Online Handwritten Mathematical Expressions Recognition by Merging Multiple 1D Interpretations

Ting ZHANG, Harold MOUCHERE, Christian VIARD-GAUDIN
IRCCyN/IVC - UMR CNRS 6597
Université de Nantes, France
ting.zhang1@etu.univ-nantes.fr, {harold.mouchere, christian.viard-gaudin}@univ-nantes.fr

***Abstract*—In this work, we propose to recognize handwritten mathematical expressions by merging multiple 1D sequences of labels produced by a sequence labeler. The proposed solution aims at rebuilding a 2D expression from several 1D labeled paths. An online math expression is a sequence of strokes which is later used to build a graph considering both temporal and spatial orders among these strokes. In this graph, node corresponds to stroke and edge denotes the relationship between a pair of strokes. Next, we select 1D paths from the built graph with the expectation that these paths could catch all the strokes and the relationships between pairs of strokes. As an advanced and strong sequence classifier, BLSTM networks are adopted to label the selected 1D paths. We set different weights to these 1D labeled paths and then merge them to rebuild a label graph. After that, an additional post-process will be performed to complete the edges automatically. We test the proposed solution and compare the results to the state of art in online math expression recognition domain.**

***Keywords*-Handwritten mathematical expression; online handwriting; 1D paths; label graph; BLSTM.**

I. Introduction

Handwritten Mathematical Expressions (MEs) recognition is a challenging problem in the field of pattern recognition because it involves both the recognition of a large set of symbol classes and symbol arrangements in a two-dimensional (2D) layout. According to the different formats of sources of handwritten ME, it can be divided into offline (images) recognition and online (inputs written on tablets) recognition. We focus on online MEs recognition in this paper. An online ME consists of one or more strokes which are sequences of points sampled from the trajectory of the writing tool between a pen-down and a pen-up.

This research domain has been boosted by the Competition on Recognition of Handwritten Mathematical Expressions (CROHME) [6], which began as part of the International Conference on Document Analysis and Recognition (ICDAR) in 2011. It provides a platform for researchers to test their methods and compare them, and then facilitate the progress in this field. It attracts increasing participation of research groups from all over the world. In this work, the provided data and tools will be used and results will be compared to participants.

Research on the recognition of math notation began in the 1960s and detailed surveys are provided in [2], [3]. Generally, ME recognition involves three interdependent tasks [3]: (1) Symbol segmentation, which consists in grouping strokes that belong to the same symbol; (2) symbol recognition, the task of labeling the symbol to assign each of them a symbol class; (3) structural analysis, its goal is to identify spatial relations between symbols and with the help of a grammar to produce a mathematical interpretation. These three problems can be solved sequentially or jointly. In the first case, the tasks are processed sequentially which means the error from the previous stage will be propagated to the next stage. The alternative solutions run these three tasks concurrently and aim at making a final decision of the formula using global information [11], [12], [13]. These approaches seem reasonable and exhibit out performances as these three problems are highly interdependent by nature. Human beings recognize symbols with the help of structure, and vice versa.

Different from the above mentioned solutions, we propose to rebuild a 2D expression with some 1D paths from a new perspective. It is possible to describe a ME using a primitive Label Graph (LG) in which nodes represent strokes and labels on the edges encode either segmentation information or layout information. The detailed description will be found in Section II. Given an handwritten expression which is a sequence of strokes, first, we build a graph which catches as many as possible edges existing in the ground truth LG, at the same time limiting the quantity of unnecessary edges. After that, 1D paths will be selected from this graph. We also expect the set of 1D paths could include all the strokes and the relationships between stroke pairs existing in both the graph and the ground truth. The advanced recurrent neural network BLSTM has been proved to outperform other classifiers in tasks like handwritten text recognition [7] and handwritten math symbol recognition [4], [5]. This strong sequence classifier is adopted to label the selected 1-D paths. We set different weights to these 1D labeled paths and then merge them to rebuild a label graph. After that, an additional post-process will be performed to complete the edges automatically. Finally, a 2D expression is rebuilt from the merged path. Grammar constraints is not included in our solution yet.

