1-1-2007

An initial evaluation of MathPad(2): A tool for creating dynamic mathematical illustrations

Joseph J. LaViola Jr.
University of Central Florida

Available online at www.sciencedirect.com
ScienceDirect
Computers & Graphics 31 (2007) 540–553
COMPUTERS & GRAPHICS
www.elsevier.com/locate/cag

Calligraphic Interfaces

An initial evaluation of MathPad^2: A tool for creating dynamic mathematical illustrations

Joseph J. LaViola Jr.

School of Electrical Engineering and Computer Science, University of Central Florida, 4000 Central Florida Blvd., Orlando, FL 32816, USA

Abstract

MathPad^2 is a pen-based application prototype for creating mathematical sketches. Using a modeless gestural interface, it lets users make dynamic illustrations by associating handwritten mathematics with free-form drawings and provides a set of tools for graphing and evaluating mathematical expressions and solving equations. In this paper, we present the results of an initial evaluation of the MathPad^2 prototype, examining the user interface's intuitiveness and the application's perceived usefulness. Our evaluations are based on both performance and questionnaire results including first attempt gesture performance, interface recall tests, and surveys of user interface satisfaction and perceived usefulness. The results of our evaluation suggest that, although some test subjects had difficulty with our mathematical expression recognizer, they found the interface, in general, intuitive and easy to remember. More importantly, these results suggest the prototype has the potential to assist beginning physics and mathematics students in problem solving and understanding scientific concepts.

Keywords: Mathematical sketching; Mathematical expression recognition; Gestural user interface; User evaluation

1. Introduction

MathPad^2 (see Fig. 1) is a pen-based, Tablet PC application prototype for creating dynamic illustrations used for exploring mathematics and physics concepts [1]. The fundamental technology behind MathPad^2 is mathematical sketching, a pen-based gestural interaction paradigm for mathematics problem solving that derives from the familiar pencil-and-paper process of drawing supporting diagrams to facilitate the formulation of mathematical expressions; however, with mathematical sketching, users can also leverage their physical intuition by watching their hand-drawn diagrams animate in response to continuous or discrete parameter changes in their written formulas [2]. Diagram animation is driven by associations that are inferred, either automatically or with gestural guidance, from recognized handwritten mathematical expressions, diagram labels, and drawing elements.

The essential goal in developing the MathPad^2 user interface was that it be as similar and fluid as pencil and paper, since mathematics and physics problems are often solved using this medium. Thus, we did not want to use any additional hardware (e.g., a modifier key or stylus button) or software (e.g., buttons) modes. Instead, we wanted all interaction to be derived from using digital ink. We developed a gestural user interface for invoking different operations in MathPad^2 because we wanted users able to work as fluidly as possible with the mathematics and drawings they create. We wanted to explore whether our choice of gestures, which by themselves are not part of pencil-and-paper interaction, are thought of as intuitive or at least complementary.

Given the foundations for MathPad^2, we performed an initial usability evaluation to gauge users' performances and reactions to the prototype to validate its design and potential benefit and determine if further, more in-depth studies are needed. More specifically, we are interested in how easy it is for users to use MathPad^2 with only a visual demonstration of how to invoke gestural operations, and in how many mistakes they make in performing various MathPad^2 tasks. We are also interested in how well subjects remember various gestural commands, since this

E-mail address: jjl@cs.ucf.edu.

0097-8493/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2007.04.008

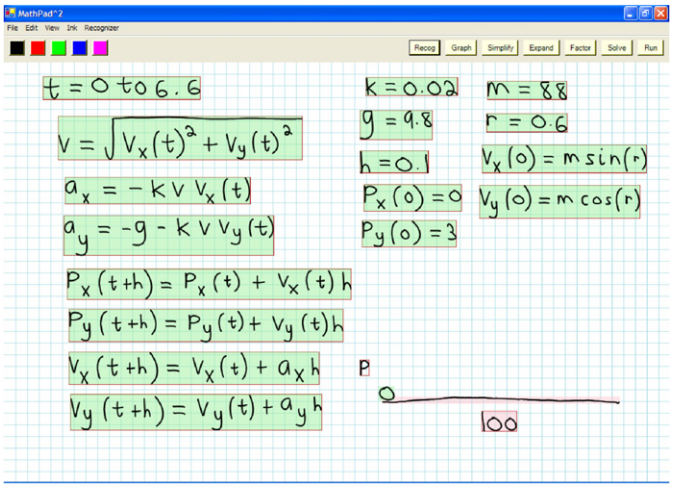

Fig. 1. A mathematical sketch, created in MathPad^2, illustrating how air drag affects a ball's 2D motion. Associations between mathematics and drawings are color-coded.

is a good indicator of intuitiveness. Using interface satisfaction [3] and perceived usefulness [4] questionnaires, we are additionally interested in whether subjects would use mathematical sketching in their work and why. We discuss our evaluation criteria and show that, although some test subjects had some difficulty with our mathematical expression recognizer, subject task performance and qualitative rankings and feedback suggest most of the interface is intuitive, and MathPad^2 is perceived as a powerful tool for exploring mathematics and physics concepts.

2. Related Work

The idea of using computers to create dynamic illustrations of mathematical concepts has a long history. One of the earliest dynamic illustration environments was Borning's ThingLab, a simulation laboratory environment for constructing dynamic models of experiments in geometry and physics that relied heavily on constraint solvers and inheritance classes [5]. Other systems such as Interactive Physics^{TM} and The Geometer's SketchPad^{TM} also let users create dynamic illustrations. Interactive Physics^{TM} uses an underlying physics engine and lets users create a variety of 2D dynamic illustrations based on Newtonian mechanics. The Geometer's SketchPad^{TM} is a general-purpose mathematical visualization tool using geometric constraints. These systems are all WIMP-based (Windows, Icons, Menus, Pointers) [6] and the resulting mode switching and loss of fluidity within the interface makes them difficult to use. Although users of these systems can visualize the dynamic behavior of their illustrations, it is difficult for them to gain a solid understanding of the underlying mathematical phenomena because they cannot write the mathematics. Since mathematical sketching uses handwritten mathematical expressions, users can leverage their knowledge of mathematical notation to create mathematical sketches. When users actually write the mathematics, they gain a better understanding of the concepts illustrated and can learn from their mistakes.

Java applets, providing both interactive and dynamic illustrations, have been developed for exploring various mathematics [7,8] and physics [9,10] principles, as well as algorithm animation [11]. However, these applets are not general, typically provide limited control over the illustration, and rarely show the user the mathematics behind the illustration. In addition, they require a traditional programming language to create the dynamic illustrations.

