International Journal on Document Analysis and Recognition (IJDAR) (2019) 22:29–39
https://doi.org/10.1007/s10032-019-00315-2

ORIGINALPAPER

Stroke order normalization for improving recognition of online handwritten mathematical expressions

Anh Duc Le[1,2] · Hai Dai Nguyen[3] · Bipin Indurkhya[4] · Masaki Nakagawa[5]

Received: 2 April 2018 / Revised: 14 November 2018 / Accepted: 7 January 2019 / Published online: 17 January 2019

Abstract
We present a technique based on stroke order normalization for improving recognition of online handwritten mathematical expressions (ME). The stroke order dependent system has less time complexity than the stroke order free system, but it must incorporate special grammar rules to cope with stroke order variations. The stroke order normalization technique solves this problem and also the problem of unexpected stroke order variations without increasing the time complexity of ME recognition. In order to normalize stroke order, the X–Y cut method is modified since its original form causes problems when structural components in ME overlap. First, vertically ordered strokes are located by detecting vertical symbols and their upper/lower components, which are treated as MEs and reordered recursively. Second, unordered strokes on the left side of the vertical symbols are reordered as horizontally ordered strokes. Third, the remaining strokes are reordered recursively. The horizontally ordered strokes are reordered from left to right, and the vertically ordered strokes are reordered from top to bottom. Finally, the proposed stroke order normalization is combined with the stroke order dependent ME recognition system. The evaluations on the CROHME 2014 database show that the ME recognition system incorporating the stroke order normalization outperforms all other systems that use only CROHME 2014 for training while the processing time is kept low.

Keywords Recognition of online handwritten mathematical expressions · X–Y cut · Stroke order normalization

Introduction

Mathematical expressions (MEs) are commonly used in education, science, business and even daily life. Math description languages such as LATEX, and math editors such as Microsoft Equation Editor are some popular methods to input MEs to a computer. However, they require users to remember the grammar of a math description language, or follow the wide and deep menus of a math editor. Recently, with the development of pen-based and touch-based tablets and smartphones, it is possible to consider entering handwritten MEs, which is more natural and user friendly. A user should be able to just write the MEs that he/she wants to input by hand, and a math recognizer would recognize and translate the inputted expression to LATEX or some other math formats automatically. But for such an approach to be practical, the ME recognizer must attain a high recognition rate, acceptable speed, and must work under as few constraints as possible.

The recognition of handwritten MEs can be divided into three main processes. First, a sequence of input strokes is segmented into hypothetical symbols (symbol segmentation), where each stroke is a sequence of coordinates from pen/touch-down to pen/touch-up. Then, each hypothetical symbol is recognized by a symbol classifier (symbol recognition). Finally, structural relations among the recognized symbols are determined and the expression structure is analyzed by parsing the recognized symbols to determine the most likely interpretation as an ME (structural analysis). The recognition problem requires not only the segmenta-

✉ Anh Duc Le
leducanh@tdtu.edu.vn

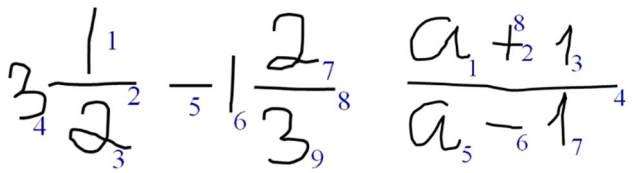
1 Division of Algorithms and Technologies for Networks Analysis, Ton Duc Thang University, Ho Chi Minh City, Vietnam

2 Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam

3 Bioinformatics Center, Institute for Chemical Research, Kyoto University, Kyoto, Japan

4 Departments of Computer Science and Cognitive Science, Jagiellonian University Cracow, Kraków, Poland

5 Department of Computer and Information Sciences, Tokyo University of Agriculture and Technology, Tokyo, Japan



(a) (b)

Fig. 1 Examples of stroke order variation in writing MEs

tion and recognition of symbols but also an analysis of two-dimensional (2D) structures and an interpretation of structural relations. Ambiguities arise in all stages of the process. Practical online handwriting ME recognition systems should be stroke order free in order to allow users to write MEs without constraints of stroke order. Figure 1 shows examples of stroke order variations for simple MEs: The stroke order is shown in small indices beside the strokes. Components in mixed fractions and integrations, superscripts, subscripts and so on are written in any order, and brackets and operators are often written later. In Fig. 1a, stroke 4 is written after its fractional expression. In Fig. 1b, stroke 8 is a delayed stroke.

Many approaches have been explored for recognizing handwritten MEs, especially in the last two decades. They are summarized in survey papers [1, 2] and a recent competition paper [3].

For symbol segmentation, symbol hypotheses net [4], candidate character lattice [5] and shape context features [6] have been proposed. For symbol recognition, elastic matching [7], HMM [8] and RNN [9] have been employed for online methods, while Adaboost, SVM and Random Forest [10] have been used for offline methods. Combinations of online methods and offline methods were presented by Garain et al. [11], Alvaro et al. [12] and Nguyen et al. [13, 14].

For the structural analysis, we will group a few recent approaches into two categories: stroke order dependent (recognition result depends on the stroke order) and stroke order free (recognition result is free from the stroke order).

For the stroke order dependent approach, the recognition problem is formulated as a search problem of the most likely ME candidate in a framework of Stochastic Context Free Grammar (SCFG) by Yamamoto et al. [15]. The search space is reduced by employing the stroke order, and a sequence of input strokes is parsed by the Cocke–Younger–Kasami (CYK) algorithm. The recognition rate is improved according to different grammatical constraints. Simistira et al. [16] and Le et al. [17, 18] employed similar approaches. Simistira et al.'s system parses segmented symbols with probabilistic SVMs and SCFG. The system by Le et al. employs a body box instead of a bounding box and introduces a set of additional productions for improving the structural analysis. The stroke order is also used to reduce the number of sub-partitions that must be considered during parsing to $O(n^2)$ and the complexity of parsing algorithm to $O(n^3|P|)$.

For the stroke order free approach, a top-down parsing algorithm was presented by Maclean et al. [19]. A shared parse forest representing all recognizable parses of the input is incrementally constructed by Unger's method. Then, the most highly ranked tree is extracted from this forest. Infeasible partitions are restricted by constraints of horizontal and vertical relations. However, the worst-case number of sub-partitions that must be considered during parsing is $O(n^4)$, and the complexity of parsing algorithm is $O(n^4|P|)$, both of which are quite large. A bottom-up parsing algorithm has also been proposed by Alvaro et al [20], which combines a formal model for online handwritten ME recognition based on 2D-SCFGs with Hidden Markov Model. The CYK algorithm is modified to parse an input ME in 2D under 2D-SCFGs, and range search is used to improve the time complexity from $O(n^4|P|)$ to $O(n^3 log n|P|)$.

