A Structural Analysis Approach for Online Handwritten Mathematical Expressions

B.Q. Huang† **and M-T. Kechadi**

School of Computer Science and Informatics,
University College Dublin, Belfield, Dublin 4, Ireland

Summary
This paper proposes a structural analysis approach for mathematical expressions based on the Attribute String Grammar and the Baseline Tree Transformation approaches. The approach consists of geometrical feature extraction, parsing structure and expression analysis steps. The algorithm for structure parsing uses baselines, which are represented by geometrical features to recursively decompose the hierarchical levels. During a hierarchy decomposition of an expression, it predicts the relationships along the symbols, and then groups the symbols into a number of sub-expressions and a set of basic units. The basic units are added into the tree and the same parsing process is performed on these sub-expressions. Once all sub-expressions are parsed, expression analysis is carried out to translate the tree into any desired syntax (e.g. Latex, Lisp). The experiments are performed to test the efficiency of this proposed structural analysis approach.

Key words:
Online handwritten mathematical expression recognition, Structural analysis, Baseline, Multilayer Perceptron Neural Networks, Support Vector Machines.

1. Introduction

One of the advantages of pen-based interfaces (e.g. pocket PC, digital tablet) is that they allow users to handwrite mathematical expressions on an electronic tablet into the word processors. Consequently, unlike traditional input interfaces, such as keyboards, pen-based interfaces are much more preferred by users for producing mathematical documents.

The process of mathematical expression recognition generally consists of three steps: segmentation, symbol recognition and structural analysis. The segmentation is used to separate symbols of an expression. Several approaches have been proposed for segmentation and symbol recognition. More details about segmentation and symbol recognition techniques and their challenges can be found in the literature [13-16, 23, 26, 27, 29, 30]. This paper focuses mainly on structural analysis.

In online handwritten mathematical expressions, the structural analysis is very challenging problem. Unlike a linear word, a mathematical expression is a two dimensional structure. For instance, a mathematical expression consists of a number of symbols, which are connected by some spatial relations, such as super and sub-scripts. The spatial relationships among the symbols are of different types, because the symbols' sizes and placements characterising them vary a lot. In addition, there are different types of symbols; each type has its own grouping rules. For instance, digits and letters can be grouped together to represent a specific meaning (e.g. 123.05 and sin); since binding operators (e.g. $\sum$, $\int$) dominate the sub-expressions around them, these sub-expressions should be grouped as individual units. For all these reasons, it is not easy to define a structural analysis technique for handwritten mathematical expressions. Few techniques for structural analysis were introduced in the literature and major ones are outlined below.

The Projection Profile Cutting [22, 28] projects pixels of an expression onto x- and y-axes to form x- and y-histograms, then it divides the expression into a number of components along the histograms. Although this method can quickly parse the structure of an expression, it fails in detecting some scripts (e.g. super/sub-scripts).

The Structure Specification Scheme [8] uses a tree to represent a mathematical expression, based on the definition of operator range, precedence and dominance. But it also cannot deal with subscripts and superscripts as operators.

The basic idea of the Graph grammar [11] was used for mathematical expressions by Lavirotte and Pottier [19]. They defined a set of rules in graph grammars to match a given condition. These rules describe the mathematical relationships and they apply a bottom-up parsing algorithm to obtain a hierarchical tree.

The Stochastic grammar [9] finds the likely parse tree of an expression. He defined that the pixels are the only terminal symbols in his grammar. For mapping non-terminal symbols to pixels, he used Hamming distance to compare the rectangle arrays of pixels at each location of the image to their corresponding templates in the font dictionary, and obtained their associated probability. Finally, the probability of a parse tree can be obtained by multiplying the probabilities for all production rules used

Manuscript received July 5, 2007
Manuscript revised July 25, 2007

in a successful parse. This approach does not need segmentation, but it is computationally expensive.

Lee et al. [21, 20] introduced the Procedural translation to recognize the structure of an expression. Their algorithm starts by grouping the symbols, which deviate from the centre of an expression into a number of units, and then re-orders this set of the units along with the remaining symbols according to the y-coordinate of the centre point of the units or symbols. Thus, the output string can be obtained by iteratively using this group-order procedure. However, this approach is not easy to maintain and extend.

Chan and Yeung [6] proposed the hierarchical decomposition method based on definite clause grammar (DCG), the left-factored rules and binding symbol pre-processing. However, this approach has a difficulty of dealing with high-level mathematical expressions, especially expressions with brackets, matrixes, etc.

The approach of the Attributed String Grammars (ASG) [2] iteratively groups the symbols of an expression from top to bottom. The idea is to divide an expression into a number of sub-expressions by comparing the symbol's position to the centre line. This process is iteratively applied to all of the sub-expressions and outputs a parsing tree. The resulting tree is put through a semantics checking process and transformed into a desired syntax. Recently, Zanibbi et al. [25] introduced a Baseline Tree Transformation approach (BST), which is based on the String Attribute Grammar approach. They parse the hierarchical levels by recursive baselines (called layout pass) in such a way as to construct a tree.

Among all these methods, the ASG technique is the most complete and popular for analysing the structure of a mathematical expression. In addition, it easily maintains any operations in such a way as to reduce the errors. However, This ASG approach depends heavily on the symbol recognition. Their spatial functions used to calculate the spatial relations between the symbols assume that these symbols are correctly recognised and some threshold values of these functions are not easy to set.

In this paper we propose an efficient structural analysis approach based on the Attributed String Grammar approach. We extract geometrical features of the symbols. Then we use a Multilayer Perceptron Network (MLP) or Support Vector Machine (SVM) to recursively find the baselines and parse the structure of an expression from left to right and from top to bottom. Finally, the obtained parsing tree is passed through an expression analyser that includes the syntax checking and semantic analysis.

