Towards a Web-based Progressive Handwriting Recognition Environment for Mathematical Problem Solving

Abstract

The emergence of pen-based mobile devices such as PDAs and tablet PCs provides a new way to input mathematical expressions to computer by using handwriting which is much more natural and efficient for entering mathematics. This paper proposes a web-based handwriting mathematics system, called WebMath, for supporting mathematical problem solving. The proposed WebMath system is based on client-server architecture. It comprises four major components: a standard Web Server, Handwriting Mathematical Expression Editor, Computation Engine and Web Browser with Ajax-based Communicator. The Handwriting Mathematical Expression Editor adopts a progressive recognition approach for dynamic recognition of handwritten mathematical expressions. The Computation Engine supports mathematical functions such as algebraic simplification and factorization, and integration and differentiation. The Web Browser provides a user-friendly interface for accessing the system using advanced Ajax-based communication. In this paper, we describe the different components of the WebMath system and its performance analysis.

Key words: architectures for educational technology system, human-computer interface, improving classroom teaching, interactive learning environments, media in education.

1 Introduction

Most high schools in Singapore provide mobile networks within its premises to support e-learning for many subjects including mathematics and science. However, these are done in the form of data repositories in its web portals for teachers to place their teaching materials, and students to retrieve the materials for learning. This form of learning is quite passive. Most of the learning materials are in the form of multiple choices for easy data input and grading. This is especially so for mathematics as the input of mathematical expressions using traditional keyboards is inconvenient and cumbersome.

The recent development of pen-based mobile devices such as PDAs and tablet PCs provides a new way to input mathematical expressions to computer using handwriting. In comparison with the traditional methods using keyboards, this approach is much more natural and efficient for entering mathematics, as human has traditionally been using pen and paper to write mathematical expressions. On the other hand, the use of computer algebra systems (CAS) such as Maple or Mathematica has become more popular nowadays especially for university students to help solve mathematical problems. But, it is still difficult for high school students to use such systems due to its mathematical syntax. With the advance of both handwriting and mathematics technologies, the current trend on integrating these two technologies to provide a handwriting mathematics environment will certainly improve the learning of mathematics for students.

There are currently a number of academic and commercial handwritten mathematical expression editors such as the Freehand Formula Entry System (FFES,

2005), Natural Log (Matsakis et al., 2003), Infity Editor (Fujimoto et al., 2007), MathJournal (XThink, 2007) and JMathNotes (Rodrguez, 2003). Some of these systems have also provided the mathematical computation capability based on computer algebra systems. However, these systems are mainly standalone applications which require the installation of the system on each individual computing device before it can be used. It is quite inconvenient for users especially when new updates of the systems are required.

To tackle this problem, we aim to develop a web-based handwriting mathematics system that provides a flexible and mobile environment for mathematical problem solving. One of the major challenges to supporting web-based handwriting mathematics is the recognition of handwritten mathematical expressions. Mathematical expressions often contain a wide range of symbols such as alphanumeric characters, Greek characters, mathematical operators, and arrows. Further, mathematical expressions are arranged in a two-dimensional structure with sophisticated grouping rules making structural analysis of mathematical expressions difficult. Another challenge is on the delivery of mathematics over the Web. Moreover, the incorporation of handwriting mathematics into a web-based environment is also not a straight-forward issue.

In this research, we have developed a web-based handwriting mathematics system, called WebMath, for supporting mathematical problem solving. A novel progressive recognition approach is proposed for dynamic recognition of handwritten mathematical expressions. And the web-based delivery of handwriting mathematics is based on client-server architecture with advanced Ajax-based (Garrett, 2005) communication. The proposed system also supports mathematical computations based on a computer algebra system. In this paper, we

present the design and implementation of the WebMath system.

The rest of the paper is organized as follows. Section 2 reviews some of the current handwriting mathematics systems. In Section 3, we discuss the progressive recognition approach for handwritten mathematical expression recognition and its performance analysis. Section 4 discusses the web-based design for progressive recognition. In Section 5, we describe the system architecture of the WebMath system and its individual components. Finally, Section 6 concludes the paper.

2 Handwriting Mathematics Systems

Currently, several handwriting mathematical expression editor systems such as Infty Editor, JMathNotes, Freehand Formula Entry System (FFES) and Natural Log have been developed. These systems are developed based on their own underlying recognition algorithms. Most of these systems are able to achieve quite satisfactory recognition results. Most of them also provide a user-friendly interface.

The Infty Editor (Fujimoto et al., 2003; Kanahori et al., 2004; Suzuki et al., 2003) supports online handwritten expression recognition. In symbol recognition, it combines a generic commercial Optical Character Recognition (OCR) engine with a character recognition engine specifically for mathematical symbols, whereas in structural analysis, it uses an optimization framework. Infty Editor has achieved good performance of 98.51% accuracy in symbol recognition and 89.6% accuracy for the overall recognition process. JMathNotes (Rodrguez, 2003) was developed based on Java. The user interface is quite

friendly and allows user to tune several parameters on Symbol Recognizer, Math Interpreter, etc. The Symbol Recognizer is implemented using the Support Vector Machine (SVM) algorithm which has achieved very low error rate (1.348%). The structural analysis is implemented based on minimum spanning tree construction and symbol dominance (Tapia and Rojas, 2003, 2005). Freehand Formula Entry System (FFES) (Smithies et al., June 1999, 2001; FFES, 2005) is another pen-based equation editor implemented in C++ and Tcl/Tk. The user interface is quite simple. In symbol recognition, FFES uses an online recognition algorithm based on nearest-neighbor classification in a feature space of approximately 50 dimensions. The structural analysis is based on a graph rewriting method (Lavirotte and Pottier, 1997; Blostein and Grbavec, 1996).