Inspired by recent successes of the attention-based encoder–decoder models in image captioning [28] and machine translation [29], Deng et al. [30], Zhang et al. [31] and Le et al. [32] have extended attention-based encoder–decoder to recognize HMEs. These recognition systems contain a CNN encoder, an attention model and an LSTM decoder. They outperformed traditional recognition systems based on CFG on the CROHME 2014 and 2016 test sets. However, they require expensive computation for training and inference. They may run on GPUs to provide cloud-based recognition services, but not on mobile devices.

Stroke order dependent, stroke order free and attention-based encoder–decoder approaches show good performance for the recent CROHME database [3]. Nevertheless, handwritten ME recognition in real environment still faces many problems with respect to the recognition rate, the recognition speed, the memory size and a user-friendly input interface. To make handwriting ME input practical, the system should allow a user to input MEs without constraints on writing style, stroke order, and should have a fast response time. The stroke order dependent approach has the advantage of a small number of sub-sequences that must be considered during parsing, which results in efficient parsing algorithms, but common stroke order variations must be incorporated in a grammar. Therefore, it cannot cope with unexpected stroke disorder. On the other hand, the stroke order free approach is robust with respect to stroke order variations but must examine an exponentially increasing number of sub-partitions as the number of strokes increases, resulting in time-consuming parsing algorithms.

Another approach is to normalize the stroke order before stroke order dependent parsing. This solution fills the gap between the stroke order dependent and the stroke order free systems and allows a stroke order dependent system to be transformed to a stroke order free system. So far, however,

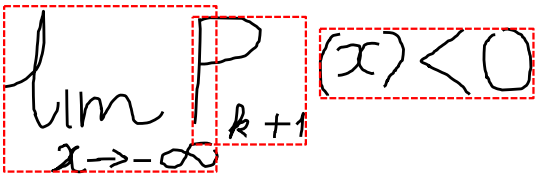


Fig. 2 Examples of a handwritten ME with overlapping structural components

little attention has been paid to this approach. Lee et al. proposed a method to recognize printed MEs [21]. They determine the vertical symbols, such as a fraction bar, symbols for the sum, the product, the integral and the root, above and below which sub-expressions of an ME are placed, and generate a symbol relation tree to represent the ME structure. Zanibbi et al. proposed a tree transformation method [22], where a baseline structure tree is constructed by determining the reading order and the operator dominance. Then, tokens consisting of multiple input symbols such as decimal numbers and function names are grouped together. Finally, they extract an operator tree that is suitable for computation. The above two methods are processed without grammar rule constraints; therefore, they are likely to misrecognize ambiguously handwritten MEs.

The X–Y cut method is a well-known top-down method for page layout analysis [23]. The document is split into smaller rectangular blocks by alternately making horizontal and vertical cuts. A cut is decided by using a threshold for the spaces between the blocks. A limitation of this method is that it only works well for recursively separable documents composed of rectangular blocks without overlap. An optimized X–Y cut has been proposed to determine page-reading order [24] by defining a score function for page layout segmentation. The layout segmentation can be seen as selecting a series of cuts in order to maximize the score function. The reading order is decided from the trace of the X–Y cut method. For handwritten MEs, however, this approach does not work well because structural components often overlap. Figure 2 shows a handwritten ME with overlapping structural components. It is hard to employ the X–Y cut method for this ME. In this paper, we propose a method for normalizing stroke order, which contains three main steps: (i) determining vertically ordered strokes by detecting vertical symbols and their upper and lower components, and then treating each component as an ME and reordering it recursively; (ii) reordering unordered strokes on the left side of a vertical symbol as horizontally ordered strokes; (iii) reordering the remaining strokes recursively by calling the stroke order normalization.

This paper is an extended and updated version of a conference paper [25] with a more elaborate and formal framework, extensive evaluation and a detailed analysis. The rest of this paper is organized as follows. The baseline system for online recognition of handwritten MEs is presented in Sect. 2. The proposed method for normalizing stroke order is described in Sect. 3. The evaluations of the proposed method and the stroke order normalization integrated system are discussed in Sect. 4, and the conclusions are presented in Sect. 5.

Overview of the recognition system

The online handwritten ME recognition problem is formulated as a search problem of the most likely interpretation of handwritten strokes. The search problem is modeled as the following formula for a candidate expression of n symbols connected by m relations and q grammar rules:

$$\begin{aligned} C = {} & \alpha_1 \sum_{i=1}^{n} \ln(P_{\mathrm{sh}}(G_i)) + \alpha_2 \sum_{i=1}^{n} \ln(P_{\mathrm{rec}}(S_i|G_i)) \\ & + \alpha_3 \sum_{k=1}^{m} \ln(P_{\mathrm{rel}}(r_k|A_k B_k)) \\ & + \alpha_4 \sum_{k=1}^{q} P_{\mathrm{gram}}\left(X_k \xrightarrow{r_k} A_k B_k\right) \end{aligned} \tag{1}$$

Here, G_i is a symbol hypothesis composed of a sequence of successive strokes; $P_{\mathrm{sh}}(G_i)$ stands for the probability of a symbol hypothesis G_i; $P_{\mathrm{rec}}(S_i|G_i)$ stands for the probability that a symbol hypothesis G_i is recognized as a symbol S_i; $P_{\mathrm{rel}}(r_k|A_k B_k)$ is the probability that two sub-expressions A_k and B_k are combined into a larger expression with a relation r_k, and $P_{\mathrm{gram}}\left(X_k \xrightarrow{r_k} A_k B_k\right)$ is the probability of a production $X_k \xrightarrow{r_k} A_k B_k$ in the grammar. The coefficients: α_1, α_2, α_3, α_4, are the weighting parameters for probabilities. These parameters are trained by genetic algorithm on a validation set. Description of each probability is presented below.

Probability of symbol hypothesis

For the segmentation, first, candidate symbol hypotheses are extracted. Then, we calculate the probability of each symbol hypothesis that can form a mathematical symbol. The following constraints are employed to reduce the number of hypotheses to be considered: (i) The maximum number of strokes for a symbol hypothesis is four, (ii) symbol hypotheses are composed of a sequence of successive strokes.