2. Structural Analysis Approach

Alternatively, based on the ASG and BST approaches, we propose a structural analysis approach. Our approach consists of geometrical feature extraction, parsing structure and expression analysis stages as shown in Figure 1(b). Given an expression the geometrical feature extraction not only extracts the bounding boxes of symbols and the centers of the boxes, but also extracts the mass centers of the symbols. In the parsing structure step, the new approach parses the hierarchy of the expression and simultaneously performs lexical checking, based on the Attribute String Grammars with the Neural Network and Support Vector Machine. Finally, the new approach performs the expression analysis. Figure 1(a) shows the BST approach [25], which consists of geometrical feature extraction, hierarchical parsing or layout pass, lexical checking and expression analysis steps.

In contrast to the BST, the new approach uses the mass points instead of the labels of symbols for predicting the spatial relationships, and our approach uses the MLP or SVM instead of using the function with a set of the thresholds for predicting the relationships between the symbols. Therefore, obtained prediction accuracy by the MLP/SVM is higher than the one obtained by using thresholds. Thus, the new approach can reduce dependency on the symbol recognition. In addition, the new approach integrates hierarchical parsing and lexical checking steps into one step -- the parsing structure phase in such a way as to avoid the errors that will take place in future parsing process. For the final steps of the both approaches, the functionality and methods of the expression analysis are the same. The details of each step of the new structural analysis approach are described in the following sections.

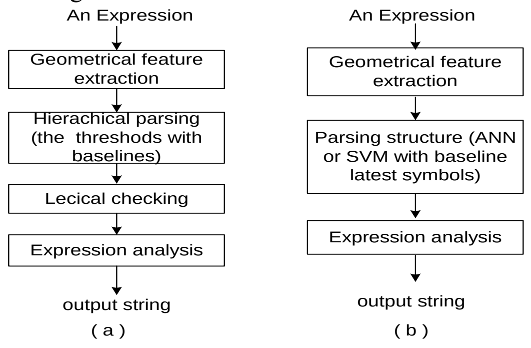
An Expression
Geometrical feature extraction
Hierachical parsing (the thresholds with baselines)
Lecical checking
Expression analysis
output string
(a)
An Expression
Geometrical feature extraction
Parsing structure (ANN or SVM with baseline latest symbols)
Expression analysis
output string
(b)

Fig. 1 (a) represents the Baseline Tree Transformation approach, (b) represents our approach.

2.1 Geometrical feature extraction

The input data to structure analysis is a set of the symbols from segmentation and symbol recognition steps. They are sorted from left to right and from top to bottom. The first operation of this structural analysis is to extract the geometrical features of the symbols of an expression. Assume that a symbol consists of N points, *s={p₁(x₁,y₁),…,p_N(x_N,y_N)}*, the geometrical features v_s of the symbol can be calculated as follows:

- Bounding box:

$$x_{\min} = \min_{1\le i\le N} x_i\ ,\ y_{\min} = \min_{1\le i\le N} y_i$$
$$x_{\max} = \max_{1\le i\le N} x_i\ ,\ y_{\max} = \max_{1\le i\le N} y_i \qquad (1)$$

- The centre point $p_c(x_c,y_c)$: the centre position of the bounding box

$$x_c = \frac{(x_{\min} + x_{\max})}{2}\ ,\quad y_c = \frac{(y_{\min}+y_{\max})}{c2} \qquad (2)$$

- Mass centre point:

$$y_m = \frac{\sum_{i=1}^{N} y_i}{N}\ ,\quad x_m = \frac{\sum_{i=1}^{N} x_i}{N} \qquad (3)$$

Given two symbols s_i and s_{i+1}, the spatial relationship between them depends on their geometrical features $F=\{V_{s_i},V_{s_{i+1}}\}$. For an expression S= {s$_i$, s$_{i+1}$,…, s$_M$}, the geometrical feature set is $F = \{V_{s_1},V_{s_2},...,V_{s_M}\}$.

2.2 Parsing Structure

There are two very important tasks that are often performed and can affect the efficiency of the whole parsing process: the determination of spatial relationships between two symbols and the grouping symbols. Therefore, the approaches to achieve these tasks are described before giving the details of the algorithm of parsing structures.

2.2.1 Determination of spatial relationships

The relationship between two symbols depends on the spatial relation between them, which is "RIGHT-UP", "RIGHT-DOWN", or "RIGHT", etc. Figure 2 shows nine types of spatial relationships based on the centre symbol, which is highlighted by the red box (e.g., for $\sqrt{x}$, the spatial relationship between the symbols 'x' and $\sqrt{\ }$ is "INTER").

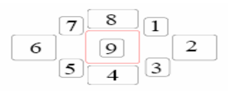

Fig.2 The layout locations: the black boxes 1, 2, 3, 4, 5, 6, 7, 8 and 9 represent that spatial relationships of "RIGHT-UP", "RIGHT", "RIGHT-DOWN", "DOWN", "LEFT-DOWN", "LEFT", "LEFT-UP", "UP" and "INNER", with respect to the red box, respectively.

Given the geometrical features of two symbols, one must predict their relationship and return one of the 9 spatial relationships. However, due to the ambiguous position of the symbols, it is not an easy task. Most of the approaches use linear functions with thresholds to inspect the horizontal and vertical displacements. However, the size of the symbols needs to be considered and also if the symbols were not recognised correctly the functions with a set of thresholds more likely give wrong spatial relationships. In addition, when using the geometrical features, the baseline function is not suitable for calculating the relationship between two symbols. MLP [24] has been successfully applied in pattern recognition. More recently, SVM, which is one of elegant machine learning methods [4, 5], has proven to be very efficient. Therefore, our structural analysis approach uses the MLP and SVM as spatial function G(.) to calculate the spatial relationships between two symbols.

- **Multilayer Perceptron Networks**

The MLP has been widely used in pattern recognition. The standard MLP is a supervised feed-forward neural network, which consists of one input layer, a number of hidden layers and one output layer. Generally, the activation function is a sigmoid function and it uses the Back-Propagation (BP) learning algorithm for training. More details about MLP can be found in [24].