Unlike the systems discussed above which are implemented as standalone applications, Natural Log (Matsakis et al., 2003; Matsakis, May, 1999) is implemented as a Java applet which can run over the Web. Natural Log also provides a user-friendly interface with tunable parameters and graphical representation of expressions. In Natural Log, symbol recognition is implemented using a statistical approach based on Gaussian Density Estimation, while structural analysis is implemented using geometric grammar.

Although the standalone implementations of mathematical expression editors such as the Infty Editor and JMathNotes are quite common, they do have some drawbacks. These systems require the installation of the applications on users' machines and they are platform dependent. And it would be quite troublesome to re-install the system when updates are required. On the other hand, the Java applet approach adopted by Natural Log is much more convenient. However, to run the system, it needs the installation of Java Runtime Environment on

the client machine.

In this research, instead of using Java applet, which is essentially an application by itself, we propose another web-based approach based on client-server architecture. To support an efficient web-based design, we use a new technology called Ajax for data communications between the client and server. In addition, we also adopt a progressive recognition approach which is able to provide immediate recognition result for each written symbol. Furthermore, rather than providing just a simple mathematical expression editor, we incorporate a computation engine into the system for mathematical problem solving.

3 Progressive Recognition

In this research, we have proposed a novel progressive recognition approach for dynamic recognition of handwritten mathematical expressions (Vuong et al., 2008). It recognizes a user's handwritten mathematical expression dynamically while he is writing that expression. This approach has an advantage that it helps a user to identify any recognition error after he has written a symbol, and enables him to correct the error immediately. As such, users do not need to wait until they finish writing the whole expression before knowing any recognition errors. Compared with traditional recognition approaches, the progressive recognition approach is much more efficient and user-friendly. To the best of our knowledge, no similar algorithms have been proposed before. Our approach is graphical-based which is fundamentally different from syntactic-based approaches such as the one proposed by Chan and Yeung (1998).

The proposed progressive recognition approach consists of three processes: Progressive Expression Partitioning, Symbol Recognition and Progressive Structural Analysis.

(1) *Progressive Expression Partitioning:* It gathers the written strokes and identifies progressively all the strokes for a symbol.

(2) *Symbol Recognition:* It recognizes the symbol from the group of strokes given by Progressive Expression Partitioning. An extended elastic matching algorithm is implemented for mathematical symbol recognition.

(3) *Progressive Structural Analysis:* It accepts the latest recognized symbol from Symbol Recognition, updates the symbol in the corresponding mathematical expression tree and returns the recognized mathematical expression to the client browser for display. The user is allowed to correct any errors made during recognition.

3.1 Symbol Recognition

In the past few decades, there have been many techniques such as Hidden Markov Model (HMM) (Kosmala and Rigoll, 1998), structural matching (Chan and Yeung, 2000b) and neural networks (Brown, 1992) proposed for symbol recognition. Most of these techniques are able to obtain quite satisfactory results. In this research, we extend the conventional elastic matching algorithm (Chan and Yeung, 1998) for symbol recognition. Elastic matching has many advantages for symbol recognition. It can achieve very high recognition rate as discussed in (Chan and Yeung, 1998), and able to cope with irregularities in writings. Further, elastic matching is fast when compared with other symbol recognition approaches.

In conventional elastic matching, it calculates the distance between input symbols and the stored sample symbols of known classes. During matching, every point in the input symbol pattern is matched against that in the sample symbol pattern in order to decide the class of the input symbol. However, the conventional elastic matching only considers Euclidean distance between points without analyzing the slopes and curvatures at those points. As such, it has difficulties in recognizing curly symbols such as "8" and "∞", as information on slopes and curvatures play a more important role than Euclidean distance in these symbols. Therefore, apart from Euclidean distance between points, the extended elastic matching algorithm also considers slope and curative information during its matching process.

The Symbol Recognition process comprises two processes: *Sample Symbols Generation* and *Recognition*.

3.1.1 Sample Symbols Generation

In *Sample Symbols Generation*, it aims to generate a database of sample symbols to be used for elastic matching purposes. The symbols are collected as a representative set of stroke data (in sequences of points with x and y coordinates) rather than images. To standardize the collected data, they are preprocessed with size normalization, smoothing, re-sampling, stroke re-orientation and stroke re-ordering. Then, symbol features such as number of strokes, the initial and end points, and the initial and end angles are extracted. Based on the extracted features, sample symbols can be categorized into different groups and stored in the symbol database. In addition, users can also specify symbols for personalization. In this case, a user can specify a written symbol

with its actual identity and store the specified symbol into his personalized symbol database which can be used together with the symbol database for elastic matching.