Probability of a symbol hypothesis $P_{\mathrm{sh}}(G_i)$ is calculated from the stroke sequence: horizontal and vertical projections of center-to-center distance of bounding boxes, stroke size difference (stroke size is the larger of height and width) and the minimum pairwise distance among all the strokes.

A Gaussian Mixture Model classifier is used for obtaining this probability.

Probability of symbol recognition

The combined recognizer composed of offline and online recognition methods is robust because it combines the advantages of both methods. Particularly, it could recognize connected strokes or cursive strokes by the online method and stroke disorders or duplicated strokes by the offline method.

Bidirectional long short-term memory Neural Network (BLSTM) and Convolution Neural Network (CNN) are used for online and offline symbol classification, respectively. Online features and local gradient features are employed to improve the accuracy of BLSTM. The maxout nonlinearity and the dropout techniques are employed to improve the performance of CNN. The combination equation is shown in (2), where β is the parameter to balance the relative weights of BLSTM and CNN. The details of this combined classifier are reported in [13].

$$P_{\text{rec}}(S_i|G_i) = \beta P_{\text{LSTM}}(S_i|G_i) + (1-\beta) P_{\text{CNN}}(S_i|G_i) \quad (2)$$

Probability of grammar rule

An SCFG is defined formally by a five-tuple $G = (N, \Sigma, R, P, S)$ where:

- N is a finite set of non-terminal symbols.
- Σ is a finite set of terminal symbols.
- R is a finite set of relations between any two sub-expressions. These relations are horizontal, upper, lower, superscript, subscript and inside.
- P is a finite set of grammar rules taking one of the following forms: $X \rightarrow a$ (terminal grammar rule), $X \rightarrow A$(non-terminal grammar rule), $X \xrightarrow{r} AB$ (non-terminal grammar rule), *with* $X, A, B \epsilon N, a \epsilon \Sigma$ and $r \epsilon R$. Each grammar rule is associated with a probability p, and the probabilities of all the rules satisfy the condition: $\forall A \in N : \sum p(A \rightarrow w) = 1$
- $S \epsilon N$ is a distinguished start symbol.

Although handwritten strokes may be ambiguous, a grammar is defined to be unambiguous. This implies that every valid grammar expression has a unique leftmost derivation. The unambiguous grammar is difficult to be defined in Chomsky Normal Form (CNF). Therefore, one more grammar rule ($X \rightarrow A$) is added for making the grammar definition easier. The details are described in [17].

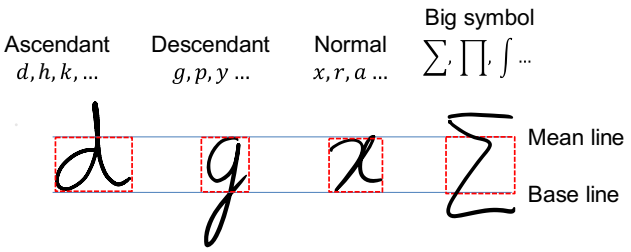


Fig. 3 Body boxes for mathematical symbols

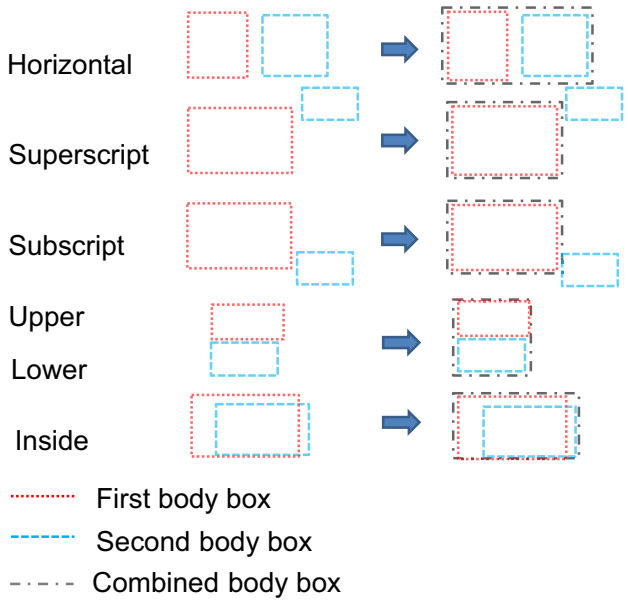
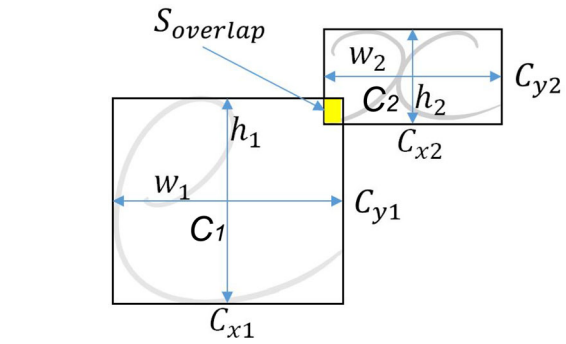


Fig. 4 Body boxes for MEs

Probability of structural relation

In some cases, the structural relations among the symbols in an ME are ambiguous even for humans. Hence, a body box, which includes the main body of each symbol/ME, is used to extract the relation [17]. The reason for employing the body box is that the structural relation is well represented by the positional relations between two body boxes rather than between two bounding boxes. It is inspired from Suzuki et al.'s work [26, 27], which uses different symbol centers for different symbol types. Symbols are classified into four groups: ascendant, descendant, normal, and big symbols. Ascendant or descendant symbols extend above the mean line or below the baseline. Normal symbols fall between the mean and the baselines. Big symbols extend beyond both the mean and the baselines as shown in Fig. 3. For each group, the body box of a symbol is the box which just covers its main body. The body box of an ME is calculated based on the structural relation between its two sub-expressions as shown in Fig. 4.

