the ICDAR 2013 test results, and our method records the shortest time at 0.2816, which is 12.4-times shorter than that of TableTrainNet. Considering that real-world applications require the handling of large volumes of Web site images, a fast processing time is required for table detection. Herein, we note that the input size of TableBank is fixed, and yields a faster computational time than the other CNN-based approach. However, it was also observed that the method using a resized input also decreases the accuracy.

Figure 7 shows the results of different methods on TOW dataset. The previous method based on CNN failed to detect tables in some cases. In the figures of the second column, the TableBank extract larger area than ground truth due to inter-class similarity, i.e., they include the box with clothes above of the table in their table area. When the border line of the table is not clear, both previous methods often failed to detect tables. The proposed method not only detect various shaped tables in images but also extract accurate bounding boxes of tables with or without border lines.

3) FAILURE CASES

Although we demonstrated that a robust and accurate detection is feasible for images from both documents and websites, limitations also exist. Typical failures of our algorithm are shown in Figure 8. When two tables are horizontally or vertically attached without an adequate space between them, our method tends to detect such tables as a single table, as shown in Fig. 8(a). Even if the characters are on both the inside and outside of the image, if they are in grid form, our method may perceive them as part of a table, as shown in Fig. 8(b).

4) DOCUMENT CONVERSION

We improved our method from simply detecting table areas to recognizing the table layout and the text it contains. The extracted positions of the features are used to recognize the layout of the table i.e., the number of rows, columns, and arrangement form of cells. Then, the position and size of the features are utilized to make a bounding box around them, as shown in Figure 9(b). With the feature bounding box, we identify the content of the text in the cells by applying the OCR engine. Using the information of the table layout and the content of the text, we convert an input image into a table.
V. CONCLUSION

Although detection methods based on deep learning have shown great success in table detection from document images, they have not been applicable to website images owing to the large amounts of variation in the different types of tables, an insufficient number of training datasets, and extremely large sized datasets requiring a long processing time. Motivated by these problems, in this study, we proposed a rule-based method for table detection from website images. We first extract the features from the images by employing an image processing method, and then identify grid patterns using only these features. Experimental results show that, for website images, our method outperforms previous methods employing deep neural networks for feature extraction and applying a classifier. Even when our method was applied to table detection in document images, the performance is slightly lower than that of previous methods but demonstrated the best performance in terms of the processing time. In addition, by applying the advantages of rule-based methods, the proposed approach can restore a document into its original form by utilizing the structures of the features used for table detection. In our future work, we aim to investigate the challenging task for detecting tables on complicated images such as our failure cases.

REFERENCES

[1] L. Hao, L. Gao, X. Yi, and Z. Tang, “A table detection method for PDF documents based on convolutional neural networks,” in Proc. 12th IAPR Workshop Document Anal. Syst. (DAS), Apr. 2016, pp. 287–292.

[2] S. Schreiber, S. Agne, I. Wolf, A. Dengel, and S. Ahmed, “DeepDeSRT: Deep learning for detection and structure recognition of table documents,” in Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR), Nov. 2017, pp. 1162–1167.

[3] B. Coutason, A. Lemaitre, D. Doer-Mann, and K. Tombre, “Recognition of tables and forms,” in Handbook of Document Image Processing and Recognition. London, U.K.: Springer, 2014, pp. 647–677.

[4] D. W. Embley, M. Harst, D. Lopresti, and G. Nagy, “Table-processing paradigms: A research survey,” Int. J. Document Anal. Recognit., vol. 8, nos. 2–3, pp. 66–86, Jun. 2006.

[5] S. Chandran and R. Kasturi, “Structural recognition of tabulated data,” in Proc. 2nd Int. Conf. Document Anal. Recognit. (ICDAR), 1993, pp. 516–519.

[6] K. Itinori, “Table structure recognition based on textblock arrangement and ruled line position,” in Proc. 2nd Int. Conf. Document Anal. Recognit. (ICDAR), 1993, pp. 765–768.

[7] S. Mandal, S. Chowdhury, A. Das, and B. Chanda, “Simple and effective table detection system from document images,” Int. J. Document Anal. Recognit., vol. 8, pp. 172–182, Jun. 2006.

[8] F. Cesarini, S. Marinai, L. Sarti, and G. Soda, “Trainable table location in document images,” in Proc. Object Recognit. Supported Interact. Service Robot., 2002, pp. 236–240.

[9] T. Kasar, P. Barlas, S. Adam, C. Chatelain, and T. Paquet, “Learning to detect tables in scanned document images using line information,” in Proc. 12th Int. Conf. Document Anal. Recognit., Aug. 2013, pp. 1185–1189.

[10] M. Fan and D. Soon Kim, “Detecting table region in PDF documents using distant supervision,” 2015, arXiv:1506.08891. [Online]. Available: http://arxiv.org/abs/1506.08891

[11] A. Gilani, S. R. Qasim, I. Malik, and F. Shafait, “Table detection using deep learning,” in Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR), Nov. 2017, pp. 771–776.

