Received February 28, 2020, accepted March 27, 2020, date of publication March 31, 2020, date of current version April 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2984627

Stroke Extraction for Offline Handwritten Mathematical Expression Recognition

CHUNGKWONG CHAN
School of Mathematics, Sun Yat-Sen University, Guangzhou 510275, China
e-mail: chsongg@mail2.sysu.edu.cn

ABSTRACT Offline handwritten mathematical expression recognition is often considered much harder than its online counterpart due to the absence of temporal information. In order to take advantage of the more mature methods for online recognition and save resources, an oversegmentation approach is proposed to recover strokes from textual bitmap images automatically. The proposed algorithm first breaks down the skeleton of a binarized image into junctions and segments, then segments are merged to form strokes, finally stroke order is normalized by using recursive projection and topological sort. Good offline accuracy was obtained in combination with ordinary online recognizers, which were not specially designed for extracted strokes. Given a ready-made state-of-the-art online handwritten mathematical expression recognizer, the proposed procedure correctly recognized 58.22%, 65.65%, and 65.22% of the offline formulas rendered from the datasets of the Competitions on Recognition of Online Handwritten Mathematical Expressions (CROHME) in 2014, 2016, and 2019 respectively. Furthermore, given a trainable online recognition system, retraining it with extracted strokes resulted in an offline recognizer with the same level of accuracy. On the other hand, the speed of the entire pipeline was fast enough to facilitate on-device recognition on mobile phones with limited resources. To conclude, stroke extraction provides an attractive way to build optical character recognition software.

INDEX TERMS Character recognition, feature extraction, offline handwritten mathematical expression recognition, optical character recognition software, stroke extraction.

I. INTRODUCTION

Mathematical expressions appear frequently in engineering and scientific documents. Since they contain valuable information, digitizing them would maximize their usability by enabling retrieval [1], integration to semantic web [2], and other automated tasks. Compared with ordinary text, mathematical expressions can present some concepts more concisely because of their two-dimensional structure. However, such compact representations are difficult to be recognized mechanically.

People are used to writing mathematical expressions on paper or blackboard, so it is inconvenience to input them by using another way. Traditional input devices like keyboards are designed for sequence of characters, although spatial relations between symbols can be represented by markups such as TeX or MathML, entering mathematical expressions by typing in a computer language is not user-friendly at all, as new users are asked to learn a new language, remember a lot of commands, and deal with miscellaneous errors. On the other hand, entering mathematical expressions with a graphical equation editor by choosing structural elements and symbols from toolboxes is inefficient for frequent users.

A recognition system turns handwritten mathematical expressions into machine manipulable syntax trees. Online recognition enables people to take notes or solve equations by writing on a touch-based device or dragging a mouse. On the other hand, offline recognition enables people to digitize existing manuscripts by scanning or record lecture notes on blackboards by taking photos. The main difference between two kinds of recognition is that trajectories are available to an online recognizer, whereas only bitmap images are given to an offline recognizer.

Online handwriting recognition systems often achieve a much higher accuracy. For example, the top three online systems correctly recognized 80.73%, 79.82%, and 79.15% of the formulas in the Competitions on Recognition of Online Handwritten Mathematical Expressions (CROHME)

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

2019 [3] respectively, whereas the top three offline systems only correctly recognized 77.15%, 71.23%, and 65.22% of them respectively. In fact, an online recognition problem can be converted to the corresponding offline problem by simply rendering the strokes. In the opposite direction, if the sequence of strokes can be recovered from a bitmap image, an online recognizer can also be applied to do offline recognition [4]. The objective of this paper is to explore the feasibility of offline handwriting recognition via stroke extraction without designing specialized online recognizers.

The main concern is that recovery of strokes is unlikely to be perfect, so it may become a single point of failure. Fortunately, good online recognition systems can tolerate some input errors, since diversity of writing habits already affected them. Furthermore, if the underlying online recognition system has been retrained with extracted strokes, it should be able to adapt to them [5]. In this case, accuracy of stroke extraction needs not be critical to that of offline recognition.

The main contributions of this paper include:

1) We propose a stroke extraction algorithm for handwritten mathematical expression, which enables reduction from offline recognition to online recognition.
2) We verify that the accuracy of the proposed system is comparable to that of other recent offline handwritten mathematical expression recognition systems, given a ready-made state-of-the-art online recognizer.
3) We demonstrate that the gap in accuracy between the proposed offline system and the underlying online system can be narrowed significantly by retraining.

The remainder of this paper is divided into four sections. Section II reviews methods to recognize online and offline mathematical expressions, as well as methods to recover trajectories. Section III describes the proposed stroke extraction algorithm for mathematical expression in detail. Section IV presents experimental results on standard datasets. Section V concludes the paper.

II. RELATED WORKS

A. ONLINE HANDWRITTEN MATHEMATICAL EXPRESSION RECOGNITION

Online handwritten mathematical expression recognition is a long-standing problem, a lot of works have been done since Anderson [6]. In the past decade, the problem attracted more and more attention because of the CROHME [7], which were held in 2011, 2012, 2013, 2014, 2016, and 2019.

Traditionally, the problem is further divided into symbol recognition and structural analysis [1]. For example, Álvaro *et al.* [8] applied hidden Markov model to recognize the symbols and parser for a predefined two-dimensional stochastic context free grammar to analyze the structure.

In order to disambiguate symbols with structural information and vice versa, several approaches were proposed to integrate symbol recognition and structural analysis closely. Yamamoto *et al.* [9] suggested parsing handwritten mathematical expression directly from strokes by using the Cocke-Younger-Kasami algorithm, so that symbol segmentation, character recognition, and structural analysis could be optimized simultaneously. Awal *et al.* [10] introduced another global approach that applies a segmentation hypothesis generator to deal with delayed strokes.

Recently, with the advances in computational power, end-to-end trainable recurrent neural networks became popular because of its ability to learn complex relations. For example, Zhang *et al.* [11] developed an encoder-decoder framework for online handwritten mathematical expression recognition named Track, Attend, and Parse (TAP), which employs a guided hybrid attention mechanism.

B. OFFLINE HANDWRITTEN MATHEMATICAL EXPRESSION RECOGNITION

On the contrary, dedicated work on offline handwritten mathematical expression recognition was almost blank in literature until very recently. An offline task was first added to CROHME in 2019 [3].

In the past, the closest problem addressed was the more constrained problem of printed mathematical expression recognition [12]. Again, in a typical system, symbols are first segmented and recognized, then the structure of the expression is analyzed [13]. For instance, in the state-of-the-art system developed by Suzuki *et al.* [14], symbols are extracted by connected component analysis and then recognized by a nearest neighbor classifier, after that, structural analysis is performed by finding a minimum spanning tree in a directed graph representing spatial relations between symbols.

Recently, Deng *et al.* [15] and Zhang *et al.* [16], [17] developed end-to-end trainable neural encoder-decoder models to translate image of mathematical expression into TeX code directly. They are quite similar to models for online recognition, except that convolutional neural networks are prepended to extract features. It should be noted that this method is so general that it can be applied to any image-to-markup problem, grammar of neither mathematical expression nor TeX is required by those systems explicitly because they can be learned from data.