Four features of D_x, D_y, H and O are extracted as shown in Fig. 5. The features D_x and D_y show the relation between



$$D_x = \frac{C_{x2} - C_{x1}}{w_1} \qquad D_y = \frac{C_{y2} - C_{y1}}{h_1}$$

$$H = \frac{h_2}{h_1} \qquad O = \frac{S_{overlap}}{h_2 \times w_2}$$

Fig. 5 Features used for determining structural relations

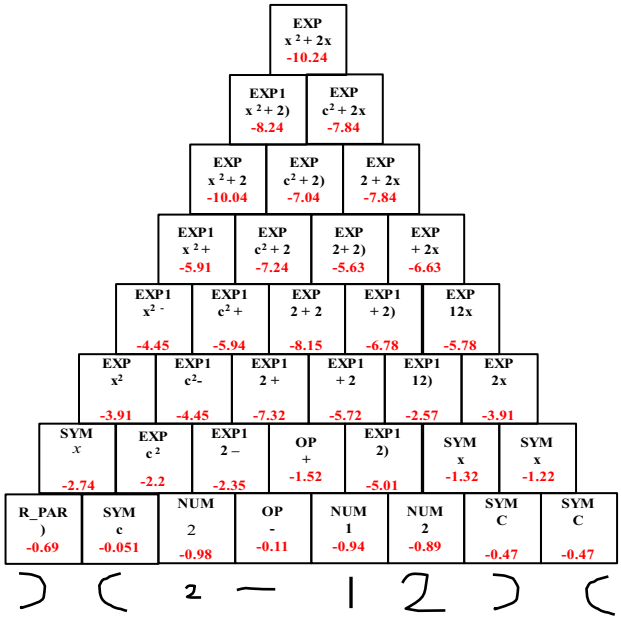


Fig. 6 Result of the parsing table for the expression x^2+2x

the horizontal centers of the two body boxes (C_{x2} and C_{x1}) and the vertical centers of the two body boxes (C_{y2} and C_{y1}), respectively. Feature H shows the ratio of the height of the two body boxes. Feature O shows the ratio of the overlap between the two body boxes. Feature D_x divides the upper, lower and inside relations into Group 1 and the horizontal, superscript, subscript relations into Group 2. Then, the features H and D_y classify the upper, lower and inside relations in Group 1 by an SVM, while the H, D_y and O features classify the horizontal, superscript and subscript relations in Group 2 by another SVM. $S_{overlap}$ is the area of overlap between the two bounding boxes, and S_2 is the area of the second bounding box. $P_{\text{rel}}(r_k|A_kB_k)$ is calculated by transforming the SVM score to a probability.

Parsing algorithm

By employing the stroke order, it is only necessary to consider sub-sequences in the time-ordered sequence of l strokes. Thus, $l(l+1)/2$ sub-sequences exist in total. The algorithm has the following two stages:

Initial stage: The CYK table is initialized by the symbol hypotheses described in Sect. 2.1. The natural logarithm of probability for each initial cell is calculated from the probabilities of symbol hypothesis, symbol recognition and grammar rule as per the following formula:

$$LP = \alpha_1 \ln(P_{\text{sh}}(G_i)) + \alpha_2 \ln(P_{\text{rec}}(S_i|\alpha)) + \alpha_4 \ln\left(P_{\text{gram}}(X \to a)\right) \quad (3)$$

Parsing stage: $X \xrightarrow{r} AB$ production rules are used to reduce two sub-expressions to a non-terminal. Then, from the reduced non-terminal, $X \to A$ production rules are used to reduce it further to another non-terminal. In each cell in the CYK table, an array of nodes is stored. Its size equals the size of the non-terminal symbol set. Each node is a representative of a non-terminal symbol, so a node can be accessed from the cell index (i_1, i_2) and the non-terminal symbol in $O(1)$. In each node, five best candidates for its non-terminal symbol are stored. The candidates for the final result are extracted from the cell $(1, l)$. The natural logarithm of probability for the combination of sub-expressions C_1 and C_2 under the grammar rule $(X \xrightarrow{r} AB)$ is calculated by the following formula:

$$LP = LP(C_1, A) + LP(C_2, B) + \alpha_3 \ln(P_{\text{rel}}(r|AB)) + \alpha_4 \ln\left(P_{\text{gram}}\left(X \xrightarrow{r} AB\right)\right) \quad (4)$$

Figure 6 shows an example of the parsing table for the ME $x^2 + 2x$. In each cell, only the best candidate which contains the non-terminal, the result and the natural logarithm of probability is shown. Blue cells are generated in the initial stage, while black cells are generated in the parsing stage. The complexity of the CYK algorithm for parsing a 2D ME is still $O(l^3|P|)$, like that of the CYK algorithm for parsing a string, and the total number of cells in the parsing table is $l(l+1)/2$ as described in [16]. However, it cannot deal with delay strokes and out-of-order input sequences.

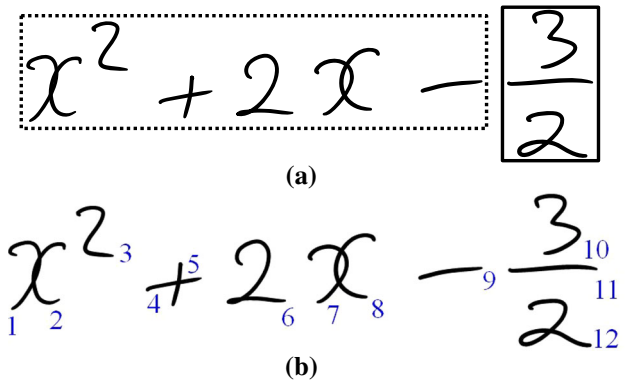


Fig. 7 Horizontal and vertical components and their desired stroke order

Stroke order normalization

Because the above parsing algorithm just recognizes the order of input sequences defined in the grammar, an input sequence must be normalized into this order. The order of input sequences defined in the grammar is called the desired stroke order.

Definition 1 Given a handwritten ME composed of l input strokes O (0_1, 0_2,…,0_l), the desired stroke order of the l input strokes is defined as follows:

- First, the ME is divided into several horizontal components (horizontal, superscript, subscript and inside components) and vertical components (upper and lower components). For each part, the desired stroke order is followed:
- The desired order of a horizontal component is the order from left to right.
- The desired order of a vertical component is defined as follows: The ME is first divided into an upper ME, a vertical symbol and a lower ME. Then, the normal order is from top to bottom (upper ME, vertical symbol, lower ME). Each of the upper and lower MEs is treated as an ME, and its desired order follows this definition recursively.
- Then, the desired stroke order of the ME is constructed by concatenating the desired stroke order of the horizontal and the vertical components from left to right.

Figure 7a shows the horizontal component (as a dotted box) and the vertical component (as a black box) of the ME. Figure 7b shows the desired stroke order of the ME.

