MathBrush: An Experimental Pen-Based Math System

George Labahn, Scott MacLean, Mirette Marzouk, Ian Rutherford, David Tausky

David R. Cheriton School of Computer Science
University of Waterloo, Waterloo, Ontario, Canada
e–mail: {glabahn,smaclean,msmarzouk,ijrutherford,datausky}@scg.uwaterloo.ca

Abstract

It is widely believed that mathematics will be one of the major applications for Tablet PCs and other pen-based devices. In this paper we discuss many of the issues that make doing mathematics on such pen-based devices a hard task. We give a preliminary description of an experimental system, currently named MathBrush, for working with mathematics using pen-based devices. The system allows a user to enter mathematical expressions with a pen and to then do mathematical computation using a computer algebra system. The system provides a simple and easy way for users to verify the correctness of their handwritten expressions and, if needed, to correct any errors in recognition. Choosing mathematical operations is done making use of context menus, both with input and output expressions.

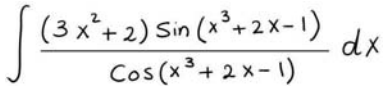
Key words: PC-tablets, Pen-based math, Computer Algebra systems

1 Introduction

Pen-based devices such as Tablet PCs have gained a significant following of users in the past few years. For the most part these devices are used for such applications as simple notetaking and annotation of documents. However there still does not seem to be any outstanding applications for such devices. Rather users seem to view the pen as a sophisticated mouse and hence does not use the full power available from these devices.

We view mathematics as being the one application that can change the current state. Of course by mathematics we mean both writing and working with mathematical expressions. In particular mathematics used with Tablet PCs need to take advantage of not only pen input but also of todays powerful computer algebra systems.

One obvious motivation for the above point of view comes from the fact that entering mathematics on a computer is problematic. For example the expression



is more natural than inputting the latex form

```
\int {\frac { \left( 3\,{x}^{2}+2 \right) \sin \left( {x}^{3}+2\,x-1 \right) }
            { \cos \left( {x}^{3}+2\,x-1 \right)}} ~ dx
```

or the maple form

```
Int((3*x^2+2)*sin(x^3+2*x-1)/cos(x^3+2*x-1),x);
```

or the mathematica form

```
Integrate[(3*x^2+2)*Sin[x^3+2*x-1]/Cos[x^3+2*x-1],x]
```

of the corresponding expression. Input for other Computer Algebra Systems (CAS) or for text processing systems provide other examples which use alternate formats and syntax to formulate a mathematical expression.

While inputting mathematics is best done using classical handwriting, it is also the case that handwriting is not well suited to actually doing the mathematics. The command representation in CAS for the above integral also allows for a simple mechanism for actually computing the integral itself. For example Maple the *value* command gives:

$$\text{value(\%)} = -\ln(\cos(x^3 + 2\,x - 1)).$$

Here % is the Maple syntax for the previous expression. Output is also a concern since handwritten experimentation with pen and paper rarely encounters problems such as massive output expressions, a common occurrence when using CAS.

Our project focuses on two components of pen-based mathematics. The first is to investigate the use of pen-based devices for mathematical computation and exploration while the second is to study the key issues when combining pen-based interfaces with CAS. We chose to build MathBrush, an experimental testbed to allow us to address issues such as input and output of mathematical expressions, the editing and manipulation of such expressions and how to interact with CAS taking the advantage their power. The intent of MathBrush is not to have a commercial product or to reinvent the wheel by reimplementing features available in CAS. Rather the intend is to provide an environment for experimenting with the various components needed for doing mathematics with pen-based devices.

The state of the art for pen-math based systems currently focuses on mathematical handwriting recognition. Examples include work by Chan and Yeung [2], the systems Infty [12], Freehand Formula Entry System (FFES) [11, 15], and MathJournal from xThink Inc (www.xthink.com). MathJournal appears to be the first commercial system for inputting mathematics on a Tablet PC but is somewhat limited, particularly when it comes to mathematical capabilities. Indeed for the most part the above systems have very basic mathematical functionality. A different approach is that of MathPad^2 [3] from Brown University which attempts to convert diagrams into mathematical formulas.

