m_{in}: A Multimodal Web Interface for Math Search

Christopher Sasarak, Kevin Hart, Richard Pospesel,[†]
David Stalnaker, Lei Hu, Robert LiVolsi, Siyu Zhu, and Richard Zanibbi
Department of Computer Science [†]School of Interactive Games and Media
Rochester Institute of Technology, 102 Lomb Memorial Drive, Rochester, NY (USA) 14623
{cms5347,kth1775,lei.hu,rjl3050,sxz8564,rlaz}@rit.edu, {pospeselr, david.stalnaker}@gmail.com

ABSTRACT

m_{in} is a web interface for constructing search queries that include mathematics, using the metaphor of an 'intelligent' blackboard. Formulas are entered using a combination of finger/mouse, keyboard, and images, with symbol recognition results shown using translucent overlays above the user's input. At the user's request, the blackboard is converted to LaTeX and inserted in the query string, and the user's symbols are repositioned and resized on the blackboard to visualize the recognized layout of symbols on baselines (writing lines). Queries may include keywords and multiple LaTeX expressions, and be submitted to a variety of search engines (e.g. Springer LaTeX Search, Wolfram Alpha). m_{in} allows non-expert users to include math expressions in queries without special codes for mathematical symbols, providing text describing a formula, or requiring the use of a template-based equation editor.

ACM Classification Keywords

H.3.3 Information Storage and Retrieval: Information Search and Retrieval, Query Formulation; H.5.2 User Interfaces: Input devices and strategies

General Terms

Design, Human Factors

Author Keywords

Mathematical Information Retrieval (MIR)

INTRODUCTION

We present a new browser-based application for formulating and issuing search queries that contain math. In our system, formulas are represented by their symbol layout, given in LaTeX. Alternatively one could search using the mathematical content of an expression, as given by an *operator tree* defining the order of operations and their scope (e.g. in Content MathML[1]). At present we are focusing on appearance-based retrieval of math, as math non-experts are better able to represent unfamiliar math by appearance than by content, and it has been suggested that mathematical experts often know names for the formulae that they use, making text search sufficient for their needs in many cases [16].

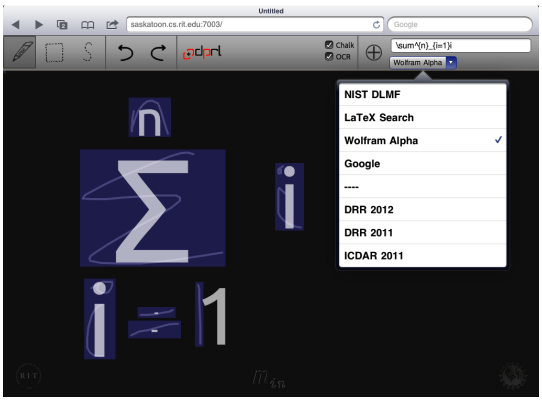

Figure 1. m_{in} running on an iPad

A variety of systems for Mathematical Information Retrieval (MIR) have been developed that allow mathematical expressions to be used directly as queries (*query-by-expression* [12]). For example, the NIST Digital Library of Mathematical Functions [7] (`dlmf.nist.gov`) allows LaTeX formulas and keywords to be included in search queries. Existing MIR systems require formulas to be entered using structure editors, where templates are used to place sub-expressions around or within operators such as fraction lines, summations or square roots [5, 8] and/or manually entering math as text using encodings such as LaTeX or MathML.

The m_{in} system[2] shown in Figure 1 provides a more natural option. The design is based on the metaphor of an 'intelligent' blackboard, where the user can quickly enter and alter expressions informally with recognition results provided unobtrusively in-place. Users may draw expressions using a combination of finger/mouse, uploaded images, and the keyboard. This combination of different input types in one system is novel, and simplifies formulating queries for mathematical information by users without specialized hardware, software, or knowledge of math encodings, such as LaTeX or MathML.

m_{in} is portable, written in Javascript and running in modern web-browsers that support Scalable Vector Graphics (SVG). Recognition algorithms for handwritten, image, and keyboard input run on remote servers which m_{in} queries. Recognition services can be easily changed based on a particular set of user's needs. An open source release of m_{in} is planned.