2167-6445/16 $31.00 © 2016 IEEE
DOI 10.1109/ICFHR.2016.42

The remainder of the paper is organized as follows. Section II introduces building a graph from handwritten ME. This section includes the conception of primitive Label Graph, the solution of generating a graph and selecting paths. Section III describes how to apply BLSTM to label 1D paths. Section IV presents recognition process, specifically make decisions at stroke level and then merge the results of 1D paths. Features for on-line ME are showed in Section V. In Section VI, we report the experiments and results. Finally, conclusions and future works are presented.

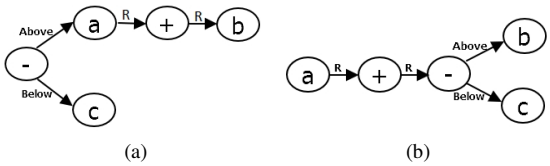
II. Build a graph

An input mathematical expression consists of one or more strokes. The sequence of strokes in an expression can be described as $S = \{s_1, ..., s_n\}$. For $i < j$, we assume s_i has been entered before s_j. In effect, S is the path following time order in a primitive (stroke) Label Graph (LG) which can model a ME. In this section, we first introduce how to represent an expression with LG and then how to build a graph from S.

A. Primitive Label Graph

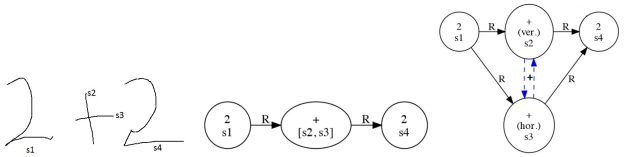
Structures can be depicted at three different levels: symbolic, object and primitive [10]. In the case of handwritten ME, the corresponding levels are expression, symbol and stroke. It is possible to describe a ME at the symbol level using a Symbol Relation Tree (SRT), while if needed to go down at the stroke level, a primitive Label Graph can be derived from the SRT. In SRT, nodes represent symbols and labels on the edges indicate the relationships between symbols. Examples can be found in Figure 1. A LG contains the same information as a SRT but at the stroke level. In LG, nodes represent strokes, while labels on the edges encode either segmentation information or layout information. As relationships are defined at the level of symbols, all strokes in a symbol have the same input and output edges. Consider the simple expression '2 + 2' written using four strokes (two strokes for '+') in Figure 2a. The corresponding SRT and LG are shown in Figure 2b and Figure 2c respectively. As Figure 2c illustrates, nodes are labeled with the class of the symbol. A dashed edge corresponds to segmentation information; it indicates that a pair of strokes belongs to the same symbol. In this case, the edge label is the same as the common symbol label. On the other hand, the non-dashed edges define spatial relationships between nodes and are labeled with one of the different possible relationships between symbols.

The spatial relationships as defined in the CROHME competition are: $Right$, $Above$, $Below$, $Inside$ (for square root), $Superscript$, $Subscript$. For the case of nth-Roots, like $\sqrt[3]{x}$, we define that the symbol 3 is $Above$ the square root and x is $Inside$ the square root. The limits of an integral and summation are designated as $Above$ or $Superscript$ and $Below$ or $Subscript$ depending on the actual position of the bounds. In addition, the label '_' is used to denote that there is no relation between two symbols.



(a) (b)

Figure 1. (a) the symbol relation tree of $\frac{a+b}{c}$; (b) the symbol relation tree of $a + \frac{b}{c}$. 'R' is for left-right relationship.



(a) (b) (c)

Figure 2. (a) '2 + 2' written with four strokes; (b) the symbol relation tree of '2 + 2'; (c) the label graph of '2 + 2'. The four strokes are indicated as s1, s2, s3, s4 in writing order. (ver.) and (hor.) are added to differentiate the vertical and the horizontal strokes for '+'. 'R' is for left-right relationship.

B. Build a graph G from S

As explained above, the input data is available as a sequence of strokes S from which we would like to obtain the final LG graph describing unambiguously the ME. In a first step, we will derive an intermediate graph G, where each node is a stroke and edges are added according to temporal or spatial properties. In order to clearly understand how to build the G graph from S, we first introduce some definitions here.

Definition 2.1 The distance between two strokes s_i and s_j can be defined as the Euclidean distance between their closest points.

$$dist(s_i, s_j) = \min_{p \in s_i, q \in s_j} \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2} \quad (1)$$

Definition 2.2 A stroke s_i is considered visible from stroke s_j if the straight line between their closest points does not cross the bounding box of any other stroke s_k.