- **Support Vector Machines**

An SVM classifier can be trained by finding a maximal margin hyper-plane in terms of a linear combination of subsets (support vectors) of the training set. If the input feature vectors are nonlinearly separable, SVM maps the data into a high dimensional feature space by using the kernel trick [4] and then classifies the data by the maximal margin hyper-plane using the following function:

$$f(x) = \operatorname{sgn}\left(\sum_{i}^{M} y_i\alpha_i\phi(x_i,x)+\delta\right) \qquad (4)$$

where M is the number of samples in the training set, x_i is a support vector with $\alpha_i > 0$, ϕ is a kernel function, x is an unknown sample feature vector, and δ is a threshold.

The parameters $\{\alpha_i\}$, can be obtained by solving a convex quadratic programming problem subject to linear constraints [5]. In practice polynomial kernels and Gaussian radial basis functions (RBF) are usually used for kernel functions. δ can be obtained by taking into account the Karush-Kuhn-Tucker condition [5], and choosing any i for which $\alpha_i > 0$ (i.e. support vectors). However, it is safer in practice to take the average value of δ over all the support vectors.

Although SVM was originally developed for binary classification, it was also used successfully to solve multi-classification problems. The experiments given in [12] indicated that the one-versus-one method developed in [18] was more suitable for practical use than the other methods.

2.2.2 Grouping Symbols

In order to reduce the errors in interpreting the symbols, and to simplify the complexity of parsing of mathematical expression hierarchies, the definition of grouping symbol is traditionally made for digits, strings, function names, and special symbols [3,7]. In this study, in order to convert the 2-D expression into its equivalent 1-D representation, the definition of grouping symbol is extended for any symbol. A group of symbols is defined as a set of consecutive symbols, with a specific meaning, according to the lexicons and the spatial relationships between the centre symbol and its neighbours. The spatial relationships is predicted by the function G(.) as detailed in the sub-section 2.2.1. If a unit represents a specific meaning, it is called a basic unit. If the spatial relationships along these consecutive symbols or are "RIGHT", it indicates that they are in the same horizontal line. For instance, 96 is only one basic unit due to the spatial relationship "RIGHT" between them; but 9^6 consists of two basic units: 9 and 6, because the spatial relationship between them is an operator “POWER”. Similarly, some letters together may form one function name as a basic unit, such as ‘cos’, ‘log’, ‘ln’, etc. For example, a_{i+2}^{x+1} can be grouped into two groups based on the symbol a: {x+1} in "RIGHT-UP" and {i+2} in "RIGHT-DOWN".

2.2.3 Algorithm

Unlike linear words, the structures of 2-D mathematical expressions are complex. The symbols of an expression are of different hierarchical levels, each of which is corresponded by a baseline. If a baseline is detected, the symbols on this baseline can be grouped as a traditional word. The lexical checking is performed by dividing these symbols into a number of basic units, which will be added into the tree. If the branches of these symbols exist, each of them is treated as a sub-expression. Thus, the whole structure of an expression can be parsed by recursively finding the baselines, and consequently the expression is represented by the tree.

Most of the baseline approaches [2,25] use the functions with thresholds to compare the y-centre (on y-axes) values of the symbol's bounding boxes. However, the baselines of the symbols not only rely on the y-centre values, but also depend on the coordinates, centre and mass points of symbols. Therefore, an expression might be incorrectly divided into a number of sub-expressions.

In order to improve the ASG or BST approaches, our approach uses the geometrical features (bounding boxes, centre and mass points) to represent the baselines, and uses the MLP or SVM to calculate the spatial relationships between the symbols and to detect the branches (grouping symbols). The detection of the symbols along a baseline does not only depend on the y-centre of the bounding box of the, but also on the geometrical features. If a number of symbols are on the same baseline, the spatial relationship among these symbols (from left to right) is "RIGHT". The basic algorithm of the our approach for parsing the structure of an expression can be described as follows: initially, consider the first symbol as the centre symbol and use its attributes to present the baseline, then use spatial function (MLP or SVM) to calculate the spatial relationships between the centre symbol and other symbols and identify the relationship between its neighbourhood. Secondly, group the symbols (excluding the centre symbol and the symbols that share the relationship "RIGHT" with the centre symbol) into different groups according to the 9 types of spatial relationships, and treat each of these groups as a sub-expression. Thirdly, the symbol that shares the relationship "RIGHT" with the centre symbol is treated as the centre symbol (change centre symbol) and the lexical checking is carried out between the former and current centre symbols, and add them to the tree. Fourthly, the same process is performed on the remaining symbols until all the symbols inputted so far have been processed. Finally, the same process (the above four steps) is applied to parse all of the sub-expressions. Consequently, the expression is represented by a parsing tree when all the sub-expressions are completed.

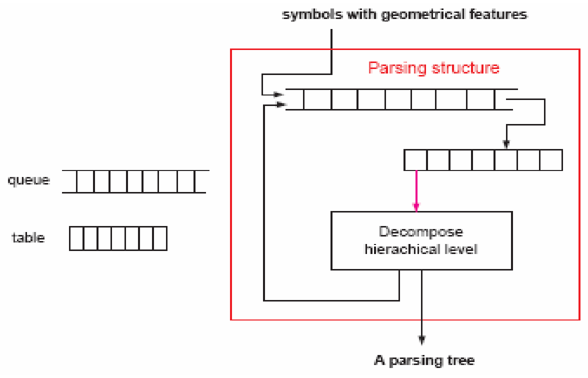
symbols with geometrical features
Parsing structure
queue
table
Decompose hierachical level
A parsing tree

Fig. 3. The process of parsing the structure of an expression.