[1] http://www.w3.org/Math/

[2] http://saskatoon.cs.rit.edu:7004

Proceedings of HCIR 2012, Cambridge MA, Oct 4-5, 2012

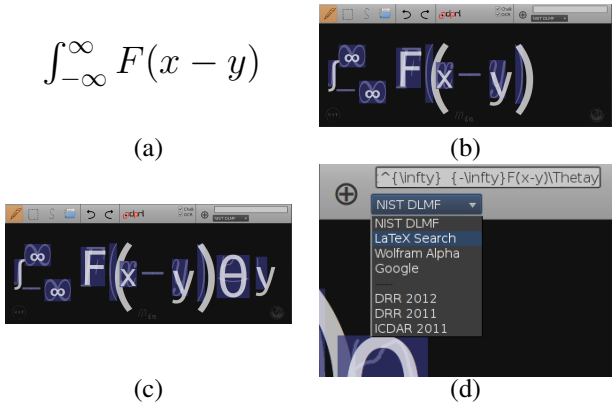

Figure 2. Entering and searching for an expression using an image file and pen strokes in m_{in}

USE CASES

As an example of where a system like m_{in} is useful, suppose that a user wants to find information about the formula shown in Figure 2(c), but does not understand it well enough to describe it in English. The formula also contains θ, which cannot be found on a keyboard.[3] Using a text-based query, it would be difficult for an inexperienced user to satisfy their information need. As shown in Figures 2 and 3, using m_{in}, they can enter the information using images and hand-drawn symbols, and then perform web searches using various search engines. Particularly if an image of the expression is available, the user can simply upload the image to m_{in}, repair the recognized structure, add additional keywords to the query if desired, and then search.

Another case where m_{in} is useful is as a tool for creating mathematical formulas for use in LaTeX documents. A user can upload the image of an expression, draw additional symbols to modify it, and then copy-and-paste the generated LaTeX into their document. Figure 2 illustrates this type of editing. Because m_{in} also incorporates an undo/redo stack the user can work freely without fear of losing their work.

RELATED WORK

Many commercial and research systems for math entry have been developed [12], and surveys of recognition algorithms for mathematical notation are available elsewhere [2, 12]. For space, we will provide just a brief overview of web-based math entry, and systems that significantly influenced the design of m_{in}. Web-based interfaces for math retrieval to date have been textual and/or template-based [5, 8], where templates for vertical structures around operators (e.g. fractions, summations, square roots) are provided to organize sub-expressions. The earliest web-based system for math entry we are aware of is the *Natural Log* system created by Matsakis in 1999 [6]. The system allowed a user to draw expressions in a Java Applet, returning results as LaTeX or MathML. Interaction was very simple, permitting just drawing and deleting pen strokes.

[3] *detexify* (http://detexify.kirelabs.org/classify.html) allows users to search for LaTeX symbol codes using hand-drawn symbols.

Concurrently, Smithies created the *Freehand Formula Entry System (FFES)* [10] which ran as an application.[4] FFES had additional operations to move strokes, and included a 'squiggle' select, allowing users to select stroke groups to correct symbol segmentation. FFES was later expanded to include additional interaction features, including displaying symbol recognition results directly above hand-drawn strokes rather than in a separate image [1], and visualizing recognized structure by 'morphing' symbols to ideal sizes and locations [14]. These changes were motivated by users' difficulty with locating recognition errors when results are displayed separately from the user's input, which has been observed multiple times in the literature [12].

Later on, Pollanen et al. produced *Xpress*, a web-based interface implemented in Ajax. Xpress was designed for creating LaTeX using keyboard and mouse input along with a modified DRACULAE parser [9]. In Xpress, a canvas was provided that typed symbols could be 'dropped' onto, and then moved and resized as desired by the user. Clicking a button would invoke DRACULAE and return a LaTeX string, along with the rendered LaTeX in a separate panel.

MATH ENTRY AND SEARCH

m_{in} incorporates and augments interface features from FFES and Xpress. To our knowledge, this is the first system to combine image, pen/finger, and keyboard input for math. m_{in} was designed to mimic the look of a blackboard. Along the top of the interface are the buttons which control the editor modes (drawing, symbol selection, and stroke/primitive selection) as well as undo/redo, and image upload. At the upper right-hand corner of the interface are the search operations. A complete list of user operations is provided in Table 1.