The vertically ordered strokes appear in vertically structured MEs such as fractions, integrations, summations and limits. Note that each vertical structure is accompanied by a particular symbol. For example, a fraction is accompanied by a fraction bar, and an integration is accompanied by an integral symbol. The vertical symbols in MEs are *fraction bar*, $\int$, Σ and *lim*. A symbol recognizer is used to detect vertical symbols. For each vertical symbol, the upper and the lower

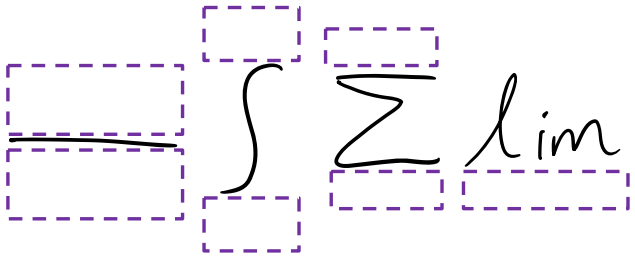


Fig. 8 Upper and lower MEs of vertical symbols

Table 1 Definition of the upper and lower rectangles

	Coordinate of upper and lower rectangles
Upper rectangle	$Top_{upper} = Top_{vertical\ symbol} - r_1 * Averaged\ Height$
	$Left_{upper} = Left_{vertical\ symbol} - r_2 * Averaged\ Width$
	$Bottom_{upper} = Top_{vertical\ symbol}$
	$Right_{upper} = Right_{vertical\ symbol} + r_2 * Averaged\ Width$
Lower rectangle	$Top_{lower} = Bottom_{vertical\ symbol}$
	$Left_{lower} = Left_{vertical\ symbol} - r_1 * Averaged\ Width$
	$Bottom_{lower} = Bottom_{vertical\ symbol} + r_1 * Averaged\ Height$
	$Right_{lower} = Right_{vertical\ symbol} + r_1 * Averaged\ Width$

rectangles, respectively, are supposed to cover its upper ME and lower ME. The upper and the lower rectangles are calculated based on the bounding box of the vertical symbol and the average height and width of the strokes within an ME.

The problem of detecting vertical symbols is their misrecognition. For example, a horizontal stroke in "=" or "5" can be recognized as a fraction bar. To make the detection robust, a verification process is added. The vertical symbol is accepted if it is accompanied by suitable upper and lower MEs. The coordinates of the upper and lower rectangles of the four vertical symbols shown in Fig. 8 are defined in Table 1, where (Top_{upper}, $Left_{upper}$) and ($Bottom_{upper}$, $Right_{upper}$) are the top-left and the bottom-right coordinates of the upper rectangle; (Top_{lower}, $Left_{lower}$) and ($Bottom_{lower}$, $Right_{lower}$) are the top-left and the bottom-right coordinates of the lower rectangle; and ($Top_{vertical\ symbol}$, $Left_{vertical\ symbol}$) and ($Bottom_{vertical\ symbol}$, $Right_{vertical\ symbol}$) are the top-left and the bottom-right coordinates of the bounding box of the vertical symbol, whereas r_1 and r_2 are the parameters for the height and the width of the rectangles. They are trained by GA from the validation set.

In order to recursively decompose an ME whose structural components may overlap, X–Y cut method is modified to determine its horizontal and vertical components as follows. Given an ME, all its vertical components are detected with

Stroke order normalization(ME)
Input: a list of strokes in free writing order.
Output: a list of reordered strokes in desired writing order.
1. S = Get the leftmost unordered vertical symbol
2. Detect the upper and lower MEs of S
  2.1 ME1 = the upper ME of S
  2.2 ME1 = Stroke order normalization (ME1)
  2.3 ME2 = the lower ME of S
  2.4 ME2 = Stroke order normalization (ME2)
3. If ((S = lim or S =Σ) && |ME1| = 0 && |ME2| >0) /*vertical symbol of lim andΣ*/
  3.1 ME3 = append(S, ME2)
  3.2 Mark all strokes in ME3 as ordered strokes
  3.3 ME4 = get un-ordered strokes on the left size of S
  3.4 ME4 = simple reorder in horizontal direction (ME4）
  3.5 Mark all strokes in ME4 as ordered strokes
4. Else if(|ME1| > 0 and |ME2| >0) /* vertical symbol of ∫, Σ, fraction bar */
  4.1 ME3 = append(ME1, S, ME2)
  4.2 Mark all strokes in ME3 as ordered strokes
  4.3 ME4 = get un-ordered strokes on left size of S
  4.4 ME4 = simple reorder in horizontal direction (ME4）
  4.5 Mark all strokes in ME4 as ordered strokes
5. Else /* not vertical symbol */
  5.1 return strokes in S, ME1 and ME2 to the list of strokes
  5.2 ME3 =∅
  5.3 ME4 =∅
6. ME5: get remaining unordered strokes
7. ME5 = Stroke order normalization (ME5）
Return ME4, ME3, ME5

Fig. 9 Stroke order normalization algorithm

their upper and lower MEs. The remaining components are considered to be horizontal components. The desired stroke order of an ME is determined by Definition 1. The proposed method detects the vertical components from left to right. The details of the stroke order normalization algorithm are presented in Fig. 9. Function *append* (ME1, ME2) extends ME1 by appending strokes in ME2 at the end. Function *simple reorder in horizontal* (ME) reorders strokes in ME from left to right by using the left-most coordinate of each stroke. In Steps 1 and 2, the leftmost vertical symbol (S) and its upper/lower MEs (ME1 and ME2) are detected. ME1 and ME2 are reordered recursively. The constraint on the size of an upper ME and a lower ME is employed to avoid misdetection of vertical symbols. In Steps 3 and 4, if S is a vertical symbol, the order is sorted from top to bottom. Unordered strokes on the left side of S are reordered in horizontal direction. In Step 5, if S is not a vertical symbol, strokes in S, ME1 and ME2 are returned to the list of input strokes as unordered strokes, while ME3 and ME4 are set as empty. In Steps 6 and 7, remaining unordered strokes are reordered recursively. The final strokes are sorted as ME4, ME3 and ME5.