Definition 2.3 We define five positional relations between two strokes s_i and s_j. The positional relation between them is simplified as the positional relation between the centers of their bounding boxes.

As shown in Figure 3, five regions ($Above$, $Below$, Sup, Sub and $Right$) are defined for stroke s_i. If the center of bounding box of s_j is in one of these five regions, for example $Right$ region, we can say s_j is in the $Right$ direction of s_i.

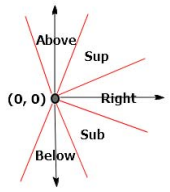


Figure 3. Five directions for a stroke s_i. Point (0, 0) is the center of bounding box of s_i. The angle of each region is $\frac{\pi}{4}$.

Definition 2.4 Let G be a directed graph in which each node corresponds to a stroke and edges are added according to the following criteria in succession.

We defined for each stroke s_i (i from 1 to n-1):

- the set of crossing strokes $S_{cro(i)} = \{s_{cro1}, s_{cro2}, ...\}$ from $\{s_{i+1}, ..., s_n\}$.
- the set of closest stroke $S_{clo(i)} = \{s_{clo}\}$ from $\{s_{i+1}, ..., s_n\} - S_{cro(i)}$.

For stroke s_i (i from 1 to n):

- the set $S_{vis(i)}$ of the visible closest strokes in five directions respectively if exist in $\{s_1, ..., s_n\} - \{s_i\} \bigcup S_{cro(i)} \bigcup S_{clo(i)}$. Here, the closeness of two strokes is decided by the distance between the centers of their bounding boxes, differently from **definition 2.2**.

Edges from s_i to the $S_{cro(i)} \bigcup S_{clo(i)} \bigcup S_{vis(i)}$ will be added to G. Finally, we check if the edge from s_i to s_{i+1} (i from 1 to n-1) exists in G. If not, then add this edge to G to ensure that the path covering the sequence of strokes in the time order is included in G.

An example is presented in Figure 4. Mathematical expression $\frac{d}{dx}a^x$ is written with 8 strokes. From the sequence of 8 strokes, the graph shown in Figure 4c is generated following the above mentioned criteria. Comparing to the ground truth, we can see all the edges in Figure 4b are included in the regenerated graph except edges from strokes 4 to 3 and from strokes 7 to 6 (dash edges). This flaw can be overcame as long as strokes 3 and 4 and edge from strokes 3 to 4 are correctly recognized. Because if strokes 3 and 4 are recognized as the same symbol, edge from strokes 4 to 3 can be completed automatically. Also, edge from strokes 7 to 6 can be added. At the same time, Figure 4c includes some unnecessary edges (dot edges) when matching to the ground truth. The graph we build should include the ground truth edges as many as possible, simultaneously reduce the number of unnecessary edges.

C. Select paths from G

The final aim of our work is to rebuild the LG of 2D expression. We propose to perform it by merging several paths from G. These paths should cover all the nodes and the edges of the ground truth LG (at least the edges of the ground truth SRT). With the correctly recognized node and edge labels, we have the possibility to rebuild a correct 2D expression. A path in G can be defined as $\Phi_i = (n_0, n_1, n_2, ..., n_e)$, where n_0 is the starting node and n_e is the end node. The node set of Φ_i is $n(\Phi_i) = \{n_0, n_1, n_2, ..., n_e\}$ and the edge set of Φ_i is $e(\Phi_i) = \{n_0 \rightarrow n_1, n_1 \rightarrow n_2, ..., n_{e-1} \rightarrow n_e\}$.

Two types of paths are selected in this work: $time$ path and $random$ path. As described in [14], $time$ path covers all the nodes but could miss some edges; $random$ path is used to catch more edges. Although this combination is not the best solution, but the simplest and the most intuitive one. A better solution will be explored in future. $time$ path starts from the first input stroke and ends with the last input stroke following the time order. For example, in Figure 4c, the $time$ path is (0, 1, 2, 3, 4, 5, 6, 7). Then, we consider several additional random paths. To ensure a good coverage of the graph we guide the random selection choosing the less visited nodes and edges. One $random$ path can be (1, 5, 6, 7).