We remark that for any pen-based system for doing mathematics there are a number of challenges. On the level of input recognition, the recognition of mathematical symbols is considerably more difficult than recognizing text input. Existing text recognizers are not suitable since they only work with a limited character set (alpha-numeric characters), and they also gain much of their strength by depending on language specific dictionaries in order to validate combinations of input characters. A significant challenge comes at the level of construction of a valid mathematical expression. Text is by nature a one dimensional input, whereas mathematical expressions are by nature two-dimensional input which makes it hard to determine an appropriate baseline. In addition, even when a valid mathematical expression is successfully recognized, there is still the problem of ambiguous text (for example $|2|$ looks very similar to 121 in handwritten text). Finally there are significant challenges at the level of display/rendering and editing. In this case problems include the need for proper line-breaking, along with additional needs for interactivity for both editing and manipulation of output expressions with a pen.

The remainder of this paper is organized as follows. In Section 2 we give the main system characteristics used for the MathBrush system, while Sections 3 to 6 provides descriptions of the five main system components for MathBrush. Section 7 gives some discussion of future work with the final section giving our conclusions.

2 Main System Characteristics

In this section we briefly discuss the main characteristics that are central for the design of the MathBrush system. For more details we refer the reader to [16]. These features include modular components, a standard interaction mechanism, ease of use, context menus and finally logging mechanism.

Central to the experimental aspect of our system are pluggable components. MathBrush has been designed so that it allows for replacing of current components with those more advanced (when available), comparing different versions of a given component, and isolating component problems. Our design carefully separates features and functionalities of different components taking in mind the standard design and functionalities of similar modules (for example we closely follow the guidelines in [5] for building a recognizer).

The use of separate, independent modules mandates the use of a standard communication mechanism. We use MathML [1] as our standard for representation of mathematical expressions and for communication with the CAS. MathML is used because it is now becoming a mathematics standard and is supported by different CAS. In addition it has support in different web browsers.

One central concern for pen-based math systems in general is ensuring ease of use for both entering and manipulating input and output of mathematical expressions. In our case, this means that the handwritten mathematics is easy to enter and requires the minimum interaction from the user to ensure that the recognition is correct. Gestures are used in a limited way and chosen to be similar to those that the user naturally uses when working with pen and paper. These gestures also need to be consistent with those from other well-designed pen-based applications. These gestures include, for example, the scratch out to delete input characters and a right click to display context menus.

Working in a pen-based environment leaves only a limited opportunity to operate with commands. This in turn allows for very little chance to do any mathematics once a handwritten mathematical expression has actually been input into the system. We have chosen to make use of dynamic context menus as done in Maple (c.f. [6]) to operate with both input and output expressions. Context menus are easy to use with a pen (considered as a mouse) and without any need for a keyboard.

An important feature we are including in our system and which is not available using pen and paper is a mechanism for logging our mathematical manipulations. By this we mean keeping track of all the user's actions while they work on a math sheet (the session that was started and is currently being worked on). When a user works on a math sheet (even on a piece of paper) they often change expressions in certain places but do not necessarily reflect these changes on all dependent expressions. As such one can end up with an inconsistent document or a document that cannot be tracked in any way. At some point of time a user often asks the question : how did one end up having a particular math sheet. The logging mechanism is available to answer such a question.

3 System Components

The five main system modules that make up MathBrush consist of a user interface, a character recognizer, a structural analyzer, a CAS interface tool and finally a mathematics rendering tool. These five system modules and their interdependencies are depicted in Figure (1). A general architecture for pen-based math systems can be found in [10].

The MathBrush user interface module receives the digital ink from the user, collects the

