Noname manuscript No.
(will be inserted by the editor)

Advancing the State-of-the-Art for Handwritten Math Recognition:

The CROHME Competitions, 2011 – 2014

Harold Mouchère · Richard Zanibbi · Utpal Garain · Christian Viard-Gaudin

Received: date / Accepted: date

Abstract The CROHME competitions have helped organizing the field of handwritten mathematical expression recognition. This paper presents the evolution of the competition over its first four years, and its contributions to handwritten math recognition, and more generally structural pattern recognition research. The competition protocol, evaluation metrics and datasets are presented in detail. Participating systems are analysed and compared in terms of the central mathematical expression recognition tasks: 1) symbol segmentation, 2) classification of individual symbols and 3) symbol relationships, and 4) structural analysis (parsing). The competition led to the development of *label graphs,* which allow recognition results with conflicting segmentations to be directly compared and quantified using Hamming distances. We introduce *structure confusion histograms* that provide frequencies for incorrect subgraphs corresponding to ground-truth label sub-graphs of a given size, and present structure confusion histograms for symbol bigrams (two symbols with a relationship) for CROHME 2014 systems. We provide a novel analysis combining results from competing systems at the level of individual strokes and stroke pairs; this virtual merging of system outputs allows us to more closely examine limitations for current state-of-the-art systems. Datasets along with evaluation and visualization tools produced for the competition are publicly available.

Keywords Handwriting recognition · Mathematical expression recognition · Competitions · Performance evaluation

1 Introduction

Research in automatic recognition of on-line handwritten mathematical expressions dates back to the 1960's. In on-line recognition, Anderson [1] developed an attributed context free grammar for recognizing hand-printed math expressions written on an input device similar to a tablet. After this initial attempt, several researchers have studied this problem at different paces. Significant research effort has been reported in the last fifteen to twenty years, in part due to on-line input devices becoming more popular. Recognition of math expressions became a requirement for preparing scientific documents on-line. Despite this increase in research activity, estimating progress became quite difficult mainly because of the lack of available benchmarking datasets and variety of evaluation metrics in use. Systems were evaluated primarily using private author-generated datasets which were not publicly available. Being unable to reproduce the results of others, researchers could not clearly judge their progress. This motivated the organization of a competition, which came to be named *CROHME: the Competition on Recognition of On-line Handwritten Mathematical Expressions* [2–5].

H. Mouchère and C. Viard-Gaudin
LUNAM/University of Nantes/IRCCyN, rue Christian Pauc, 44306 Nantes, FRANCE
Tel.: +332-40-68-30-82 E-mail: harold.mouchere@univ-nantes.fr / christian.viard-gaudin@univ-nantes.fr

R. Zanibbi
Dept. Computer Science, Rochester Institute of Technology, 1 Lomb Memorial Drive, Rochester, NY, USA
Tel.: +1-585-475-5023, E-mail: rlaz@cs.rit.edu

U. Garain
Computer Vision and Pattern Recognition (CVPR), Indian Statistical Institute, 203, B.T. Road, Kolkata 700 108, INDIA Tel.: +91.33.25.75.28.60 E-mail: utpal@isical.ac.in

Since CROHME's inception, researchers have been gradually attracted towards the event. The number of participating systems increased from five to eight in four years. Industrial research groups showed interest in and participated in CROHME. Participating systems have used several methodologies, and CROHME has provided a nice platform for comparing these methods and bringing out the relative merits of individual approaches. Each CROHME developed an original test dataset which is separated from a training dataset. Evaluation was done by the organizers and subsequently checked by participating teams for consistency and to correct minor errors (e.g. in output file formats). In this way, CROHME has documented progress in the area, and the concerned research community is now aware of advances in terms of shareable resources, capabilities of different methods, and evaluation results. So far, CROHME has influenced a particular community which had been conducting their research in a sporadic manner to work using a more consistent and scientific approach.

In this paper we present the evolution of CROHME over the last four years, along with its contributions to handwritten math recognition research and related areas. These include the datasets, different tasks, evaluation metrics, tools, participating systems, and finally, analysis of results using both standard and novel methods. Preparation of CROHME data involves several issues including the number of allowable symbols, two-dimensional structures, and coding of data. The CROHME datasets started with a relatively simple set of samples in CROHME-2011, gradually adding more complicated expressions over the next three years. Figure 1 shows some real samples from the last CROHME training set which show the difficulty of the recognition tasks: symbol segmentation, symbol recognition, spatial relation recognition and expression parsing.

While the complexity of expressions to recognize increased with each competition, the recognition of complete expressions is very difficult. As a result, the competition tasks were modified to consider sub-tasks. This gave research groups options to participate in the event as per their convenience. For example, recognition of matrices (being a difficult task) was introduced as a new task in the fourth year, along with a separate competition for isolated symbol recognition.

Expression recognition results can be evaluated in various ways, and evaluation metrics have undergone a number of changes in the past 10-15 years. Initially, simple measures like number of correctly recognized symbols or structures were used [6]. Researchers found these measures insufficient for characterizing local recognition errors, and subsequently, tree-matching based methods [7], and later label graph-based evaluation developed for CROHME [8] were introduced. Along with the competition tasks, the CROHME evaluation metrics evolved over time.

In addition to a detailed discussion of the main features of CROHME competitions, we will address the following questions. 1) Why is mathematical expression recognition a difficult problem? 2) Is it possible to specify which expressions are more complex than others? 3) How does one build a representative corpus of expressions? 4) How do we detect the weakness and specific errors made by a recognition system? 5) Can we merge results from different recognition systems? 6) As mathematical expressions are a two dimensional language, can we detect which symbols configurations are more susceptible to being recognized incorrectly?

The structure of this paper is as follows. Section 2 describes the CROHME competition (tasks, corpus, datasets and protocol). Section 3 discusses evaluation metrics used and developed for the competitions. Section 4 presents the participating systems and the major methodologies systems use. New evaluation results and critical analysis of the results are presented in Section 5. Section 6 outlines the impact of the competition and issues to be addressed in future.

2 The CROHME Competition

In this Section, we present how the CROHME competitions have been organized. After defining the different tasks in the first subsection, section 2.2 explains how we chose and collect the math expressions. Then section 2.3 analyses the datasets considering its complexity. The last subsection explains the submission and testing protocol of the competition.

2.1 CROHME Task Definition

Mathematical Notation. It is difficult to define mathematical notation precisely [9–11]. [1]

The precise semantics of a formula depends on its domain of use, and the way in which an author chooses to define, and possibly re-define parts of their notation. In this way, mathematical notation is highly dialectical, and is in essence a form of *natural* visual language [12].

For the CROHME competitions, samples of formulae from algebra and calculus have been used, which is traditional in math recognition research. We assume all input strokes belong to exactly one symbol - this removes symbols connected by a ligature (e.g. '2x' written

[1] A well-written history of mathematical notation is available [11]

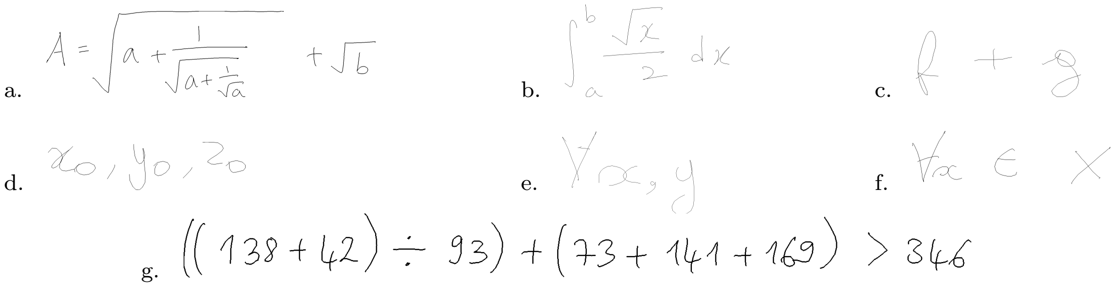
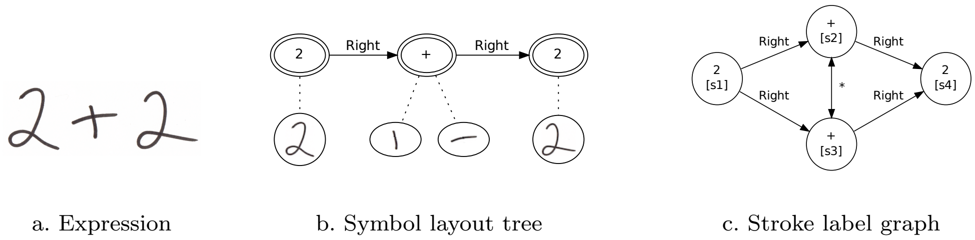


Figure 1 Handwritten formulae from the CROHME training dataset.



a. Expression b. Symbol layout tree c. Stroke label graph

Figure 2 A simple handwritten expression (a) and its interpretation as represented by a *symbol layout tree* (b) and *stroke label graph* (c). The expression contains four strokes, labeled $s1 - s4$ in time order. The symbol layout tree represents spatial relationships between symbols, here with handwritten strokes associated with each node shown by dotted edges. The stroke label graph represents the same information, using strokes rather than symbols as nodes, and using both relationship (e.g. *Right*) and stroke merge (*) labels on edges.

with one stroke). Function names are treated as individual symbols (e.g. $\cos, \sin, \tan, \lim$). Figure 2a shows a handwritten expression, along with a *symbol layout tree* representing the spatial relationships between symbols in Figure 2b. The correspondences of handwritten strokes to symbols in the layout tree are shown using dotted edges. Symbol layout trees have the leftmost symbol on the main writing at the root of the tree, with symbols (nodes) connected by spatial relationships. For CROHME, spatial relationships include *Right, Above, Below, Superscript, Subscript* and *Inside* (for square roots). LaTeX and Presentation MathML[2] encode symbol layout trees without the correspondence of strokes to symbols. We call these *symbolic* encodings. Allowable expressions for different tasks are defined using LaTeX string grammars as described below.