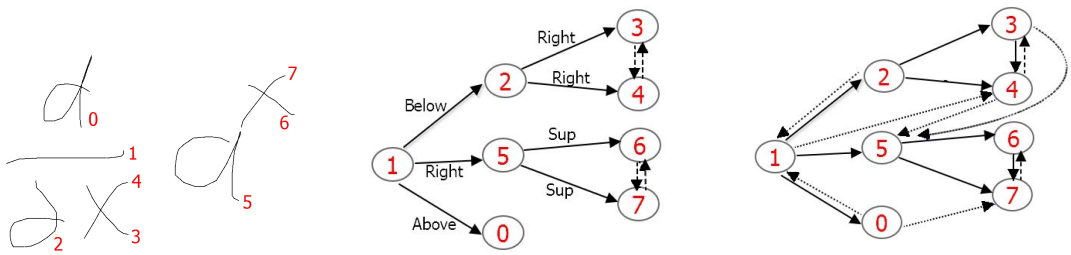
III. BLSTM Inputs

Followed, a strong sequence labeler should be chose to label $time$ or $random$ paths. The advanced recurrent neural network BLSTM combines the advantages of BRNN and LSTM and thus it offers access to long range context in two directions. In the nature of things, this advanced architecture outperforms other models in several tasks [7], [4], [5].

In order to visualise the RNN and understand how it operates, we unfolded it along the input sequence and illustrated a part of the unfolded one in Figure 5. In order to be easier to illustrate, only one forward layer is presented. Here, each node represents a layer of network units at a single time-step. The input at a single time-step is a feature vector; for the whole process, the input is a time sequence of feature vectors. The output at a single time-step is a probability vector of which each element is the probability of belonging to some class; the overall output is a sequence of probability vectors. The same weights (w1, w2, w3) are reused at every time-step.

To feed the inputs of the BLSTM, it is important to scan as well the points belonging to the strokes themselves (on-paper points) and also the points separating one stroke from the next one (in-air points). We expect that the on-paper points will be labeled with corresponding symbol labels and that the in-air points will be assigned with one of the possible edge labels. Thus, besides re-sampling points from strokes, we also re-sample points from the straight line which links two strokes, as can be seen in Figure 6. In the rest of this paper, $strokeD$ and $strokeU$ are used to indicate a re-sampled pen-down stroke and a re-sampled pen-up stroke for convenience.

Given each path, we first re-sampled points both from visible strokes and between two successive visible strokes in the path. 1D unlabeled sequence can be described as $\{strokeD_1, strokeU_2, strokeD_3, strokeU_4, ..., strokeD_K\}$ with K is the number of re-sampled strokes. Note that if s



(a) (b) (c)

Figure 4. (a) $\frac{d}{dx}a^x$ is written with 8 strokes; (b) the LG from ground truth; (c) the graph built using our method, dot edges are unnecessary and dash edges are missed compared to the ground truth.

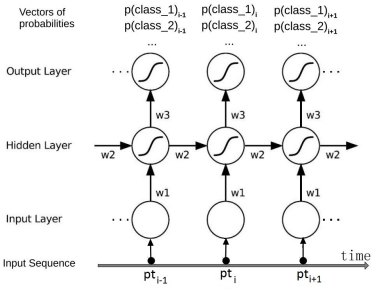


Figure 5. An unfolded single-directional recurrent network. For each time-step, the input is feature vector extracted from one point; the output is the probabilities of this point belonging to different classes.

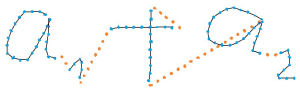


Figure 6. The illustration of on-paper points (blue) and in-air points (red) in $time$ path.

is the number of visible strokes in this path, $K = 2 * s - 1$. Ground-truth of each point is required for BLSTM training process. The labels of the points from $strokeD_i$ should be assigned with the label of the corresponding node in LG; The label of the points from $strokeU_i$ should be assigned with the label of the corresponding edge in LG. If no corresponding edge exists, the label $NoRelation$ will be defined as '_'.

In our case, at a time-step, the input to the BLSTM is the feature vector extracted from one point; the output is the probabilities of this point belonging to different classes.