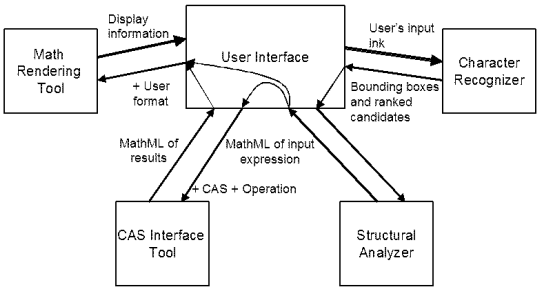


Figure 1: System Components

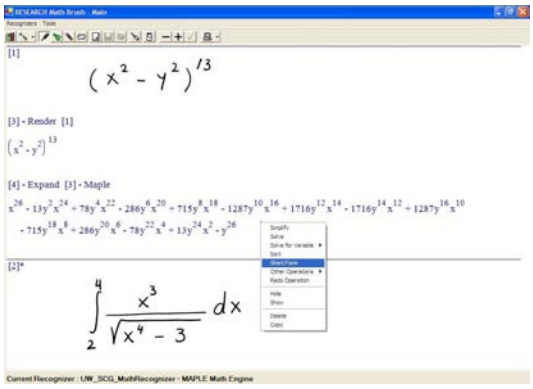


Figure 2: User Interface

user's interactions and commands (via context menus), and ultimately renders the results back to the user. The interface module sends the collected ink to the character recognizer. The character recognizer detects different characters and generates a set of bounding boxes. For each box it generates a set of candidate characters and the recognition confidence associated with each candidate. It passes this information back to the interface module. The interface module displays the recognition results to the user and allows for correction of the results (in our case, by choosing from a drop down menu containing the alternative candidates). The interface module then passes the bounding boxes, their character candidates -after applying user's corrections- to the structural analyzer. The structural analyzer processes the information from the character recognizer and constructs a well formed mathematical expression. Presentation MathML corresponding to this expression is then generated by the analyzer and is passed back to the interface module. The interface module sends the resulting MathML representation together with whatever operation was specified by the user to the CAS interface tool. This tool is used to interact with the target CAS system and returns the computed results generated from the CAS represented as a presentation MathML expression. The presentation MathML and the format defined by the user are sent to a MathML rendering tool, which generates a set of boxes and characters for the interface module to display. These system components are described in detail in the following sections.

4 Top Level User Interface

The top level of MathBrush is the user interface where the user inputs handwritten mathematical expressions, corrects any mistakes, interacts with the CAS and plays with the resulting computation or expression. For input the user can write multiple expressions in the math sheet and use the context menu for recognition. The user interface displays the recognition results in a separate, nearby window preserving the original relative location of the input ink strokes. This is used to make it easy for a user to decide on the characters that need to be corrected. Corrections, when needed, are accomplished by using a drop down menu of assorted alternate candidates. The expression is then rendered.

Once an expression is rendered a dynamic context menu (menu items change according to the expression type) for output can be used to choose mathematical operations and have the chosen CAS perform the intended operations. Figure (2) shows an example of the use of context menus for operating on output. Results coming back from a CAS are rendered by the top level user interface.

Finally, the interface also gives the user the ability to tailor their environment to their

own preferences. Such preferences include specification of CAS, recognition strategy (after each stroke, after a pause or when user requests), various formats for the input ink and of the rendered results, and so on.

5 Character Recognizer and Structural Analyzer of Input

The mechanism for recognizing handwritten mathematics and its conversion to a valid mathematical expression is handled through the character recognizer and then the structural analyzer. In this section we give a brief description of these two components as currently used in the MathBrush system.

5.1 Character Recognizer

The character recognizer [4] used in MathBrush is an implementation of existing methods found in the literature. We chose to create such a recognizer for practical purposes, since this allowed us to both experiment and to have full control of this process. On the other hand there was no obvious candidate that was available and which could fulfill our requirements.

The character recognizer module receives the ink from the interface module and generates, as most character recognizers do, a set of bounding boxes, with each box including a set of candidate characters along with their recognition confidence values. The character recognizer uses a symbol database that contains samples for each symbol.

The character recognizer [16] involves three phases: stoke preprocessing, segmentation and finally matching. The preprocessing of strokes includes stroke joining, re-sampling, trimming, smoothing, and normalization. The second phase is segmentation where the input is broken into distinct stroke groups. Segmentation is done by first estimating the likely number of strokes which make up the input symbol by using the geometry of strokes to generate proximity and stack hints[16]. This is then followed by a process of feature extraction. Features such as: width, height, and angle between end points are extracted from the input and compared to the feature extracted from the database A *resolution matrix* is used to eliminate the possibility of early reporting of symbols with few strokes included in other symbols. For example, this helps to prevent the reporting of F,- or L,= instead of a correct E.

Once the stroke preprocessing and segmentation phases have been completed the characters are ready to be compared to a database of symbols. The recognition phase combines a number of matching procedures to obtain a final score or confidence value. These procedures include basic elastic matching, deformable template matching and finally structural chain code matching.

We achieved improvement in our character recognition by personalizing and training the symbols database. Tablet PCs tend to have a single primary user and training the database on his handwritting has a great impact on the recognition accuracy. It's always possible to include more than one user profile on a single machine and load or unload them on the fly as needed.

5.2 Structural Analyzer

A structural analysis of an input expression involves the process of converting a set of symbols into a mathematical expression. The mathematical expression can then be passed to a CAS for evaluation or computation. The structural analyzer used in MathBrush is described in [8]. In this subsection we give a flavour of some of the techniques used in the analyzer.