Input and Output. Stroke data is provided to participants in CROHME InkML (XML) files. Each stroke has a unique identifier and sequence of (x,y) locations. Participants must parse CROHME InkML files with stroke data, and then return one of two output formats, depending upon the competition instance. For CROHME 2011-2013, a CROHME InkML file was generated that contained the symbol layout tree in Presentation MathML, along with annotations for the correspondence of symbols to stroke groups. In CROHME 2013 - 2014, Comma Separated Variable (.csv) files for *label graphs* (described below) could be returned instead.

CROHME Competition Tasks. Math notation is domain specific, and the difficulties of recognizing expressions depend upon which symbols and symbol relationships are used. For the competition we define several difficulty levels, from simple expressions on a single writing line (e.g. 'x + 1') to matrix expressions. Each level is called a *task* defined by a context free grammar defining a set of legal LaTeX expressions for the task. These grammatical definitions mean that the expression sets are infinite, which is one difficulty of ME recognition. To parse LaTeX strings we use the PEP[3] parser which is a free implementation of the Earley algorithm. Table 1 shows the increasing difficulty of the five tasks. As can be observed, the symbol set has gradually increased in size and now includes 101 symbols. This enables a good coverage of different scientific domains, while at the same time increasing recognition difficulty due to more easily confused classes, such as for digits '0' and '1' with letters '*O*' and '*l*'. Furthermore the allowable symbol layouts have increased over time. Only six atomic spatial relationships between two symbols are used (Above, Below, Superscript, Subscript, Inside (for square roots), Adjacent at Right), but allowing additional structures increases the difficulty of the task

[2] `http://www.w3.org/Math/`

[3] `http://www.ling.ohio-state.edu/~scott/#projects-pep`

by increasing the number of legal expressions. For example, in task 1 nested fractions and superscripts on function names are not allowed, but in the task 2 these constraints are removed. Table 1 shows the number of production rules used to define the LaTeX grammar. The increasing number of rules reflects more a higher specificity in the modeling of the language than an increase of the complexity of the language. Nevertheless, allowing more spatial relationships and symbols makes expressions more diverse and complex. The task grammars are provided in a package available with the CROHME dataset from the TC11 website.

2.2 Corpus Construction

Before CROHME existed, researchers used to experiment with their own private datasets. Hence fair comparisons between systems were hardly possible. So, one outcome of the CROHME project was to deliver high quality datasets. The main issues to build a dataset are related to relevance, completeness, and correctness of the data. Specifically for the domain of mathematical expressions many parameters are involved: set of symbols (size, identity) and their distribution; corpus composition, extracted from a realm of discourse (algebra, thermodynamic, mechanic ...); number of writers, number of expressions written per writer and number of times a given expression is written.

CROHME organizers used existing resources from different labs already working on ME recognition. In 2011 and 2012 the organizers merged several existing datasets from different laboratories: CIEL [13] and HAMEX [14] (University of Nantes), MfrDB [15] (Czech Technical University), ExpressMatch [16] (University of Sao Paulo), datasets from KAIST and CVPR/ISI and in 2013 MathBrush [17] (University of Waterloo). The expressions (stroke data and ground-truth) coming from these datasets have been used directly by doing file format conversion and adding missing ground-truth.

As explained in the introduction these existing datasets suffer from a lack of representative situations. For example in the MathBrush dataset [17] the expressions are generated randomly from a grammar. In [16] a small corpus of 56 expressions has been written by 25 writers. In [13] only 36 different expressions are selected to cover different scientific domains, then each expression is written by different writers. In each such case, the final corpus is not representative of real usage.

Since CROHME 2013, we have designed a corpus better reflecting the current use of mathematical expressions. Scientific books or courses are interesting sources with lots of ME, but mostly, it is very domain specific, using only a subset of symbols and relations. Instead, we used the wikipedia French pages and the wikibooks English pages as a source of effective MEs. In wikipedia pages, the math expressions are delimited by "math" tags. We have extracted more than 164 000 ME which are our pool of expressions. In addition, some filtering was introduced to control the content of the corpus:

- removing duplicate expressions: popular expressions such as $sin^2x+cos^2x=1$, will be present on several pages.
- controlling LaTeX string length: all expressions will have between 3 and 50 LaTeX symbols. For example, the two following strings "x^2" and "x_i", have a length of 3, but with only two printable symbols.
- defining a valid LaTeX symbol set: a list of acceptable LaTeX symbols and commands is defined.
- validating with a grammar parsing tool: a LaTeX grammar is defined and only successfully parsed expressions are accepted in the corpus.
- controlling the symbol frequency: the symbol frequency in the test set should be the same as in the corresponding training set.

In a real context, the symbol frequency is domain dependant. In CROHME, the symbol frequency in the test sets is made similar to the training sets to be representative. In the first years of the competition (2011 and 2012) the expressions were selected in the available datasets considering only the presence of selected symbols. It was sufficient because the number of the different symbols was low and they corresponded to the most frequent ones; so the sub-sets (training and test) were already balanced. However, when the number of symbols increases, we are getting closer to the tail of the distribution and when expressions are chosen among a huge set (like wikipedia or wikibook corpus) the choice cannot be completely randomized. As a result, in the test set of task 3 in 2012, most expressions have been randomly chosen from the wikipedia FR corpus but some expressions have been added manually to balance the symbol frequency.

In a more systematic way, since CROHME 2013, an algorithm has been used to build a corpus with the same frequency of symbols as in a reference corpus. We used an iterative algorithm which compares the current frequency of each term in the corpus under construction with regards to the targeted frequency of the same term in the reference corpus. At each iteration, the algorithm sorts with decreasing costs the candidate expressions from a pool of expressions. The cost $C_{c,r}(e)$ of a candidate expression (e), eq. 1, has been defined as the sum of its term costs. A term cost $cost_{c,r}(t_i)$, eq. 2, being defined as the negative log ratio of frequency terms t_i in the current dataset c with regards to the reference

Table 1 CROHME Expression Grammars (Tasks). *Year*: competitions where grammar/task was introduced, generally the tasks are used during two consecutive years; *Grammar*: grammar/task identifier used in the competitions; *Symbols*: number and list of used symbols in the corpus; *#P*: number of production rules; *Additions*: expressions added to the corpus with expression samples.

Year	Grammar	Symbols	#P	Additions (with examples)
2011	1 / I	36 symbols : $abcdeiknxyz0123456789\phi\pi\theta$ $+ - \pm \sin \cos \neq \leq > = ()\sqrt{}$	38	No nested exprs. in fractions or sub/superscript $x^2 + y^2 > 1$ $\sqrt{b^2 - 4ac}$
	2 / II	56 symbols, 20 added: $ABCFj\alpha\beta\gamma\infty$ $\div \times \sum \log \tan \ldots \geq \rightarrow \lim \int !$	60	No recursion limits; complex structures included $\sqrt{1+\frac{1}{\sqrt{2}}} + \sqrt{1-\frac{1}{\sqrt{2}}}$ $\lim\limits_{x \rightarrow \frac{\pi}{2}+0} \tan x = -\infty$
2012	3 / III	75 symbols, 19 added: $\{\}[]XY < tfgmrp/,.$ $\exists\forall\in$	95	Set operators and brackets $\forall x \in X$ $\left[\frac{2}{3}x^{\frac{3}{2}}\right]_0^1$
2013	4 / IV	101 symbols, 26 added: $EGHILMNPRSTV$ $hloqsuvw\vert'\sigma\Delta\lambda\mu$	155	nth-root $\sqrt[4]{648+648}+8$
2014	matrix / IV-matrix	101 symbols	168	Matrices within and containing expressions $A = \begin{pmatrix} 3 & 1 \\ 4 & 0 \end{pmatrix}$ $\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$

dataset r. $f_d(t_i)$ is the term frequency of a symbol t_i in d, which can be an expression or a full corpus.

$$C_{c,r}(e) = \sum_{t_i \in e} cost_{c,r}(t_i) \tag{1}$$

$$cost_{c,r}(t_i) = log(f_r(t_i)) - log(f_c(t_i)) \tag{2}$$

Thus a candidate expression which contains many symbols that are under-represented in the current corpus will be ranked before a candidate expression which contains many symbols that are over-represented. A first version of this protocol has been defined in 2013 and the presented version has been used in 2014. Note that this approach can be extended to use the frequencies of other criteria such as the equation length or the spatial relation types. In 2014 the selection takes also into account the size of expressions: the size value is processed as other symbols, its frequency is used to compute the cost $C_{c,r}(e)$ of an expression.

Once the expressions are selected using the previous criteria, they are written once by writers from the different organising labs (University of Nantes, RIT, ISI).

The next section presents several statistics to illustrate the diversity of the dataset and the coverage of the test set with regards to the training set.

2.3 Dataset Properties

A detailed analysis of the corpus used in the main task (task 4) of the competition is provided in this section. It compares the frequencies of each symbol, of each spatial relations and the complexity of the expressions using different indicators across the training subsets and the test sets (2013, 2014). Note that the training part has been the same in 2013 and 2014 competition, only the test sets have been renewed. Thus we compare the statistics from the training part to test parts 2013 and 2014.

Table 2 shows the frequencies of some symbols in the test sets in 2013 and 2014 compared to the training set. We can see that some symbols are very frequent (like '$-$', '1' and '+') and other are very rare (like '$\in$', '$\forall$' and '$\exists$') in all sets. Even if the term frequencies are quite well respected in both test sets, 2014 test better respects the proportions of the different symbols with regards to the training set.