IV. Recognition

As Figure 5 shows, since the RNN outputs local classifications for each point, some form of post-processing need to be done. The connectionist temporal classification (CTC) [9] technology is a good choice for sequence transcription tasks. It outputs the probabilities of the complete sequences directly but do not provide the alignment between the inputs and the target labels. In our case, we need the labels of strokes to obtain a LG. Thus, we propose to make decisions on stroke ($strokeD$ or $strokeU$) level instead of sequence level (as CTC) or point level. As BLSTM can access the contextual information from two directions in an unlimited range, the output probability of each point is not a local decision. With the same method taken by Alex Graves for isolated handwritten digits recognition using a multidimensional RNN with LSTM hidden layers in [8], we choose for each stroke the label which has the highest cumulative probability over the entire stroke. Suppose that p_{ij} is the probability of outputting the i_{th} label at the j_{th}time step. Then, for each stroke k, the probability of outputting the i_{th} label can be computed as $ps_{ik} = \sum_{j=1}^{|s_k|} p_{ij}$, where $|s_k|$ is the number of points of stroke k. We sort the normalized ps_{ik} and only keep the top n probable labels with the accumulative probability ≥ 0.8. Note that n is maximum 3 even though the accumulative probability of top 3 labels is not up to 0.8.

Each stroke belongs at least to one path, but possibly to several paths. Hence, several recognition results can be available for a single stroke. At this stage, we propose to compute the probability $P_s(l)$ to assign the label l to the stroke s by summing all path Φ_i with the formula:

$$P_s(l) = \frac{\sum_{\Phi_i} W_{\Phi_i} * P_{\Phi_i,s}(l)\mathbb{1}_S(s)\mathbb{1}_{label(s,\Phi_i)}(l)}{\sum_{\Phi_i} W_{\Phi_i}\mathbb{1}_S(s)} \quad (2)$$

$$S = n(\Phi_i) \cup e(\Phi_i) \quad (3)$$

$$\mathbb{1}_S(i) = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

W_{Φ_i} is the weight for path Φ_i and $label(s, \Phi_i)$ is the set of candidate labels for stroke s from path Φ_i, $1 \leq |label(s, \Phi_i)| \leq 3$. If stroke s exists in path Φ_i, but $l \notin label(s, \Phi_i)$, in this case, $P_{\Phi_i,s}(l)$ is 0. The classifier answers that there is no possibility to output label l for stroke s from path Φ_i. We still add W_{Φ_i} into the normalized factor of $P_s(l)$. If stroke s does not exist in path Φ_i, the classifier's answer to stroke s is unknown. And we should not take into account this path Φ_i. Thus, W_{Φ_i} will not be added into the

normalized factor of $P_s(l)$. After normalization, the label with the maximum probability is selected for each stroke.

Finally, post-processing should be done in order to rebuild a valid LG, i.e. adding edges. We first look for the segments (symbols) using connected component analysis: a connected component where nodes and edges have the same label is a symbol. With regards to the relationship between two symbols, we choose the label having the maximum accumulative probability among the edges between two symbols. Then, according to the rule that all strokes in a symbol have the same input and output edges and that double-direction edges represent the segments, some missing edges can be completed automatically.

V. Experiments

A stroke is a sequence of points sampled from the trajectory of a writing tool between a pen-down and a pen-up at a fixed interval of time. Then an additional re-sampling is performed with a fixed spatial step to get rid of the writing speed. The number of re-sampling points depends on the size of expression. For each path, we re-sample with $10 \times (length/avrdiagonal)$ points. Here, $length$ refers to the length of all the strokes in the path (including distance between successive strokes) and $avrdiagonal$ refers to the average diagonal of the bounding boxes of all the strokes in an expression.

Subsequently, we compute five features per point, which are quite close to the state of art [4], [11]. For every point $p(x, y)$ we obtained 5 features [sinθ, cosθ, sinϕ, cosϕ, PenUD]. The detailed description can be seen in [14].

We use the RNNLIB library[1] for training a BLSTM model. For each training process, the network having the best classification error on validation data set is saved. Then, we test this network on the test data set. The Label Graph Evaluation library (LgEval) [1] is adopted to evaluate the recognition output. The detailed description of network architecture and configuration can be found in [14]. This configuration has obtained good results in both handwritten text recognition [7] and handwritten math symbol classification [4], [5].