Figure 3 details the parsing structure component shown in Figure 1. It consists of one queue and one table. The queue stores a number of expressions and sub-expressions. The elements in the queue are accessed based on the first-in-first-out rule. The symbols of an expression with their geometrical features are stored in the table from left to right. The symbols along with their geometrical features of an entire expression, which have already been recognised in previous steps, are inserted into the queue. Before initiating the parsing process one has to read one expression or sub-expression into the table. The output of this process is a parsing tree. This tree is composed of a number of nodes and each node consists of a set of symbols S, one spatial relationship representing a baseline, and a set of children nodes V = $\{v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8,v_9\}$. These children nodes are used to collect the branches, so that they are called branch nodes. The nodes v_1, v_3, v_4, v_5, v_7, v_8, and v_9 are used to collect the symbols of the block 1, 3, 4, 5, 7, 8 and 9, respectively (see Fig. 2). Thus, any sub-expression is represented by a node. The process of parsing the structure of an expression can be described by two phases and they are described below:

- **Phase one (Preparation Phase):**

When an expression is ready to be parsed, a new node is created to record its symbols and submitted to the queue. If the expression is not sub-expression, its node is considered as the root. The process starts by selecting one node from the queue and stores its symbols along with its geometrical features in the table, from left to right.

- **Phase two (Parsing Tree Construction):**

The aim of this phase is to detect all the hierarchical levels of an expression by identifying all the branches of the tree and to perform lexical checking. The symbols with the geometrical features are sequentially explored from left to right in the table. The centre symbol S_c is defined and well as its neighbours. Initially, S_c is the first written symbol of the expression. If the relationship with one of its neighbours is "RIGHT", then that symbol becomes the centre. If one of the special symbols (e.g. divide) shares the relationship "DOWN" with the centre, the centre symbol S_c becomes a branch symbol, and the special symbol becomes the centre S_c. The symbols in the main hierarchical level (baseline) are grouped into several basic units and added to the parse tree. Each branch is treated as a sub-expression and submitted to the queue so that the process will restart again from the first phase. Assume that the centre symbols are stored in a list W. In order to clearly describe the algorithm for parsing an expression, we explain in the following the steps of the algorithm without considering the special symbols (e.g. divide, $\sum$, $\int$ and $\surd$) and then extend it to any special symbol:

1) For the initialisation, the first symbol in the table is selected as the centre S_c.
2) Add the symbol S_c to the list W. If S_c is the last symbol, go to step 4. If S_c is not the last symbol, the symbols after the centre S_c are sequentially selected to calculate the spatial relationships with the centre symbol S_c, until the symbol S_r that shares relationship "RIGHT" with the centre symbol S_c is found (or the end of the symbol list S_e is reached).
3) If there is any symbol between S_c and S_r (or from the symbol after S_c to S_r, according to the spatial relationships, the symbols between S_c and S_r (or from the symbol after S_c to S_e) are grouped into block 1, 3, 4, 5, 7, 8 and 9 respectively (see Fig. 2). Then go to step 4. If there is no symbol between S_c and S_r, then S_r becomes the centre and go to step 2.
4) Group the symbols in W into basic units according to a given lexicon. Thus, new nodes are created (one for each unit) to load these basic units. The spatial relationship "RIGHT" is assigned for these nodes. The parent of the new nodes is the node from which the symbols were attached to (centre symbol). If a new node that holds the last basic unit is used to represent that unit, then the list W is empty, and S_r becomes the centre symbol and go to step 2. Simultaneously, these children nodes are submitted into the queue during the first phase. Consequently, the process will recursively decompose the next hierarchical level of the expression.

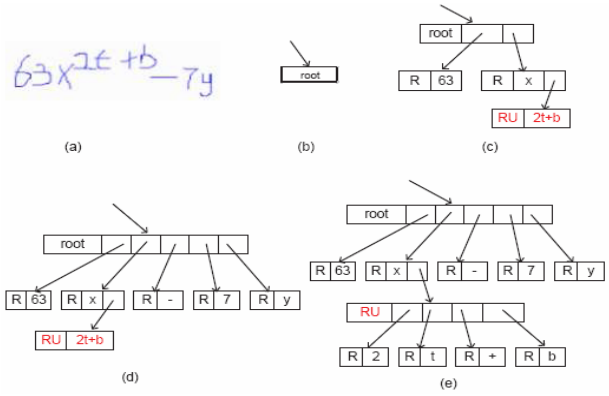

Fig.4. Parsing the structure of an expression that has no special symbols. (a) is the online handwritten expression; (b), (c), (d) and (e) are the instant trees. The symbols "R" and "RU" represent the spatial relationship "RIGHT" and "RIGHT-UP", respectively.

The process is used to decompose the hierarchical levels of the expressions without any special symbols. Fig. 4 shows that the expression $63\,x^{2t+b} - 7\,y$ is decomposed and its hierarchy is represented by a parse tree. According to phase 1, a root is created (see Fig. 4(b)), and all the symbols {6, 3, x, 2, t, +, b, -, 7, y} are moved into the table. In phase 2, according to steps 2 and 3, sequentially select the symbols "6, 3", and "x" as the centres, and add them to the list W (W={63x}). When x is the centre, the symbols "2, t, +", and "b" that are between x and – are identified as superscript. According to step 4, based on the given lexicon, the symbols "63x" are grouped into two basic units: "63" and "x". Two new nodes are then created to load these two basic units (see Fig. 4 (c)). Therefore, a child of the node that contains the symbol x is created to hold the superscript "$2t+b$". The nodes that contain the basic units are the children of the root. Similarly, the remaining symbols "-7y" define three basic nodes and are added to the root as children (see Fig. 4(d)). Consequently the hierarchical level that consists of those basic units is detected. Other levels are found by submitting the sub-expression using the same process recursively. Fig. 4(e) shows the whole parsing tree that represents the hierarchy of the given expression.