Table 1. Operations in m_{in}. Operations available only for desktop machines appear with an asterisk (*). Primitives are hand-drawn strokes, connected components in images, and typed characters

Tasks	Operations
INPUT	Draw (mouse/finger)
	*Type (keyboard entry)
	*Upload Image File
SELECTION	Rectangle (Symbol) Selection
	'Squiggle' (Primitive) Selection
EDITING	Undo/Redo
	Move Primitives/Symbols
	Delete Primitives/Symbols
	Pinch-to-Resize (*iPad only*)
	Hold/Touch to Join Primitives into a Symbol
	Symbol Recognition Correction Menu
	Convert to LaTeX; Insert in Query String
VISUALIZATION	Align Symbols (on LaTeX generation)
	Change Visible Blackboard Layers (Chalk/OCR)
SEARCH	Select Search Engine
	Edit Search String
	Search (Enter)

[4] FFES and DRACULAE are available as open source: http://www.cs.rit.edu/ rlaz/ffes/

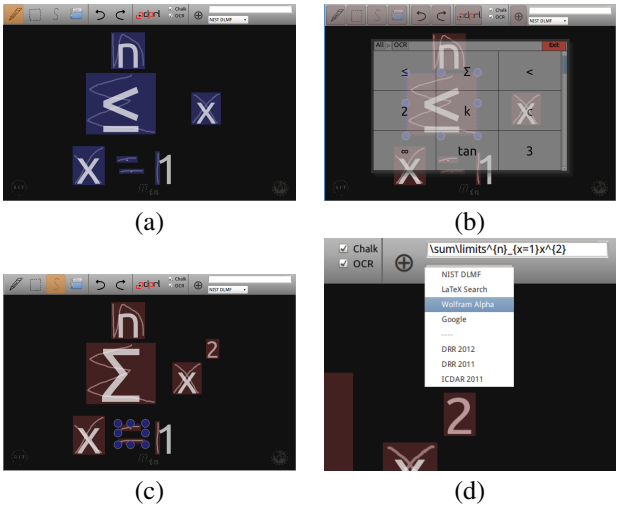

Figure 3. Fixing symbol classification and segmentation errors

An example of these operations is shown in Figure 2. Figure 2(a) shows an image that was uploaded into m_{in} using the folder button. Figure 2(b) shows the m_{in} interface after the image has been broken into connected components and the screen has been populated with them.

Components can be moved by switching into either Rectangle Selection mode or 'Squiggle' Selection mode. In Rectangle selection mode, a rectangle is drawn around the symbols to select. Squiggle selection mode works similarly, but is more granular because it can be used to select individual connected components or strokes in a symbol, by clicking and dragging the mouse or finger over the strokes. For example, for a handwritten $+$, the horizontal bar could be selected independently of the vertical one.

The image in Figure 2(a) is a convolution formula, but is missing the $\Theta(y)\mathrm{d}y$ portion. Using the Pen Mode of m_{in}, it is possible to fill in the missing parts of the formula as shown in Figure 2(c).

In Figure 3(a) a formula has been drawn, where the Σ symbol has been recognized as a $\leq$ symbol. This can be fixed by clicking on the object in any mode and holding. After a few seconds a correction menu appears as shown in (b); the user can then select the correct symbol, as shown in (c). (c) also demonstrates adding characters to the blackboard from the keyboard. A '2' was added to the blackboard and then moved into position to make the equation inside the summation x^2.

Another problem with the formula shown in Figure 3 is that the two strokes of the '$=$' symbol have been displayed as two separate '$-$' symbols. While the DRACULAE parser [13] can parse this correctly (see Figure 1), users may prefer to see '$=$'. To fix this, both strokes can be selected using either Rectangle Selection or Squiggle Select as shown in Figure 3(c). Clicking/touching and holding on the group selection for a second will cause the symbols to be merged together and recognized as a single symbol.

Pressing the $\oplus$ button beside the query invokes a *style-preserving morph* [11] that repositions and resizes symbols to visualize detected structure, and the recognized formula is inserted into the query as a LaTeX string; this is shown in Figure 3(d). For the convolution example the user might add 'convolve' to the search string to focus the search.