C. STROKE EXTRACTION

Stroke extraction was studied for offline signature verification [18] and East Asian character recognition [19]. A typical stroke extractor detects candidates of sub-strokes first and then reassembles them into strokes by resolving ambiguities. Sub-strokes can be detected by breaking down the skeleton or approximating the image with geometrical primitives such as polygonal chains.

Lee and Pan [18] designed a set of heuristic rules to trace the skeleton. Boccignone *et al.* [20] tried to reconstruct strokes by joining the pair of adjoining sub-strokes having the smallest difference in direction, length, and width repeatedly. Doermann *et al.* [21] proposed a general framework to integrate various temporal clues.

Jäger [22] reconstructed strokes by minimizing total change in angle between successive segments within a stroke.

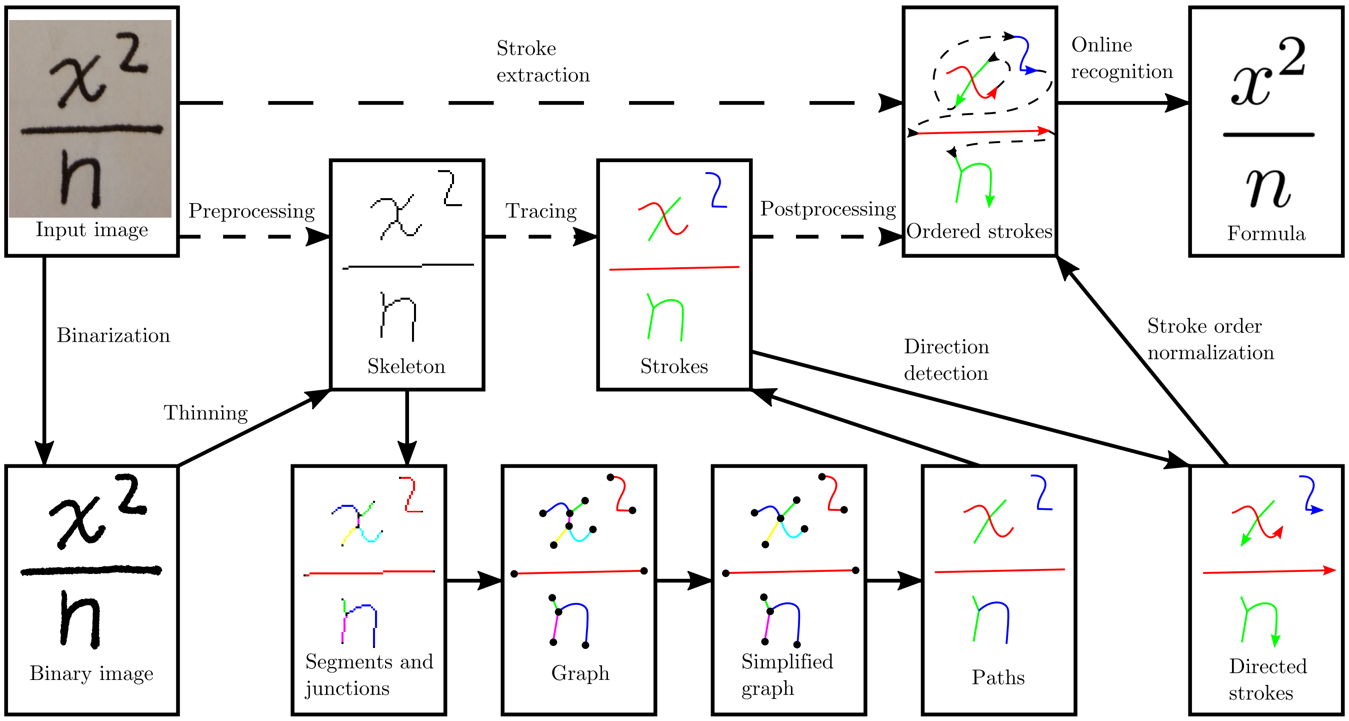


FIGURE 1. **Outline of the proposed method. Boxes represent intermediate results, solid line arrows show the recognition pipeline, and broken line arrows indicate stages.**

Lau *et al.* [23] selected another cost function taking distance between successive segments and directions of the segments into account. Unfortunately, this kind of formulations is essentially traveling salesman problem which is NP-complete, so computing the optimum efficiently may not be possible if there are more than a few sub-strokes.

In order to prevent explosion of combinations, Kato and Yasuhara [24] restricted themselves to single-stroke script subjecting to certain assumptions on junctions, so that strokes can be extracted by traversal of graph. Nagoya and Fujioka [25] extended the technique to multi-stroke script under assumptions on how strokes are intersected.

Nevertheless, the effectiveness of stroke extraction to offline recognition has remained largely untested. Evaluations of existing stroke extraction methods were often performed visually or quantitatively with their only metrics on small private datasets. Resulting accuracy of offline recognition was seldom reported and limited to single character recognition, where the challenge of symbol segmentation was not addressed. Moreover, they strongly rely on specially designed structural matching methods, which can tolerate a variety of deformations, so they may not work well with ordinary methods for online recognition.

III. OFFLINE TO ONLINE REDUCTION

A. OVERVIEW

Figure 1 outlines the proposed offline handwritten mathematical expression recognition system. In order to bridge between offline and online recognition, a stroke extraction algorithm is required to convert a bitmap image to a sequence of strokes, so that online recognition engines are applicable afterward. The proposed stroke extraction method consists of three stages: preprocessing, tracing, and postprocessing.

In the preprocessing stage, binarization and thinning are applied to increase the signal-to-noise ratio. Adaptive binarization distinguishes textual components from noisy background. Thinning reduces the amount of foreground pixels while roughly preserving the shape of strokes.

In the tracing stage, the set of strokes is reconstructed by finding a set of paths on a graph based representation of skeleton, in which vertexes and edges represent junctions and segments respectively. Ideally, each edge belongs to exactly one path and strokes are smooth, so a greedy search does the trick. In reality, some edges should be ignored because they correspond to artefacts created during thinning. On the other hand, some edges should be traversed twice because they belong to double-trace strokes, which are sometimes found in symbols such as “n”. Therefore, noise reduction is applied before the search and restoration of double-traced strokes is performed after the search.

In the postprocessing stage, stroke direction detection and stroke order normalization are applied to recover remaining temporal information. Although mathematical expressions have more complex spatial structures than ordinary text, people usually write from top left to bottom right. Therefore, heuristic rules often produce acceptable results.

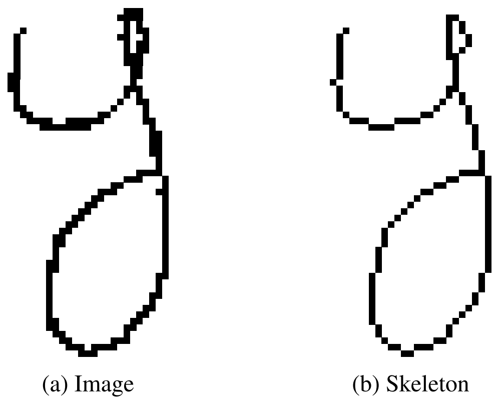


FIGURE 2. Example of thinning. The binary image is transformed into the skeleton with single pixel thickness. They are topologically equivalent to each other.