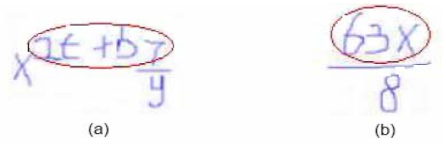

Fig.5. Two examples of expression where the special symbol "-" was written after the numerators. (a) the superscript of x and the numerator "-" are separated. (b) the hierarchical level of the numerator "63x" is swapped with the hierarchical level of the symbol "-".

However, when an expression has special symbols, the above process cannot be directly used to decompose the hierarchy of the expression, because the special symbols have their own grouping rules, which are different from the other symbols. If the numerator or max-limitation is written before the special symbol (e.g. "-", "∫", "√" or "∑") in an expression, the above process will incorrectly group the symbols, resulting in wrong hierarchical levels of the parsing tree. The expressions in which the numerator was written before the symbol "-", Fig. 5(a) shows that the superscript of the symbol "x" will contain the numerator that actually belongs to the symbol "-", and Fig. 5(b) shows that the expression will be incorrectly decomposed. Therefore, the above process has to be modified to handle special symbols. This can be achieved by modifying step 2 as follows:

2) When the centre symbol s_c is determined, it is firstly added to the list W. If s_c is the last symbol, go to step 4. Otherwise, the special relationships between its neighbours are identified. During the identification of the centre neighbours, the following steps are performed:

a) If the end of the symbol list s_e is reached, go to step 3.

b) If the symbol s_r that shares relationship "RIGHT" with the centre symbol s_c is found, and s_r is not the special symbol "-" or "∑", go to step 3.

c) When the symbol s_r that shares relationship "RIGHT" with the centre symbol s_c is found, and s_r is the special symbol "divide" or ∑", move the location of s_r in the table before the symbols that actually shares relationship "ABOVE" with s_r so that it can correctly be grouped with the centre symbol s_c. The location of special symbol s_r in the table is moved before the symbols and it is dealt with as follows: Firstly, according to the spatial relationships, find the set of the symbols s_{sa} that contains the symbols, which actually belong to the symbol s_r neighbourhood. Secondly suppose that s_r is the centre and use G(.) to calculate the spatial relationships between the symbol s_{sa}. Finally place the location of s_r before the symbols that share the spatial relationship "ABOVE" with s_r (see Fig. 6(c)). Change the location of the symbol s_r in the table, go to step 3.

d) If the symbol s_r is a special symbol (e.g. "divide", "∫" or "∑") and it is located under the centre symbol s_c, create two new nodes to swap the hierarchical levels. One new node is a child of the other. For these two new nodes, suppose that the parent and the child nodes are n_p and n_c respectively, and let the parent of the node which contains the centre symbol be P_c. The procedure of swapping hierarchical levels is as follows: Firstly, the special symbol is assigned to node n_p, the spatial relationships "RIGHT" and "UP" are assigned to node n_p and n_c respectively. Secondly, move all of the children from node P_c to node n_c as children. Thirdly, let node n_p become a child of the parent of node P_c. Fourthly, select this special symbol as the centre s_c, and go back to the start of the procedure.

By replacing the old step 2 by the new step, we obtained a robust process for decomposing an expression and constructing its corresponding parsing tree. Fig. 6 gives an example for decomposing the hierarchy of a mathematical expression that is written from left to right and from top to bottom. Fig. 6(a) is an online handwritten mathematical expression. Fig. 6 (c) shows that the position of symbols "7" and "-" in the table are swapped, when the symbol "x" that shares the spatial relationship "RIGHT" with the symbol "-" is found. Consequently, the symbol "-" is located earlier than symbol "7", and the symbols are easily separated into two parts: "2t+b" and "7", which belong to the symbol "x" and "-" respectively (see Fig. 6(d) and 6(e)). Fig. 6(d) shows the node that contains the symbols "2t+b" is a child of the node that contains the symbol "x". Fig. 6(e) shows that the node that contains the symbol "-" is the same hierarchical level with the node that contains "x". Fig. 6(f) shows that the hierarchy of the sub-expression "2t+b" is parsed and translated into a tree. Fig. 6(g) shows that the whole hierarchical tree of the expression, and the node that contains the symbol "-" is the grandparent for the nodes that contain "63", "x" or "-". Therefore, the hierarchical level that "-" locates is higher than the one for the numerator $\frac{63\ x^{2t+b}}{7\ y}$.

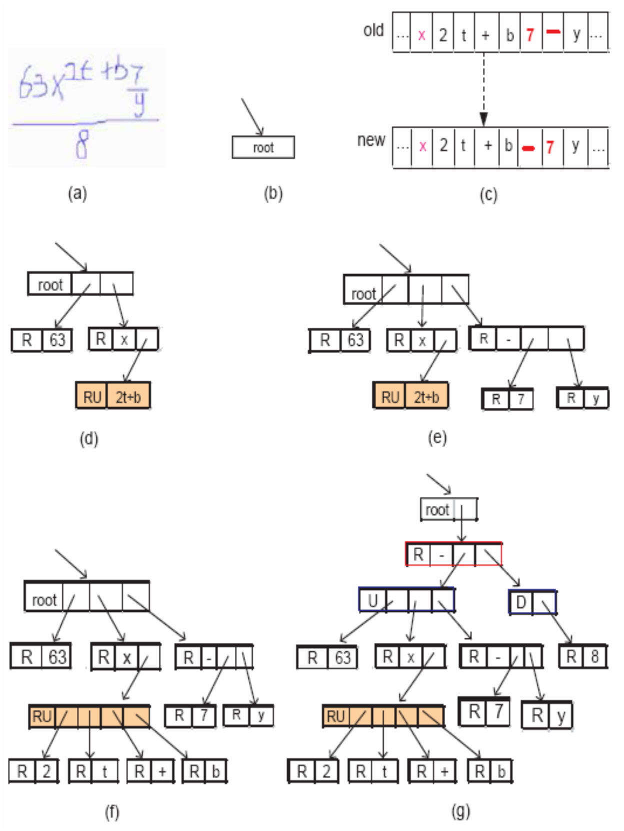

Fig.6. Parsing the structure of an expression with special symbols. (a) is the online handwritten expression. (c) shows the order of the symbols from the old table to the new one. (b), (d), (e), (f) and (g) are the instant trees. In the first field of the nodes, the symbols "R", "RU", "D" and "U" represent the spatial relationships "RIGHT", "RIGHT-UP", "DOWN" and "UP" respectively.