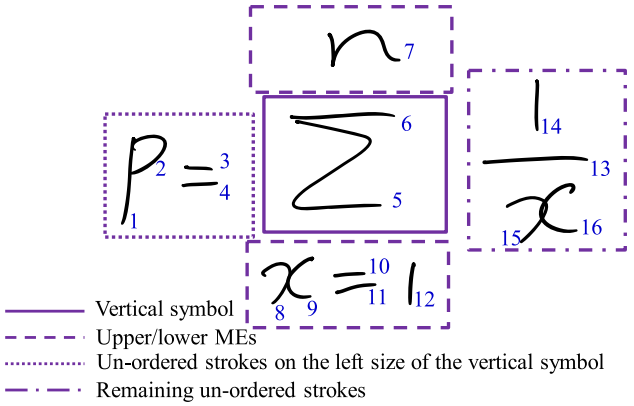


Fig. 10 An example of running the stroke order normalization algorithm

Figure 10 shows an example of running the stroke order normalization algorithm. First, the lower and the upper strokes of the leftmost symbol = are recognized as fraction bars in vertical symbols. However, they are misdetections as they do not have both an upper ME and a lower ME. Second, the next leftmost unordered vertical symbol is determined as Σ. Then, the upper and lower MEs of Σ are determined as n and $x = 1$, respectively. They are reordered recursively by calling the stroke order normalization. The order is sorted from top to bottom and marked as ordered strokes. Third, unordered strokes on the left side of Σ are determined as $P=$. Strokes on the left side of Σ are horizontally ordered strokes because Σ is the leftmost unordered vertical symbol. They are sorted in the horizontal direction and marked as ordered strokes. Finally, unordered strokes of 1, a fraction bar and x are reordered by calling the stroke order normalization recursively. The result of the stroke order normalization is P, $=$, n, Σ, x, $=$, *1*, *1*, *a fraction bar and x*.

Evaluation

The proposed stroke order normalization is evaluated as follows. First, the symbol recognizer is evaluated for detecting vertical symbols. Then, the stroke order normalization is evaluated. There are no stroke order ground truths for MEs, so a small database is selected to provide these ground truths for evaluating the stroke order normalization. Next, integration of the stroke order normalization into a stroke order dependent recognition system is evaluated for the full database. Finally, the integrated system is evaluated on artificially stroke-disordered datasets to demonstrate that it is stroke order free.

Databases

CROHME 2014 was a contest for online handwritten ME recognition algorithms organized at ICHFR 2014. It allowed

Table 2 Performance of symbol recognition on CROHME 2014 test set

	Test set	Vertical symbol set
Num. of symbols	9999	1082
Rec. rate (%)	90.70	99.26

the performance of our proposed system to be compared with others under the same conditions. There were seven participants. The CROHME 2014 database employed for the contest contained 8836 MEs for training and 986 MEs for testing. The number of symbol classes was 101, including many similar symbols such as $\{C, c\}$, $\{X, x, \times\}$, and so on. Isolated symbols from CROHME 2014 were extracted for training and evaluating symbol recognition. It contained 120,341 symbols for training and 9999 symbols for testing.

A validation set was extracted by taking 10% of the training set, and the remainder was used for training.

A small database was selected from the testing set for evaluating the stroke order normalization. It contained 100 MEs (50 MEs were consistent with our normalization while 50 MEs were inconsistent with our normalization). The stroke order ground truths were labeled manually. We call this small database as Small CROHME 2014 test set.

Artificially stroke-disordered datasets were prepared by selecting 25%, 50%, 75% and 100% of MEs from CROHME 2014 test set and randomizing their stroke orders.

All the experiments were performed on an Intel(R) Core(T7) i7 2.93-GHz CPU with 4.0-GB memory.

Experiments

The first experiment evaluated the performance of symbol recognition. Table 2 shows the recognition rate on the testing set and the vertical symbol set extracted from it. The vertical symbols in the CROHME 2014 test set were *fraction bar*, $\int$, Σ, and *limit*. We obtained a recognition rate of 99.26% for the vertical symbol set. This is sufficiently high to justify the reordering method based on the recognition of vertical symbols.

The second experiment evaluated the performance of stroke order normalization on Small CROHME 2014 test set. As mentioned above, the CROHME 2014 test set does not contain stroke order ground truths, and it is tedious to provide it. However, we could extrapolate the performance of the stroke order normalization on the CROHME 2014 test set by estimating it from the performance on the Small CROHME 2014 test set. Table 3 shows the result of stroke order normalization on the Small CROHME 2014 test set. There were some failed cases because of misdetecting vertical symbols and sub-expressions. Figure 11 shows two such examples. A weakness of the stroke order normalization was found to be in the detection of upper and lower MEs when they lie outside their rectangles.

Table 3 Performance of stroke order normalization on Small CROHME 2014 test set

	Rate of stroke order normalization (%)
Consistent with our normalization (50 samples)	96
Inconsistent with our normalization (50 samples)	92

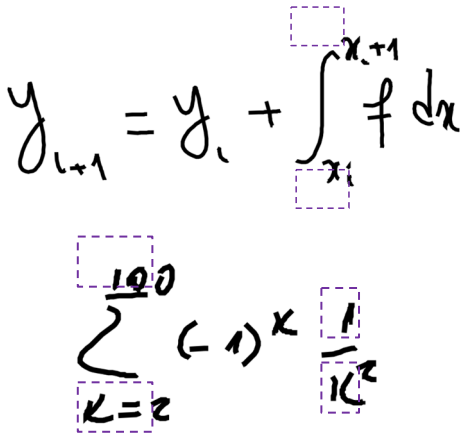


Fig. 11 Two examples of failed stroke order normalization (Upper and lower MEs lie outside the upper and lower rectangles (shown as red boxes)

We checked the stroke order manually to classify the MEs into consistent and inconsistent stroke order groups. CROHME 2014 test set contains 851 MEs which are consistent with our normalization and 135 MEs which are inconsistent. The expected rate of stroke order normalization was calculated with the following formula:

$$\text{ExpectedRate} = \frac{N_{\text{CO}} * \text{Rate}_{\text{CO}} + N_{\text{IO}} * \text{Rate}_{\text{IO}}}{N_{\text{CO}} + N_{\text{IO}}} \quad (5)$$

where N_{CO} and N_{IO} are, respectively, the number of MEs that are consistent and inconsistent with our normalization. Rate_{CO} and Rate_{IO} are, respectively, the rate of stroke order normalization for consistent and inconsistent order MEs. The expected rate of stroke order normalization would be:

$$(851 * 96\% + 135 * 92\%)/(851 + 135) = 95.89\%$$

The third experiment evaluated the performance of the recognition system incorporating the stroke order normalization (System IX) on the CROHME 2014 test set in order to compare it with the stroke order dependent system (System VIII), and the three best systems that participated in the CROHME 2014 competition. System I was developed