Input for the analyzer is a set of bounding boxes along with a set of candidates for each box ranked by their recognition confidence values. The analyzer makes use of additional information

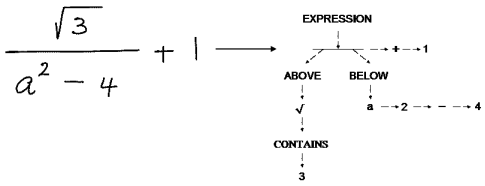


Figure 3: Baseline Tree

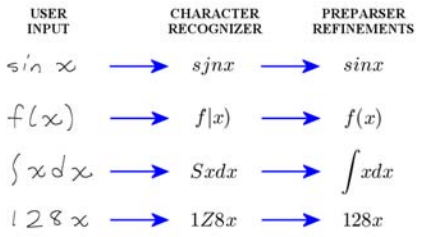


Figure 4: Pre-Parsing Examples

stored in a symbol database and includes, for each character candidate, a unique structural type which defines the symbol's expected positioning on a baseline and one or more possible semantic types, which define grammar rules to be applied to a symbol during expression parsing. The structural analyzer uses the character data and the symbols information in the database to generate a valid math expression. The resulting expression is then converted to presentation MathML and sent back to the interface module for rendering.

Our structural analysis framework [8] consists of four consecutive phases: determination of layout, pre-parsing, structural grouping and final parsing. The phase that determines the layout uses only the bounding box information to determine the symbols' relative locations in order to construct an initial baseline tree. See Figure (3).

Pre-parsing makes use of expected mathematical content to refine and correct some character candidates. Examples include the matching of brackets, integral matching with differential symbols, and determination of function names and numbers (c.f. Figure (4)).

Structural grouping finds baselines in an expression and refines character candidates by fitting characters onto these baselines. This is the main phase in the structural analysis. This process works by first estimating a baseline position based on the candidates and using information from the symbol database. This in turn generates a structural confidence for each candidate which, when combined with character recognizer confidence values is then used to update the candidates list and their probabilities. This process is repeated until the candidate list stabilizes. The final parsing phase involves a final set of refinements making use of a database of likely expressions in order to select the best result.

6 CAS Interface and Rendering Tools

The final two components of MathBrush involve the interaction with a CAS and a rendering of the final mathematical expressions. The CAS interface tool receives the presentation MathML of a math expression from the interface module together with the operation the user wants to perform on that expression and the CAS system the user wants to use. This module forms the

appropriate command and sends it to the CAS. It passes the presentation MathML coming back from the CAS to the interface module for rendering. Currently the CAS interface tool supports interaction with Maple and can be easily extended to work with any CAS that supports MathML.

The Math rendering tool takes as its input a presentation MathML expression passed from the interface module together with any user-requested formats and provides instructions on rendering the expression in the user interface. The tool itself generates a set of boxes with each of the characters to be displayed in that box for the interface to display. A database that stores the operators and external entities dictionary as defined by the W3C consortium [1] is accessed during rendering. A line breaking algorithm has been implemented with the strategy of trying to keep a sub-tree together. For example, it tries to fit a fraction in a line (preferably the current line) and if not then it displays it as numerator/denominator). The rendering algorithm also takes care of stretching certain operators (for example, brackets, square roots, integral and summation operators) in order to fit multi-lined operands if needed. The bounding boxes generated use the font and output format provided by the user.

7 Extensions and Future Work

Work is underway to further improve and experiment with the MathBrush system. For example, there are still many classes of expressions not yet recognized by the system, including for example matrices. Matrices offers a significant set of challenges, both in recognition of such objects and in manipulating the elements inside them. We also plan to investigate the use of guided input such as ruled lines in order to help the character recognizer to distinguish between such things as lower and upper case letters and super- and sub-scripts. Such guided input will also help to establish first approximations of baselines for improved performance for the structural analyzer. In addition, the design of the structural analyzer contains many parameters. We expect to do further experiments with MathBrush on the training of these parameters in order to obtain optimal values. The use of the system has shown that the user most likely corrects the recognition results before rendering. This makes the job of the structural analyzer easier and it makes it better to focus on correcting other structural errors.

In parallel to the above work, it is also our plan to improve recognition accuracy by using a different approach to the recognition problem. We plan to investigate the replacement of our recognizer/analyzer components with a single entity which is based on the use of graphical probabilistic models, in particular Bayesian networks or their extensions. These allow the recognizer to reason naturally like a human, providing the most reasonable guesses within any context. As these models are intuitive, they allow us to improve them or to extend them easily. These models can also be trained easily, which allows them to adapt to individual users.