2.3 Expression analysis

In practice, poorly written and joined characters are common in a mathematical expression. To achieve an acceptable recognition rate, the contextual knowledge has to be used for online handwritten mathematical expression recognition, including syntactical and semantic knowledge. After the parsing structure, the 1-D representation is converted into a parsing tree and then the whole expression is analysed, which includes syntax checking and semantic analysis. In the current research, the completely established parsing tree is converted into Latex format [1]. At present, only a small set of grammar expressions for Latex is created. However, this set can be easily extended by adding new rules to the grammar.

3. Experimental results and Discussion

Two datasets are used in the experiments: one dataset is used to train the spatial functions (the SVM and MLP), the other is used to verify the whole structural analysis approach. These two datasets were collected by four researchers using a Trust tablet 200 connected to a Dell Optiplex GX620 PC. The dataset for training the spatial functions contains 1422 patterns (called spatial relation data). The other dataset consists of 112 expressions (40 short expressions and 72 long expressions). These 112 expressions are not only used to verify the efficiency of the proposed parsing approach, they are also used to generate 1822 spatial relation patterns for testing the efficiency of the spatial functions. These 1822 patterns are obtained by segmenting the symbols of the expressions. In the following, we introduce the segmentation and symbol recognition approaches used the experiments. This is followed by the experimental results outlining both the efficiency of our technique as well as comparing it to the existing approaches.

3.1 Segmentation and Symbol Recognition

Initially, an expression consists of a number of strokes, and each stroke is simply represented by a sequence of points. Symbol recognition that includes segmentation and identifies symbols was carried out first. In this paper, segmentation and symbol identification are simultaneously performed. The segmentation groups the strokes into a number of symbols by observing the centre points and the output confidence of the recognizer. Generally, a symbol consists of less than 4 strokes. Therefore, every 4 consecutive strokes are taken into consideration. The process of the stroke grouping is carried out as following: (1) calculate the bounding box with the centre point of each stroke according to Eq.1, (2) select two consecutive strokes stroke_{i+1} and stroke_{i+2} from the left. If the centre point of one stroke is inside the bounding box of the other stroke, or the two centre points of the two strokes are near the same location, the two strokes are combined into one hypothetical symbol S_{one}. If the centre of the third stroke $stroke_{i+3}$ is inside the bounding box of the hypothetical symbol S_{one} or the bounding box of the stroke stroke_{i+3} covers the centre of the symbol S_{one}, the third stroke $stroke_{i+3}$ is merged into the symbol S_{one}. When bounding boxes of the strokes do not cover the stroke $stroke_{i+3}$ are different symbols, the highest confidence (probability) from HMM-SNN outputs is selected to group the strokes $\{stroke_i$, i=1,2,3,4$\}$ into a number of symbols. This technique of grouping strokes into symbols by selecting the highest confidences can be found in [30]. When the symbol is found, the next four stokes will be considered. Thus the recursive segmentation process is used to group all the strokes in an expression into a number of symbols. Once the segmentation is done, the HMM-SNN method is used to identify the unknown symbols. The detail of the HMM-SNN method is given in [13, 14]. The symbol recognizer based on HMM-SNN can classify English characters, digits, common mathematical Greek symbols and operators. The information and the label of classes of symbols are passed from the symbol recognition stage to the structure analysis stage.

3.2 Experiments of Structural Analysis

The MLP and SVM were used as the spatial functions to predict the spatial relationships between the symbols of expressions, which is detailed in Section 2.2.1. The MLP with one hidden layer was trained based on the 1422 patterns. The number of input neurons is as the same as the number of the dimensions of a geometrical feature vector. The number of output neurons is the number of classifications of the spatial relationships. The number of hidden neurons of the MLP is equal to the average of the numbers of input and output neurons. 16, 13 and 9 are the numbers of the input, hidden and output neurons, respectively. The sigmoid function is selected as the activation function for the MLP. BP learning algorithm with the learning rate 0.3 and the maximum 500 cycles was used to train the MLP. The 4 folds for the Cross-Validation [10] were used to stop training the MLP. The highest cross-validation accuracy of 96.19% was achieved.

With the training data and 4 folds for the Cross-Validation, the SVM were trained to find the separating decision hyperplane that maximizes the margin of the classified training data. Two sets of values: the regularization term $\mathrm{C}\in\{2^8,2^7,\ldots,2^{-5}\}$ and $\sigma\in\{2^8,2^7,\ldots,2^{-4}\}$ of RBF were attempted to find the best parameters for multi-classification of the spatial relationships between symbols. All together, 169 combinations of C and σ were evaluated in the experiment. The optimal parameter set (C,σ) yielding a maximum classification accuracy was $(2^2,2^6)$. The achieved accuracy of the Cross Validation is 95.98%.

The 1822 testing patterns were used to test the MLP, the SVM and the threshold function of Zanibbi et. al. [25]. the MLP, the SVM and the threshold function achieved the accuracy of 92.81%, 95.77% and 81.28%, respectively (See the Table 4). The confusion matrixes of the classification of spatial relations are showed in the Tables 1, 2 and 3. These tables were obtained by the MLP, the SVM and the spatial function with the thresholds, respectively. In the tables, the spatial relationships "RIGHT-UP", "RIGHT-DOWN", "LEFT", "RIGHT", "DOWN", "LEFT-UP", "LEFT-DOWN", "UP" and "INNER" are represented by 1, 2, 3, 4, 5, 5, 6, 7, 8 and 9, respectively (as in Figure 2). The shows what spatial relationships are used often. The spatial relationships "RIGHT-UP", "RIGHT", "DOWN", "UP" and "RIGHT-DOWN" are the most often used. The spatial relationships "LEFT-UP" and "LEFT-DOWN" are not often used. The tables also show that what spatial relation are easily identified into the wrong one by the spatial functions.