Table 4 Performance of the stroke order normalization with the recognition system on CROHME 2014 test set (%)

System	Measurement			
	Sym Seg	Sym Seg + Rec	Rel Tree	Exp Rec
System I	93.31	86.59	84.23	37.22
System III	98.42	93.91	94.26	62.68
System VII	89.43	76.53	71.77	26.06
System VIII	89.85	83.21	71.55	35.80
System IX	91.90	84.86	79.75	37.63

by Alvaro et al., in which symbol recognition, segmentation and structural analysis were globally determined using the Stochastic Context Free Grammar. System III was developed by Myscript, which handled segmentation, recognition and interpretation concurrently in order to produce the best candidates. This system also included a statistical language model to evaluate the contextual probabilities between symbols in the equation. It was trained on about 30,000 additional handwritten ME samples collected from writers in different countries. System VII was developed by Mouchere et al., and it simultaneously optimized expression segmentation, symbol recognition, and 2D structure recognition under the restriction of an expression grammar. During the training phase, symbol hypotheses were generated without using a language model. The dynamic programming algorithm found the best segmentation and recognition of the input. The classifier learned both the correct and incorrect segmentations. Finally, a bottom-up parsing process combined sub-expressions into a larger expression under 2D grammar.

The four factors measured in the evaluation test were: *Sym Seg* as symbol segmentation, *Sym Seg + Rec* as symbol segmentation and recognition, *Rel Tree* as structural analysis (termed "relation tree"), and *Exp Rec* as expression recognition as listed in Table 4. With the reordering method, our new system was found to improve all the measures and it outperformed all other systems that use only CROHME 2014 for training (Except System III, which used about 3.5 times more training patterns than the CROHME training set, and hundreds of thousands of equations for estimating statistical language model). The recognition rate was found to improve from 35.80 to 37.63% although this difference was not verified by paired t-test with $P < 0.001$. The competition details are described in [11]. The average processing time for stroke order normalization was only 1.42 ms per ME (the average number of strokes is 14). This result shows that the time cost of reordering is negligible.

Table 5 shows the recognition rates of the baseline system and the stroke normalization integrated system on two sub-datasets (consistent and inconsistent with our normalization). The recognition rate on the consistent dataset is almost same (37.37% vs 37.13%) while that on the inconsistent dataset is largely improved (25.93% vs 40.74%).

Table 5 Performance of the stroke order dependent and the stroke order normalization integrated systems on consistent and inconsistent stroke order sub-datasets (%)

System	System Exp Rec	
	Consistent with our normalization	Inconsistent with our normalization
System VIII	37.37	37.13
System IX	25.93	40.74

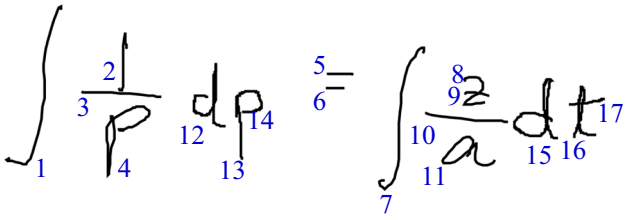
Table 6 Performance of the stroke order dependent and the stroke order normalization integrated systems on artificially stroke-disordered MEs (%)

System	Ratio of stroke-disordered MEs			
	25%	50%	75%	100%
System VIII	27.59	18.76	10.24	0.00
System IX	34.79	34.58	33.98	33.57

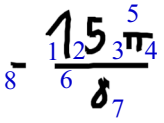
The final experiment was to confirm that the integrated system is stroke order free on the artificially stroke-disordered datasets. Table 6 shows the expression recognition rate for four levels of artificially stroke-disordered datasets. The recognition rate of the stroke order dependent system (System VIII) fell rapidly to 0% as the proportion of disordered MEs increased, while the recognition rate of the stroke order normalization integrated system (System IX) fell only a little. Since symbol hypotheses are generated by combining a sequence of successive strokes, the system cannot detect a vertical symbol if stroke order of the vertical symbol is not successive. With the stroke order normalization, the recognition rate was maintained, which shows that the integrated system is almost stroke order free.

The paired t test was employed to verify the significance of the performance. The integrated system was found to be significantly better than the stroke order dependent system on each of the four levels of artificially stroke-disordered datasets with $P < 0.001$.

Figure 12 shows two examples recognized correctly by the stroke order normalization integrated system, but misrecognized by the stroke order dependent system in CROHME 2014 test set. The stroke order of each ME is shown under it. The first symbol "*d*," the second symbol "*p*" in Fig. 9a) and the symbol "−" in Fig. 9b) are written in unusual stroke orders. The stroke order dependent system cannot recognize them, but the stroke order normalization integrated system can recognize these MEs.



(a) Stroke order: ∫, *1*, *fraction bar*, *p*, =, ∫, *z*, *fraction bar*, *a*, *d*, *p*, *d*, *t*



(b) Stroke order: *1*, *5*, *π*, *fraction bar*, *8*, -

Fig. 12 MEs recognized correctly by the reordering method

Conclusion

We presented here a method for normalizing the stroke order of online handwritten MEs. Our method detects vertical symbols, and upper and lower MEs to divide input strokes into vertical and horizontal strokes. Unordered strokes on the left side of vertical symbols are reordered as horizontal-order strokes. The upper ME, the lower ME and the remaining strokes are reordered recursively. It is a simple but effective approach to recognizing unexpected variations in the stroke order. This stroke order normalization is integrated into a stroke order dependent recognition system. Experiments on the CROHME 2014 database show that the stroke order normalization is able to sort-disordered strokes with 92% accuracy with a small side effect, and it is expected to achieve an accuracy of 95.89% in reordering and the stroke order normalization integrated system improves the recognition rate from 35.80 to 37.63%. The integrated system is verified as stroke order free in the artificially stroke-disordered datasets. In addition, the processing time of the reordering method is negligible.