2.1. Special-purpose languages

Special-purpose languages have also been developed to create dynamic illustrations. For example, Feiner, Salesin, and Banchoff developed DIAL, a diagrammatic animation language for creating dynamic illustrations of mathematical concepts [12]. Brown and Sedgewick [13] developed BALSA, one of the first systems for interactive algorithm animation. Stasko developed the XTANGO [14] and SAMBA [15] animation systems that use high-level scripting languages to create dynamic illustrations, with algorithm animation the focus. Squeak, based on the SmallTalk programming language, is a more modern system for creating dynamic illustrations using a high-level scripting language [16]. Visual languages for creating dynamic illustrations have been developed as well [17–19]. Although these languages are powerful and let users create a variety of dynamic illustrations, they require users to learn a new language and do not take advantage of the naturalness of a pencil-and-paper interaction approach. In contrast, mathematical sketching requires minimal learning, since users already know how to write mathematical expressions.

2.2. Gesture-based interfaces

MathPad^2 utilizes a modeless gestural user interface. Gestural user interfaces have been used in a variety of different applications. For example, Damm et al. used a gestural user interface in their Knight system, a tool for cooperative objected-oriented design [20], and Gross used gestures for creating and editing diagrams for conceptual 2D design [21,22]. In the 3D domain, Zeleznik et al. used gestures for rapid conceptualizing and editing of approximate 3D scenes [23] and Igarashi et al. used gestures in creating free-form 3D models [24]. In other examples, Forsberg et al. used gestures in musical score creation [25] and Landay and Myers developed a gesture-based system for prototyping user interfaces [26]. In addition, electronic whiteboard systems for informal presentations and meetings using gestural interaction have been developed [27,28].

While these gestural interfaces have worked well in their particular applications, they have two important drawbacks. First, they require mode switching to invoke different gestures or to switch between gesturing mode

and drawing mode. Mode switching in these applications, whether accomplished using different mouse buttons [23], keyboard buttons [29], the stylus barrel button [30], and virtual buttons on the computer screen [31], often disrupts users' cognitive interaction flow. Second, these applications often have limited drawing domains: they focus on one particular type of drawing input (e.g., just free-form drawing [22] or gestures for only creating simple 3D geometric primitives [23]). Unlike these applications, mathematical sketching strives for a modeless gestural interface that allows fluid transitions among drawing free-form shapes, writing mathematics, and performing gestural actions.

There has been some recent work on building modeless gestural interface techniques. For example, Saund et al. use an overloaded mouse drag selection technique in an image-editing application that lets users select image material using click selection, a selection rectangle, or a lasso selection based on where they clicked and their selection paths [32]. This approach uses the inferred-mode interaction protocol [33] by examining the pen trajectory and context to determine if object selection or drawing is intended and uses a button for dealing with ambiguities (this approach is similar to Igarashi's suggestive interface techniques [34]). Although this technique is indeed modeless (when the button is not needed), it is limited in scope compared with mathematical sketching because *all* gestural interactions in mathematical sketching are modeless, not merely a subset.

2.3. Pen-based dynamic illustration

In addition to the WIMP and programmatic approaches to making dynamic illustrations, pen-based systems have also been developed. For example, the ASSIST system, developed by Alvarado, lets users sketch diagrams that are recognized as drawing primitives and sent to a mechanical engineering software package for simulation [35]. A similar system lets users sketch drawings of simple vibratory mechanical systems; the system recognizes the primitives and creates a dynamic illustration of the simulation [36].

The key to these systems is that they use domain knowledge about Newtonian mechanics and recognize users' sketches as specific primitives. Thus, although these systems provide powerful illustrations of physics and mathematics concepts, they are limited in their domain knowledge and in hiding the underlying mathematical formulations from the user. Since mathematical sketching uses mathematics as its primary method of telling the system how drawings should behave, our approach is more general and users can create more types of dynamic illustrations.

Pen-based systems have also been developed for other types of dynamic illustration. For example, Pickering et al. developed a system for sketching football plays, simulating them, and then creating a dynamic illustration of the play outcome [37]. Other pen-based systems have been developed for creating traditional animations [38–40].

2.4. Computational and symbolic math engines

The primary focus of mathematical software systems such as Mathematica™, Maple™, MathCad™, and Matlab™ has been entering mathematics for computation, symbolic mathematics, and illustration. Graphing calculators and the myriad of educational math software applications (see Tall [41] for some examples) can be considered smaller versions of these systems. These tools can create dynamic illustrations using mathematics as input. However, the mathematical notation used in these systems is one-dimensional, requiring unconventional notation for concepts that would be intuitive in 2D handwritten mathematics. In addition, these systems do not let the user create diagrams in a natural pencil-and-paper style.

2.5. Mathematical expression recognition and applications

Finally, there has been a significant amount of work in mathematical expression recognition systems that let users enter 2D handwritten mathematics [42–45]. Work in this area began as early as the mid-1960s with expression recognition systems and algorithms developed by Anderson [46] and Martin [47]. However, only a few of these systems go beyond just developing recognition technology. For example, Chan and Yeung [48] developed a simple pen-based calculator, while xThink Inc. developed MathJournal™, a system designed to solve equations, perform symbolic manipulation, and make graphs. MathJournal is the closest in spirit to mathematical sketching because its animation controls let users write down and recognize mathematics, make drawings, and assign the mathematics to the drawings.[1] However, a key limitation of MathJournal's animation control is that users must keyframe their animations (typically providing a starting and ending frame), making the user interface less fluid and contravening how users would make diagrams with pencil and paper. In addition, MathJournal's animation control lacks the iteration and conditional constructs, diagram rectification, and modeless gestural user interface that mathematical sketching supports.

3. The MathPad^2 user interface

To make mathematical sketches in MathPad^2, users write down mathematics, make drawings, and make associations between the two. Additionally, users can invoke mathematical tools such as graphing, function evaluation, and equation solving to help create and manipulate their sketches. In this section, we describe

[1]This functionality did not emerge till after mathematical sketching [1] was first published.

how users perform these tasks with MathPad^2 's modeless gestural user interface. A summary of the commands is found in Fig. 2.

When designing our modeless gestural interface, we wanted the gestures not to interfere with the entry of drawings or equations and still be direct and natural enough to feel fluid. To accomplish this, we use context sensitivity to determine what operations to perform with a single gesture. We also use the notion of punctuated gestures, compound gestures with one or more strokes and terminal punctuation, to help disambiguate gestures from mathematics and drawings. We also wanted to ensure that gestures which seem logical for more than one command should be used for all of those commands. For example, if a particular gesture makes sense for two or three different operations, then we want that gesture to invoke all those operations. More details on the design of and methodology behind these gestures can be found in [1,2].