3.1.2 Recognition

In *Recognition*, user input symbol is also preprocessed and features are extracted as in *Sample Symbols Generation*. After that, the input symbol is matched against each sample symbol using the extended elastic matching algorithm. Note that only those symbols in the same classified group with the input symbol are used for matching. This helps to reduce the size of the sample set of symbols to be matched against the input symbol. In addition, we also pass the input symbol to the Microsoft Tablet SDK (Microsoft, 2007) for symbol recognition. The Microsoft Tablet SDK is a character recognition library provided by Microsoft for Windows Tablet operating system. It is able to recognize standard handwriting symbols including alphabetical characters, digits and common arithmetic operators. However, it is unable to recognize mathematical symbols and Greek characters. As such, the extended elastic matching algorithm is used for recognizing mathematical related symbols, while the Microsoft Tablet SDK is used for recognizing standard characters. Finally, based on the confidence level of the two matching results, we decide the identity of the input symbol.

3.2 Progressive Expression Partitioning

A mathematical symbol may comprise more than one stroke. It is necessary to identify the strokes that belong to one symbol before Symbol Recognition.

Expression partitioning aims to partition strokes correctly into symbols. The main challenge of expression partitioning is that it does not have any obvious rules to group the strokes that belong to a symbol. Furthermore, mathematical expressions are in two-dimensional structure which further complicates the expression partitioning process. For example, the bounding box technique is difficult to deploy for symbols such as the $\sqrt{\ }$ sign which usually contains other symbols within its structure. Spatial distance feature cannot be applied for expressions containing superscripts or subscripts due to its small font sizes.

In our approach, when there is a new stroke written, this new stroke will be grouped with some previously written strokes to form different stroke groups. We assume that users always complete all strokes of one symbol before writing the next symbol, and they do not make any corrections to previously written symbols. Therefore, we only group consecutively written strokes. We further assume that all users are well-informed about this assumption. Our experiments show that users can easily adapt their writing style to what the system requires in just a few minutes of getting familiar to the system. After a list of stroke groups is created, each group is passed to *Symbol Recognition*. The stroke group with the highest confidence will be chosen and the corresponding symbol is returned.

3.3 Progressive Structural Analysis

Structural analysis is the process of analyzing the 2-dimensional structural of mathematical expressions to create a structural representation of the expressions. As mathematical symbols are usually arranged in a complex two-dimensional structure, possibly of different sizes and in recursive manner, this

makes structural analysis a challenging problem even when all symbols are recognized correctly. In structural analysis, many issues such as symbol relationships, symbol grouping rules and context-dependent symbol groups need to be tackled.

Many techniques have been investigated for structural analysis for handwritten mathematical expression recognition. These include grammar-based approaches (Chou, 1989; Fateman et al., 1996; Chan and Yeung, 2000a; Toyota et al., 2006), tree transformation (Zanibbi et al., 2002), Hidden Markov Models (HMMs) (Kosmala and Rigoll, 1998) and Minimum Spanning Tree (Tapia and Rojas, 2003, 2005). Some of the techniques such as the grammar-based approaches are slow, while others are sensitive to users' writing errors. In the Progressive Structural Analysis (PSA) approach, it recognizes progressively a user's handwritten mathematical expression while he is writing that expression.

In mathematical expressions, input symbols are related. The relationships could be superscript like a^2, subscript like b_3, and above and under like $\sum_{a}^{b}$, etc. And one symbol can form different relationships with other different symbols. There could also be different ways of interpreting a relationship between two specific symbols. Therefore, we need to identify the most plausible relationship between input symbols. In addition, mathematical expressions are organized in a recursive structure. One expression may contain symbols and in turn each symbol may contain other expressions as its superscript, subscript, prescript, etc.

In PSA, we focus on determining the relationships between two symbols and the grouping of related symbols. To do this, we define Mathematical Expres-

sion Tree (MET) to represent mathematical expressions. In the PSA approach, whenever a new symbol from the input mathematical expression is processed, the relationship and grouping properties of that symbol will be identified. These properties are then updated into the MET of the corresponding expression.

Figure 1 shows the overall process of the PSA approach which consists of three processes: *Related Symbol Identification, MET Update* and *Representative Format Conversion*. Examples of MET update process for the expression "a^2+b^2" are also illustrated. The shaded nodes are *expression nodes*. The root node of a MET is always an expression node. All other expression nodes represent sub-expressions in that expression. Every expression node except the root has a symbol node as its parent. The unshaded nodes are *symbol nodes*. Each symbol node represents one symbol. Every symbol node has an expression node as its parent. All symbols represented by sibling symbol nodes have row relationship. These symbols are on the dominant baseline of their parent expression.

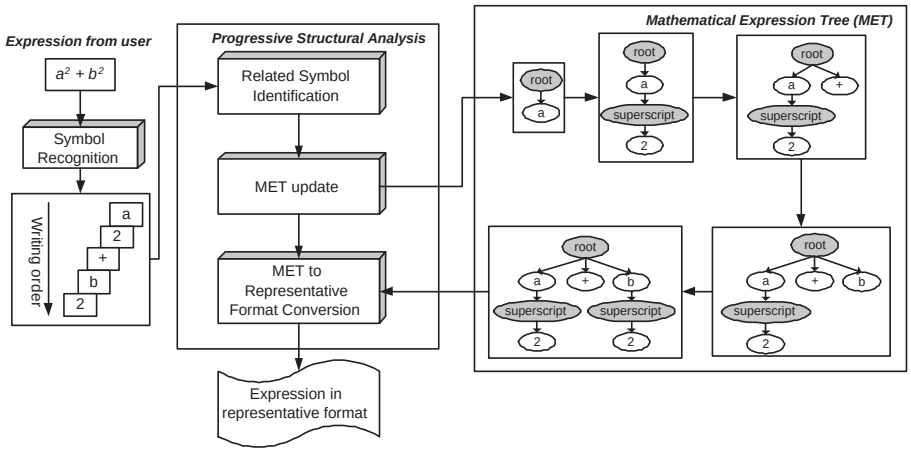

Fig. 1. Overall process of Progressive Structural Analysis.