A search engine can be selected using the drop-down box next to the text box as shown in Figure 3(d). Pressing enter in the search box or iPad keyboard will make the browser navigate to a search for that term in a new tab. Currently support is provided for Google, Wolfram Alpha, NIS DLMF, and Springer LaTeX Search. After a search has been issued, the user may return to the m_{in} tab in their browser which has been left in the state it had before the query was issued, allowing the query to be reformulated by the user.

SYSTEM ARCHITECTURE

The system architecture is shown in Figure 4. m_{in} is unaware of how recognition is performed because it only sends requests. This makes it easy to adapt m_{in} to other notations such as chemical formulas, flowcharts, or musical script.

m_{in} stores and displays stroke, text and image data using SVG. m_{in} then passes this information on to one or more recognition servers. As shown in Figure 4, these recognition systems can be arbitrarily complex, provided they accept simple XML representations of strokes or images and return recognition results in another simple XML format.

For images, image data is sent through three processes before being returned to m_{in} (shown as '1-NN OCR'). First is the connected components extractor which divides a binarized version of the image into contiguous blobs. The segmenter determines which connected components are to be considered part of one symbol, and finally a nearest-neighbor classifier [17] assigns classes to segmented symbols, and returns this information to m_{in} for display. A second symbol recognizer is used for recognizing strokes drawn directly on the blackboard, using Hidden Markov Models [3, 4].

The production of LaTeX and alignment of symbols to visualize recognized layout is produced using a DRACULAE parser (described earlier). Symbol identities and locations (as bounding boxes) are passed to a DRACULAE web service. The DRACULAE outputs contain information about the sizes and location for symbols to be used when 'morphing' the expression to visualize the detected baselines of the expression.

As shown in Figure 4, Search requests are made by embedding the query string in the URL for a search engine. New search engines may be added by providing a new option to the drop-down menu and a function to create the necessary search URL syntax.

It is easy to integrate new recognizers with m_{in}. m_{in} has different JavaScript objects for representing each of keyboard entered symbols, drawn symbols, and uploaded images. These objects share a common API, and keep a reference to the appropriate recognition server. When it is time for m_{in} to classify items on the blackboard, it groups objects by the server they need to be classified by. This way, m_{in} can send the symbols for a particular server in one request. In order to add a new primitive type to m_{in}, code must be pro-

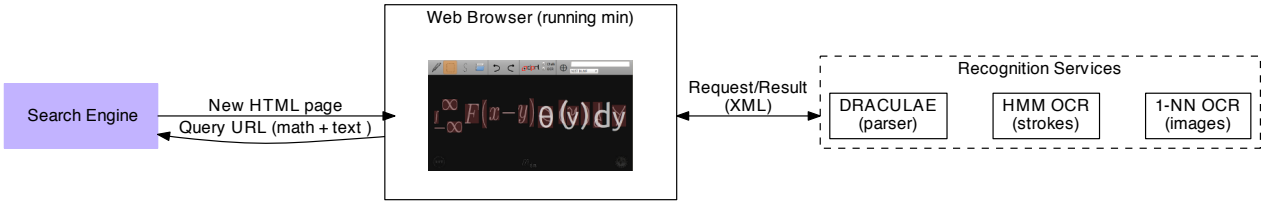

Figure 4. m_{in} Architecture. Queries are strings containing text and LaTeX. Web services provide symbol recognition and parsing, with requests and results represented in XML

vided for managing the object on the blackboard, for producing an XML representation of the primitive, and to provide a reference to the server that this type of primitive should be classified by.

PRELIMINARY FEEDBACK AND FUTURE WORK

Our goal in developing m_{in} is to allow non-experts to be able to search for information about math in a way that is intuitive and simple. The integration of m_{in} with math search engines make it easy to have an edit-search-edit work-flow. m_{in} also has ramifications for retrieval in other domains: the system uses web services for pattern recognition and search, and these may be easily exchanged for others, even for other notations (e.g chemical diagrams, or musical notation).

Some search engines do not accept LaTeX input natively, and it would be useful to add a framework for producing different encodings for symbol layout (e.g. Presentation MathML). We are currently developing such a system, using a common structure representation [15]. We also plan to represent expressions in a query using a drop-down list, with multiple expressions represented and edited concurrently.