To write mathematical expressions, users simply write them down using the stylus as if they were using pencil-and-paper. To have the system recognize a mathematical expression, users must lasso the expression and make a tap inside the lasso. Recognized symbols are presented to users in their own handwriting since MathPad^2 has handwriting samples from individual users as a result of our writer-dependent mathematical expression recognition engine. When users move the stylus over the bounding box of the recognized mathematical expression, a green button appears in the box's lower right corner, and when pressed, shows whether the expression was parsed correctly. If a mathematical expression is recognized incorrectly, users can simply erase the offending symbols using a scribble erase gesture followed by a tap and then re-recognize the expression. Users can also tap on a recognized symbol to get a list of alternates. If there is a parsing error with the mathematical expression, users can lasso the offending symbols and interactively move them to a new location where the complete expression will be reparsed.

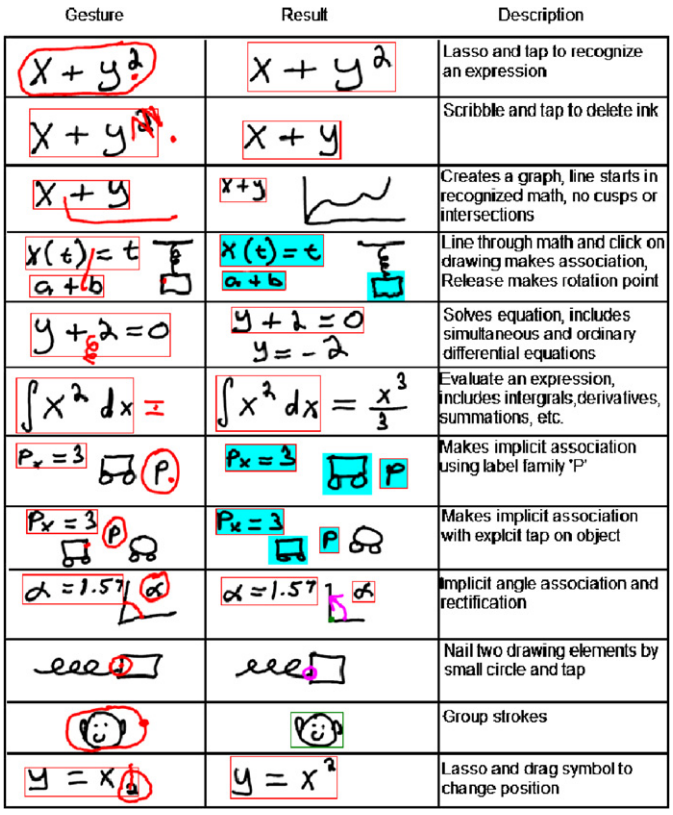

Fig. 2. MathPad^2 's gestural commands. Gesture strokes in the first column are shown here in red. In the second column, cyan-highlighted strokes provide association feedback (the highlighting color changes each time a new association is made), and magenta strokes show nail and angle association/rectification feedback.

Users make drawings in the same way they write mathematical expressions except that the ink strokes need not be recognized. We refer to these ink strokes as drawing elements and they can be grouped together to form composite drawing elements. Users lasso the drawing elements they want to compose and make a tap on the lasso line. Tapping on the lasso line distinguishes this operation from recognizing mathematical expressions. Users can also nail drawing elements together by drawing a small circle over them and making a tap inside the circle. Nailing drawing elements together lets users make stretchable objects. Note that the drawn circle must not completely contain any drawing elements in order to be recognized as a nail gesture. This constraint distinguishes it from the gesture for making composite drawing elements and recognizing mathematical expressions.

One of the most important components of MathPad^2 is the ability to associate mathematics to drawing elements so they know how to behave during an animation. Users can make associations either explicitly or implicitly. Users make explicit associations by simply drawing a line through the bounding boxes of all the necessary mathematical expressions and tapping on a particular drawing element. As the stylus hovers over drawing elements, they highlight to give users feedback about which drawing element they will select. These types of associations are "explicit" because the user has to physically select the mathematics with the stylus to make an association with a drawing. Implicit associations are made by labeling a drawing element with a variable name or constant value and can be either point or angle associations. Point associations are made in the same way that mathematical expressions are recognized except the tap is made on the drawing element instead of inside a lasso. Angle associations are made by drawing an angle arc and label. Then users lasso the label and make a tap whose location on the arc determines the *active line*—the line attached to the arc that will move when the angle changes. The apex of the angle is then marked with a green dot, and the active line is indicated with an arrowhead on the angle arc. In either case, MathPad^2 uses the label to find all of the required mathematical expressions that should be associated to the drawing element. These types of associations are "implicit" because the user simply has to label the drawing and the

system uses it to find the required mathematics needed to make an association.

Finally, MathPad[2] provides users with a mathematical toolset for graphing and evaluating functions as well as solving equations. Users graph functions by simply drawing a sufficiently long, smooth line with no self-intersections, starting inside the bounding box of a recognized mathematical expression, intersecting any other functions along the way, and ending outside all expression bounding boxes. This gesture creates a graph control widget where users can view plots of the functions the graph gesture has intersected and also change the domain and range of the functions by writing down the values and pressing the update button.

Users evaluate mathematical expressions such as integrals, summations, and derivatives by writing an equal sign to the right of the expression and making a tap inside the equal sign's bounding box. The results are then displayed to the right of the drawn equal sign. Users solve single, simultaneous, or ordinary differential equations, by making a squiggly gesture (see Fig. 2). This gesture is identical to the graphing gesture except the line must contain two self-intersections. The results are then displayed underneath the last intersected equation.

4. Experiment 1: mathematical expression recognition

The first part of gauging MathPad[2] 's overall usability is to determine how well our mathematical expression recognizer performs on users' handwritings. Our mathematical expression recognizer consists of a symbol recognizer and a 2D parser. The symbol recognizer uses a pairwise AdaBoost classifier along with the Microsoft handwriting recognizer, and the 2D parser uses a context-free grammar with 2D rules for determining symbol dependencies. More information on our recognizer can be found in [2].

4.1. Experimental design and tasks

Before subjects could participate in the recognition tasks, they all provided writing samples so the mathematical symbol recognition portion of the mathematical expression recognizer could learn their particular writing styles.[2] Each subject used a training application and was shown how to write on the screen, how to erase ink (using the scribble erase gesture), and how to store writing samples in the system. Subjects then wrote each symbol 20 times. The first 10 were written normally and the second 10 were written as small as possible to help detect when small symbols are written. Subjects provided writing samples for 48 different symbols including $a-z$, $0-9$, Σ, (,), $-$, $\sqrt{}$, $\int$, {, $<$, $>$, $+$, $\neq$, and *else*.[3] We chose this particular symbol set because it is the minimal set needed to create mathematical sketches or use tools that MathPad[2] supports.