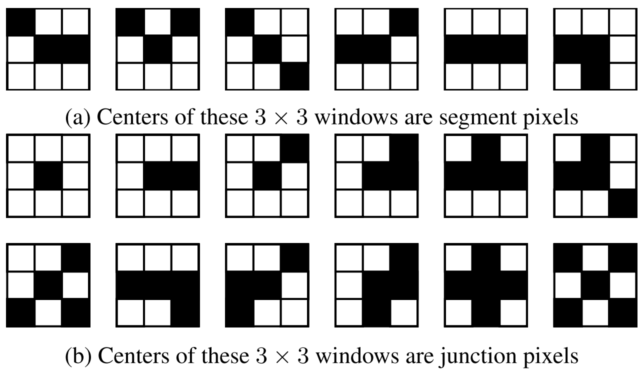

(a) Centers of these 3 × 3 windows are segment pixels
(b) Centers of these 3 × 3 windows are junction pixels

FIGURE 3. Examples of classified foreground pixels. The classification is a local operator.

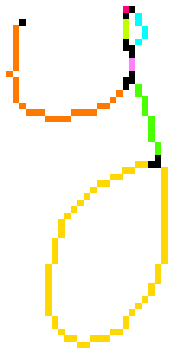


FIGURE 4. Examples of segments and junctions. Segments are filled with different colors and junctions are all in black.

B. PREPROCESSING

The input image is often degraded and contains details which are not important for stroke extraction, so binarization and thinning are applied successively to normalize it.

The input image is first converted to a grayscale image by averaging the color channels (possibly weighted), then it is converted to a binary image by applying Sauvola's method [26]. Compared with global thresholding techniques such as Otsu's method [27], such a local adaptive approach addressed commonly seen degradations including uneven illumination and random background noises. Although binarization has been studied in depth, it cannot separate textual components of the mathematical expressions from all realistic images perfectly, so additional steps may be required by the use cases. For example, mathematical expression localization can be applied to remove text next to an expression, line detection can be performed to hide grid lines, and mathematical morphology can be used to restore connectivity of strokes, but they are out of the scope of this paper.

Skeleton of the binarized image is obtained by using a thinning method by Wang and Zhang [28], which is a variant of the original method by Zhang and Suen [29] but preserves the shape of diagonal strokes better. Figure 2 compares an image with its skeleton.

For printed document recognition, skew detection and correction are often performed. However, they should not be applied to a single mathematical expression because the number of symbols may not be enough to estimate the angle reliably. To make thing worse in the present situation, symbols from a handwritten formula need not stick to a single baseline, so expressions like "x^{x^x}" may fool skew estimators based on line detection like Hough transformation.

C. DECOMPOSITION OF SKELETON

Graph based representation of skeleton is the key data structure of the proposed tracing procedure, the graph is constructed by decomposing the skeleton into segments and junctions.

A foreground pixel having exactly two other foreground pixels in its 8-neighborhood where the two pixels are not 4-neighbor of each other is called a segment pixel. Other foreground pixels are called junction pixels. Figure 3 illustrates the rule.

A connected component of the set of segment pixels is called a segment. On the other hand, a connected component of the set of junction pixels is called a junction. The set of segments and the set of junctions can be computed by using any standard algorithm for connected component analysis [30]. Figure 4 shows an example of decomposed skeleton.

For each segment S_i, its pixels can be listed in a way such that successive pixels are 8-neighbor of each other, more formally,

$$S_i = \{p_{i,1}, \ldots, p_{i,\ell_i}\}$$

where $p_{i,k}$ is in the 8-neighborhood of $p_{i,k-1}$ for $k = 2, \ldots, \ell_i$. If $p_{i,1}$ is in the 8-neighborhood of p_{i,ℓ_i}, the segment is topologically a circle and does not touch any other junction or segment unless $\ell_i = 1$; otherwise, the segment is topologically a line segment, $p_{i,1}$ touches exactly one junction and so do p_{i,ℓ_i}, other pixels in the segment never touch any other segment or junction.

For the sake of consistency, a "junction" is imposed to each looped segment to ensure that every segment has a start pixel and an end pixel, in addition, each touches a junction.

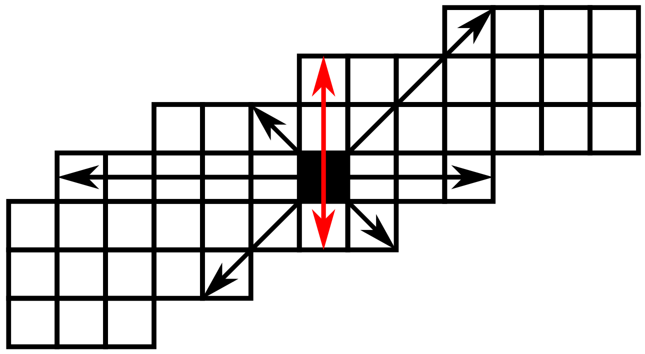


FIGURE 5. Example of stroke width estimation. Squares represent foreground pixels and arrows represent directional runs of the solid pixel. Estimated stroke width of the solid pixel is the length of the red arrow, which is the shortest one among the four.

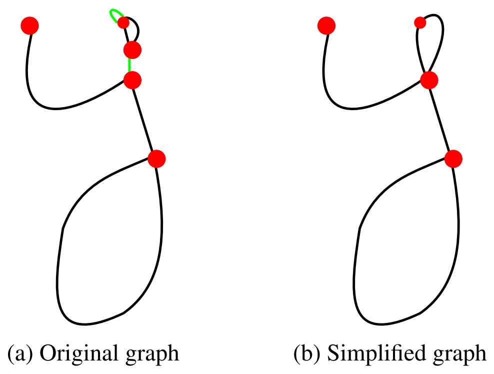


FIGURE 6. Example of noise reduction. The original graph contains two edges coming from short segments, which are draw in green. One of them is a self-loop, it is removed in the simplified graph. The other one has two different end points, they are merged in the simplified graph.

Therefore, a junction can be considered as a vertex in the sense of graph theory, whereas a segment can be considered as an edge connecting two (possibly the same) vertexes. Furthermore, a path in this undirected graph corresponds to a possible trace of ink in the input image, a connected component in this graph corresponds to a connected component of the skeleton. Figure 6a shows the graph coming from the same example as in Figure 4.

D. NOISE REDUCTION

Subtle features such as salt and pepper noises in the binarized image can affect the skeleton, salt noises result in really short segments whereas pepper noises result in isolated junctions. In addition, thinning may introduce distortions. Since they can distract stroke extractor and recognition engine, they should be discarded from the graph. Absolute threshold values are not used because they do not work in all resolutions. Observed that stroke width is uniform in a piece of handwriting, it is chosen to be a reference length.

Stroke width transformation is an image operator that assigns an estimated stroke width to each foreground pixel. It was proposed for scene text detection [31], where strokes are considered as contiguous pixels having approximately constant stroke width locally. Using a straightforward viewpoint, stroke width of a pixel can be estimated by the minimum length of the four directional runs [32] passing through it as shown in Figure 5. Under the above definition, stroke width transformation can be computed in linear time with respect to the size of binary image by caching the numbers of successive foreground pixels found in certain directions.

For each set of pixels, its width is estimated by the maximum stroke width among its pixels. Furthermore, the tip size of the pen is estimated by the average stroke width over all the segments. Now, the rules to reduce noises can be stated:

1) For each edge with a length smaller than a multiple of the average stroke width, remove it from the graph and merge its end points.
2) For each vertex with degree 0 and a width less than a multiple of the average stroke width, remove it from the graph.