The symbol frequencies do not allow to evaluate the complexity of the different ME sets. Mainly the complexity of an expression is based on the complexity of its MathML tree, that is why we present in Figure 3 some statistics extracted from the MathML trees of the three sets:

- the maximum depth of the trees ignoring the `<mrow >`[4] elements, defined as degree of nestedness DoN in [18]: it represents how nested are the expressions (superscript in a superscript or fraction of subscripted symbols...), a depth of 0 is a one line expression;
- the sum of baselines: it counts the number of times a sub-expression is not on the same baseline as its mother expression, for example $x^2 + y^2$ and x^{y^2} and $\frac{x}{y}$ have 3 baselines;

[4] The MathML `<mrow>` element is used to group subexpressions, which usually contain one or more operators with their respective operands. This element renders as a horizontal row containing its arguments.

Table 2 Symbol frequencies for training and test sets of Task 4 in CROHME 2013 and 2014. Symbols are sorted by decreasing frequency in the training set. Only most and least frequent symbols are shown.

	Train 2013/2014	**Test 2013**	**Test 2014**
-	7940 (9.254%)	440 (7.233%)	910 (9.083 %)
1	6219 (7.248%)	314 (5.162 %)	721 (7.196 %)
2	6195 (7.220%)	338 (5.556%)	715 (7.136 %)
+	5409 (6.304%)	267 (4.389 %)	622 (6.208 %)
x	5042 (5.876%)	261 (4.291 %)	587 (5.859 %)
(	3945 (4.598%)	295 (4.850 %)	458 (4.571 %)
)	3939 (4.591%)	294 (4.833 %)	458 (4.571 %)
=	3611 (4.209%)	319 (5.244%)	434 (4.332 %)
a	2475 (2.885%)	137 (2.252 %)	279 (2.785 %)
3	2458 (2.865%)	117 (1.923 %)	289 (2.885 %)
n	2239 (2.610%)	140 (2.301 %)	267 (2.665 %)
0	1795 (2.092%)	128 (2.104 %)	214 (2.136 %)
$\sqrt{}$	1793 (2.090%)	86 (1.414 %)	213 (2.126 %)
y	1765 (2.057%)	82 (1.348 %)	225 (2.246 %)
4	1635 (1.906%)	77 (1.266 %)	183 (1.827 %)
b	1599 (1.864%)	81 (1.332 %)	185 (1.846 %)
z	1074 (1.252%)	49 (0.806 %)	120 (1.198 %)
d	1063 (1.239%)	89 (1.463 %)	119 (1.188 %)
5	1003 (1.169%)	49 (0.806 %)	122 (1.218 %)
		...	
{	69 (0.080%)	6 (0.099 %)	7 (0.070 %)
}	69 (0.080%)	6 (0.099 %)	7 (0.070 %)
>	56 (0.065%)	9 (0.148 %)	7 (0.070 %)
σ	52 (0.061%)	14 (0.230 %)	13 (0.130 %)
μ	46 (0.054%)	17 (0.279 %)	7 (0.070 %)
Δ	35 (0.041%)	21 (0.345 %)	5 (0.050 %)
λ	27 (0.031%)	4 (0.066 %)	7 (0.070 %)
$\in$	14 (0.016%)	9 (0.148 %)	3 (0.030 %)
$\forall$	8 (0.009%)	3 (0.049 %)	2 (0.020 %)
$\exists$	4 (0.005%)	2 (0.033 %)	4 (0.040 %)

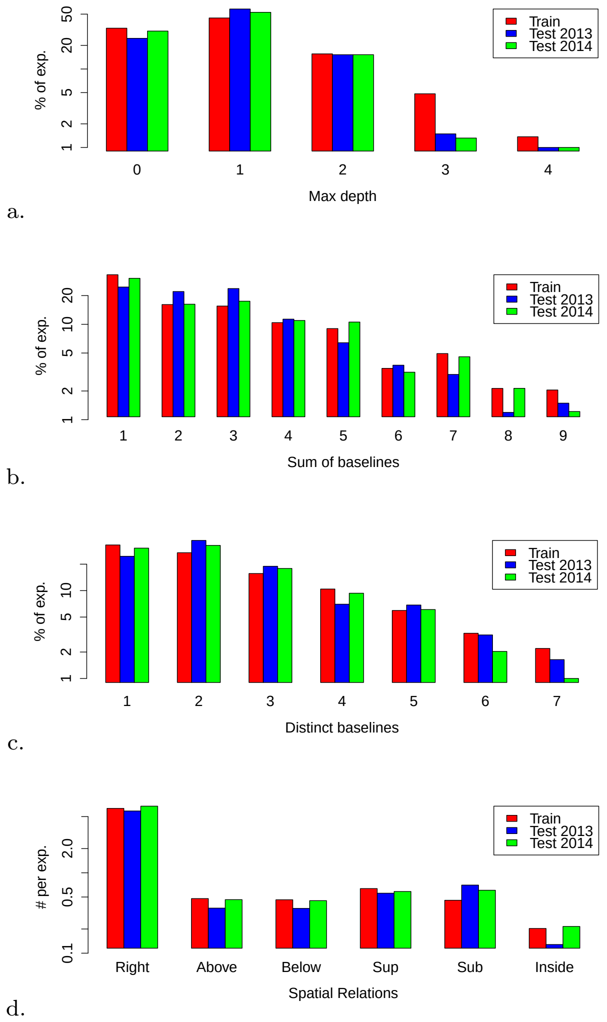


Figure 3 Complexity statistics for training (Red) and test sets for Task 4 in CROHME 2013 (Blue) and 2014 (Green). In a-c, maximum symbol layout tree depths, total number of baselines ('sum of baselines') and distinct baseline histograms are presented in order of increasing complexity (i.e. tree depth and number of baselines). Spatial relationship frequencies are presented in terms of the average number of occurences per expression. Log-scale are used for the y-axes.

- the number of distinct baselines, defined as geometric complexity GC in [18]: in the previous metric some baseline can have the same level, this metric counts the number of different levels which are used in the full expression, e.g. $x^2 + y^2$ uses 2 distinct baselines but x^{y^2} and $\frac{x}{y}$ use 3 distinct baselines;
- the spatial relation frequency: the previous metrics are quite independent of the nature of the spatial relations, this last metric counts the number of time each spatial relation is used.

For each of these 4 statistics, the frequency is showed (using a log scale) by normalizing the counter by the number of expressions in the corresponding dataset. We can see that about one third of the datasets are simple one line expressions (max depth of 0 and number of baseline or distinct baseline equal to 1 for 33% of expressions in the training set, 25% in test set 2013 and 30% in test set 2014). Then half of expressions are not nested expressions (max depth of 1 for 45%, 58% and 53% of expressions). Probably these expressions are covered by the expressions with 2, 3 or 4 baselines (from 42% to 57% of the expressions) and 2 or 3 distinct baselines (from 43% to 56% of expressions). Furthermore, the tails of the histograms have been cut, thus some expressions which are not shown here are very complex (about 2% to 5%) with a max depth of more than 4, more than 9 baselines and more than 8 distinct baselines. Finally the spatial relation frequencies show that the most frequent relation is *Right* with 6 relations per expression in average, then *Superscript* (about 0.5 superscript per expression) followed by the *Subscript* relation (also about 0.5 per expressions) and *Above* and *Below* (0.4 per expressions). The relation *Inside* used only in square roots is rarer with 0.2 per expression.

2.4 Competition Protocol

Each year there was something new in the competition (a new task, a new evaluation tool or a new file format) thus we tried to give enough time to participants to submit a system taking into account these updates. Having enough time was a key point because the training and testing data have to be created for the competition and this is a time consuming process.

After the call for participation, we provide the registered participants with the evaluation tools, the file formats, the task definitions (i.e. the grammars and symbol lists) and an initial training set. During the next months we eventually update the training set depending on our tests and the participant feed-backs. Then the participants have to submit a first draft system about four months after the call. This first submission allows to test the systems in real conditions and to fix some technical issues (mostly OS configurations or file format problems). Detailed results on the training set were sent back to participants to detect the problems. The final deadline is about 5 months after the call for participation. Then the participants have to submit their final systems, one per task. During the next 2 weeks the organizers run each participating system on each test dataset. After the competition, the test data is available on demand and submitted on TC11 website.

This protocol requires a huge effort from the organization committee but it has several advantages. Firstly it allows good collaboration and discussion between participants and organizers as several exchanges are necessary. Furthermore, this protocol is quite fair as no participant see the test set before the final submission (except for the organization teams). This protocol allows also organizers to correct some minor issues in the test data or evaluation tools after the final submissions, e.g. correct ground-truth errors in the test set or add new metrics and features in the evaluation tools.

3 Evaluation Metric and System Rankings

Evaluation of structural pattern recognition systems is often difficult because of the interaction between detected objects and their relationships. For example, when a relationship between one correct and one incorrectly segmented symbol is detected, how this should be evaluated? Obviously the relationship is incorrect because one of the symbols is incorrect, but can we quantify partially correct structure somehow? To address this question, a number of new metrics and a new structure representation were developed for CROHME. In particular, new stroke-based metrics allow partially correct recognition results to be located and measured precisely [8, 19]. We discuss these metrics below.

3.1 LaTeX, MathML, CROHME InkML and Symbol-Level Evaluation

Expression and Structure Recognition Rates. Prior to CROHME, it was common to evaluate math recognizer performance using expression recognition rates, e.g. the percentage of LaTeX formula strings matching ground truth. The metric is simple to understand and provides a useful global performance metric. Expression rate has been used to rank participating systems in all of the CROHME competitions, first using canonicalized Presentation MathML (insuring that identical expressions with different MathML representations match), and later label graphs, taking symbol segmentation into account (see the next Section). However, an absolute expression rate does not characterize partially correct recognition.

Previous attempts to quantify partially correct recognition in symbolic encodings have been made. The EMERS [7] metric is a tree edit distance using an Euler string representation for MathML trees. Symbol errors are weighted based on their distance from the main writing line (baseline) of the expression. The IMEGE metric [20] is an image-based comparison of LaTeX, using the number of matching pixels in a rendered LaTeX string with the associated ground truth LaTeX rendering. To prevent missing valid subexpressions that have been 'shifted' in the rendered image, small image distortions are permitted.