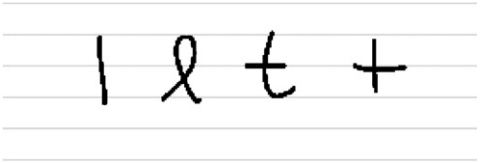

Fig. 3. Example of symbols that needed to be written as shown.

To guide subjects in symbol training, they were given a symbol sheet showing a handwritten version of each symbol. The sheet showed how certain symbols should be written. While, in general, subjects could write symbols, however, they wanted, it is very difficult to distinguish among certain characters. Therefore, we asked subjects to provide writing samples for symbols such as "1", "l", "t", and "+" exactly as they were shown (see Fig. 3) on the symbol sheet so they were distinguishable. Note that subjects were encouraged to put down the stylus after 15–20 symbols so their hands would not tire. It took subjects an average of 50 min to enter their writing samples.

After the training phase, subjects were run through the mathematical symbol accuracy test. Subjects were instructed to write each symbol 12 times when prompted by the training application, then click on a button to commit their test data. We asked subjects to perform this task for several reasons. First, we wanted a normalized test so that each symbol would have equal weight in determining the recognizer's accuracy. Second, we wanted to evaluate the recognizer's accuracy without requiring subjects to write symbols with widely varying sizes, as occurs frequently when writing mathematical expressions (e.g., subscripts, integration limits). Finally, from our observations of how people write mathematics, we wanted to look for any significant differences between the symbol recognizer's performance when subjects write a symbol repeatedly versus writing them in mathematical expressions. Subjects were encouraged to take short breaks after every 10–15 symbols. On average, subjects took about 25 min to complete this test.

After a short break, subjects were run through the mathematical expression accuracy test. Before taking the test, subjects were asked to write a few simple trial expressions such as x^2, x_1, $\frac{3}{5}$, $\sqrt{xyz}$, and $\int_0^2 x\,dx$. These expressions gave subjects a feel for the recognizer and an idea of how it parses mathematical expressions. Note that we purposely kept subjects' practice time to a minimum to reduce any adaptation they might make to the recognizer. Since we designed the parsing rules to be general, it was

[2]Any new user of the system must provide handwriting samples to train the recognizer since it is writer dependent.

[3]The "." and ":" symbols are also part of the symbol set but subjects need not provide samples for them.

important to evaluate how well subjects' written expressions fit within those rules.

Subjects wrote 36 different mathematical expressions (see Appendix A), writing each one when prompted by the training application. The mathematical expressions were chosen from a set used in Chan's recognition experiments [49] plus expressions of our own design. The test expressions range from simple to complex and are representative of the types of expressions users would write in MathPad^2. Subjects were encouraged to take breaks after writing 10–15 expressions so their hands would not tire. On average, subjects took about 45 min to complete this test.

4.2. Participants

Eleven subjects (seven males and four females) participated in the mathematical symbol and expression recognizer accuracy study. Their ages ranged from 19 to 38, and all were students or staff of Brown University. Subjects included computer science students and research staff members as well as physics and applied mathematics majors. Only one subject was left-handed. Six subjects had never used a Tablet PC before, while one had used a PDA and the other four had used Tablet PCs extensively. Of the 11 subjects, 2 of them were considered experts in using our mathematical expression recognizer, 2 of them had used our recognizer only in passing, and the remaining 7 subjects had never used the recognizer. We wanted to have two subjects with expert knowledge of our recognizer to provide a benchmark for how well users could perform with extensive training and use of the recognition engine. All subjects were paid $30 for their time and effort.

4.3. Evaluation measures

We use two measures for evaluating the mathematical expression recognizer. The first measure is a simple symbol accuracy metric, defined as the number of correct symbols divided by the total number of symbols written. This metric is used in the mathematical symbol recognizer test and the mathematical expression recognizer test. The second is a parsing accuracy metric, defined as the number of correct parsing decisions made divided by the total number of parsing decisions. This metric is similar to the R_o metric in [50] and is used only in the mathematical expression recognizer test.

As an example of calculating how many parsing decisions the algorithm needs to make, consider $y = 2x/\sqrt{x}$. The algorithm must make a total of seven parsing decisions. First, it needs to decide if the horizontal line is a fraction delimiter or a minus sign (one decision). Second, it must determine what symbols are above and below the fraction line (four decisions). Third, since there is a square root sign, the algorithm must determine what symbols should be operated on by the square root (one decision). Finally, the algorithm must decide whether the x in the numerator is a superscript of the 2 or simply multiplied by it (one decision).[4] Note that we do not include whether symbols are to the left or right of the "=" sign as part of our parsing accuracy metric. Including them would overly bias the metric in a positive direction, since we have found that our parser has little if any trouble making decisions on whether symbols are to the left or right of "=".

Table 1
Symbol recognition accuracy from the symbol test (SYM), expression test (EXP), and the their combination (TOT)

	Mathematical symbol recognition accuracy		
	SYM (%)	EXP (%)	TOT (%)
Mean	95.7	94.5	95.1
Std. dev.	2.77	2.49	2.65

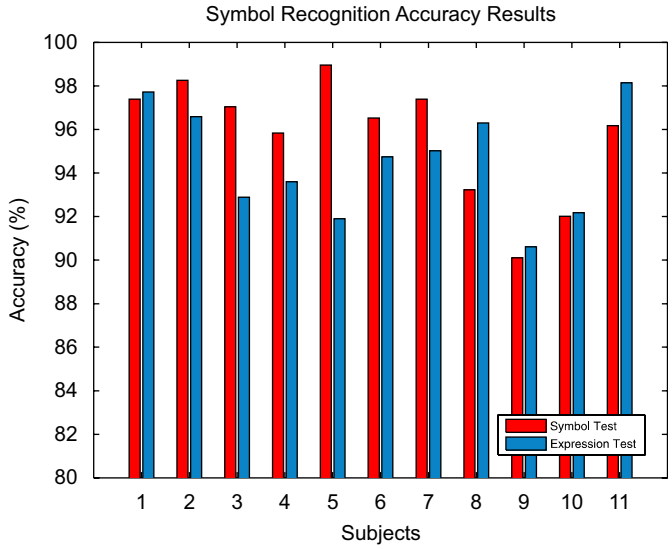

Fig. 4. The accuracy of our symbol recognizer in the symbol and expression tests for each subject.

4.4. Results and discussion

Table 1 shows the overall recognition accuracy results for the mathematical symbol test (SYM) and the symbol accuracy component of the mathematical expression test (EXP) for our recognizer. Each subject wrote 576 symbols in the mathematical symbol test and 703 symbols in the mathematical expression test. A total of 14,069 symbols were used to test the recognizer with an overall accuracy of 95.1%. Fig. 4 shows the recognition accuracy results for both tests on a subject-by-subject basis. There is no significant difference between SYM and EXP ($t_{20} = 1.07$, $p > 0.29$), indicating that the recognizer deals

[4] A subscripted number is not possible in our parsing algorithm.

equally well with isolated symbols and recognizing symbols in the context of mathematical expressions.