Figure 6b shows the simplified graph coming from the same example as in Figure 4.

E. STROKE TRACING

Clearly, an isolated vertex in the graph represents a dot in the mathematical expression, possibly a decimal point or part of a character like "i". Therefore, a stroke containing a single point is extracted for each vertex with degree 0. In addition, a path in the skeleton graph indicates a candidate of stroke. Although there may be multiple ways to combine the edges into paths, some combinations are more likely to form strokes of a mathematical expression written by human being. Here are some heuristic principles:

- The total number of strokes should be minimal. Since letting the pen to leave the paper requires additional time, an unicursal way is preferred.
- The difference in directions between two successive segments should be as small as possible. Since turning suddenly requires slowing down, a fluent stroke is better to be smooth.

Subjecting to these considerations, each edge is assigned to exactly one path by a bottom up clustering. Initially, each edge forms a path on its own. While there is a pair of paths having a common end point, choose a pair such that the angle between them is the minimum, then merge them into one path. Repeat the procedure until no path can be merged.

It should be noted that the two principles may not always agree. If the number of strokes is considered more important, its minimum can be obtained by merging each path with circuits that have a common vertex with it, just like the algorithm that search for an Eulerian path.

F. RESTORATION OF DOUBLE-TRACED STROKES

Sometimes, a segment should be shared by more than one strokes or appeared in a stroke multiple times due to reentry

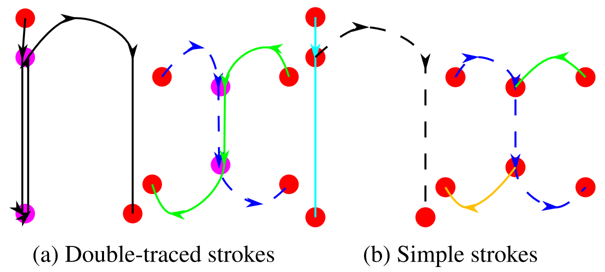


FIGURE 7. **Examples of double-traced strokes. In a commonly seen way to write the expression "*nx*", the edges between two magenta vertexes are traversed twice. The simple tracing procedure cannot recover the strokes correctly.**

during writing as shown in Figure 7a. The tracing procedure above would handle such cases incorrectly by producing too many strokes as in Figure 7b.

In order to fix the double-traced strokes, a search for shared segments is needed, so that they can be used to reconnect separated strokes. Candidates of shared segments should meet the following criteria:

- The segment has two different end points, which are vertexes in the graph with an odd degree. Otherwise, the number of vertexes in the graph having an odd degree does not decrease when it is doubled.
- Each end point of the segment is also an end point of a path given by subsection III-E and the angle between them is not close to $\pi/2$. This condition can prevent the two strokes of the symbol "T" from being merged.

G. POSTPROCESSING

The tracing stage does not determine all dynamic information, so stroke direction detection and stroke order normalization are performed afterward. The convention that people write from left to right and from top to bottom is used to resolve ambiguities.

Firstly, directions of the strokes should be detected. The stroke tracing procedure gives rise to an ordering of points within a stroke naturally, however, the opposite ordering may also make sense. A simple rule is sufficient to determine the direction of each stroke in most cases. Let the coordinates of the first and the last point of a stroke be (x_{start}, y_{start}) and (x_{end}, y_{end}) respectively, then the list of points should be reversed if

$$2x_{end} + 3y_{end} < 2x_{start} + 3y_{start}.$$

Finally, the order that people write down the strokes should be recovered. Although some mathematical expression recognition systems are stroke order free [8], [10], others use stroke order to prune the search space, so they are sensitive to stroke order [33], [34]. In fact, stroke order normalization is a way to turn a stroke order dependent recognizer into a stroke order free one [35].

There are possibly multiple orders to write down the same formula. For example, someone prefers to write down the square root sign first, but the others write down the radicand first. Therefore, it is not always possible to recover the original order, what can be done is to assign a reasonable order. In some cases, mistakes made by the stroke extractor can also be viewed as a kind of normalization and may in fact enhance performance by eliminating unusual stroke order.

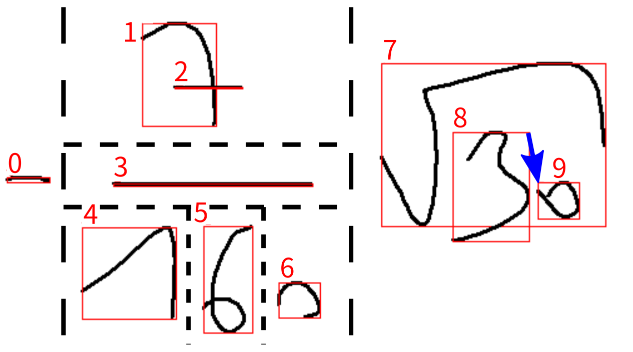


FIGURE 8. **Example of stroke order normalization. Vertical and horizontal broken lines separate groups generated by the recursive X-Y cut, the blue arrow represents a precedence relation being used in the topological sort, and the red numbers show the estimated order of strokes.**

A hierarchical approach is applied to sort the strokes. To begin with, strokes are grouped by recursive projection, then the groups are sorted in a left to right and top to bottom manner. However, recursive X-Y cut cannot determine the order of symbols inside a square root, so strokes inside each group are sorted by a topological sort afterward. A stroke T_i precedes another stroke T_j if one of the following conditions holds:

- T_i is on the left of T_j, where their projection to the y-axis (but not the x-axis) intersect;
- T_i is on top of T_j, where their projection to the x-axis (but not the y-axis) intersect.

Further ambiguities are resolved by using the coordinates of the top left corner of the bounding boxes. Figure 8 illustrates how strokes are sorted.

IV. EVALUATION

A. DATASETS AND EVALUATION PROTOCOL

The proposed system was evaluated on CROHME datasets, which are standard for mathematical expression recognition. Table 1 summarizes the datasets. For each mathematical expression in a dataset, list of points in each stroke is provided together with ground truth. In addition to MathML representation of the expression, correspondence between symbols and strokes is also provided. Since a bitmap image of mathematical expression can be obtained by rendering the strokes, the datasets can be used to evaluate offline recognition systems as well. Following the settings of task 2 (offline handwritten formula recognition) in CROHME 2019, formulas were rendered at resolution of 1000×1000 pixels by using the script provided by the organizers. However, it should be noted that rendered images are different from scanned or camera captured mathematical expressions in the level of background noises, although removing them is another research topic.

TABLE 1. Number of expressions in CROHME datasets.

Year	Training	Validation	Test
2014	8836	671	986
2016	8836	986	1147
2019	9993	986	1199

Aligning with the offline task in CROHME 2019 [3], expression level metrics computed from the symbol level label graphs of formulas were used to evaluate the proposed system. Structure rate measures the percentage of recognized expressions matching the ground truth if all the labels of symbols are ignored. Expression rate measures the percentage of recognized expressions matching the ground truth up to a certain number of labeling mistakes on symbols or spatial relations. On the other hand, stroke classification rate, symbol segmentation rate, symbol recognition rate, and metrics based on stroke level label graph are inapplicable because offline recognition does not produce correspondence between symbols and strokes.