- *Related Symbol Identification.* After the latest symbol S is recognized from

Progressive Expression Partitioning, the related symbol R of S will then be identified. The related symbol R is a symbol that has already been inserted into the MET before S is written. It is useful for inferring the position in MET that S will be inserted into. For example, in Figure 1, 'a' is the related symbol of '+', and '+' is the related symbol of 'b'. Assuming that '+' is the latest input symbol and its position in the MET has not yet been identified. If 'a' is identified as the related symbol of '+', then from their relationship (i.e., row relationship), it can be inferred that they share the same parent expression. Thus, '+' is assigned as the next sibling of 'a' in its position in the MET.

To determine the related symbol, we first find the previous symbol P which is written just before S chronologically. If P is not found, there is no related symbol R. It means that S is the first input symbol. If P can be found, then we create a list of symbols residing at the path from the root of the MET to P. Each symbol in the list will be tested on its likelihood of being S's adjacent neighbor or S's direct ancestor based on its relationship with S (e.g., row, superscript, subscript, inside, above or under). Finally, the symbol with the highest likelihood is chosen as the related symbol of S.

- *MET Update*. In MET Update, the position of the latest input symbol S in the MET will be determined. If S is the first symbol of the expression, then S will be assigned as the root of MET. Otherwise, S will be assigned as adjacent neighbor or direct child of R in MET according to its relationship with the related symbol. In addition, this process also groups meaningful consecutive symbols in MET into mathematical units such as trigonometric functions of sin, cos and tan.
- *MET To Representative Format Conversion*. The last step in the PSA approach is to convert MET into a representative format such as MathML

or Latex. These formats help to render the expression graphically for viewing. In WebMath, we convert MET into the MathML format, as we display mathematical expressions on Web browsers.

3.4 Performance Analysis

To evaluate the effectiveness of the proposed progressive recognition approach, we have conducted an experiment using a number of different expressions extracted from the "CRC standard Math tables and formulae" (Zwillinger, 2003). These expressions are grouped into six domains, namely Elementary Algebra, Number Theory, Trigonometric Functions, Geometry, Differential Calculus and Integration. Each domain contains ten expressions of different types except Number Theory which contains only three expressions. This is because that the expressions in Number Theory provided by the "CRC standard Math tables and formulae" are quite simple and similar. Thus, only one example from each group of similar expressions was selected which resulted in only three expressions in this domain. There are totally fifty-three expressions in the six domains. Some examples are shown in Table 1.

Ten users were involved in this experiment. Each user was first given a short introduction on the WebMath system. Then, they spent ten minutes to get familiar with the functions provided by WebMath and experiment it with a few simple expressions of different types. These simple expressions are different from those used for testing in the experiment. After that, each user was asked to write all the expressions from the test data set of expressions, with each expression written only once. To evaluate the performance, we measure the accuracy of the combined *Symbol Recognition* and *Progressive Expression*

Table 1

Example mathematical expressions.

Domain	*Example*
Elementary Algebra	$\frac{b^2c^2-4b^3d-4ac^3+18abcd-27a^2d^2}{a^4}$
	$(a \pm b)^4 = a^4 \pm 4a^3b + 6a^2b^2 \pm 4ab^3 + b^4$
	$a^4 + b^4 = \left(a^2 + \sqrt{2}ab + b^2\right)\left(a^2 - \sqrt{2}ab + b^2\right)$
Number Theory	$\left(\frac{a}{n}\right) = \prod_{i=1}^{k} \left(\frac{a}{p_i}\right)^{b_i}$
	$x_k + y_k\sqrt{d} = \left(x + y\sqrt{d}\right)^k$
	$\frac{x}{1-x^2} + \frac{1}{1-x^4} + \frac{x^2}{1-x^4} = \frac{1}{1-x}$
Trigonometric Functions	$\sin(\alpha \pm \beta) = \sin\alpha\cos\beta \pm \cos\alpha\sin\beta$
	$\sin 2\alpha = 2\sin\alpha\cos\alpha = \frac{2\tan\alpha}{1+\tan^2\alpha}$
	$\cos\frac{\alpha}{2} = \pm\sqrt{\frac{1+\cos\alpha}{2}}$
Geometry	$\frac{y-y_1}{x-x_1} = \frac{y_0-y_1}{x_0-x_1}$
	$\left(\frac{kx_1+(100-k)x_0}{100}, \frac{ky_1+(100-k)x_0}{100}\right)$
	$\frac{1}{4}\sqrt{4p^2q^2 - \left(b^2+d^2-a^2-c^2\right)^2}$
Differential Calculus	$\frac{d}{dx}\left(\frac{1}{u}\right) = -\frac{1}{u^2}\frac{du}{dx}$
	$\lim_{x\to a}\frac{f(x)}{g(x)} = \lim_{x\to a}\frac{f'(x)}{g'(x)}$
	$\frac{d^2}{dx^2}\left(f\left(u\right)\right) = \frac{df}{du}\left(u\right)\cdot\frac{d^2u}{dx^2} + \frac{d^2f}{du^2}\left(u\right)\cdot\left(\frac{du}{dx}\right)^2$
Integration	$\int \frac{1}{\sqrt{x^2\pm a^2}}dx = \log\left(x + \sqrt{x^2 \pm a^2}\right)$
	$\int_a^b f\left(x\right)dx + \int_b^c f\left(x\right)dx = \int_a^c f\left(x\right)dx$
	$\int_0^\infty \frac{x^{p-1}}{1+x}dx = \frac{\pi}{\sin p\pi}$

Partitioning, and *Progressive Structural Analysis*.