It is difficult to compare our recognizer's performance with that of other recognizers in the literature because many recognizers do not have reported accuracy numbers and those that do use different test data, test on different numbers of symbols, and break up their results in different ways. However, we can make some rudimentary comparisons with other recognizers. Li and Yeung [51] achieved 91% accuracy for lower- and upper-case letters and digits. Chan and Yeung [52] reported 97.4% accuracy for upper- and lower-case characters, while Connell and Jain [53] achieved 86.9% accuracy for lower-case characters and digits. Scattolin and Krzyzak [54] reported 88.67% accuracy for digits. Garain and Chaudhuri [55] reported 93.77% accuracy for 198 different symbols. Finally, Matsakis [43] claimed 99% accuracy for 60 symbols for a single user. From these results, we believe that our symbol recognizer has comparable accuracy to other recognizers.

The parsing component of our mathematical expression recognizer made correct parsing decisions 90.8% of the time with standard deviation of 4.47%. Six hundred and three parsing decisions had to be made for all 36 mathematical expressions per subject. Thus, 6633 parsing decisions were used to evaluate the parsing component. In the best case, the parser achieved an accuracy as high as 99.2% and in the worst case as low as 83.6% (see Fig. 5). The variability of these results stems from the writer-independence of the parsing component. Some subjects performed very well with our parsing rules while others had more difficulty, indicating that more flexible parsing rules are probably required. However, with more practice, subjects adapted better to these parsing rules. Subjects 1 and 2 in Fig. 5 are a benchmark for our parsing system because they are considered experts with our mathematical expression recognizer. Thus, accuracies between 95% and 99% should be possible across all subjects with adequate use. Having training data on how users write mathematical expressions could also improve the overall accuracy of the parsing component.

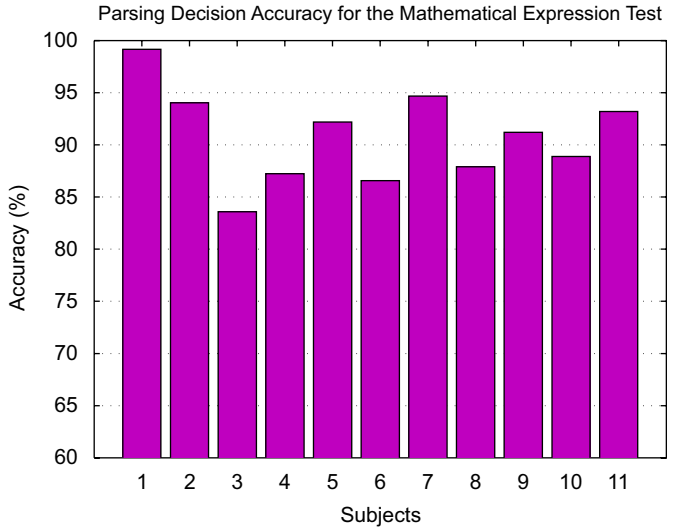

Fig. 5. Parsing decision accuracy across subjects.

Comparing other mathematical expression parsing systems with our own is difficult, especially since many reported results are from experiments that assume the symbols coming into the parsing component are 100% correct. Therefore, making comparisons using these parsing systems is inappropriate. Both Fukuda et al. [56] and Chang and Yeung [50] conducted parsing experiments similar to ours. Fukuda et al. achieved parsing accuracies of 98.46% with accuracy measured by the number of correctly parsed spatial relationships between mathematical symbols. In addition, they restricted subjects to write mathematical expressions in the correct left-to-right order. Chang and Yeung achieved accuracies of over 99% using 600 mathematical expressions as test data, but their automatic error detection and correction system boosted accuracy dramatically. From these results, we believe that the parsing component of our mathematical expression recognizer works well in some cases but definitely needs improvement.

5. Experiment 2: MathPad^2 evaluation

5.1. Experimental design and tasks

The goal of our initial usability experiment is to get users' reactions to the prototype to validate the user interface design and its potential benefit as well as determine if further, more in-depth studies are needed. More specifically, we wanted to evaluate the intuitiveness of MathPad^2 's user interface and gauge the perceived usefulness of the tool. Writing down mathematical expressions and making drawings is a fairly intuitive task, and although our gestural commands need to be taught, we felt they were designed so that they should be easy to understand given simple demonstrations of their use.

In the experiment, subjects must complete six tasks representing common interactions that a student or teacher would perform with MathPad^2. Before a subject performs each task, the experimenter shows the subject how to perform the required gestures for that task via demonstration only. Tasks 1–3 were designed to test how well users were able to use the graph, equation solving, and expression evaluation gestures. First, subjects were shown how to write and recognize mathematical expressions using the lasso and tap gesture, how to erase ink using the scribble erase gesture, and how to use the correction user interface (i.e., erasing and re-recognizing a symbol, tapping on a symbol to show alternates, moving a symbol for reparsing). Then, they were shown how to perform each task specific gesture or command. For task 1 (Graphing), after being shown the required gestural commands, the subjects write, recognize, and then graph $y = x$, $y = x^3$, and $y = \cos(x)e^x$. Then subjects change $y = x^3$ to $y = x^2$, graph the function, and change the function's domain from $-5 \ldots 5$ to $0 \ldots 8$. For task 2 (Equation Solving),

subjects write down and recognize $x^2 - 16x + 13 = 0$ and solve the equation. Next, subjects write and recognize $x^2y + 2y = 4$ and $3x + y = 2$ and solve this set of simultaneous equations. For task 3 (Expression Evaluation), subjects write down the following expressions and evaluate them:

- $\int_0^2 x^2 \, dx$
- $y = \int x^2 \cos(x) \, dx$
- $\frac{dy}{dx}$
- $\frac{d^2y}{dx^2}$
- $\sum_{l=0}^{5} (l-1)^2$.

In all tasks, subjects were instructed to use the correction user interface if the recognizer incorrectly recognizes symbols or expressions.

Tasks 4–6 were designed to let users make mathematical sketches and evaluate whether they prefer to use implicit or explicit associations. Task 5 also was designed to evaluate how well subjects can make nails. Note that only task 4 required subjects to write down the necessary mathematical expressions. Tasks 5 and 6 used prewritten mathematical expressions because we felt having them write and recognize these expressions was not needed, given the many expressions they had already written in the mathematical expression recognition study (see Section 4). However, with task 4, we wanted to see how well subjects could make a mathematical sketch from beginning to end. For these tasks (4–6), the experimenter described the sketches to each subject orally.