B. INTEGRATION WITH A READY-MADE ONLINE RECOGNIZER

In the first experiment, the proposed stroke extractor[1] was combined with version 1.3 of MyScript Math recognizer,[2] which is a state-of-the-art online handwritten mathematical expression recognition system, to form an offline recognizer. The online recognizer was customized with a grammar to eliminate candidates of symbols and constructs that never appeared in CROHME datasets. Table 2 shows the customized grammar, where "$<start>$" is the start symbol.

Table 3 shows experimental results on CROHME 2014 test set. The first seven are online recognition systems participated in CROHME 2014 [36]. All of them except MyScript were only trained on the official training set. The proposed system outperformed all participated systems in CROHME 2014 except MyScript. Since MyScript Math recognizer itself has evolved over the past few years, MyScript Interactive Ink version 1.3, which is up-to-date as of this writing, was evaluated in the online setting too. On the other hand, im2markup [15], Watch, Attend, and Parse (WAP) [16], CNN-BLSTM-LSTM [37], Paired Adversarial Learning (PAL) [38], and Multi-Scale Attention with Dense Encoder (MSD) [17] are offline recognition systems, the proposed procedure achieved a better performance than them. Noted that the proposed system used additional training data indirectly because it was based on MyScript, im2markup also relied on a large dataset of printed formulas, other compared offline recognizers only required the official training data.

[1]The proposed stroke extractor is publicly available as a free software at https://github.com/chungkwong/mathocr-myscript and https://github.com/chungkwong/mathocr-myscript-android.

[2]Documents of the recognizer is available at https://developer.myscript.com/docs/interactive-ink/1.3/overview/about/.

TABLE 2. Customized grammar for CROHME datasets.

$$
\begin{array}{rcl}
<start> & ::= & <term> \\
<term> & ::= & <operand> \\
 & | & <symbol2> \\
 & | & <symbol4> \\
 & | & <term><term> \\
 & | & \frac{<term>}{<term>} \\
 & | & <operand>_{<term>} \\
 & | & <operand>^{<term>} \\
 & | & <operand>_{<term>}^{<term>} \\
 & | & <symbol3> \\
 & & \quad <term> \\
 & | & \quad <term> \\
 & | & <symbol4> \\
 & & \quad <term> \\
<operand> & ::= & <symbol1> \\
 & | & \sqrt{<term>} \\
 & | & \sqrt[<term>]{<term>} \\
 & | & <symbol5><term><symbol6> \\
<symbol1> & ::= & `0' \mid `1' \mid `2' \mid `3' \mid `4' \mid `5' \mid `6' \mid `7' \mid `8' \mid `9' \\
 & | & `A' \mid `B' \mid `C' \mid `E' \mid `F' \mid `G' \mid `H' \mid `I' \mid `L' \\
 & | & `M' \mid `N' \mid `P' \mid `R' \mid `S' \mid `T' \mid `V' \mid `X' \mid `Y' \\
 & | & `a' \mid `b' \mid `c' \mid `d' \mid `e' \mid `f' \mid `g' \mid `h' \mid `i' \mid `j' \mid `k' \\
 & | & `l' \mid `m' \mid `n' \mid `o' \mid `p' \mid `q' \mid `r' \mid `s' \mid `t' \mid `u' \\
 & | & `v' \mid `w' \mid `x' \mid `y' \mid `z' \mid `\Delta' \mid `\alpha' \mid `\beta' \mid `\gamma' \\
 & | & `\theta' \mid `\lambda' \mid `\pi' \mid `\sigma' \mid `\varphi' \mid `\phi' \mid `\mu' \mid `<' \mid `>' \mid `!' \\
 & | & `/' \mid `\infty' \mid `\mathrm{sin}' \mid `\mathrm{cos}' \mid `\mathrm{tan}' \mid `\mathrm{lim}' \mid `\mathrm{log}' \\
<symbol2> & ::= & `+' \mid `-' \mid `\pm' \mid `\times' \mid `\div' \mid `\cdot' \mid `=' \mid `\prime' \mid `,' \mid `.' \\
 & | & `\ldots' \mid `\rightarrow' \mid `\exists' \mid `\in' \mid `\neq' \mid `\leq' \mid `\geq' \mid `\forall' \\
<symbol3> & ::= & `\sum' \mid `\int' \mid `\mathrm{lim}' \\
<symbol4> & ::= & `\sum' \mid `\int' \\
<symbol5> & ::= & `(' \mid `[' \mid `\{' \mid `|' \\
<symbol6> & ::= & `)' \mid `]' \mid `\}' \mid `|'
\end{array}
$$

TABLE 3. Recognition performance on CROHME 2014 test set.

System	Expression rate			Structure rate
	Exact	≤ 1 label error	≤ 2 label errors	
Online recognizer				
São Paulo	15.01	22.31	26.57	-
RIT, CIS	18.97	26.37	30.83	-
RIT, DRPL	18.97	28.19	32.35	-
Tokyo	25.66	33.16	35.90	-
Nantes	26.06	33.87	38.54	-
Politècnica de València	37.22	44.22	47.26	-
MyScript	62.68	72.31	75.15	-
MyScript 1.3	69.47	78.30	81.03	82.86
Offline recognizer				
Harvard, im2markup	39.96	-	-	-
USTC, WAP	44.4	58.4	62.2	-
CAS, PAL	47.06	63.49	72.31	-
TDTU, CNN-BLSTM-LSTM	48.78	63.39	70.18	-
USTC, MSD	52.8	68.1	72.0	-
Proposed+MyScript 1.3	58.22	71.60	75.15	77.38

Table 4 shows experimental results on CROHME 2016 test set. The first five are online recognition systems participated in CROHME 2016 [39]. MyScript was trained on about 30,000 extra handwritten expressions, Wiris included a language model which was trained on the Wikipedia formula corpus, the others only used the official training set. As expected, MyScript had a higher accuracy than the proposed system in all the metrics because the later was based on the former. The proposed system significantly outperformed all the remaining participated systems in CROHME

TABLE 4. Recognition performance on CROHME 2016 test set.

System	Expression rate			Structure rate
	Exact	≤ 1 label error	≤ 2 label errors	
Online recognizer				
Nantes	13.34	21.02	28.33	21.45
São Paolo	33.39	43.50	49.17	57.02
Tokyo	43.94	50.91	53.70	61.55
Wiris	49.61	60.42	64.69	74.28
MyScript	67.65	75.59	79.86	88.14
MyScript 1.3	73.06	82.30	87.10	88.58
Offline recognizer				
USTC, WAP	42.0	55.1	59.3	-
TDTU, CNN-BLSTM-LSTM	45.60	59.29	65.65	-
USTC, MSD	50.1	63.8	67.4	-
Proposed+MyScript 1.3	65.65	77.68	82.56	85.00

TABLE 5. Recognition performance on CROHME 2019 test set.