With regards to data set, we use CROHME 2014 train set (8834 expressions) for training, CROHME 2013 test set (670 expressions) for validation and CROHME 2014 test set (983 expressions) for test. validation The output layer size in this experiment is 108 (101 symbol classes + 6 relationships + $NoRelation$). For each expression, we extract the $time$ path and 6 $random$ paths. Totally, 3 classifiers are trained with only $time$ path, only 6 $random$ paths and $time$ path plus 6 $random$ paths respectively in order to evaluate the effect of the training set content. These classifiers are denoted as CLS_T, CLS_R and CLS_{T+R}. We use these different classifiers to label the different types of paths extracted from the test set as presented in Table I. In exp.1, only the labeled $time$ path is used to rebuild a 2D expression. For other experiments, $time$ path and $random$ paths are merged to obtain a final result. The weight of $time$ path is set to 0.4 and each random path is 0.1.

[1]Graves A. RNNLIB: A recurrent neural network library for sequence learning problems. http://sourceforge.net/projects/rnnl/.

Table I
The strategies to label different types of paths.

exp.	Path Type	
	Time	Random
1	CLS_T	
2	CLS_T	CLS_T
3	CLS_T	CLS_R
4	CLS_{T+R}	CLS_{T+R}

Table II
The symbol level evaluation results on CROHME 2014 test set, including the experiment results in this work and CROHME 2014 participant results (Top 4 by recall of Segments).

exp.	Segments (%)		Seg + Class (%)		Tree Rels. (%)	
	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.
1	92.30	85.15	83.70	77.22	58.40	74.27
2	92.32	85.28	84.37	77.94	67.79	58.34
3	92.77	85.99	85.17	78.95	67.79	67.33
4	91.51	83.48	82.96	75.67	66.95	62.06
system	CROHME 2014 participant results (Top 4)					
III	98.42	98.13	93.91	93.63	94.26	94.01
I	93.31	90.72	86.59	84.18	84.23	81.96
VII	89.43	86.13	76.53	73.71	71.77	71.65
V	88.23	84.20	78.45	74.87	61.38	72.70

Table III
The expression level evaluation results on CROHME 2014 test set, including the experiment results in this work and CROHME 2014 participant results (Top 2)

exp.	correct (%)	$\leq$ 1 error	$\leq$ 2 errors	$\leq$ 3 errors
1	11.80	19.33	26.55	31.43
2	8.55	16.89	23.40	29.91
3	13.02	22.48	30.21	35.71
4	11.19	19.13	26.04	31.13
system	CROHME 2014 participant results (Top 2)			
III	62.68	72.31	75.15	76.88
I	37.22	44.22	47.26	50.20

The evaluation results on symbol level are provided in Table II including recall (Rec.) and precision (Prec.) rates for symbol segmentation (Segments), symbol segmentation and recognition (Seg+Class), spatial relationship classification (Tree Rels.). A correct spatial relationship between two symbols requires that both symbols are correctly segmented and with the right relationship label. As presented, the results for Segments and Seg+Class do not show a big difference among 4 experiments. It explains that $time$ path is enough to give good results and $random$ paths contributes little. With regard to Tree Rels., Rec. of exp.(2 3 4) is improved compared to exp.1 because $random$ paths catch some ground truth edges which are missed by $time$ path; but Prec. rate declines which means that $random$ paths also cover

some edges which are not in ground truth LG. Unexpectedly, these excess edges are not labeled as $NoRelation$. Among 4 experiments, exp.3 outperforms others for all the items. Thus, it is a better strategy to use CLS_T for labeling $time$ path and use CLS_R for $random$ path. Our results are comparable to the results of CROHME 2014 because the same training and testing data sets are used. The second part of Table II gives the symbol level evaluation results of the top 4 participants in CROHME 2014 sorting by the recall rate for correct symbol segmentation. The best Rec. of Segments and Seg+Class reported by CROHME 2014 are 98.42% and 93.91% respectively. Ours are 92.77% and 85.17%, both ranked 3 out of 8 systems (7 participants in CROHME 2014). Our solution presents competitive results on symbol recognition task and segmentation task, but not on relationship detection and recognition task. However, our proposal still shows promising results because Tree Rels. recognition rate will be improved as long as a high quality graph is built and a good path selecting method is adopted. It is also a part of our future work.