The fourth task (Bouncing Ball) has subjects create a complete mathematical sketch of an object bouncing along the ground. Subjects write and recognize the four mathematical expressions shown in Fig. 6, make a drawing with a horizontal line representing the ground and a composite drawing element consisting of three circles drawn near the start of the horizontal line. Next, subjects write the number 20 and associate it to the horizontal line. Finally, subjects associate the mathematics to the composite drawing element, either choosing an explicit association or using an implicit association with the letter "p" as a label, and run the sketch. Note that if MathPad^2 fails to recognize subjects' mathematical expressions after several attempts, we provide them with prewritten expressions. However, we do not make them aware of this when the instructions for this task are given.

The fifth task (Oscillator) had subjects create a mathematical sketch illustrating damped harmonic oscillation. The experimenter instructs subjects to first draw a line and make seven nail gestures along that line. This subtask does not have anything to do with the mathematical sketch itself, but gives us additional accuracy data on how well subjects can perform the nail gesture. Subjects make a drawing consisting of a horizontal line, a spring underneath the line, and a box underneath the spring (see Fig. 7). Subjects then use two nail gestures to nail the horizontal line to the spring and the spring to the box. Next, subjects associate the mathematics to the box, using an explicit or implicit association with the letter "y" as a label, and run the sketch.

In the last task (2D Motion), subjects created a mathematical sketch illustrating 2D projectile motion subject to air resistance (see Fig. 1). Subjects draw a horizontal line and a ball near the left side of the horizontal line. They then associate the number 100 to the horizontal line. Finally, subjects associate the mathematics to the ball, using an explicit or implicit association with the letter "p" as a label, and run the sketch. After all six tasks are completed, subjects answer a post-questionnaire.

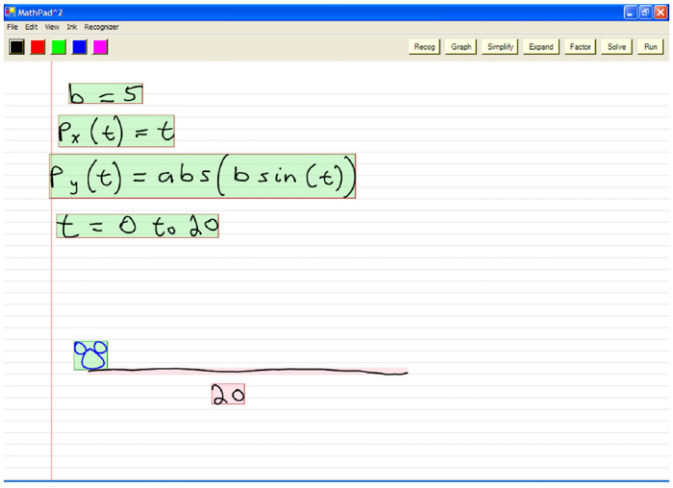

Fig. 6. The fourth task in the MathPad^2 usability test.

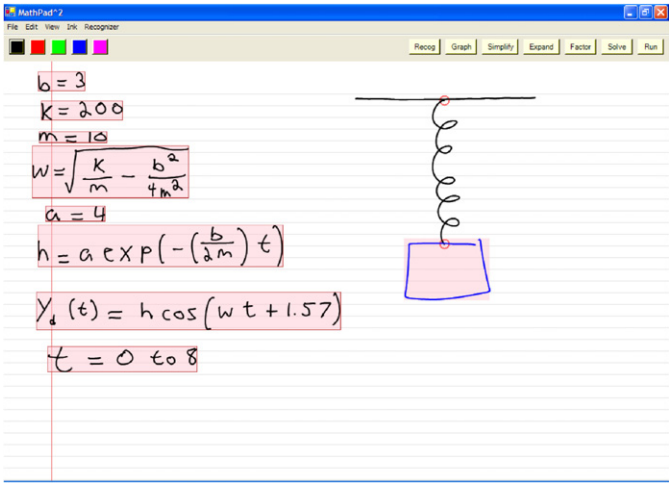

Fig. 7. Subjects create a damped harmonic oscillator in the fifth task.

5.2. Participants

Seven subjects (four men and three women) participated in our usability evaluation. These subjects also participated in experiment 1 (subjects 5–11) but were not the subjects who had prior experience with Tablet PCs or initial familiarity with our mathematical expression recognizer. Subjects were recruited from the Brown University undergraduate population and were either physics or applied mathematics majors. We chose this particular user population because MathPad^2 was designed for mathematics and physics students. Subjects' ages ranged from 19 to 23 and all were right-handed; only one had used a pen-based computer before (a PDA). All seven subjects were asked prior to the study if they had used mathematical software before and which packages: six subjects answering yes and had used a variety of different packages including Matlab, Mathematica, and Maple. All seven subjects were paid $30 for their time and effort.

5.3. Evaluation measures

We evaluate MathPad^2 's usability using quantitative and qualitative data from subjects' task performances and from a post-questionnaire. As subjects performed the six experimental tasks, the experimenter recorded important information about subjects' performances in completing each task, the decisions they made, and counted their mistakes.[5] Performance is characterized by whether subjects can complete each task and how well they do on each subtask. Therefore, the experimenter records whether or not subjects make the appropriate gestures correctly and, if so, whether on the first attempt. Knowing how well subjects perform gestural operations on their first attempt is an important measure because it tells us how easy the gestures are to make and remember. The experimenter also records subjects' choices of implicit and explicit associations in tasks 4–6 so as to get a quantitative metric for their preferences.

After subjects have completed all six tasks they are given a post-questionnaire designed to get their reactions to the MathPad^2 user interface and its perceived usefulness as well as assess how well they remember certain gestures. The post-questionnaire consists of four parts. The first and second parts are adapted from Chin's Questionnaire for User Interface Satisfaction [3] and asks subjects to rate MathPad^2 's user interface as a whole and its individual components. The third part of the post-questionnaire, the recall test, asks subjects to show what gestures they would use for six different operations. The fourth part of the post-questionnaire was adapted from the Perceived Usefulness portion of Davis's questionnaire for user acceptance [4] and asks whether subjects would use MathPad^2 in their work. After subjects answer the post-questionnaire, the experimenter reviews it with them to make sure their answers are clear and to elaborate further on any specific parts of MathPad^2.

[5]We could have used a more sophisticated recording setup, such as videotaping subjects during the experiment. However, subjects used a Tablet PC that was connected to a monitor so the experimenter could easily see what subjects were doing without hindering them during trials. In addition, the experimenter's recording task was simple enough to be done in real time.