For CROHME a simpler and faster-to-compute metric was used to count partially correct expressions. For each system we produced a list giving the percentage of expressions with matching MathML tree but with at most n incorrect symbol and relationship labels. The rate for $n = 0$ is the percentage of test expressions correctly recognized, followed by expression rates for increasing numbers of labeling errors. For expressions where the node and edge structure of the layout trees do not match, no number of relabelings can correct the expression. For example, '$2x_{+1}$' has two label disagreements with '$2x + 7$' (the subscript for '+', and '1' not matching '7'), but these differences can be corrected with two relabeling operations. Conversely, the expression '$2^x + 1$' cannot match '$2x + 7$' because the '2' has a different number of child nodes. Expression rates for $n \leq 3$ symbol and relationship label errors are reported in each CROHME competition.

For CROHME 2012 we also proposed a new formula structure ($STRUCT$) expression rate, which matches layout tree structure, ignoring symbol labels and stroke segmentation. For example, if the expression '$x^2 - 1$' is recognized as '$2^a + b$,' it will be considered as correct. As another example, recognizing the expression in Figure 2 as '5 x 5' will also be considered correct. In this way, formula structure detection can be evaluated separately from symbol segmentation and classification.

Symbol and Relationship Detection. The correspondence between strokes and symbols provided in CROHME InkML files allows us to compute the percentage of target symbols correctly segmented (*SYM_Seg* in CROHME 2011 to 2013), and the number of correctly segmented and classified target symbols (*SYM_Rec* in CROHME 2011 to 2013). These are measures of symbol *recall.* One can also measure the *precision* of recognized symbols, which is the percentage of correctly detected or detected and classified symbols. During CROHME 2014, we realized that symbol relationships can also be evaluated using recall and precision, measured over pairs of symbols with a relationship and their associated strokes. Relationships are detected correctly if both symbols in the relationship are correctly segmented and have a defined relationship in ground truth, and correctly detected and classified if the relationship labels also agree.

3.2 Label Graphs and Stroke-Based Metrics

Label Graphs. Beginning with CROHME 2013 a second structural representation called *stroke label graphs* was used [19], as illustrated in Figure 2c. A label graph defines structure using a directed graph over handwritten strokes. Strokes (nodes) are labeled by their associated symbol, and directed edges between stroke pairs are labeled by either a spatial relationship between two symbols (e.g. 'Right'), no relationship ('_') or a pair of directed edges for strokes belonging to the same symbol ('*'). For legibility, 'no relationship' edges are omitted in Figure 2c, but Figure 2c represents a *complete* graph with every nodes and directed edge labeled. An adjacency matrix over strokes with labels may be used [8]; the number of labels in a label graph is given by n stroke labels, plus $2\binom{n}{2}$ directed edge labels, giving n^2 labels in total. In Figure 2c, $4^2 = 16$ labels are defined.

Every possible symbol layout tree for a set of input strokes can be represented by a label graph, along with partial and illegal interpretations (e.g. a forest of symbol layout trees). Symbol-level evaluation metrics may also be computed directly from label graphs [8]. Symbols are defined by cliques of '*' (merge) labels or labels that match labels for attached strokes - we use '*' edges in Figure 2. Using these stroke cliques, we can recover the symbol layout tree from a label graph after recovering symbols. For efficiency, in our tools we use connected component analysis over stroke labels to identify the grouping of strokes into symbols.

Hierarchical structure can be represented using sets of labels for nodes and edges, where labels for each level of structure are disjoint [5,8]. One can then filter for labels from a specific 'level.' This is how we obtained results at different levels for the matrix recognition task in CROHME 2014. Label graphs may be easily combined and/or filtered in various ways, as we do to identify which expressions can be correctly recognized if all correct node and edge labels from participant systems are joined in Section 5. Non-hierarchical structure can also be represented with label graphs.

Label Hamming Distances. We can use label graphs to compute *exact* disagreement for input strokes between two interpretations. This resolves the problem with symbolic encodings such as LaTeX being unable to put disagreeing segmentations at the symbol level into correspondence. We define Hamming distances for stroke labels (ΔC), conflicts of segmentation at individual edges (ΔS, where only one of two graphs have a 'merge strokes' label) and edge relationship type conflicts (ΔR). The absolute (total) Hamming distance ΔB is the sum of these component label errors ($\Delta B = \Delta C + \Delta S + \Delta R$). We also defined a variation that weights segmentation edges lower to compensate for their frequency (ΔE).

Stroke-Level Error Analysis. Label graphs permit highly detailed error analyses: now, when a symbol is mis-segmented by a system, we can determine precisely which strokes were grouped correctly or incorrectly for the target symbol. This additional information allows us to make new analyses such as that presented in Section 5.3, where we enumerate small sub-graphs in ground-truth (e.g. for pairs of related symbols) and then count the number of times different errors are made for each. We can now count and visualize incorrect label graphs representing any combination of classification, segmentation and relationship mis-labelings for strokes in a given target sub-graph.

4 Participating Systems

In this Section, we provide a summary of techniques that have been used in the CROHME competition [2–5]. Results from different systems for the CROHME 2014 competition are discussed in the next Section. For brevity, the identifiers shown in Table 3 are used to identify participants. Table 3 also provides references pertaining to participant systems. Relatively current surveys of techniques for recognizing typeset and handwritten mathematical expressions are available [6, 9, 10, 21, 22] along with a more general introduction [23].

Table 3 CROHME Participants (2011-2014). Related work cited by groups are provided in the *Refs.* column. The rightmost portion of the table shows which competitions each group participated in.

			CROHME			
Id	Research Group	Refs.	2011	2012	2013	2014
Ath	Athena Research Center (Greece)	[24]	✓	✓		✓
Czt	Czech Technical University (Czech Republic)	[25, 26]			✓	
Mys	MyScript/Vision Objects (France)			$\boxed{\checkmark}$	$\boxed{\checkmark}$	$\boxed{\checkmark}$
Nan	University of Nantes, IRCCyN (France)	[13, 14, 27, 28]	✓	✓	✓	✓
Ria	Rochester Institute of Technnology (USA)	[29–35]	✓	✓	✓	✓
Rib	Rochester Institute of Technology Imgaging Science (USA)	[31, 35, 36]				✓
Sab	Sabanci University (Turkey)	[24, 37]	✓	✓	✓	
Sap	University of São Paulo (Brazil)	[16, 38]			✓	✓
Tok	Tokyo University of Agriculture and Techn. (Japan)	[39–41]			✓	✓
Val	Universitat de Politècnica de Valencia	[42–44]	$\boxed{\checkmark *}$	✓	$\boxed{\checkmark *}$	$\boxed{\checkmark *}$
Wat	University of Waterloo (Canada)	[45, 46]		✓		

$\boxed{\checkmark}$ winning system $\boxed{\checkmark *}$ winning system trained only on CROHME training data

4.1 Language Models

Here we briefly summarize the symbolic, spatial and/or temporal constraints used by CROHME systems. These are used to validate hypotheses and/or constrain search in participant systems.

Symbols. For symbol classes, some participants represent all CROHME symbol classes separately, while others (e.g. *Wat*) use a smaller set of classes along with rules to tokenize (i.e. combine) symbols into larger symbols, e.g. representing $\leq$ using routines to combine a horizontal line below a $<$ into $\leq$, or individual letters into a function name (e.g. for 'cos,' which is a symbol class in CROHME). *Nan* was the first to include a 'reject' class to represent invalid segmentations. To make computation feasible, nearly all systems make a restriction upon the maximum number of strokes in a symbol (most commonly 4 strokes).

Spatial Relationships between Symbols. Many systems estimate the typographic class of symbols, representing where a symbol sits on the writing line. Commonly these include ascenders (e.g. 'd'), descenders (e.g. 'y') and symbols lying between the writing line and center line or 'x-line' (e.g. 'a'). These typographic classes are then used to constrain the locations of symbols in particular relationships (e.g. subscript, vs. adjacent-at-right vs. superscript). Mathematical types are also used to constrain relationships (e.g. rejecting subscripted digits (2_1), and superscripts or subscripts for '+'). Another important element is how spatial regions are defined. Many methods partition space around a symbol using rectangular regions, and then test for relationships of neighboring symbols or sub-expressions that lie within these regions. Alternatively, neighboring symbols or strokes are used, often with a maximum distance for valid neighbor relationships.

Expression Grammars. Two generalizations of context-free grammars have been used to define legal expressions by participants.[5] The first generalization is two-dimensional context-free grammars that allow horizontal, vertical, and scripted concatenation in production rules *(Nan, Tok, Val, Wat)*. In all cases, the grammars incorporate fuzzy *(Wat)* or probabilistic weights; some incorporate these in the grammar production rules, but more commonly the score for an interpretation is defined using symbol and relationship classification scores (e.g. in a linear combination) to avoid biasing scores toward smaller trees. The second generalization is probabilistic graph grammars, where production rules may have graphs on the left and right-hand side production rules to represent more complex patterns *(Sab)*. These systems enumerate a space of possible layout trees and then return the highest-ranked interpretation as output. To make this tractable, low-confidence symbol and relationship types are pruned during parsing (e.g. considering only top-10 symbol classes).

Three systems *(Ria, Rib, Sap)* use simple 'baseline' grammars representing only the horizontal adjacency of symbols on a writing line, and rectangular spatial regions around baseline symbols [48]. Parsing with these grammars involves a greedy top-down search, locating the main baseline of the expression and then recursively locating the main baseline in each sub-region. These grammars do not consider mathematical types, and do not insure that operators have all of their arguments.

[5] The first example of such a grammar for math recognition was presented by Chou in the late 1980's [47].