System	Expression rate			Structure rate
	Exact	≤ 1 label error	≤ 2 label errors	
Online recognizer				
TUAT	39.95	52.21	56.54	58.22
MathType	60.13	74.40	78.57	79.15
PAL-v2	62.55	74.98	78.40	79.15
Samsung R&D 2	65.97	77.81	81.73	82.82
MyScript 1.3	77.40	85.82	87.99	88.82
MyScript	79.15	86.82	89.82	90.66
Samsung R&D 1	79.82	87.82	89.15	89.32
USTC-iFLYTEK	80.73	88.99	90.74	91.49
Offline recognizer				
TUAT	24.10	35.53	43.12	43.70
Univ. Linz	41.49	54.13	58.88	60.02
SCST-USTC	62.14	75.06	78.23	78.32
PAL-v2	62.89	74.98	78.40	79.32
Proposed+MyScript 1.3	65.22	78.48	83.07	84.90
PAL	71.23	80.31	82.65	83.82
USTC-iFLYTEK	77.15	86.82	88.99	89.49

2016 without access to original strokes. On the other hand, WAP [16], CNN-BLSTM-LSTM [37], and MSD [17] are also offline recognition systems, the proposed system outperformed them. Although the indirect use of additional training data may contribute to that, good performance was obtained with a less powerful online recognizer which used the official data only, the details are presented in the next subsection.

The proposed system participated in CROHME 2019 [3]. Table 5 shows the final results of the competition. The proposed system was ranked the third place in the offline task. Noted that Samsung R&D 1, PAL, MyScript 1.3, and MathType used additional training data, USTC-iFLYTEK included a language model which was trained on an external dataset.

Although the accuracy of the proposed offline recognizer was good compared with other offline recognizers, it was still much lower than that of the underlying online recognizer. If the stroke extractor worked perfectly, the two should be the same. The gap in expression rate was much larger than the gap in structure rate, the observation indicates that structural analysis is less sensitive to errors in stroke extraction than symbol recognition.

In order to narrow the gap, one should bring output of the stroke extractor and expected input of the online recognizer closer. There are two ways to achieve that. The first way is to improve the stroke extractor, so that it can recover written strokes exactly. The second way is to retrain the online recognition engine, so that it can adapt to the artificial strokes.

Improving the stroke extractor seems to be the most obvious choice, but it is hard. Adding more heuristic rules would lead to serious maintainability issues, whereas marginal benefit is diminishing. Therefore, many researchers had given up in this direction. Developing data driven stroke extractors is possible, but it would defeat the purpose. If heavy models such as convolutional and recurrent neural networks are applied, it is pointless to predict trajectories instead of formulas themselves.

Retraining the online recognition engine is a realistic choice. If the recognizer has seen artificial strokes given by the stroke extractor during training, it should be able to learn what they mean. By applying this technique, the sense of stroke extracted is decoupled from the way written by human beings, thus the stroke extractor needs not worry too much about corner cases. Unfortunately, MyScript Math recognizer was not trainable by user, so the next experiment was switched to another online recognizer.

C. INTEGRATION WITH A TRAINABLE ONLINE RECOGNIZER

In the second experiment, the proposed stroke extractor was combined with TAP,[3] which is a published online handwritten mathematical expression recognition system [11], to form an offline recognizer. The paper introducing TAP combined three online models, three offline models, and three language models to get the best results, but ensemble modeling was not used in this experiment, since the objective is to check if a pure online model can adapt to the artificial strokes.

Evaluation was performed on the CROHME 2016 dataset, only the official training set and validation set were used to train and validate the models respectively. The encoder of TAP consumes an 8-dimensional feature vector for a point at each time step, the decoder of TAP predicts a TeX token at each time step, the loss function involves alignment between points and TeX tokens. When trained on written strokes, trace points and alignment were available from the dataset directly. When trained on artificial strokes, trace points were extracted from rendered images and alignment were estimated with Hausdorff distance. In both cases, TeX code for each formula was converted from annotation in MathML. Since a trained model may depend on initialization, three models were trained on the same training set with different initial

[3]The original version of TAP is available at https://github.com/JianshuZhang/TAP. Our version of programs, datasets, and pretrained models are available at https://github.com/chungkwong/mathocr-tap.

TABLE 6. Recognition performance on CROHME 2016 test set given different training data.

Training/validation	Test	Expression rate			Structure rate
		Exact	≤ 1 label error	≤ 2 label errors	
Written	Written	43.68	55.88	61.29	62.60
Written	Extracted	23.63	38.71	47.17	51.79
Extracted	Extracted	43.07	56.67	62.95	64.95

weights, then the one resulting in the highest expression rate on the validation set was selected to be tested on the test set.

Table 6 shows experimental results on CROHME 2016 test set. The model which had been trained on written strokes performed much better on written strokes than on extracted strokes, the same phenomenon was already observed with MyScript. However, the model which had been trained on extracted strokes leaded to a much better offline recognizer, its accuracy almost caught up the online recognizer. The gap in expression rate between them was only 0.61%, noted that the gap in expression rate between the best online system and the best offline system participated in CROHME 2019 was 3.58%. The results verify that an online recognition system can adapted to the artificial strokes well by retraining. Another implication is that the essential difficulty between online recognition and offline recognition may be smaller than it was thought.

Although the offline recognizer built upon TAP was not as good as the one built upon MyScript, there is room for improvement. For example, language models, ensemble models, augmented datasets, and expanded datasets were not applied in this experiment, whereas they were commonly used by other state-of-the-art mathematical expression recognition systems [3] to boost the accuracy significantly [11], [37].

D. EFFICIENCY

In the third experiment, efficiency of the proposed system was examined on devices ranging from low-end mobile phones to powerful GPU server. Table 7 shows details of those devices.

Table 8 shows the averaged time used to recognize 1147 rendered mathematical expressions in the test set of CROHME 2016. Although the stroke extraction algorithm was implemented on CPU and not optimized for speed, it was still much faster than online recognition on all the devices tested. A modern low-end mobile phone (Phone 2) took about one second to recognize an image on average, whereas a clearly outdated mobile phone (Phone 1) took about two seconds. Although it is noticeable, the speed should be acceptable for on-device offline handwritten mathematical expression recognition, if users feel that online recognition is already fast enough.

TABLE 7. Specifications of tested devices.

Device	Component	Specification
Phone 1	CPU	Cortex-A7 4 cores @ 1.3 GHz
	RAM	512MB
Phone 2	CPU	Cortex-A53 2 cores @ 2.0 GHz + 6 cores @ 1.45 GHz
	RAM	3GB
Ultrabook	CPU	Intel® Core™ m3-6Y30 4 cores @ 0.90GHz
	RAM	4GB
Desktop	CPU	Intel® Core™ i5-7500 4 cores @ 3.40GHz
	RAM	8GB
Server	CPU	Intel® Xeon® Gold 6271C 2 cores @ 2.60GHz
	RAM	32GB
	GPU	NVIDIA® Tesla® V100 16GB

TABLE 8. Time used to recognize 1147 expressions in CROHME 2016 dataset.

Device	Recognizer	Execution time per expression (s)		
		Stroke extraction	Online recognition	Total
Phone 1	MyScript	0.894	1.441	2.335
Phone 2	MyScript	0.222	0.979	1.201
Ultrabook	TAP	0.019	6.023	6.042
Desktop	TAP	0.009	0.466	0.475
	WAP	-	-	6.839
Server	TAP	0.029	0.132	0.162
	WAP	-	-	0.364

WAP,[4] which is a native offline recognizer [16], was slower than the proposed procedure on all the device tested. Like other modern optical character recognition systems, WAP uses a convolutional neural network which is computationally intensive [40] to extract features, so the speed relies heavily on the availability of powerful GPU. Since online recognition engines are usually less resource-hogging than native offline recognition engines, on-device online recognition is already integrated into various note-taking applications and input methods nowadays, whereas optical character recognition functionality usually relies on cloud based services. Therefore, stroke extraction provides an attractive way to improve user experience by avoiding unpredictable network latency and privacy issues.