In *Symbol Recognition* and *Progressive Expression Partitioning*, the accuracy is calculated as the proportion of mis-recognized symbols from the total number of symbols written by users. In *Progressive Structural Analysis*, the accuracy is calculated as the proportion of mis-recognized expressions from the total number of expressions written by users. To make this process independent of *Symbol Recognition*, we assume that symbols are recognized correctly. To ensure this, after writing each stroke, the users are required to correct manually the mis-recognized symbols with the correct ones. These incorrectly recognized symbols are not counted to the overall performance.

Table 2

Experimental results.

	Accuracy(%)	
	Symbol Recognition and Progressive Expression Partitioning	Progressive Structural Analysis
Elementary Algebra	98.7	94
Number Theory	98.5	90
Trigonometric Functions	96.2	97
Geometry	97.3	93
Differential Calculus	96	97
Integration	94.6	96
All Domains	96.67	94.53

The experimental results are given in Table 2. In *Symbol Recognition* and *Progressive Expression Partitioning*, the average recognition rate for all domains is 96.67%. This is the recognition rate achieved without using personalization. If personalization is used, the recognition rate can further be improved to a higher value even up to 100% for some users. As shown from Table 2, Elementary Algebra and Number Theory have achieved the highest recognition rates of 98.7% and 98.5% respectively. It is reasonable as most of the symbols used in these domains are mainly common alphanumeric symbols. Differential Calculus and Integration have achieved low recognition rates of 96% and 94.6% respectively. It is because these domains contain more complex mathematical symbols such as ∞, $\int$, π, etc.

In *Progressive Structural Analysis*, the average accuracy for all domains is 94.53%. Among the six domains used in the experiment, Trigonometric Functions and Differential Calculus have achieved the highest recognition rate of 97%, while Number Theory has the lowest recognition rate of 90%. On average, the recognition rates for all the domains are above 90%. We have also observed that expressions in domains with lower recognition rates are more complex than other domains in terms of the number of symbols and their mathematical structures.

4 Web-based Design for Progressive Recognition

As mentioned in Section 2, there are a number of standalone handwritten expression recognition systems. However, to our knowledge, there is no client-server web-based system available for online handwriting mathematical computation. The web-based design offers a number of advantages. Firstly, it offers

a high level of mobility. Users can run the web-based handwriting system anytime and anywhere as long as the computer supports Web browsers and the Internet. Secondly, it does not require a specific system or operating system. Therefore, it can run on any browser-supported machines. Thirdly, all program codes are stored at the server. As such, any updated version of the system can be run when the user invokes the system. Finally, it is safe to the clients' machines as no installation is required at the clients.

In the classic "click and wait" web applications, user actions trigger an HTTP request back to a web server. The server performs some processing such as retrieving data, formatting the data, communicating with legacy systems, and then returns the updated HTML page to the client. As a result, the entire web page needs to be reloaded in order to display the returned results. This mechanism has two main drawbacks: (1) Even when the amount of submitted data and returned data is small, the entire web page which may include images, sound or video clips needs to be reloaded. This wastes a lot of bandwidth. (2) Users are unable to do any other useful tasks while the page is being reloaded that may take a few seconds up to a few minutes.

Due to the above drawbacks, traditional Web is not suitable for applications which require instantaneous response and continual data flow between the client and the server. The Java Applet approach adopted by Natural Log helps to provide a partial solution to this problem. In this approach, the Java Applet code at the server is downloaded and executed at the client machine. Although this approach offers mobility and instant update, it does not provide users with the look and feel Web page. Furthermore, this approach requires Java Runtime Environment to be installed on client machines which is not always possible. In this paper, we have implemented the communication mechanism

between the client and server in the WebMath system using Ajax (White, 2006). The term Ajax stands for "Asynchronous JavaScript and XML". It allows all interactions between the client and server to be carried out in the background, thereby overcoming the drawbacks of the traditional Web. It basically uses the XMLHttpRequest object to create a tunnel from the client's browser to the server and transmit information back and forth without having to refresh the page to support asynchronous data retrieval. The data can be transmitted in XML format since it may be in complex structure rather than plain text. Ajax has recently become one of the popular techniques for developing Web applications. Many popular web sites such as the Google Maps (Google, 2007b), Google Mail (Google, 2007a), Google Suggest (Google, 2007c), Zuggest (Shanahan, 2006) and Meebo (Meebo, 2007) have used Ajax for their implementations. Our approach makes the system behave exactly like a Web page with the advantages of Web application such as mobility, instant update while maintaining the look and feel style. In addition, it only requires a Web browser and network connections, and thus ensures client machines' safety. Note that our system is not targeted to run on the entire Web. Instead, it is intended to be used at organizational levels such as universities, schools, companies etc where reliable networks are available.