4.2 Preprocessing Stroke Data

The stroke data provided in CROHME comes from a variety of countries, and from a number of devices including electronic whiteboards, tablet computers, writing tablets, and Anoto pens that capture physical pen strokes.[6] Stroke sampling rates and coordinate systems differ greatly between devices and capture platforms. Stroke points are represented using integers or real-values, and sometimes include negative coordinates. Participants have used a variety of approaches to handle this, including normalizing expression size (e.g. fixing the expression height at 1.0 and then maintaining the relative width of the original stroke data), and resampling strokes to compensate for differences in stroke resolution arising from sampling and writing speed (faster movement yields fewer samples). Stroke resampling techniques have included Line Density Projection Interpolation *(Tok)*, uniform distance stroke resampling *(Ria, Rib, Nan)* and Cubic splines *(Ria in 2013 and 2014)*.

4.3 Training

Participants have used a variety of approaches to setting system parameters for pre-processing, language models and recognition. Systems using stochastic context-free grammars have had their rule probabilities tuned both manually and algorithmically (using the Inside-Outside algorithm), and a fuzzy-based grammar system had membership functions that also required tuning *(Wat)*. A number of systems trained their recognition modules stage-wise (e.g. first symbol segmentation and/or classification, spatial relationship classification, and then parsing/search parameters). For example, each module of the *Val* system is trained independently [43]: the isolated symbol recognizer (HMM or BLSTM in [44]) using extracted isolated symbols and spatial relation classifier (SVM) from pairs extracted from the training expressions. A pair of research groups (as *Nan*) utilized an explicit reject class in their classifiers, and trained classifiers directly from expression data, dynamically generating instances for the 'reject' class within a global training architecture. Some participating systems were trained on additional data alongside the provided CROHME training set *(Czt, Tok 2013, Sap 2013)*, or on a completely different training set *(Mys)*.

4.4 Recognition Operations

The recognition operations can be split into four primary sub-tasks: 1) segmenting symbols, 2) classification of symbols, 3) classification of spatial relationships, and 4) parsing expression symbols and structure (i.e. the search strategy or processing pipeline). Even if some systems try to perform two or more steps simultaneously in 'holistic' approaches, they all extract information for these sub-tasks.

Symbol Segmentation. Segmentation involves grouping strokes into symbols. To reduce the number of segmentations considered, participants always use some form of continuity constraints. Symbols drawn with multiple strokes have a temporal and/or a spatial continuity: the strokes are close in time or in space. Once the space of segmentation hypotheses is defined, systems can enumerate possible segmentations before evaluating each one. Some participants design a dedicated recognizer for symbol segmentation. However, most participants use symbol recognition results to guide segmentation: if the segment corresponds to a symbol with high confidence, then the hypothesis is kept. The reject option is between these two ideas: modeling invalid segments using the reject class, and then allowing this to be considered alongside concrete symbol classes (i.e. not filtering segmentations detected as invalid from the hypothesis space).

After using symbol recognition results to evaluate segmentations, some participants select a segmentation of strokes into symbols, but most delay this decision until later processing. Depending upon the parsing method, there are two ways to delay segmentation: first, in a lattice which stores candidate segmentations, or secondly allowing the parsing process to enumerate possible segmentations. If parsing is bottom-up (e.g. CYK), these two methods for considering candidate segmentations are equivalent, as all valid segments needed to be computed, evaluated and stored.

Symbol Classification. A wide variety of features and classification algorithms were used for symbol classification. Even if decisions can be delayed (e.g. to the parsing stage), all participants consider symbol recognition as an isolated task - i.e. the features, classification algorithms and training are completed separately from other recognition steps.

We can distinguish two families of feature types: on-line features which use the sequence of points in strokes, and off-line features which use the image of a symbol candidate. The advantage of on-line features is that they use information unavailable in an image such as timing data, exact pen position, and paths in crossing or overwriting strokes. However, this additional inform-

[6] http://www.anoto.com/

ation leads to more variability for some classes due to differing stroke orders, variations in writing speed, etc. Using both on-line and off-line features can be useful, and participants using this combination obtain the best results in isolated symbol recognition results. This fusion of on-line and off-line features can be done early at the feature level (e.g. by concatenation of feature vectors) or later using a classifier combination.

The classification algorithms employed are diverse. We can distinguish classifiers that use features sensitive to time (Nearest Neighbor with template matching or DTW, Hidden Markov Models, Recurrent Neural Network) and those considering symbol shape using off-line features (MLP, SVM, Decision trees). Some classifiers include a reject class, allowing invalid symbols to be identified.

Spatial Relationship Classification. Spatial relationships can be detected using different elements, either through symbol to symbol relationships, or sub-expression to sub-expression relationships. Early solutions use intuitive features based on relative positions of bounding boxes for symbols and/or sub-expressions, or the position of a symbol relative to a symbol on a detected baseline. These raw features have the drawback of not taking into account the shape of symbols. A single bounding box arrangement can be shared by different symbol relationships (e.g. consider the bounding boxes for a handwritten A^2 vs. pa). One way to mitigate this problem is to use the typographic class of the symbols within a relationship, e.g. descenders, ascenders, and large operators (e.g. $\int$).

In the final year of the competition, layout features have been defined using shape contexts *(Val, Ria, Rib)* which provide visual information absent in the lower-resolution bounding box-based features. From a given layout feature, it is possible to recognize the spatial relations with a classifier (e.g. SVM or set of thresholds) or define weights/costs that can be used in evaluating interpretations during parsing. Similar to simple classification hypotheses, geometric or probabilistic weights can be used to generate a space of possible structural interpretations, allowing multiple interpretations to be considered during parsing. The *Wat* system is unique in that the 'Above' relationship is not considered, reducing spatial relationship classification from a 6-class to a 5-class problem.

Parsing. Most participants define the formula recognition problem as a global optimization, requiring the minimization of a cost or maximization of a joint probability for symbols and relationships. Formula rank scores are defined by a combination of symbol, relationship and production rule confidences (e.g. using a linear combination or geometric mean). It is natural to compute these scores and use them in the parsing process.

Participants using a Context Free Grammar-based language model employ the Cocke-Yonger-Kasami (CYK) parsing algorithm. We can split these participants into two groups, based on how they initialize the CYK table. *Nan, MyS, Czt* generate symbol hypotheses and use them as the leafs (terminals) of the parsing table. The second group (*Tok, Val*) use strokes as terminals, but they have to fill multiple lines of the table during initialization - the first line for one stroke symbol candidates, the second line for two stroke symbol candidates, up to the maximum number of strokes allowed in a symbol. The resulting search space between these initialization strategies can be similar, particularly when the temporal stroke-order is preserved. The main difference comes from how symbol hypotheses are generated (see the discussion above). Probabilistic Graph Grammars (*Sab*) have also been used, using a parsing algorithm similar to CYK: initialized by symbol hypotheses, merging symbols into sub-expressions by applying grammar rules, and stopping when all strokes are consumed. One advantage of this approach is that grammar rules are not restricted to Chomsky Normal form and defined using graphs, making the grammars more intuitive to define and read.

These approaches are bottom-up algorithms. Conversely, a successful top-down parsing strategy has been proposed by *Wat*. He defines a Relational CFG parser using Unger's algorithm. Each production rule explicitly defines how subexpressions are arranged, whether horizontally or vertically. In this approach, the 'Above' relationship is not directly detected. Indeed vertical structures are parsed from top to bottom with 'Below' relations. The complexity of the stroke partitioning in this algorithm (polynomial) is reduced by terminal reordering before rule applications.

Remaining approaches use recursive baseline detection from a fixed symbol segmentation (*Ath, Ria, Sap*). Once symbols on a baseline have been detected, symbols located in vertical relationships with baseline symbols are detected. We can distinguish systems using Minimum Spanning Trees (MST) identifying vertical structures before right, superscript or subscript relationships (*Ria 2013-2014, Rib*). While relatively fast, these baseline-based approaches are less accurate than the CFG-based approaches. They employ much less strict language models that permit invalid interpretations, and are greedy algorithms with limited backtracking.

5 Results and New Challenges

As the detailed results using the official metrics described in section 3.2 are available in the competition papers, we choose to present the results following a different point of view in order to address the following questions: (i) how difficult is the recognition task? (ii) how do the participating systems behave towards this complexity? (iii) is it difficult to do better than what the participating systems produced? and (iv) what are the issues still remained unsolved?

In this section, we focus on the last competition results for analyzing which type of expressions still need attention. After this analysis we show how to use the proposed tools and framework based on the label graph as described in section 3.2 to merge results from different systems and do a high level error analysis.

5.1 Expression Level Results with Regards to Expression Complexity

Table 4(a) counts the number of correct expressions from the last test set (Part 4 test set 2014) which are well recognized by each system. For example, out of 986 expressions, the best performing system recognized 618 expressions (62.15%) and the system ranked last recognized 148 (15.05%). While checking expression wise performance of the systems, we note that 707 expressions are recognized by at least one system and 279 expressions are never recognized. It indicates that there are 89 expressions (707 - 618 = 89) which are not recognized by the best system but are recognized by some of the other systems. The second row shows the number of expressions which are recognized only by the corresponding system (and not by any of the other systems). It shows that each system has a different behavior and they all have their own strengths and weaknesses. Furthermore, if an oracle is available to choose the right system for each expression, theoretically it would be possible to reach at a recognition rate of 707/986 = 71.7%. The merging of the decisions coming from different systems is discussed in the next section.