5.4. Results and discussion

5.4.1. Task performance results

For the first three tasks, subjects were able to write and recognize all of the mathematical expressions fairly easily. In some cases, they had to use the correction user interface to fix recognition errors, generally getting MathPad^2 to recognize their expressions on the second or third attempt. Twenty-seven out of 28 graphing operations (4 per subject) were completed on the first attempt. Subjects also had to change the domain of a graph; they all completed this operation on the first attempt. Twelve out of 14 equation solving operations (2 per subject) were completed on the first attempt. The other two equation solves were correctly performed on the second attempt. Thirty-four out of 35 expression evaluations (5 per subject) were completed on the first attempt. One subject, however, did have difficulty in getting MathPad^2 to recognize $\mathrm{d}^2y/\mathrm{d}x^2$ and even after multiple attempts was not able to evaluate the expression.

All seven subjects were able to complete tasks 4–6 making the dynamic illustrations. Subjects also had no difficulty in making the drawings for each task and only once did a subject have trouble making a composite drawing element. In the Bouncing Ball task, 12 out of 14 associations were completed on the first attempt and 8 of them were done implicitly. Three subjects did have difficulty in getting MathPad^2 to recognize the required mathematical specification for the Bouncing Ball task and, after multiple attempts (about 10 min), were given prewritten expressions. The difficulty was not in symbol recognition, but in expression parsing. Two of these subjects had parsing decision accuracies below 90% in the mathematical expression test while the other subject's accuracy was 92%. This result provides evidence indicating that higher parsing decision accuracy is needed. In the Spring task, 56 out of 63 nails (7 per subject) were completed on the first attempt. Most of the remaining nails were completed on the second attempt. However, one subject required several attempts to make the necessary nails and had to recreate the drawing after inadvertently erasing part of it when erasing an incorrectly recognized nail. Subjects had to make one association in this task, and all seven were completed on the first attempt explicitly. For the 2D motion task, subjects completed 12 out of 14 associations on the first attempt with all of them made implicitly. One subject did have some difficulty with the implicit associations and needed several attempts to make them correctly.

Overall, subjects did well on all six tasks, considering they had no hands-on training beforehand. Their first

Table 2
A breakdown of test subjects' first attempt gesture performance

	First attempt gesture performance summary		
	Completed	Total	Percentage (%)
Graphing	27	28	96
Equation Solving	12	14	86
Exp. Evaluation	34	35	97
Nails	56	63	88
Associations	31	35	89
Total	160	175	91.4

attempt performances are summarized in Table 2. Subjects had no difficulty in making a lasso and tap to recognize mathematical expressions or in using the scribble erase gesture. In only one case did a subject not complete part of a task and this was due to MathPad^2 's inability to recognize an expression correctly. Subjects completed 160 out of the 175 gestural operations correctly (91.4%) on their first attempt. This number is high considering that subjects had not practiced any of the gestural commands. One subject did have some difficulty with implicit associations due to problems with making taps. The greatest problem subjects had with the six tasks was obtaining correctly recognized expressions in certain situations. That three out of the seven subjects required prewritten mathematics for the Bouncing Ball task shows that the mathematical expression recognizer needs improvement.

5.4.2. Post-questionnaire results

Overall reaction. Table 3 summarizes subject's overall reaction to MathPad^2. It shows that they had a positive reaction to the prototype. When subjects were asked why they chose their rankings, most asserted that MathPad^2 works well, is easy to use, and would be very useful for students in a classroom setting and/or doing homework problems. One subject was "amazed at the application's power". Two subjects claimed MathPad^2 was easy to use but could be frustrating when it had trouble recognizing their handwriting; this frustration explains why the second and third rankings in Table 3 are slightly below the first and fourth rankings.

Ease of use. Subjects rated different parts of the MathPad^2 user interface from 1 (easy) to 7 (hard). Table 4 summarizes these results and shows that subjects found the tasks they had to perform easy to do. Subjects gave recognizing expressions the highest average ranking, indicating the fact that some users had trouble getting MathPad^2 to recognize their handwriting. When asked about their ranking, they stated that the gesture for recognizing mathematical expressions (i.e., lasso and tap) was easy to do, but the results of the recognition operation led them to choose a worse ranking on the easy (1) to hard (7) scale.

Association preference. All seven subjects preferred explicit associations, claiming they were easier to remember and simpler and faster to perform. However, they did say that when associations need to be made with a drawing element and a large set of mathematical expressions, the implicit method is more appropriate. We can thus conclude that both association methods have their place in mathematical sketching.

Table 3
Subjects' average ratings of their overall reaction to MathPad^2 on a scale from 1 to 7

	Overall reaction to MathPad^2	
	Mean	Std. dev.
Terrible = 1, wonderful = 7	6.42	0.54
Difficult = 1, easy = 7	5.57	0.98
Frustrating = 1, satisfying = 7	5.57	1.13
Dull = 1, stimulating = 7	6.14	0.38

Table 4
Subjects' average ratings of ease of use for different components of the MathPad^2 user interface (scale: 1 = easy, 7 = hard)

	MathPad^2 user interface ease of use	
	Mean	Std. dev.
Writing mathematics	1.43	0.97
Recognizing mathematics	2.57	1.81
Graphing functions	1.0	0.0
Solving equations	1.0	0.0
Evaluating expressions	1.0	0.0
Grouping drawing elements	1.57	0.79
Making associations	1.71	0.76
Making nails	1.57	0.59

Correction user interface. Five out of the seven subjects tested found the correction user interface helped them. The two subjects who said no claimed that the alternate lists gave them no help in correcting recognition errors. One subject wanted more choices to appear in the alternate lists, especially in the equation alternate list.

Positive and negative UI aspects. Most subjects identified the most positive aspect as its ability to quickly make drawings move as described by mathematical equations. Two subjects claimed that solving equations was one of the user interface's most positive aspect. One subject thought that the best part of MathPad^2 's user interface was the scribble erase command; another subject said the user interface's simplicity was its most positive aspect. Three subjects stated that getting MathPad^2 to recognize certain symbols and equations correctly was the most negative aspect of the user interface. Two subjects stated that the lack of interactive feedback for implicit associations was a significant drawback, and one subject stated that a negative aspect was the time necessary to get used to the gestural commands. Finally, two subjects said that MathPad^2 's user interface had no negative aspects.

Table 5
Subjects' average ratings of the perceived usefulness of MathPad^2 in their work (scale: 1 = unlikely, 7 = likely)

	MathPad^2 perceived usefulness	
	Mean	Std. dev.
Accomplish tasks faster	5.14	1.95
Improve performance	4.71	2.36
Increase productivity	5.0	1.91
Enhance effectiveness	5.14	2.04
Easier to do work	5.57	1.90
Useful in work	5.42	2.37

Overall ease of use. On average, subjects gave MathPad^2 a 1.86 (1 equals easy and 7 equals hard) with a standard deviation of 0.69. When they were asked to explain their ratings, two dominant themes emerged. First, subjects found the interface easy to use and remember, but were in some cases frustrated by problems in mathematical expression recognition. However, the subjects who had trouble with recognition all felt it would improve with more practice. Those subjects were also asked if they would still use MathPad^2 in spite of their recognition problems; they all said they could deal with these problems because of the functionality MathPad^2 would give them. Second, subjects felt the interface was easy to use once it was explained, a result that helps to validate our demonstration-based teaching protocol.