E. ANALYSIS OF THE STROKE EXTRACTION METHOD

In the last experiment, the proposed stroke extraction method was analyzed. To evaluate the necessity of the optional components, each of them was removed from the proposed system, resulting accuracy was compared to that of the complete system, the decrease in expression rate measured the importance of the removed component.

Table 9 shows recognition performance of the modified systems on CROHME 2016 test set, where MyScript Math recognizer was applied to recognize extracted strokes. Despite the simplicity of the rule for stroke direction detec-

[4]WAP is available at https://github.com/JianshuZhang/WAP.

TABLE 9. Recognition performance of the modified systems on CROHME 2016 test set.

The only component removed from the proposed system	Expression rate Exact	≤ 1 label error	≤ 2 label errors	Structure rate
Stroke direction detection	46.56	63.64	70.53	74.63
Restoration of double-traced strokes	53.97	66.87	71.93	73.76
Noise reduction	58.94	72.36	79.51	80.91
Stroke order normalization	59.98	73.50	78.38	81.34
Customized grammar	61.99	74.72	79.69	82.65
-	65.65	77.68	82.56	85.00

tion, it greatly boosted the accuracy of offline recognition by allowing a more accurate classification of symbols. Noted that the symbol recognizer of MyScript employed both static and dynamic features, so it could recognize some strokes even if their directions are wrong. Restoration of double-traced strokes was quite important because it enabled a more reliable symbol segmentation, where broken strokes might lead to oversegmentation. Noise reduction, stroke order normalization, and customized grammar also contributed to the overall accuracy.

The evaluation may inspire the development of other stroke extraction methods. Although handcrafted rules are used in several steps and they are probably not the best ones, experimental results indicate that the steps tackled problems which matter the accuracy of recognition. Therefore, stroke extraction methods designed for offline recognition in the future should address those problems too, especially if retraining of the underlying online recognizer is not allowed.

V. CONCLUSION

In this paper, a stroke extraction algorithm is proposed for handwritten mathematical expression, so that an offline recognizer can be built upon an online one. A proof-of-concept implementation of the proposed stroke extractor is publicly available as a free software. Given a ready-made state-of-the-art online recognition engine, good offline accuracy was achieved on CROHME datasets. Given a trainable online recognition system, an offline recognizer with the same level of accuracy was constructed by retraining it with extracted strokes. Noted that the underlying online recognizers are not specially designed for extracted strokes.

The proposed approach is especially preferable for real-time use cases on devices with limited resources. Since online recognizers generally occupy less memory and run faster than native offline recognizers, stroke extraction provides an efficient way to implement on-device offline recognition on mobile phones and tablets.

Stroke extraction is a general methodology to offline handwriting recognition. Besides handwritten mathematical expression, the same approach can be applied to other types of handwriting such as chemical expression, musical notation, and diagram in principle. Examining if specialized stroke extractors are needed for different types of handwriting is a future work.

Developing independent recognition systems for online and offline handwriting may no longer be necessary, since stroke extraction allows advances on online recognition to be propagated immediately to the offline case. Instead, online recognition system makers can enter the offline market without abandoning existing investments. Therefore, the potential of reduction from offline recognition to online recognition is justified.

REFERENCES

[1] R. Zanibbi and D. Blostein, "Recognition and retrieval of mathematical expressions," *Int. J. Document Anal. Recognit.*, vol. 15, no. 4, pp. 331–357, Dec. 2012.

[2] M. Marchiori, "The mathematical semantic Web," in *Proc. Int. Conf. Math. Knowl. Manage.*, Bertinoro, Italy, Feb. 2003, pp. 216–224.

[3] M. Mahdavi, R. Zanibbi, H. Mouchère, C. Viard-Gaudin, and U. Garain, "ICDAR 2019 CROHME + TFD: Competition on recognition of handwritten mathematical expressions and typeset formula detection," in *Proc. Int. Conf. Document Anal. Recognit. (ICDAR)*, Sydney, NSW, Australia: IEEE, Sep. 2019, pp. 1533–1538.

[4] H. Nishida, "An approach to integration of off-line and on-line recognition of handwriting," *Pattern Recognit. Lett.*, vol. 16, no. 11, pp. 1213–1219, Nov. 1995.

[5] P. M. Lallican, C. Viard-Gaudin, and S. Knerr, "From off-line to on-line handwriting recognition," in *Proc. 7th Int. Workshop Frontiers Handwriting Recognit.*, L. Schomaker and L. Vuurpijl, Eds. Amsterdam, The Netherlands: International Unipen Foundation, Sep. 2000, pp. 303–312.

[6] R. H. Anderson, "Syntax-directed recognition of hand-printed two-dimensional mathematics," in *Proc. Interact. Symp. Syst. Experim. Appl. Math.*, 1968, pp. 436–459.

[7] H. Mouchère, R. Zanibbi, U. Garain, and C. Viard-Gaudin, "Advancing the state of the art for handwritten math recognition: The CROHME competitions, 2011–2014," *Int. J. Document Anal. Recognit.*, vol. 19, no. 2, pp. 173–189, Jun. 2016.

[8] F. Álvaro, J.-A. Sánchez, and J.-M. Benedí, "Recognition of on-line handwritten mathematical expressions using 2D stochastic context-free grammars and hidden Markov models," *Pattern Recognit. Lett.*, vol. 35, pp. 58–67, Jan. 2014.

[9] R. Yamamoto, S. Sako, T. Nishimoto, and S. Sagayama, "On-line recognition of handwritten mathematical expressions based on stroke-based stochastic context-free grammar," in *Proc. 10th Int. Workshop Frontiers Handwriting Recognit.*, G. Lorette, Ed. La Baule, France: Suvisoft, Oct. 2006, pp. 249–254.

[10] A.-M. Awal, H. Mouchère, and C. Viard-Gaudin, "A global learning approach for an online handwritten mathematical expression recognition system," *Pattern Recognit. Lett.*, vol. 35, pp. 68–77, Jan. 2014.

[11] J. Zhang, J. Du, and L. Dai, "Track, attend, and parse (TAP): An end-to-end framework for online handwritten mathematical expression recognition," *IEEE Trans. Multimedia*, vol. 21, no. 1, pp. 221–233, Jan. 2019.

[12] U. Garain and B. B. Chaudhuri, *OCR Printed Math. Expressions*. London, U.K.: Springer, 2007, pp. 235–259.

[13] K.-F. Chan and D.-Y. Yeung, "Mathematical expression recognition: A survey," *Int. J. Document Anal. Recognit.*, vol. 3, no. 1, pp. 3–15, Aug. 2000.

[14] M. Suzuki, F. Tamari, R. Fukuda, S. Uchida, and T. Kanahori, "INFTY: An integrated ocr system for mathematical documents," in *Proc. Acm Symp. Document Eng.*, C. Roisin and E. V. Munson, Eds. Grenoble, France: ACM, Nov. 2003, pp. 95–104.