Let us now focus on the remaining 279 expressions which are never recognized. Figure 4 shows the repartition of these expressions with regard to the complexity criteria defined in section 2.3. The histograms show that the CROHME data still have challenging expressions to be recognized. If we consider the number of symbols per expression as a complexity measure, more than 41% expressions having more than 11 symbols are never recognized. The expressions with less number of symbols are nicely recognized and only 13% of the expressions with less than 5 symbols are never recognized. The recognition rate for expressions with 6 to 10 symbols is 22%. If we consider more structural criteria, the proportion of never recognized expressions always increases with the complexity. It rises from 13% of expressions with only one baseline to 19% with 2 baselines, 33% with 3 baselines and then between 32% and 57% for more complex expressions. This is more evident if we take into account only the number of distinct baselines: the proportion of never recognized expressions triples from 13% to 39% when expressions have 3 baselines like a subscript and a superscript in the same time or just a simple fraction. The last histogram (maximum depth) well sums up the situation: one third of the test set (˜300 exp.) are simple and quite well recognized (only 13% are never recognized exp.); one half (˜500 exp.) has a first level of difficulties and are still difficult to recognized (31% are never recognized) and more complex expressions are still challenging.

5.2 Merging System Outputs

In the previous section we showed that if it is possible, thanks to an oracle, to choose the right recognition result among the answers from the 7 participants, we could reach the recognition rate of 71.7% (see Table 4, 707 among the 986 expressions from the test set 2014). This is the best we can do with late decision merging (expression level). The label graphs allow stroke level and stroke pair level evaluation of the different systems (see Section 3.2). We can use these label graphs to perform an early merging of the system decision. Indeed as we did at the expression level, we can merge all local decisions using an oracle which knows the right answer. To implement this theoretical merging we use the multi-label version of the label-graph with a specific metric to compare the resulting graph with the ground-truth. Let us describe the main outline of this process. First a multi-label graph is built for each expression by aggregating the output files from each participant. Then these multi-label graphs are compared with the ground-truth: for a solution (stroke and edge labels) if the correct label is present among all decisions, we keep only this solution else we replace it by an error label. Finally, these merged and corrected label graphs are compared again by using the standard evaluation tool to extract recognition rate and error counters.

The right part of Table 4 shows expression recognition rate for the best possible early merging of each local decision (strokes and relations) for 2 or more participants. As the contribution of each participant is not the same, we progressively add the systems sorted by their global recognition rate: increasing (the two worst

Table 4 CROHME 2014 Expression Rates (a) before and (b) after merging system results. In (a), the second row shows the number of expressions recognized only by the corresponding system. In (b), the number of correctly recognized expressions obtained after merging all stroke labeling decisions from 2 to 7 systems are shown, both in order of increasing and decreasing expression recognition rate as shown at the left.

a. CROHME 2014 Test Set Expression Rates

	Mys	Nan	Ria	Rib	Sap	Tok	Val	Any Sys.
Correct	618	257	187	187	148	253	367	707
Unique	164	5	3	3	0	8	23	-

b. Expression Rates After Label Merging

#Syst.	Increasing	Decreasing
2	307 (31.14%)	716 (72.62%)
3	422 (42.80%)	751 (76.17%)
4	563 (57.10%)	774 (78.50%)
5	668 (67.75%)	786 (79.72%)
6	727 (73.73%)	800 (81.14%)
7	810 (82.15%)	

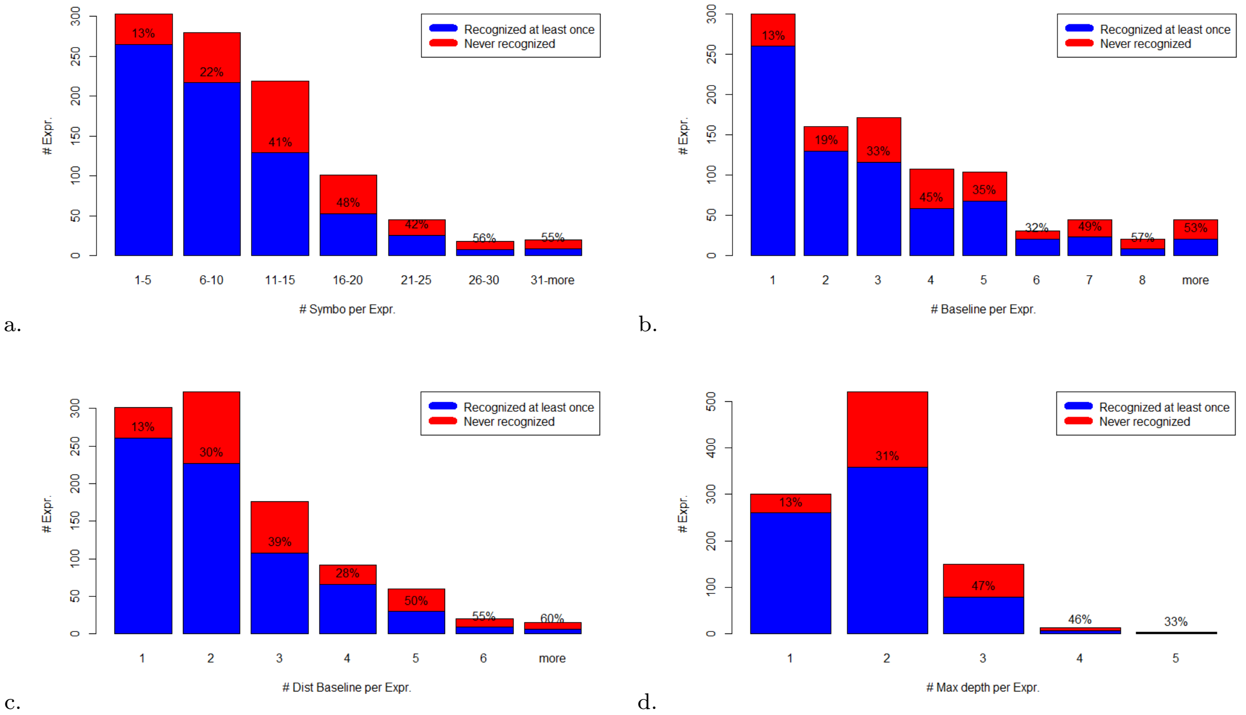


Figure 4 Frequency of Expressions with Differing Complexities. Complexity is characterized by (a) number of symbols, (b) number of baselines, (c) number of distinct baselines, and (d) maximum layout tree depth. Frequencies are shown for the full 2014 test set (red + blue bars), and for expressions recognized correctly by none of the participants (red). Percentages on red bars give the ratio of unrecognized expressions to the all expressions belonging to the column/parameter.

systems are merged first) or decreasing (the two best systems are merged first).

The first observation is that each system contributes to the merging, indeed the number of correct expressions always increases whatever the order of the systems. Table 4, in addition, shows that one system has no expressions which are recognized only by it, but at the stroke and relation level it participates significantly (10 expressions added in the last step of the decreasing order). The second point is that this merging strategy can reach a better recognition rate of 82.15% which is much more than the 71.7% by considering only the expression level results. It means that among the 279 expressions which are never recognized, 103 expressions can be recognized using an early merging of the system results. Some expressions are partially recognized by several systems and the correct expression can be retrieved from these partial results. Note that no language model has been used in this merging process. It means that if some errors introduce completely wrong structures, even by correcting the stroke labels, it will be difficult to retrieve the correct structure. Nevertheless, once correct and incorrect nodes and edges are detected, we can measure the primitive and object level metrics. The globally merged system incurs 214 errors on stroke labels and 600 on edge labels which are only a quarter of the errors shown by the best system results (845 errors on strokes and 2489 edges, cf. [5]). If we consider the object level metrics, the object recall rate is 98.32% (well segmented and well recognized) and relation recall rate is 97.13% (well detected with valid objects and relationships). Compared to the system *Mys*, the symbol recognition has been increased more than spatial relation recognition (+4.41% vs. +2.87%). These corrections allow recovering half of their miss-recognized expressions (-47%, from 368 to 176 wrong expressions).

5.3 Bigram Error Analyses

As described in section 3.2 the CROHME metrics are based on counting the errors at stroke level (stroke labels or relation label between strokes) or object level. However, it also possible to count the errors at sub-graph level without a big increase of complexity. In this section, we show what can be done at the bigram level where two symbols are linked by a spatial relation.

Starting from the ground-truth object tree, we enumerate each pair of symbols which are connected in this tree. Using corresponding strokes, it is possible to check if this sub-graph is correct or not in the recognized expression. One drawback of this approach is that sometimes one primitive level error can lead to several wrong bigrams because the corresponding symbol appears in several sub-graphs. Nevertheless this counting allows a high level error analysis. Figure 5 presents a small capture of the HTML file generated by this error analysis tool applied to *Nan* system. For this system, a total of 4056 errors are counted for 1489 different object bigrams. Figure 5 shows the errors for only one particular bigram pattern $x+$. For a particular symbol bigram many different conditions at stroke level exist in the test set. For instance, the bigram '$x+$' appears with 4 different conditions in the test set (from 2 strokes to 5 strokes). In Figure 5, the wrong labels are drawn in red. With the 4-stroke case (row 1), out of the 34 errors which appear, 7 errors concern a missing spatial relation between x and $+$, 4 errors concern a wrong spatial relation (*Sub* instead of *Right*, x_+), missing nodes (*ABSENT* label) happen twice; in the 3-strokes cases (row 2), 2 errors concern wrong symbol and spatial relation label (n_+) and 1 error accumulates a wrong segmentation and wrong labels (x_1-), . . .

Table 5 presents the most frequent bigram errors for each system and for all systems together. For each system, the three most frequent errors are selected (in bold font) and can be compared with errors from other systems. Figure 6 shows one example which explains one occurrence of the error on the bigram +1 for the system *Nan*. Furthermore, this figure shows one possible visualization of the errors in the corresponding label graph: five strokes are used (numbered in nodes from 0 to 4), 3 nodes have the correct label (blue ones) and 2 are incorrect (wrong segmentation of the x); all edges are misrecognized (the correct label is in brackets, the label '_' stands for 'no edge').