References

1. Chan, K., Yeung, D.: Mathematical expression recognition: a survey. Int. J. Doc. Anal. Recognit. **3**, 3–15 (2000)
2. Zanibbi, R., Blostein, D.: Recognition and retrieval of mathematical expressions. Int. J. Doc. Anal. Recognit. **15**, 331–357 (2012)
3. Mouchere, H., Viard-Gaudin, C., Zanibbi, R., Garain, U.: ICFHR 2014 competition on recognition of on-line handwritten mathematical expressions (CROHME 2014). In: International Conference Frontiers in Handwriting Recognition, pp. 791–796 (2014)
4. Lehmberg, S., Winkler, H.J., Lang, M.: A soft-decision approach for symbol segmentation within handwritten mathematical expressions, International conference on acoustics, speech, and signal processing, vol. 6, pp. 3434–3437, Atlanta (1996)
5. Toyozumi, K., et al.: A study of symbol segmentation method for handwritten mathematical formula recognition using mathematical structure information. In: International Conference on Pattern Recognition, vol. 2, pp. 630–633, Cambridge (2004)
6. Hu, L., Zanibbi, R.: Segmenting handwritten math symbols using adaboost and multi-scale shape context features. In: International Conference on Document Analysis and Recognition, pp. 1180–1184, Washington (2013)
7. MacLean, S., Labahn, G.: Elastic matching in linear time and constant space. In: IAPR Workshop on Document Analysis Systems (2010)
8. Hu, L., Zanibbi, R.: HMM-based recognition of online handwritten mathematical symbols using segmental K-means initialization and a modified pen-up/down feature. In: International Conference on Document Analysis and Recognition, pp. 457–462, Beijing (2011)
9. Alvaro, F., Sanchez, J.A., Benedi, J.M.: Classification of on-line mathematical symbols with hybrid features and recurrent neural networks, International Conference on Document Analysis and Recognition, pp. 1012–1016, Washington (2013)
10. Davila, K.M., Ludi, S., Zanibbi R.: Using off-line features and synthetic data for on-line handwritten math symbol recognition, International Conference on Frontiers in Handwriting Recognition, pp. 323–328, Crete (2014)
11. Garain, U., Chaudhuri, B.B.: Recognition of online handwritten mathematical expressions. IEEE Trans. Syst. Man Cybern. Part B Cybern. **34**, 2366–2376 (2004)
12. Alvaro, F., Sanchez, J.A., Benedi, J.M.: Offline features for classifying handwritten math symbols with recurrent neural networks. In: International Conference on Pattern Recognition, pp. 2944–2949, Stockholm (2014)
13. Nguyen, H.D., Le, A.D., Nakagawa, M.: Deep neural network for recognizing online handwritten mathematical symbols. In: IAPR Asian Conference on Pattern Recognition, pp. 121–125 (2015)
14. Nguyen, H.D., Le, A.D., Nakagawa, M.: Recognition of online handwritten math symbols using deep neural networks, IEICE Transactions on Information and Systems, vol. E99.D, pp. 3110–3118 (2016)
15. Yamamoto, R., Sako, S., Nishimoto, T., Sagayama, S.: Online recognition of handwritten mathematical expressions based on stroke-based stochastic context-free grammar. In: International Workshop on Frontiers in Handwriting Recognition, pp. 249–254, La Baule, France (2006)
16. Simistira, F., Katsouros, V., Carayannis, G.: Recognition of online handwritten mathematical formulas using probabilistic SVMs and stochastic context free grammars. Pattern Recognit. Lett. **53**, 85–92 (2015)
17. Le, A.D., Nakagawa, M.: A system for recognizing online handwritten mathematical expressions by using improved structural analysis. Int. J. Doc. Anal. Recognit. **19**, 305–319 (2016)
18. Le, A.D., Phan, T.V., Nakagawa, M.: A system for recognizing online handwritten mathematical expressions and improvement of structure analysis. In: IAPR Workshop on Document Analysis Systems, pp. 51–55 (2014)
19. MacLean, S., Labahn, G.: A new approach for recognizing handwritten mathematics using relational grammars and fuzzy sets. Int. J. Doc. Anal. Recognit. **16**, 139–163 (2013)
20. Alvaro, F., Sanchez, J., Benedi, J.: Recognition of on-line handwritten mathematical expressions using 2D stochastic context-free grammars and hidden markov models. Pattern Recognit. Lett. **35**, 58–67 (2014)
21. Lee, H.-J., Wang Lee, J.-S.: Design of a mathematical expression understanding system. Pattern Recognit. Lett. **18**, 289–298 (1997)
22. Zanibbi, R., Blostein, D., Cordy, J.R.: Recognizing mathematical expressions using tree transformation. IEEE Trans. Pattern Anal. Mach. Intell. **24**(11), 1455–1467 (2002)

23. Nagy, G., Seth, S.: Hierarchical representation of optically scanned documents. In: International Conference on Pattern Recognition, pp. 347–349, Montreal, Canada (1984)
24. Meunier, J.: Optimized *XY*-cut for determining a page reading order. In: International Conference on Document Analysis and Recognition, pp. 347–351, Seoul, Korea (2005)
25. Le, A.D., Nguyen, H.D., Nakagawa, M.: Modified *X*–*Y* cut for reordering strokes of online handwritten mathematical expressions. In: IAPR Workshop on Document Analysis Systems, pp. 233–238, Greece (2016)
26. Eto, Y., Suzuki, M.: Mathematical formula recognition using virtual link network. In: International Conference on Document Analysis and Recognition, pp. 430–437, USA (2001)
27. Aly, W., Uchida, S., Suzuki, M.: Identifying subscripts and superscripts in mathematical documents. Math. Comput. Sci. **2**, 195–209 (2008)
28. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y.: Show, attend and tell: neural image caption generation with visual attention. In: International Conference on Machine Learning, vol. 37, pp 2048–2057 (2015)
29. Luong, T., Pham, H., Manning, C.: Effective approaches to attention-based neural machine translation. In: The 2015 Conference on Empirical Methods in Natural Language Processing (2015)
30. Deng, Y., Kanervisto, A., Ling, J., Rush, A.: Image-to-Markup Generation with Coarse-to-Fine Attention, pp. 980–989. ICML, Pittsburgh (2017)
31. Zhang, J., Du, J., Zhang, S., Liu, D., Hu, Y., Hu, J., Wei, S., Dai, L.: Watch, attend and parse: an end-to-end neural network based approach to handwritten mathematical expression recognition. Pattern Recognit. **71**, 196–206 (2017)
32. Le, A.D., Nakagawa, M.: Training an end-to-end system for handwritten mathematical expression recognition by generated patterns. In: International Conference on Document Analysis and Recognition, pp. 1056–1061 (2017)