Table III shows the recognition rates at the global expression level with no error, and with at most one to three errors in the labels of LG. This metric is very strict. For example one label error can happen only on one stroke symbol or in the relationship between two one-stroke symbols; a labeling error on a 2-strokes symbol leads to 4 errors (2 nodes labels and 2 edges labels). The recognition rate with no error on CROHME 2014 test set is 13.02%. The best one and worst one reported by CROHME 2014 are 62.68% and 15.01%. With regard to the recognition rate with ≤ 3 errors, 4 participants are between 27% and 37% and our result is 35.71%. Considering that there is no grammar constraint included in this moment, the results are understandable and will be improved once this defect is overcame.

VI. Conclusion

We recognize 2D handwritten mathematical expressions by merging multiple 1D labeled paths in this paper. Given an expression, we propose an algorithm to generate a graph using both temporal and spatial orders among strokes. Then, different types of paths are selected from the graph. As a strong sequence labeller, BLSTM is used to recognize these paths. Finally, we merge the results from different paths to rebuild a math expression. Currently, we do not use any grammar knowledge (even local grammar) to recognize expressions. Our proposal presents competitive results on symbol recognition task and segmentation task, promising results on relationship recognition task. Undeniably, the recognition rate is low at the global expression level at this moment. However, it will be improved certainly as the graph and paths catch more ground truth edges, i.e. more variations of stroke order.

In future, we will evaluate the built graph with comparing to the ground truth and then modify it in order to cover more and more ground truth edges and reduce unnecessary edges meanwhile. A better strategy for selecting paths will be explored also. Later, the grammar model will be integrated to the recognition system.

Acknowledgment

We would like to thank the China Scholarship Council for supporting PhD studentship at Nantes university.

References

[1] Mouchère H, Viard-Gaudin C, Zanibbi R, et al. *ICFHR 2014 Competition on Recognition of On-line Handwritten Mathematical Expressions (CROHME 2014).* ICFHR. 2014: 6 p.

[2] Chan K F, Yeung D Y. *Mathematical expression recognition: a survey.* IJDAR, 2000, 3(1): 3-15.

[3] Zanibbi R, Blostein D. *Recognition and retrieval of mathematical expressions.* IJDAR, 2012, 15(4): 331-357.

[4] Álvaro F, Sánchez J A, Benedí J M. *Classification of on-line mathematical symbols with hybrid features and recurrent neural networks.* 2013 12th ICDAR. IEEE, 2013: 1012-1016.

[5] Álvaro F, Sánchez J A, Benedí J M. *Offline Features for Classifying Handwritten Math Symbols with Recurrent Neural Networks.* 2014 22nd ICPR. IEEE, 2014: 2944-2949.

[6] Mouchère H, Zanibbi R, Garain U, Viard-Gaudin C. *Advancing the state of the art for handwritten math recognition: the CROHME competitions, 2011-2014.* IJDAR, 2016, 19(2): 173-189.

[7] Graves A, Liwicki M, Fernández S, et al. *A novel connectionist system for unconstrained handwriting recognition.* IEEE Transactions on PAMI, 2009, 31(5): 855-868.

[8] Graves A. *Supervised sequence labelling with recurrent neural networks.* Heidelberg: Springer, 2012.

[9] Graves A, Fernández S, Gomez F, et al. *Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks.* Proceedings of the 23rd ICML. ACM, 2006: 369-376.

[10] Zanibbi R, Mouchère H, Viard-Gaudin C. *Evaluating structural pattern recognition for handwritten math via primitive label graphs.* In IS&T/SPIE Electronic Imaging, International Society for Optics and Photonics, 2013, 865817-865817.

[11] Awal A M, Mouchère H, Viard-Gaudin C. *A global learning approach for an online handwritten mathematical expression recognition system.* PRL, 2014, 35: 68-77.

[12] Álvaro F, Sánchez J A, Benedí J M. *Recognition of on-line handwritten mathematical expressions using 2d stochastic context-free grammars and hidden markov models.* PRL, 2014, 35: 58-67.

[13] Álvaro F, Sánchez J A, Benedí J M. *An integrated grammar-based approach for mathematical expression recognition.* PR, 2016, 35: 135-147.

[14] Zhang T, Mouchère H, Viard-Gaudin C. *Using BLSTM for Interpretation of 2D Languages - Case of Handwritten Mathematical Expressions.* CORIA-CIFED, 2016, 217-232.