It is interesting to note that the systems have different behaviors but also share some common difficulties. The two sub-expressions $x+$ and $\frac{1}{}$ are the most frequent errors, maybe because these 4 symbols are the most frequent ones (see table 2). Furthermore $x+$, $x\times$ and $\times x$ are very frequent, it can be explained because these bigrams combine several difficulties. They are multi-strokes, thus at first, there is difficulty in segmentation phase. These symbols can easily be confused keeping a valid structural subexpression (if xx is recognized instead of $x\times$ it could still be a valid expression). The $-=$ and $=-$ sub expressions are difficult to recognize because the horizontal lines can be merged to build an extended fraction bar. For example, $= \, -$ can be recognized as $\frac{-}{}$. Note that the horizontal bar can be a fraction bar or a minus sign. It is surprising that the most of frequent symbol bigram errors have left-right relation. The only complex structure is $\frac{1}{}$ and several errors are possible. For instances, merging of the symbol in a $+$ sign or in a digit 1 written with a horizontal stroke.

If we look at the merged system results, we can say that the most difficult bigrams to recognize are $\sum^n$ and $\sum_i$ (mainly confusion between relations *Above* with *Sup* and *Below* with *Sub*). Among the presented cases, the three most frequent errors use spatial relations with complex symbols ($\sum$, fraction bar) and the other types use very confusing symbols $\times$ and x. Actually 303 different bigrams are listed in the full error list of the merged system and 705 are listed in the error list for the best participating system, i.e, *Mys*.

6 Open Problems and Conclusion

The analysis of CROHME results shows two important aspects. Firstly, the community of CROHME has proposed efficient solutions to solve the problem for simple cases: small expressions with simple structures have only about 13% of errors. The second point is that the complex expressions are still challenging. However, complex expressions do not heavily degrade the overall performance of the systems as a system can easily achieve accuracy like 30% just by recognizing simple expressions present in the dataset. The error analysis shows that symbol segmentation and recognition are still difficult in the context of complex expressions. The problem is not in the symbol classifier (as shown in the isolated symbol classification task in CROHME 2014)

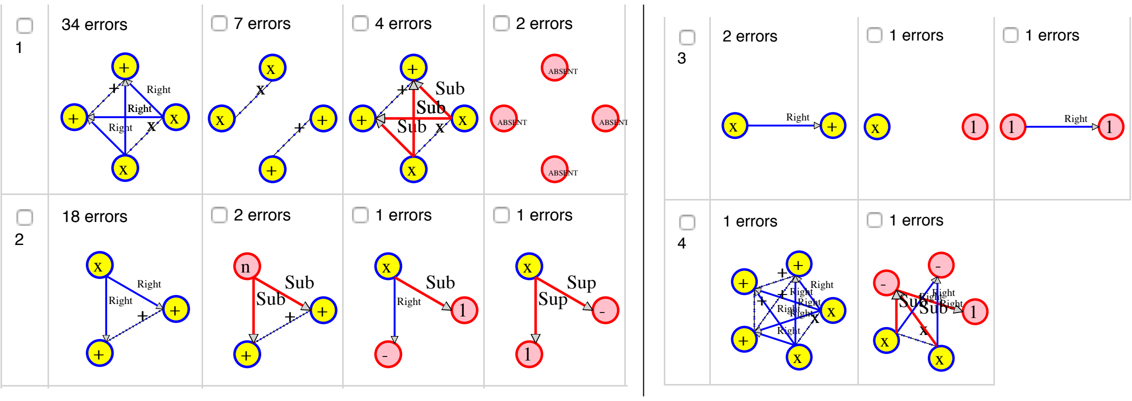
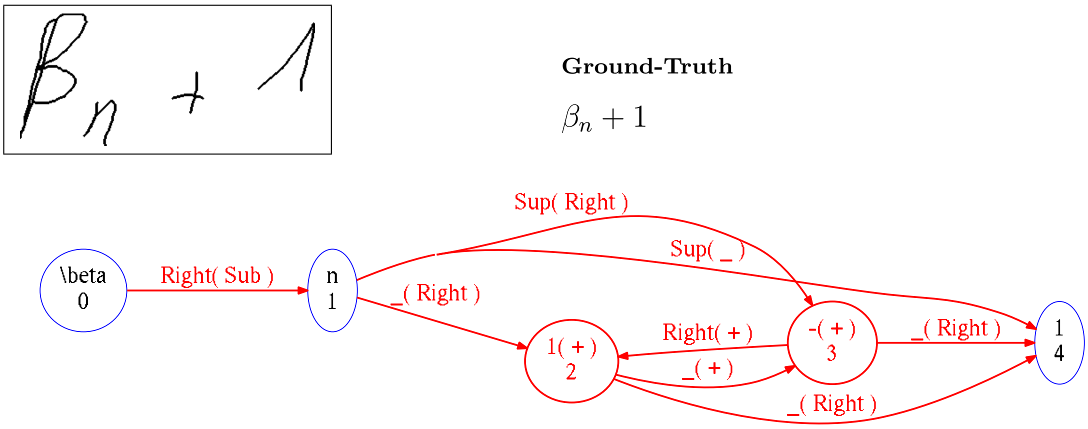


Figure 5 Structure Confusion Histogram for Target '$x+$.' These results are represented in an HTML file, computed using the *Nan* outputs for the 2014 test set. The first column shows the different stroke-level ground-truth graphs and total number of misrecognitions, numbered: 1 (4 strokes), 2 (3 strokes), 3 (2 strokes), and 4 (5 strokes). The remaining columns in each row show specific errors and their counts for each ground truth stroke pattern. The 'tick' (selection) boxes allow individual files in which errors occur to be selected and exported. A number of confusions have been omitted for space.

Table 5 Symbol Bigram Error Counts (CROHME 2014). Errors are shown for each system, globally for all systems (All) and using the merged label output from all systems. The top-3 errors in each column are shown in bold.

	Number of Errors								
Bigram	**All**	**Merged**	**Mys**	**Nan**	**Ria**	**Rib**	**Sap**	**Tok**	**Val**
$\frac{1}{}$	358	1	5	27	**60**	**91**	**108**	**52**	15
$x+$	285	1	11	**55**	41	41	56	**55**	**26**
$(x$	224		5	**32**	**44**	32	52	**50**	9
$+1$	224		7	**34**	29	44	**67**	38	5
$=-$	209		6	20	34	49	**60**	22	18
$=1$	180			17	33	**52**	46	26	5
00	160			31	**53**	31	20	17	5
$-=$	157	1		17	21	**55**	39	15	7
$x\times$	145	6	**24**	22	20	24	22	10	**23**
$\times x$	141	5	**23**	20	19	24	22	10	**23**
$f($	136		4	30	21	18	21	19	**23**
$\frac{}{n}$	130	**8**	8	16	21	28	24	21	12
$\sum^{n}$	114	**12**	**17**	17	17	12	17	17	17
$\sum_{i}$	63	**8**	9	9	9	12	11	9	9



Ground-Truth

$\beta_n + 1$

Output

βn^{-1^1}

Figure 6 Misrecognized Expression from *Nan* System and Corresponding Label Graph Error Visualization. The five input strokes are numbered from 0-4, appearing at the bottom of each node. Red nodes and edges represent incorrect labels, with the correct label shown in brackets. This sample illustrates one error for the '+1' pattern: the left-right link between the '+' and the '1' is wrong (*Sup* instead of *Right*) and '+' is over-segmented into '−1.' *Note:* on edges, '_' represents no relationship.

but in taking into account a more global or local context. Furthermore, we show that having efficient tools to analyze complex elements like math expressions is important. They can highlight some specific problems which can be hidden in the mass of information. The merging experiment shows that the different systems have different strengths which are often complementary. Thus, it is important that the community would continue to share the tools, datasets and the part of their systems in order to capitalize the efforts done by each one. Finally, it could be concluded that the organization of CROHME is indeed a huge effort which leads to setting up a full framework for handwritten math recognition and it has a tremendous contribution for advancement of the related research field.

To conclude, exciting perspectives for this field of research can be suggested. One of the most promising one would be to incorporate in a deeper way statistical language models. Using basic n-gram, or more elaborated skip-gram, models defined either at the symbol or at the sub-expression level seems appealing. Such models have proved to be very efficient to improve text recognition performances. Of course, the adaptation is not straightforward because of the nature of the 2D structures which are not present in regular texts. Extending this concept, it may be tempting to get rid of the use of formal CFG to guide the interpretation stage, and to consider more semantic approaches, so that mathematical concepts could be revealed. Following this idea, it would be interesting to look in the direction of Continuous Bag of Words for computing vector representations of sub-expressions.

Acknowledgements We thank Drs. Kim and Kim (KAIST, South Korea), for their help during the first CROHME. Part of this research is done under the National Science Foundation (USA) Grant No. IIS-1016815.