5 System Architecture

Figure 2 shows the overall architecture of the WebMath system which consists of the following components: Handwriting Mathematical Expression Editor, Computation Engine, Web Browser with Ajax-based Communicator. These components are organized in client-server architecture according to its func-

tionalities.

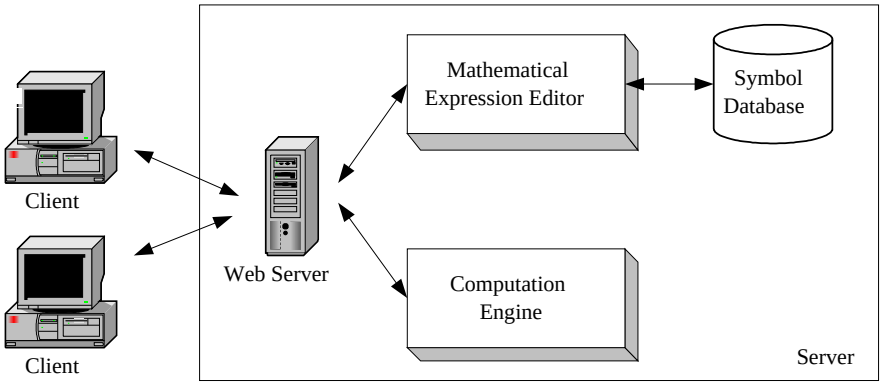

Fig. 2. System architecture of WebMath.

The client computers provide a user-friendly interface for users to enter mathematical inputs and view the computation outputs based on Web browsers. Figure 3 shows the user interface of the WebMath system which contains three windows: Expression Window, Recognition Window and Computation Window.

5.1 Handwriting Mathematical Expression Editor

The Handwriting Mathematical Expression Editor is the major component of the WebMath system for supporting progressive recognition. It receives data inputs from the client's Web browser for processing and recognition. As discussed in Section 3, we use the extended elastic matching algorithm for symbol recognition and progressive structural analysis algorithm for structural analysis. A symbol database which stores the sample symbols is also maintained at the server for symbol recognition purpose. The user input expressions are displayed graphically in the Expression Window of the user interface, while the recognition results are displayed in the Recognition Window.

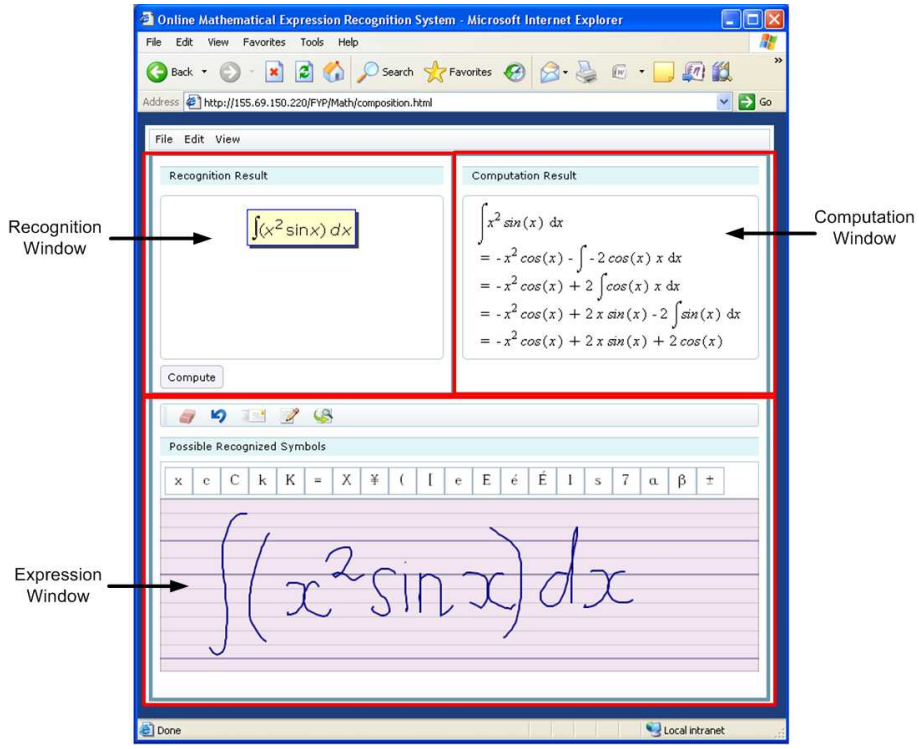

Fig. 3. User interface.

5.2 Computation Engine

The Computation Engine performs the calculation of the input mathematical expressions. The computation functions are provided by Maple (Maplesoft, 2007). Many mathematical functions such as algebraic simplification and factorization, and differentiation and integration are supported. The results are displayed in the Computation Window of the browser interface through the Mathematical Markup Language (MathML) representation (W3C, 2007a).

5.3 Web Browser with Ajax-based Communicator

To support efficient communication between the client and server, Ajax is adopted for the implementation. Ajax helps maintain continual data flow be-

tween the client and server without the need of refreshing the web page on the browser. It makes the web-based system behave exactly like a desktop application. Figure 4 shows the Ajax-based interactions between the client and server in WebMath for expression recognition and mathematical computation.