Table 1. Confusion Matrix (rows: teaching input, columns: classification) obtained by the MLP.

C	1	2	3	4	5	6	7	8	9
1	335	3	0	22	0	0	0	0	0
2	1	401	0	62	0	0	1	0	1
3	0	0	43	1	0	1	2	0	0
4	2	13	0	561	0	0	0	0	0
5	1	0	0	0	160	0	0	8	0
6	0	0	0	0	0	2	0	0	0
7	0	0	0	0	0	0	0	0	0
8	0	0	0	1	8	3	0	148	0
9	0	0	0	1	0	0	0	0	41

Table 2. Confusion Matrix (rows: teaching input, columns: classification) obtained by the SVM.

C	1	2	3	4	5	6	7	8	9
1	342	3	0	15	0	0	0	0	0
2	9	451	0	4	0	0	0	0	2
3	0	0	45	1	0	0	1	0	0
4	3	16	0	556	0	0	0	0	1
5	0	0	0	0	163	0	0	6	0
6	0	0	0	0	0	2	0	0	0
7	0	0	0	0	0	0	0	0	0
8	0	0	0	0	16	0	0	144	0
9	0	0	0	0	0	0	0	0	42

Table 3. Confusion Matrix (rows: teaching input, columns: classification) obtained by the baseline function with the thresholds.

C	1	2	3	4	5	6	7	8	9
1	115	4	0	19	0	0	0	222	0
2	41	411	0	10	1	0	1	0	2
3	0	0	43	1	0	3	0	0	0
4	4	12	0	556	3	0	0	0	1
5	0	2	0	0	165	1	1	0	0
6	0	0	0	0	0	2	0	0	0
7	0	0	0	0	0	0	0	0	0
8	1	1	0	0	3	7	0	148	0
9	0	0	0	0	0	0	0	1	41

The above results were obtained by the MLP and SVM for the determination of the spatial relationship between only two symbols. However, they do not show mean that two-dimensional expressions can be correctly converted into their equivalent one-dimensional representation with the same high recognition rates (92.81% and 95.77%). Therefore, each entire expression in testing dataset was used to verify the efficiency of the proposed approach. Each expression data was converted into its equivalent one-dimensional syntactical representation that is the Latex text. Figure 7 shows some idea examples of expressions that are used for testing and training. Unlike the above classification problem, it is very difficult to use machines to verify the structure of an entire expression. Therefore, we directly verify whether the expression is correctly parsed by the proposed approach here.

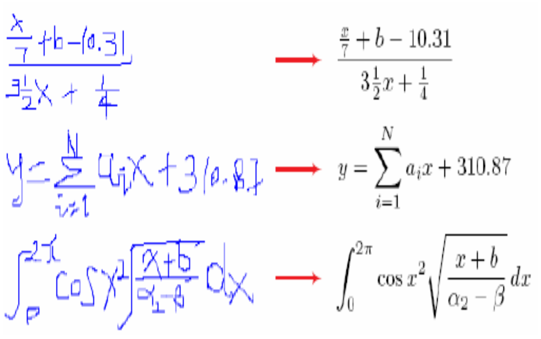
Fig. 7.Samples for correct recognition results

Table 4. The recognition results by different analysis approaches.

Methods	spatial Functions G(.) Rates %	40 Short Expressions	72 Long Expressions	Total 112 Expressions
		Rec. Rates %	Rec. Rate %	Rec. Rate %
BST	81.28	72.50	63.89	66.96
ASG-MLP	92.81	90.00	83.33	85.71
ASG-SVM	95.77	97.50	86.11	90.18

The BST of Zanibbi et. al. [25] and our proposed structural analysis approach with the spatial functions (them MLP and SVM) were used to recognize the structures of the 112 expressions. The recognition results of the approaches are showed in Table 4. In the Table 4, the recognition rates from our approach are higher than the one from BST approach (85%, 90.18% > 66.96%). From this table, the approaches more successfully recognize the short (or simple) expressions than the long (or complex) ones. This table shows that the spatial functions can influence or affect the entire the analysis process (e.g. structural parsing). For the new proposed approach, if the spatial function can achieve higher recognition rate of the spatial relationships between symbols, the entire approach can also achieve higher recognition rate of the expression's structures. For example, the analysis approach with the SVM (AGS-SVM) achieved higher accuracy than the one with the MLP (AGS-MLP). Because a single incorrect spatial relationship between two symbols can cause incorrect recognition of the whole structural of an expression, the recognition rates of 85.71% and 90.18% on the expressions are less than the recognition rates of 92.81% and 95.77% on the spatial relation data.

3. Conclusion

In this paper we propose an approach for structural analysis of online handwritten mathematical expressions to reduce the limitation of Attribute String Grammar and the Baseline Tree Transformation approaches, which strongly depend on the symbol recognition. Given an expression, the proposed approach extraction attributes (bounding boxes and mass points) of the symbols in the expression, then parse the structure of the expression to build up a tree, and finally perform expression analysis to translate the tree into any syntax desired language. In the parsing structure stage which is the most important process, the algorithm finds the centre symbols on the baselines which are presented by centre symbol's attributes, then uses the MLP or SVM to predict the relationships between the centre symbol and other symbols, and groups these symbols into a number of sub-expressions (branches) and a number of basic units. Finally, the basic units added to the tree and the sub-expressions are recursively decomposed by this process. The capability of this approach is demonstrated in terms of the achieved recognition rate on the created datasets from the tablet. The experimental results show the approach is well suited for structural analysis of online handwritten mathematical expressions.