Currently our focus is also on investigation of editing and manipulation of output expressions making use of a pen. Such actions are typically done with the use of pen and paper and are natural for systems such as MathBrush. We expect to continue with doing such operations in-place. We also expect that editing and manipulation requirements will require alternate representations for our rendered expressions. Having full control over how our output expression is rendered gives an additional reason why we have preferred to work with an interface which is independent of current in CAS interfaces.

8 Conclusions

MathBrush is a system for allowing users to experiment with mathematical computation combining pen-based systems along with CAS. It is designed for experimentation, allowing for

replacement of important components such as the character recognizer or structural analyzer (or both in case of recognition/analyzers that involve feedback loops), interchange one CAS with a different CAS, and so on. It has been constructed in order to allow actual users to make use of such a pen-based system. The availability of a system such as MathBrush allows for investigation of many issues related to using pen-based math systems.

References

[1] D. Carlisle, P. Ion, N. Poppelier, R. Miner (editors), R. Ausbrooks, S. Buswell, S. Dalmas, S. Devitt, A. Diaz, R. Hunter, B. Smith, N. Soiffer, R. Sutor, S. Watt. *Mathematical Markup Language (MathML) Version 2.0*, W3C Recommendation (2001), http://www.w3.org/TR/2001/REC-MATHML2-20010221.

[2] K-F. Chain and D-Y Yeung, *Recognizing on-line handwritten alphanumeric characters through flexible structural matching* Pattern Recognition, 32(7), pp. 1099-1114 (1999)

[3] J.J. LaViola Jr and R.C. Zeleznik, *MathPad^2: A system for the creation and exploration of Mathematical sketches.* ACM Transactions on Graphics. Special Issue: Proceedings of 2004 SIGGRAPH 432-440 (2004)

[4] S. MacLean, *The MathBrush character recognizer*, Internal report for Symbolic Computation Group, 20 pages, (2005)

[5] Microsoft Recognizer Guidelines, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/tpcsdk10/lonestar/appendix/tbconcustomrecognizer.asp

[6] M. Monagan, K.O. Geddes, K.M. Heal, G. Labahn, S.M. Vorkoetter, J. McCarron and P. DeMarco, *Maple Advanced Programming Guide* (2005)

[7] M. Revow, C. Williams and G. Hinton, *Using generative models for handwritten digit recognition*, IEEE Transactions Pattern Analysis and Machine Intelligence 18(6), pp. 592-606 (1996)

[8] I. Rutherford, *Structural Analysis for Pen-Based Math Input Systems*, MMath Thesis, School of Computer Science, University of Waterloo, Waterloo, Canada. (2005)

[9] P. Scattolin, *Recognition of Handwritten Numerals Using Elastic Matching.* Master's thesis, Computer Science Department, Concordia University Montreal (1993)

[10] E. Smirnova and S.M. Watt, *A Context for Pen-Based Mathematical Computing*, Proceedings of the 2005 Maple Summer Workshop, pp. 409-422. (2005)

[11] S. Smithies, *Freehand Formula Entry System*, Master's thesis, University of Otago, Dunedin, New Zealand (1999).

[12] M. Suzuki, F. Tamari, R. Fukuda, S. Uchida, T. Kanahori, *Infty- an integrated OCR system for mathematical documents*, Proceedings of ACM Symposium on Document Engineering 2003, Grenoble, Ed. C.Vanoirbeek, C.Roisin, E. Munson, pp.95-104 (2003)

[13] C.C. Tappert, *Cursive Script Recognition by Elastic Matching*, IBM Journal of Research and development 26(6), pp. 765-771 (1982)

[14] Bo Wan and S.M. Watt, *An Interactive Mathematical Handwriting Recognizer for the Pocket PC*, Proc. International Conf. on MathML and Math on the Web (MathML 2002), June 28-30 2002, Chicago USA.

[15] R. Zanibbi, D. Blostein and J.R. Cordy, *Aiding manipulation of handwriting mathematical expressions through style preserving morphs*, Graphics Interface 2001, pp 127-134 (2001)

[16] G. Labahn, S. MacLean, M. Marzouk, I. Rutherford and D. Tausky, *A preliminary report on the MathBrush Pen-Math System*, Proc. Maple Conference 2006, pp. 162-178 July 2006, Waterloo, Canada.