References

1. R. H. Anderson, "Syntax-directed recognition of hand-printed two-dimensional mathematics," in *Symposium on Interactive Systems for Experimental Applied Mathematics: Proceedings of the Association for Computing Machinery Inc. Symposium*, (New York, NY, USA), pp. 436–459, ACM, 1967.
2. H. Mouchère, C. Viard-Gaudin, D. H. Kim, J. H. Kim, and G. Utpal, "Crohme2011: Competition on recognition of online handwritten mathematical expressions," in *Proceedings of the 11th International Conference on Document Analysis and Recognition (ICDAR)*, (Beijing, China), 2011.
3. H. Mouchère, C. Viard-Gaudin, D. H. Kim, J. H. Kim, and G. Utpal, "Icfhr 2012 - competition on recognition of on-line mathematical expressions (crohme 2012)," in *Proceedings of the 13th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, (Bari, Italy), 2012.
4. H. Mouchère, C. Viard-Gaudin, R. Zanibbi, U. Garain, D. H. Kim, and J. H. Kim, "Icdar 2013 crohme: Third international competition on recognition of online handwritten mathematical expressions," in *Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR)*, (Washington, DC, USA), August 2013.
5. H. Mouchère, C. Viard-Gaudin, R. Zanibbi, and G. Utpal, "Icfhr 2014 - competition on recognition of on-line mathematical expressions (crohme 2014)," in *Proceedings of the 14th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, (Creta, Greece), 2014.
6. K. Chan and D. Yeung, "Mathematical expression recognition: a survey," *International Journal on Document Analysis and Recognition*, vol. 3, no. 1, pp. 3–15, 2000.
7. K. Sain, A. Dasgupta, and U. Garain, "Emers: a tree matching-based performance evaluation of mathematical expression recognition systems," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 14, no. 1, pp. 75–85, 2011.
8. R. Zanibbi, H. Mouchère, and C. Viard-Gaudin, "Evaluating structural pattern recognition for handwritten math via primitive label graphs," in *IS&T/SPIE Electronic Imaging*, pp. 865817–1 – 865817–11, International Society for Optics and Photonics, 2013.
9. D. Blostein and A. Grbavec, "Recognition of mathematical notation," in *Handbook of Character Recognition and Document Image Analysis*, pp. 557–582, World Scientific Publishing Company, 1997.
10. R. Zanibbi and D. Blostein, "Recognition and retrieval of mathematical expressions," *Int'l J. Document Analysis and Recognition*, vol. 15, no. 4, pp. 331–357, 2012.
11. F. Cajori, *A History of Mathematical Notations*. Chicago, Illinois: The Open Court Publishing Company, 1929. 2 vols.
12. K. Marriott, B. Meyer, and K. B. Wittenburg, "Visual language theory," ch. A Survey of Visual Language Specification and Recognition, pp. 5–85, New York, NY, USA: Springer-Verlag New York, Inc., 1998.
13. A.-M. Awal, H. Mouchère, and C. Viard-Gaudin, "Towards handwritten mathematical expression recognition," in *Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1046–1050, 2009.
14. S. Quiniou, H. Mouchère, S. Saldarriaga, C. Viard-Gaudin, E. Morin, S. Petitrenaud, and S. Medjkoune, "Hamex - a handwritten and audio dataset of mathematical expressions," in *Proceedings of the 11th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 452–456, 2011.
15. J. Stria, M. Bresler, D. Průša, and V. Hlavc, "Mfrdb: Database of annotated on-line mathematical formulae.," in *Proceedings of the 13th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 542–547, 2012.
16. F. D. J. Aguilar and N. S. Hirata, "Expressmatch: a system for creating ground-truthed datasets of online mathematical expressions," in *Proceedings of 10th IAPR International Workshop on Document Analysis Systems (DAS)*, pp. 155–159, IEEE, 2012.
17. S. MacLean, G. Labahn, E. Lank, M. Marzouk, and D. Tausky, "Grammar-based techniques for creating ground-truthed sketch corpora," *International Journal*

on Document Analysis and Recognition (IJDAR), vol. 14, no. 1, pp. 65–74, 2011.

18. U. Garain and B. B. Chaudhuri, "A corpus for ocr research on mathematical expressions," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 7, no. 4, pp. 241–259, 2005.
19. R. Zanibbi, A. Pillay, H. Mouchere, C. Viard-Gaudin, and D. Blostein, "Stroke-based performance metrics for handwritten mathematical expressions," in *Proceedings of the 11th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 334–338, IEEE, 2011.
20. F. Ávaro, J.-A. ánchez, and J.-M. Benedí, "An image-based measure for evaluation of mathematical expression recognition," in *Pattern Recognition and Image Analysis* (J. Sanches, L. Micó, and J. Cardoso, eds.), vol. 7887 of *Lecture Notes in Computer Science*, pp. 682–690, Springer Berlin Heidelberg, 2013.
21. U. Garain and B. Chaudhuri, *OCR of Printed Mathematical Expressions*, pp. 235–259. Springer, 2007. 10.1007/978-1-84628-726-8_11.
22. E. Tapia and R. Rojas, "A survey on recognition of on-line handwritten mathematical notation," tech. rep., Free University of Berlin, January 2007.
23. D. Blostein and R. Zanibbi, *Processing Mathematical Notation*, ch. 5.6. Handbook of Document Image Processing and Recognition, Springer Reference, Springer, 2014.
24. F. Simistira, V. Katsouros, and G. Carayannis, "A template matching distance for recognition of on-line mathematical symbols," in *Proceedings of the 11th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, (Montréal), pp. 415–420, 2008.
25. J. Stria and D. Průša, "Web application for recognition of mathematical formulas," in *Proc. Conf. Theory and Practice of Information Technologies*, (Vrátna dolina, Slovak Republic), pp. 47–54, 2011.
26. J. Stria, M. Bresler, D. Průša, and V. Hlavác, "Mfrdb: Database of annotated on-line mathematical formulae," in *Proceedings of the 13th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 542–547, 2012.
27. A.-M. Awal, H. Mouchère, and C. Viard-Gaudin, "Improving online handwritten mathematical expressions recognition with contextual modeling," in *Proceedings of the 12th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 427–432, 2010.
28. A.-M. Awal, H. Mouchère, and C. Viard-Gaudin, "A global learning approach for an online handwritten mathematical expression recognition system," *Pattern Recognition Letters*, vol. 35, no. 1, pp. 68 – 77, 2014. Frontiers in Handwriting Processing.
29. L. Hu and R. Zanibbi, "HMM-based recognition of online handwritten mathematical symbols using segmental k-means initialization and a modified pen-up/down feature," in *Proc. Int'l Conf. Document Analysis and Recognition*, (Beijing, China), pp. 457–462, Sept. 2011.
30. L. Hu, K. Hart, R. Pospesel, and R. Zanibbi, "Baseline extraction-driven parsing of handwritten mathematical expressions," in *Proc. Int'l Conf. Pattern Recognition*, (Tsukuba Science City, Japan), pp. 326–330, Nov. 2012.
31. L. Hu and R. Zanibbi, "Segmenting handwritten math symbols using AdaBoost and multi-scale shape context features," in *Proc. Int'l Conf. Document Analysis and Recognition*, (Washington, USA), pp. 1180–1184, 2013.
32. Y. Eto and M. Suzuki, "Mathematical formula recognition using virtual link network," in *Proc. Int'l Conf. Document Analysis and Recognition*, (Seattle, USA), pp. 762–767, 2001.
33. K. Davila, S. Ludi, and R. Zanibbi, "Using off-line features and synthetic data for on-line handwritten math symbol recognition," in *Proceedings of the 14th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, (Crete, Greece), pp. 323–328, 2014.
34. R. Zanibbi, D. Blostein, and J. R. Cordy, "Recognizing mathematical expressions using tree transformation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 11, pp. 1455–1467, 2002.
35. F. Alvaro and R. Zanibbi, "A shape-based layout descriptor for classifying spatial relationships in handwritten math," in *ACM Symp. Document Engineering*, (Florence, Italy), pp. 123–126, 2013.
36. M. Liwicki and H. Bunke, "Feature selection for hmm and blstm based handwriting recognition of whiteboard notes," *International Journal on Pattern Recognition and Artificial Intelligence (IJPRAI)*, vol. 23, no. 5, pp. 907–923, 2009.
37. M. Celik and B. Yanikoglu, "Mathematical formula recognition using a 2D stochastic graph grammar," in *Proc. Int'l Conf. Document Analysis and Recognition*, (Beijing, China), pp. 161–166, 2011.
38. F. Julca-Aguilar, N. Hirata, C. Viard-Gaudin, H. Mouchère, and S. Medjkoune, "Mathematical symbol hypothesis recognition with rejection option," in *Proceedings of the 14th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, (Crete, Greece), pp. 500 – 504, 2014.
39. D. Le, T. V. Phan, and M. Nakagawa, "A system for recognizing online handwritten mathematical expressions and improvement of structural analysis," in *Proceedings of 11th IAPR International Workshop on Document Analysis Systems (DAS)*, (Tours, France), 2014.
40. B. Zhu, J. Gao, and M. Nakagawa, "Objection function design for MCE-based combination of on-line and off-line character recognizers for on-line handwritten Japanese text recognition," in *Proceedings of the 11th International Conference on Document Analysis and Recognition (ICDAR)*, (Beijing, China), pp. 594–599, 2011.
41. A. Lee and M. Nakagawa, "A tool for ground-truthing on-line handwritten mathematical expressions," in *International Graphonomics Society Conf.*, (Nara, Japan), 2013.
42. F. Alvaro, J. Sanchez, and J. Benedi, "Recognition of printed mathematical expression using two-dimensional stochastic context-free grammars," in *Proc. Int'l Conf. Document Analysis and Recognition*, (Beijing, China), pp. 1225–1229, 2011.
43. F. Alvaro, J. Sanchez, and J. Benedi, "Recognition of online handwritten mathematical expressions using 2D stochastic context-free grammars and Hidden Markov Models," *Pattern Recognition Letters*, vol. 35, pp. 56–67, 2014.
44. F. Alvaro, J. Sanchez, and J. Benedi, "Offline features for classifying handwritten math symbols with recurrent neural networks," in *Proc. Int'l Conf. Pattern Recognition*, p. (to appear), 2014.
45. G. Labahn, E. Lank, S. MacLean, M. S. Marzouk, and D. Tausky, "Mathbrush: A system for doing math on pen-based devices," in *Proc. Document Analysis Systems*, (Nara, Japan), pp. 599–606, 2008.
46. S. MacLean and G. Labahn, "A new approach for recognizing handwritten mathematics using relational grammars and fuzzy sets," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 16, no. 2, pp. 139–163, 2013.

47. P. A. C. Chou, "Recognition of equations using a two-dimensional stochastic context-free grammar," in *Visual Communications and Image Processing IV* (W. A. Pearlman, ed.), vol. 1199 of *SPIE Proceedings Series*, pp. 852–863, 1989.
48. R. Zanibbi, D. Blostein, and J. R. Cordy, "Recognizing mathematical expressions using tree transformation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, pp. 1455–1467, 2002.