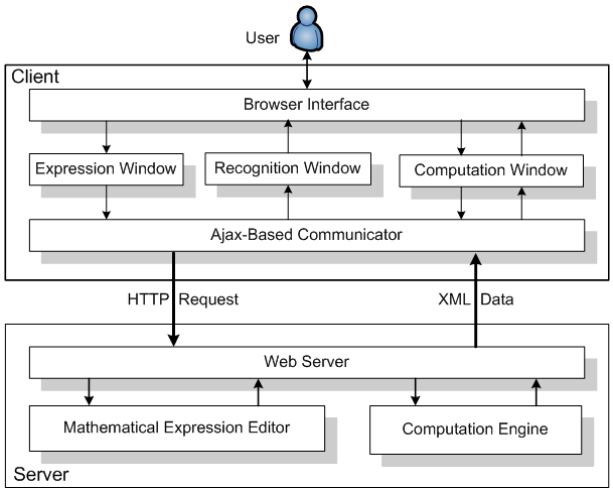

Fig. 4. Ajax-based Communicator.

5.3.1 Expression Recognition

The Expression Window accepts user inputs on handwriting mathematical expressions from either a mouse or an electronic pen. It is responsible for capturing input data including strokes (represented by a sequence of points) and user interactions such as pen-down, pen-up and pen-move events. In addition, it is also responsible for displaying output data in graphical form on the Web browser. To capture strokes and user interactions, an event-driven JavaScript object is created. It is able to capture all interactions between users and the client, convert the pen-move data into sequences of points and initiate the communication with the server. It should be noted that the WebMath system supports progressive recognition. Thus, whenever a user creates a stroke data

(a pen-up event occurs), the JavaScript object collects data for that stroke and sends it to the server for recognition. To support graphical display, SVG (Scalable Vector Graphics) (W3C, 2007b) is embedded inline with HTML Web pages. SVG is an XML standard which enables drawing on the Web. Netscape and Firefox browsers have built-in features to render SVG while Internet Explorer requires an additional plug-in.

During expression recognition, the stroke data is sent from the Expression Window to the Ajax-based Communicator after a pen-up event. Then, the communicator creates an HTTP request which encapsulates the stroke data and sends it to the server. At the server side, the stroke data in the new request is combined with all the previously written strokes from that particular user. They are then processed by Symbol Recognition, Progressive Expression Partitioning and Progressive Structural Analysis to generate the progressive recognition result. After that, the server stores the new stroke data into a temporary cache for subsequent processing and returns the recognition result in XML format back to the Ajax-based Communicator. The recognition result is then displayed in the Recognition Window.

The Recognition Window renders mathematical expression graphically. Since basic HTML does not support the rendering of mathematical expressions in graphical forms, we use the Mathematical Markup Language (MathML) (W3C, 2007a) representation to display mathematical expressions. Similar to SVG, MathML is also treated as an external object. It is supported by Firefox, but using it with Internet Explorer requires an additional plug-in.

5.3.2 Mathematical Computation

In mathematical computation, it operates in a very similar way as in expression recognition. The input expression and computation command are sent from the Computation Window to the Ajax-based Communicator. These data are then transmitted to the server by a HTTP request. The Computation Engine at the server performs the calculation. The results are then sent back to the communicator. It is then displayed graphically on the Computation Window.

The Computation Window supports a number of computation commands such as algebraic factorization and simplification, and differentiation and integration. The Computation Engine provides the computational capability of the WebMath system. In our design, we use Maple (Maplesoft, 2007) for the implementation. To do this, the input mathematical expression and the corresponding command are sent to Maple, and the resultant expression is calculated and returned to the client. Since Maple supports step-by-step problem solving, the WebMath system is also able to offer this functionality to provide a step-by-step solution to the given mathematical problem.

6 Conclusions

In this paper, we have presented the design and implementation of WebMath, a web-based handwriting mathematics system. The web-based design enhances mobility, platform independence and system security compared with standalone applications. The proposed WebMath system adopts a progressive recognition approach to provide dynamic recognition of handwritten mathematical expressions. This technique enables users to observe the recognition

process and make immediate corrections if any errors are occurred during the recognition process. The WebMath system provides a user-friendly interface that supports the input of mathematical expressions, and mathematical expression recognition and computation over the Web. Mathematical computation is provided by a computer algebra system called Maple. This feature can be used as a problem solving tool for high school mathematics. The WebMath system has been evaluated which has achieved very satisfactory performance.

It is possible to embed the WebMath system into a tutoring and learning system for mathematics. The system first generates a problem based on the curriculum it stores. Students could use WebMath to work out the solutions. The tutoring system could then compare its solution with that of the student and performs a diagnosis based on the differences. After giving feedback, the system reassesses and updates the student skills model and the entire cycle is repeated. As the system is assessing what the student knows, it is also considering what the student needs to know, which part of the curriculum is to be reinforced, and how to present the material. By engaging student-centered learning and providing a handwriting mathematics environment, WebMath can help students to practice mathematical problems in their own pace. As such, WebMath is useful in supporting high school students in mathematical problem solving.

References

Blostein, D., Grbavec, A., 1996. Recognition of mathematical notation. Handbook on Optical Character Recognition and Document Analysis, World Scientific Publishing Company 22.

Brown, E., 1992. Applying neural networks to character recognition. Http://www.ccs.neu.edu/home/feneric/Papers/charrecnn.pdf.

Chan, K., Yeung, D., 1998. Elastic structural matching for on-line handwritten alphanumeric character recognition. In: International Conference on Pattern Recognition (ICPR). Brisbane, pp. 1508–1511.

Chan, K., Yeung, D., 2000a. An efficient syntactic approach to structural analysis of on-line handwritten mathematical expressions. Pattern Recognition 33 (3), 375–384.