We have had some informal feedback from users of m_{in} at an annual public event held at our institution. At that time it was possible to move recognized symbols in the Pen mode, preventing new strokes from being drawn within the bounding box of a recognized symbol. This confused many users at the event, and we also saw that better affordances are needed to suggest the 'click-and-hold' actions used to re-segment symbols and correct classification. We are in the process of carrying out formal studies of both math search use cases and the usability of m_{in}, to identify opportunities for improving the design of the interface.

ACKNOWLEDGMENTS

The idea to encapsulate pattern recognition algorithms in web services came from a discussion with Prateek Sarkar. Meridangela Gutierrez-Jhong suggested having a drop-down list for multiple expressions in m_{in}. This material is based upon work supported by the National Science Foundation (USA) under Grant No. IIS-1016815.

REFERENCES

1. D. Blostein, E. Lank, A. Rose, and R. Zanibbi. User interfaces for on-line diagram recognition. In *Graphics Recognition: Algorithms and Applications*, volume 2390 of *LNCS*, pages 92–103, 2002.
2. K.-F. Chan and D.-Y. Yeung. Mathematical expression recognition: A survey. *IJDAR*, 3:3–15, 2000.
3. L. Hu, R. Hart, R. Pospesel, and R. Zanibbi. Baseline extraction-driven parsing of handwritten mathematical expressions. In *Proc. ICPR*, Tsukuba Science City, Japan, 2012. (to appear).
4. L. Hu and R. Zanibbi. HMM-based recognition of online handwritten mathematical symbols using segmental k-means initialization and a modified pen-up/down feature. In *Proc. ICDAR*, pages 457–462, Beijing, 2011.
5. M. Kohlhase and I. Sucan. A search engine for mathematical formulae. In *Proc. AISC '2006*, number 4120 in LNAI, pages 241–253. Springer Verlag, 2006.
6. N. Matsakis. Recognition of handwritten mathematical expressions. Master's thesis, MIT, Cambridge, MA, May 1999.
7. B. Miller and A. Youssef. Technical aspects of the digital library of mathematical functions. *Annals of Mathematics and Artificial Intelligence*, 38(1):121–136, May 2003.
8. R. Munavalli and R. Miner. Mathfind: a math-aware search engine. In *Proc. SIGIR*, pages 735–735, Seattle, 2006.
9. M. Pollanen, T. Wisniewski, and X. Yu. Xpress: A novice interface for the real-time communication of mathematical expressions. In *Proc. Work. Mathematical User-Interfaces*, Linz, Austria, 2007.
10. S. Smithies, K. Novins, and J. Arvo. A handwriting-based equation editor. In *Proc. Graph. Int.*, pages 84–91, Kingston, Ontario, Canada, June 1999.
11. R. Zanibbi, J. Arvo, K. Novins, and K. Zanibbi. Aiding manipulation of handwritten mathematical expressions through style-preserving morphs. In *Proc. Graph. Int.*, pages 127–134, Ottawa, 2001.
12. R. Zanibbi and D. Blostein. Recognition and retrieval of mathematical expressions. *IJDAR*, (to appear).
13. R. Zanibbi, D. Blostein, and J. R. Cordy. Recognizing mathematical expressions using tree transformation. *TPAMI*, 24:1455–1467, 2002.
14. R. Zanibbi, K. Novins, J. Arvo, and K. Zanibbi. Aiding manipulation of handwritten mathematical expressions through style-preserving morphs. In *Proc. Graph. Int.*, pages 127–134, Ottawa, Canada, 2001.
15. R. Zanibbi, A. Pillay, H. Mouchère, C. Viard-Gaudin, and D. Blostein. Stroke-based performance metrics for handwritten mathematical expressions. In *Proc. ICDAR*, Beijing, China, 2011.
16. J. Zhao, M.-Y. Kan, and Y. L. Theng. Math information retrieval: user requirements and prototype implementation. In *Proc. JCDL '08*, pages 187–196, Pittsburgh, PA, USA, 2008.
17. S. Zhu, L. Hu, and R. Zanibbi. Math symbol recognition and retrieval using turn function and image grid features. *(submitted for publication)*, 2012.