[15] Y. Deng, A. Kanervisto, J. Ling, and A. M. Rush, "Image-to-markup generation with coarse-to-fine attention," in *Proc. 34th Int. Conf. Mach. Learn.* D. Precup and Y. W. Teh, Eds. Sydney, NSW, Australia: PMLR, Aug. 2017, pp. 980–989.

[16] J. Zhang, J. Du, S. Zhang, D. Liu, Y. Hu, J. Hu, S. Wei, and L. Dai, "Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition," *Pattern Recognit.*, vol. 71, pp. 196–206, Nov. 2017.

[17] J. Zhang, J. Du, and L. Dai, "Multi-scale attention with dense encoder for handwritten mathematical expression recognition," in *Proc. 24th Int. Conf. Pattern Recognit. (ICPR)*, D. Lopresti and R. He, Eds. Beijing, China: IEEE, Aug. 2018, pp. 2245–2250.

[18] S. Lee and J. C. Pan, "Offline tracing and representation of signatures," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 4, pp. 755–771, Jul. 1992.

[19] C.-L. Liu, I.-J. Kim, and J. H. Kim, "Model-based stroke extraction and matching for handwritten Chinese character recognition," *Pattern Recognit.*, vol. 34, no. 12, pp. 2339–2352, Dec. 2001.

[20] G. Boccignone, A. Chianese, L. P. Cordella, and A. Marcelli, "Recovering dynamic information from static handwriting," *Pattern Recognit.*, vol. 26, no. 3, pp. 409–418, Mar. 1993.

[21] D. S. Doermann and A. Rosenfeld, "Recovery of temporal information from static images of handwriting," *Int. J. Comput. Vis.*, vol. 15, nos. 1–2, pp. 143–164, Jun. 1995.

[22] S. Jager, "Recovering writing traces in off-line handwriting recognition: Using a global optimization technique," in *Proc. 13th Int. Conf. Pattern Recognit.*, M. E. Kavanaugh and B. Werner, Eds. Vienna, Austria: IEEE, Aug. 1996, pp. 150–154.

[23] K. K. Lau, P. C. Yuen, and Y. Y. Tang, "Stroke extraction and stroke sequence estimation on signatures," in *Proc. Object Recognit. Interact. Service Robots*, R. Kasturi, D. Laurendeau, and C. Suen, Eds. Quebec City, QC, Canada: IEEE, Aug. 2002, pp. 119–122.

[24] Y. Kato and M. Yasuhara, "Recovery of drawing order from single-stroke handwriting images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 9, pp. 938–949, Sep. 2000.

[25] T. Nagoya and H. Fujioka, "Recovering dynamic stroke information of multi-stroke handwritten characters with complex patterns," in *Proc. Int. Conf. Frontiers Handwriting Recognit.*, J. E. Guerrero, Ed. Bari, Italy: IEEE, Sep. 2012, pp. 722–727.

[26] J. Sauvola and M. Pietikäinen, "Adaptive document image binarization," *Pattern Recognit.*, vol. 33, no. 2, pp. 225–236, Feb. 2000.

[27] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-9, no. 1, pp. 62–66, Jan. 1979.

[28] P. S. P. Wang and Y. Y. Zhang, "A fast and flexible thinning algorithm," *IEEE Trans. Comput.*, vol. 38, no. 5, pp. 741–745, May 1989.

[29] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Commun. ACM*, vol. 27, no. 3, pp. 236–239, Mar. 1984.

[30] L. He, X. Ren, Q. Gao, X. Zhao, B. Yao, and Y. Chao, "The connected-component labeling problem: A review of state-of-the-art algorithms," *Pattern Recognit.*, vol. 70, pp. 25–43, Oct. 2017.

[31] B. Epshtein, E. Ofek, and Y. Wexler, "Detecting text in natural scenes with stroke width transform," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* San Francisco, CA, USA: IEEE, Jun. 2010, pp. 2963–2970.

[32] K.-C. Fan and W.-H. Wu, "A run-length-coding-based approach to stroke extraction of Chinese characters," *Pattern Recognit.*, vol. 33, no. 11, pp. 1881–1895, Nov. 2000.

[33] F. Simistira, V. Katsouros, and G. Carayannis, "Recognition of online handwritten mathematical formulas using probabilistic SVMs and stochastic context free grammars," *Pattern Recognit. Lett.*, vol. 53, pp. 85–92, Feb. 2015.

[34] A. D. Le and M. Nakagawa, "A system for recognizing online handwritten mathematical expressions by using improved structural analysis," *Int. J. Document Anal. Recognit.*, vol. 19, no. 4, pp. 305–319, Dec. 2016.

[35] A. D. Le, H. D. Nguyen, B. Indurkhya, and M. Nakagawa, "Stroke order normalization for improving recognition of online handwritten mathematical expressions," *Int. J. Document Anal. Recognit.*, vol. 22, no. 1, pp. 29–39, Mar. 2019.

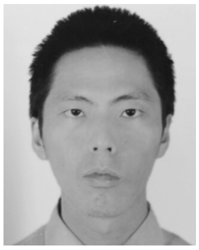
[36] H. Mouchere, C. Viard-Gaudin, R. Zanibbi, and U. Garain, "ICFHR 2014 competition on recognition of on-line handwritten mathematical expressions (CROHME 2014)," in *Proc. 14th Int. Conf. Frontiers Handwriting Recognit.*, J. E. Guerrero, Ed. Heraklion, Greece: IEEE, Sep. 2014, pp. 791–796.

[37] A. D. Le, B. Indurkhya, and M. Nakagawa, "Pattern generation strategies for improving recognition of handwritten mathematical expressions," *Pattern Recognit. Lett.*, vol. 128, pp. 255–262, Dec. 2019.

[38] J.-W. Wu, F. Yin, Y.-M. Zhang, X.-Y. Zhang, and C.-L. Liu, "Image-to-markup generation via paired adversarial learning," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, M. Berlingerio, F. Bonchi, T. Gärtner, N. Hurley, and G. Ifrim, Eds. Dublin, Ireland: Springer, Sep. 2018, pp. 18–34.

[39] H. Mouchere, C. Viard-Gaudin, R. Zanibbi, and U. Garain, "ICFHR2016 CROHME: Competition on recognition of online handwritten mathematical expressions," in *Proc. 15th Int. Conf. Frontiers Handwriting Recognit. (ICFHR)*, J. E. Guerrero, Ed. Shenzhen, China: IEEE, Oct. 2016, pp. 607–612.

[40] D. Zhelezniakov, V. Zaytsev, and O. Radyvonenko, "Acceleration of online recognition of 2D sequences using deep bidirectional LSTM and dynamic programming," in *Proc. 15th Int. Work-Conf. Artif. Neural Netw.*, I. Rojas, G. Joya, and A. Catala, Eds. Gran Canaria, Spain: Springer, Jun. 2019, pp. 438–449.



CHUNGKWONG CHAN was born in Hong Kong, in 1992. He received the B.S. degree in mathematics and applied mathematics from Sun Yat-Sen University, Guangzhou, China, in 2015, where he is currently pursuing the Ph.D. degree in mathematics. His current research interests include mathematical expression recognition, information retrieval, and natural language processing.