Chan, K., Yeung, D., 2000b. Mathematical expression recognition: a survey. International Journal of Document Analysis and Recognition 3 (1), 3–15.

Chou, P., 1989. Recognition of equations using a two-dimensional stochastic context-free grammar. In: Pearlman, W. (Ed.), Visual Communications and Image Processing IV. Vol. 1199 of SPIE Proceedings Series. pp. 852–863.

Fateman, R., Tokuyasu, T., Berman, B., Mitchell, N., 1996. Optical character recognition and parsing of typeset mathematics. Journal of Visual Communication and Image Representation 7 (1), 2–15.

FFES, 2005. Freehand Formula Entry System. Http://www.cs.queensu.ca/drl/ffes/.

Fujimoto, M., Fujiyoshi, M., Kanahori, T., Kawane, F., Komada, T., Malon, C., Ohtake, N., Suzuki, M., Uchida, S., Yamaguchi, K., Yuan, C., 2007. Infty Project. Http://www.inftyproject.org/en/.

Fujimoto, M., Kanahori, T., Suzuki, M., 2003. Infty editor - a mathematics typesetting tool with a handwriting interface and a graphical front-end to openxm servers. Computer Algebra Algorithms, Implementations andApplications RIMS Kokyuroku vol.1335, 217226.

Garrett, J., 2005. Ajax - a new approach to web applications. Daptivepath.com, adaptivepath.com/publications/essays/archives/000385.php.

Google, 2007a. Google Mail. Http://gmail.google.com.

Google, 2007b. Google Maps. Http://maps.google.com.

Google, 2007c. Google Suggest. Http://www.google.com/webhp?complete=1&hl=en.

Kanahori, T., Fujimoto, M., Suzuki, M., 2004. Authoring tool for mathematical documents infty. In: Proceedings of the Workshop on Mathematical User Interfaces.

Kosmala, A., Rigoll, G., 1998. On-Line Handwritten Formula Recognition Using Statistical Methods. In: International Conference on Pattern Recognition (ICPR). Brisbane, pp. 1306–1308.

Lavirotte, S., Pottier, L., 1997. Optical formula recognition. In: 4th International Conference on Document Analysis and Recognition (ICDAR). Vol. 1. pp. 357–361.

Maplesoft, 2007. Maple. Http://www.maplesoft.com/products/maple/index.aspx.

Matsakis, N., May, 1999. Recognition of handwritten mathematical expressions, master's thesis. Massachusetts Institute of Technology, Cambridge, MA, http://www.ai.mit.edu/projects/natural-log/papers/matsakis-MEng-99.pdf.

Matsakis, N., Miller, E., Viola, P., 2003. Natural Log. Http://www.ai.mit.edu/projects/natural-log/index.shtml.

Meebo, 2007. Meebo. Http://www.meebo.com.

Microsoft, 2007. Microsoft Windows XP Tablet PC Edition. Http://msdn2.microsoft.com/en-us/library/ms840465.aspx.

Rodrguez, E. T., 2003. JMathNotes. Http://page.mi.fu-berlin.de/ tapia/JMathNotes/.

Shanahan, F., 2006. Zuggest. Http://www.francisshanahan.com/zuggest.aspx.

Smithies, S., Novins, K., Arvo, J., 2001. Equation entry and editing via handwriting and gesture recognition. In: Behaviour and Information Technology,.

pp. 53–67.

Smithies, S., Novins, K., Arvo, J., June 1999. A handwriting-based equation editor. In: Fourth International Conference on Document Analysis and Recognition. pp. 357–361.

Suzuki, M., Tamari, F., Fukuda, R., Uchida, S., Kanahori, T., 2003. Infty - an integrated ocr system for mathematical documents. pp. 95–104.

Tapia, E., Rojas, R., 2003. Recognition of on-line handwritten mathematical expressions using a minimum spanning tree construction and symbol dominance. In: 5th International Workshop on Graphics Recognition, Recent Advances and Perspectives (GREC). LNCS 3088. pp. 329–340.

Tapia, E., Rojas, R., 2005. Recognition of on-line handwritten mathematical expressions in the e-chalk system - an extension. In: 8th International Conference on Document Analysis and Recognition. pp. 1206–1210.

Toyota, S., Uchida, S., Suzuki, M., 2006. Structural analysis of mathematical formulae with verification based on formula description grammar. In: 7th International Workshop on Document Analysis Systems. LNCS 3872. Nelson, New Zealand, pp. 153–163.

Vuong, B.-Q., Hui, S.-C., He, Y., 2008. Progressive structural analysis for dynamic recognition of on-line handwritten mathematical expressions. Pattern Recognition Letters 29 (5), 647–655.

W3C, 2007a. MathML. Http://www.w3.org/Math/.

W3C, 2007b. Scalable Vector Graphics (SVG). Http://www.w3.org/Graphics/SVG/.

White, A., 2006. Measuring the benefits of ajax. Developer.com, www.developer.com/xml/article.php/3554271.

XThink, 2007. Math Journal. Http://www.xthink.com/MathJournal.html.

Zanibbi, R., Blostein, D., Cordy, J., 2002. Recognizing mathematical expres-

sions using tree transformation. IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (11), 1455–1467.

Zwillinger, D., 2003. CRC Standard Math Tables and Formulae, 31st Edition. CRC Press, Boca Raton.