Further research should use a huge dataset to evaluate the effectiveness of this approach. Comparative study with other algorithms will be pursued in the near future. More new rules to the expression grammar (e.g. based on Latex representation) should be added to this structural analysis system. In addition, further research should focus on the segmentation.

References

[1] http://www.latex-project.org/.
[2] R. H. Anderson. Syntax-directed recognition of hand printed two-dimensional mathematics. Ph.D. dissertation: Determent Eng. Appl. Phys., Harvard Univ., 1968.
[3] D. Blostein and A. Grbavec. Recognition of Mathematical Notation, chapter 22. World Scientific Publishing Company, 1996.

[4] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In Pro. The 5th Annual ACM Workshop on Computational Learning Theory, pages 144-152, Pittsburgh, PA, July 1992. ACM Press.

[5] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2):121-167, 1998.

[6] Kam-Fai Chan and Dit-Yan Yeung. Towards efficient structural analysis of mathematical expressions. In SSPR/SPR, pages 437-444, 1998.

[7] Kam-Fai Chan and Dit-Yan Yeung. Mathematical expression recognition: a survey. IJDAR, 3(1):3-15, 2000.

[8] S. K. Chang. A method for the structural analysis of two-dimensional mathematical expressions. Information sciences, 2:253-272, 1970.

[9] P.A. Chou. Recognition of equations using a two dimensional stochastic context-free grammar. In In SPIE11 Conference on Visual Communications and Image Processing, volume 1199, pt. 2, pages 852-863, Philadel phia, PA, 1989.

[10] Herwig Friedl and Erwin Stampfer. Cross-Validation. 2001. http://citeseer.ist.psu.edu/article/friedl01crossvalidation.htm.

[11] A. Grbavec and D. Blostein. Mathematics recognition using graph rewriting. In Document Analysis and Recognition, Proceedings of the Third International Conference on, volume 1, pages 417-421, Philadelphia, PA, Aug 1995.

[12] C.-W. Hsu and C.-J. Lin. A comparison of methods for multi-class support vector machines. IEEE Transactions on Neural Networks, 13:415-425, 2002.

[13] B. Q. Huang and M-T. Kechadi. An hmm-snn method for online handwriting symbol recognition. In Interna tional Conference on Image Analysis and Recognition, pages 897-905, Povoa de Varzim, Portugal, Sept. 2006. LNCS 4142.

[14] B. Q. Huang and M-T. Kechadi. A hybrid hmm-svm method for online handwriting symbol recognition. In The 6th International Conference on Intelligent Systems Design and Applications (ISDA'06), pages 887-891, Jian, Shandong, China, Oct. 2006. IEEE Computer Society.

[15] M. Koschinski, H.J. Winkler, and M. Lang. Segmentation and recognition of symbols within handwritten mathematical expressions. In ICASSP, pages 2439-2442, 1995.

[16] A. Kosmala, S. Lavirotte, L. Pottier, and G. Rigoll. On-line handwritten formula recognition using hidden markov models and context dependent graph grammars. In 5th International Conference on Document Analysis and Recognition, Bangalore, India, 1999.

[17] A. Kosmala and G. Rigoll. Recognition of online handwritten formulas. In 6th Int. Workshop on Frontiers in Handwriting Recognition (IWFHR), Taejon, Korea, 1998.

[18] U.H.G. Krebel. Pairwise classi¯cation and support vector machines. In B. SchOlkopf, C.J.C. Burges, and A.J. Smola (Eds.), editors, Advances in kernel methods: Support vector learning, pages 255-268, Cambridge, MA, 1999. MIT Press.

[19] S. Lavirotte and L. Pottier. Optical formula recognition. In In Proc. 4th Int. Conf. on Document Analysis and Recognition, volume 1, pages 357-361, Ulm, Germany, 1995.

[20] His-Jain Lee and Jiumn-Shine Wang. Design of a mathematical expression recognition system. In The 3rd International conference on Document analysis and recognition, pages 1084-1087, 1995.

[21] H.J. Lee and M.C. Lee. Understanding mathematical ex pressions using procedure-oriented transformation. Pattern Recognition, 27(3):447-457, 1994.

[22] N. Okamoto and A. Miuazawa. An experimental implementation of a document recognition system for papers containing mathematical expressions. In Structural Document Image Analysis, pages 36-53, New York, 1992. Springer Verlag.

[23] R. Plamondon and S. Srihari. Online and offline handwriting recognition: A comprehensive survey. IEEE PAMI, 22(1):63-84, 2000.

[24] D. E. Rumelhart and J. L. McClelland. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, volume 1 and 2. MIT Press, Cambridge, MA, 1986.

[25] R.Zanibbi, D. Blostein, and J.R. Cordy. Recognising mathematical expressions using tree transformations. IEEE Tran. on Pattern Analysis and Machine Intelligence, 24(11), November 2002.

[26] J. Serra. An overview of character recognition focused on off-line handwriting. IEEE Trans., 31(1):216-233, May 2001.

[27] C. C. Tappert, C. Y. Suen, and T. Wakahara. The state of the art in online handwriting recognition. IEEE Trans. Pattern Anal. Mach. Intell., 12(8):787-808, 1990.

[28] H. M. Twaakyondo and M.Okamoto. Structure analysis and recognition of mathematical expressions. In Pro. of 3rd Int. Conf. on Document Analysis and Recognition, volume 1, Montreal, Canada, 1995.

[29] H. J. Winkler and M. Lang. Online symbol segmentation and recognition in handwritten mathematical expressions. Volume 4, pages 3377-3380, Munich, Germany, 1997.

[30] H.J. Winkler and M. Lang. Symbol segmentation and recognition for understanding handwritten mathematical expressions. In A. Dwnton and S. Impedovo, editors, Progress in Handwriting Recognition, World Scientific, volume 4, pages 407-412, Singapore, 1997.