[12] H.-H. Chen, S.-C. Tsai, and J.-H. Tsai, “Mining tables from large scale HTML texts,” in Proc. 18th Conf. Comput. Linguistics, 2000, pp. 166–172.

[13] Y.-S. Kim and K.-H. Lee, “Detecting tables in Web documents,” Eng. Appl. Intell. Syst., vol. 18, no. 6, pp. 745–757, Sep. 2005.

[14] Y. Wang and J. Hu, “A machine learning based approach for table detection on the Web,” in Proc. 11th Int. Conf. World Wide Web (WWW), 2002, pp. 242–250.

[15] B. Krápl and M. Herzog, “Visually guided bottom-up table detection and segmentation in Web documents,” in Proc. 11th Int. Conf. Document Anal. Recognit., Aug. 2009, pp. 223–232.

[16] R. Smith, “An overview of the tesseract OCR engine,” in Proc. 7th Int. Conf. Document Anal. Recognit. (ICDAR), Sep. 2007, pp. 269–273.

[17] F. Shafait and R. Smith, “Table detection in heterogeneous document images,” in Proc. 5th IAPR Int. Workshop Document Anal. Syst., 2010, pp. 65–72.

[18] R. Smith, “Learning regular hidden Markov models in document analysis: Table detection,” in Proc. 10th Int. Conf. Document Anal. Recognit., 2009, pp. 843–847.

[19] F. Cesarini, S. Marinai, L. Sarti, and G. Soda, “Trainable table location in document images,” in Proc. 12th Int. Conf. Document Anal. Recognit., Aug. 2013, pp. 1449–1453.

[20] G. Cavallin, “Sviluppo di un algoritmo di apprendimento automatico per l’identificazione dei dati contenuti in polizze assicurative,” M.S. thesis, Politecnico di Milano, Milan, Italy, 2005, pp. 610–614.

[21] M. Li, L. Cui, S. Huang, F. Wei, M. Zhou, and Z. Li, “TableBank: Table benchmark for image-based table detection and recognition,” 2019, arXiv:1903.01949. [Online]. Available: http://arxiv.org/abs/1903.01949

[22] E. Crestan and P. Pantel, “Web-scale table census and classification,” in Proc. 4th ACM Int. Conf. Web Search Data Mining (WSDM), 2011, pp. 545–554.

[23] M. Gobel, T. Hassan, E. Oro, and G. Orsi, “ICDAR 2013 table competition,” in Proc. 12th Int. Conf. Document Anal. Recognit., Aug. 2013, pp. 1449–1453.

[24] J. Eberius, K. Braunschweig, M. Hentsch, M. Thiele, A. Ahmadov, and W. Lehner, “Building the dresden web table corpus: A classification approach,” in Proc. IEEE/ACM Int. Symp. Pattern Recognit. (CVPR), Jul. 2017, pp. 1492–1500.

[25] N. Siegel, N. Lourie, R. Power, and W. Ammar, “Extracting scientific figures with distantly supervised neural networks,” in Proc. 18th ACM/IEEE Joint Conf. Digit. Libraries, May 2018, pp. 223–232.

[26] J. Hu, “Detecting tables in HTML documents,” in Proc. Int. Workshop Document Anal. Syst. Berlin, Germany: Springer-Verlag, 2002, pp. 249–260.

[27] A. C. E. Silva, “Learning rich hidden Markov models in document analysis: Table location,” in Proc. 10th Int. Conf. Document Anal. Recognit., 2009, pp. 843–847.

[28] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 1492–1500.

[29] G. Bradski and A. Kaehler, Learning OpenCV: Computer Vision With the OpenCV Library. Newton, MA: O’Reilly Media, 2008.

[30] N. Siegel, N. Lourie, R. Power, and W. Ammar, “Extracting scientific figures with distantly supervised neural networks,” in Proc. 18th Int. Conf. World Wide Web, 2005, pp. 1000–1001.

[31] B. Krápl, M. Herzog, and W. Gatterbauer, “Using visual cues for extraction of tabular data from arbitrary HTML documents,” in Proc. 14th Int. Conf. World Wide Web, 2005, pp. 1000–1001.

[32] J. Kim, H. Hwang: Rule-Based Method for Table Detection in Website Images
JIHU KIM received the B.S. degree from the Department of Software, Gachon University, Seongnam, South Korea, in 2020, where he is currently pursuing the master’s degree. His research interests include computer vision and machine learning.

HYOSEOK HWANG (Member, IEEE) received the B.S. degree in mechanical engineering from Yonsei University, Seoul, South Korea, in 2005, the M.S. degree in robotics, in 2009, and the Ph.D. degree from the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejon, South Korea, in 2017. He was a Research Scientist with the Samsung Advanced Institute of Technology (SAIT), Samsung Electronics, Suwon, South Korea, from 2009 to 2018. Since 2018, he has been an Assistant Professor with the Department of Software, Gachon University, Seongnam, South Korea. His research focused on computer vision, machine learning, which spans over object detection, 6D pose estimation and 3D reconstruction for intelligent robot, augmented reality, and health care.