Gesture recall test. Subject were asked how to invoke gestural commands for graphing, solving equations, evaluating expressions, recognizing a mathematical expression, making nails, and making implicit associations. This part of the questionnaire took place about 5–10 min after they used MathPad^2. Subjects answered 38 out of the 42 recall questions correctly (six per subject) for a recall rate of 90%. Of the four questions subjects answered incorrectly, three subjects missed the equation solving gesture (squiggle) and one missed the expression evaluation gesture (equal and tap). The 90% recall rate indicates that subjects had little difficulty remembering MathPad^2 gestures a short time after they used the prototype,[6] except for the equation solving gesture. Even though three out of the seven subjects forgot the equation solving gesture, they still claimed it was easy to use based on their mean ranking in Table 4.

Likely usage. Table 5 summarizes subjects' ratings on the different "perceived usefulness" statements, on a scale of 1 (unlikely) to 7 (likely). Most subjects would use MathPad^2 in their work. When asked to explain their ratings, four subjects stated that the application would help them to do their classwork and obtain a better understanding of problems and concepts. However, there was no consensus on whether MathPad^2 would speed their understanding of these problems and concepts. One subject said that the ability to quickly solve equations and make graphs would be very beneficial. Two subjects said they did not think they would use MathPad^2 in its current form in their work (explaining the high standard deviations in Table 5). Both of these subjects work in theoretical physics, one in optics and the other in modern physics. However, one of these subject stated she would have used MathPad^2 during beginning physics classes while the other stated he would use MathPad^2 if it had support for light ray and optics diagrams. Finally, all seven subjects felt the application would be a good tool for teachers of introductory mathematics and physics classes.

5.5. Discussion

The results of our initial MathPad^2 usability study suggest that, based on our evaluation criteria, the MathPad^2 user interface is, in general, intuitive with subjects picking up the interface with relative ease. With only minimal training, most gestures are easy to remember and use. However, if we examine the first attempt task performance results (Table 2) in conjunction with the recall test from our post-questionnaire, we see that the equation solving gesture has the lowest first attempt accuracy and was the most difficult to remember. This indicates that this gesture is not as intuitive as the others. Additionally, if we look deeper into users' preferences for making associations, we see that they preferred explicit associations and of the four associations that were not completed on their first attempt, all four were implicit. Again, this result suggests that explicit associations are more intuitive than implicit ones. However, given subjects chose to make implicit associations in all cases for the 2D Motion task and explicit associations in all cases for the Spring task, implicit associations seem to be more appropriate for simple labels and for associations with a large number of mathematical expressions. First attempt performance for making nails was also a bit lower than expected, but we feel this might have been an implementation issue. In terms of perceived utility, subjects think the application is a powerful tool that beginning physics and mathematics students could use to help solve problems and better understand scientific concepts.

Most subjects performed the tasks with little trouble, while a few had some difficulty, stemming primarily from problems with mathematical expression recognition. However, these subjects also said they were willing to accept these recognition problems, given what MathPad^2 can offer them. This result is somewhat contrary to our expectations about the negative impact of our mathematical expression recognizer on MathPad^2 usability. Nevertheless, we need better mathematical expression recognition that will perform robustly across a larger user population.

[6]These results do not extend to the subject's long-term recall of MathPad^2 gestures. Further studies are needed to explore this issue but, we expect, given extended use of MathPad^2 and the small size of the gesture set, subjects would not have difficulty remembering the gestures between application sessions.

Although these results do not tell us how much more accurate the recognizer needs to be, its clear that a mean accuracy of 90.8% for making correct parsing decisions is too low. A better correction user interface could also go a long way to helping with users' frustrations when incorrect recognitions occur. In addition, more interactive feedback is needed for implicit associations, and the equation solving gesture should be redesigned.

Although the results of our initial evaluation are positive, we recognize it can be argued that there are two limitations with our study. First, we only used seven test subjects. We could have had more subjects, but we felt that seven was appropriate for an initial evaluation of MathPad^2 and its gestural interface, given one of our main goals was to determine whether larger studies were needed. Second, we did not compare MathPad^2 's user interface with any other interface metaphors. Although this could be considered a limitation, our goal in this evaluation was to determine how well users could use the MathPad^2 interface, not whether it was better than any other interface. For this work, we feel our experimental design was suited to answering our intended questions. However, as we perform future usability tests to gain a deeper understanding of the benefits of mathematical sketching, we will need more comparative experimental designs with larger subject numbers.

Given our evaluation results, we plan to make improvements to MathPad^2 by adding more functionality and improving the weaker points of the interface as well as improving the parsing component of our mathematical expression recognizer. Given the generally positive results of our evaluation, we are confident in pursuing further MathPad^2 experimentation. One such study will involve testing the MathPad^2 interface under more realistic conditions (as opposed to laboratory conditions). For example, we would like to test MathPad^2 's interface when subjects have a real task to perform for its own sake rather than for the sake of an experimental evaluation. Exploring the accuracy of our mathematical expression recognizer when subjects are writing rapidly is another worthwhile pursuit. We also plan to explore the pedagogical benefits of MathPad^2 in a summative evaluation where students will use MathPad^2 as part of a mathematics or introductory physics course.

6. Conclusion

We have presented an initial evaluation of MathPad^2, a prototype application for making dynamic illustrations using the mathematical sketching paradigm, to test its intuitiveness and perceived utility. Our evaluation suggests that MathPad^2 's user interface is generally intuitive, although some parts of the interface need to be reevaluated. Additionally, the MathPad^2 application is perceived to be a powerful tool for exploring mathematics and physics concepts. Although some of our test subjects had some difficulty with getting the system to recognize their mathematical expressions, they still gave positive feedback about MathPad^2 and would use it regardless of these issues because of its functionality. These results also support future MathPad^2 development and the need for longer term evaluations.

Acknowledgments

Thanks to the anonymous reviewers for their valuable suggestions for improving this paper. Special thanks goes to Robert Zeleznik and Andries van Dam for valuable discussions. This work is sponsored in part by a gift from Microsoft.

Appendix A. Mathematical expressions used in experiment 1

The following equations were used in the mathematical expression recognition accuracy study:

$$(ab)^x = a^x b^x, \tag{A.1}$$

$$\frac{a^x}{a^y} = a^{x-y}, \tag{A.2}$$

$$y^3 + 3py + q = 0, \tag{A.3}$$

$$y = x + \frac{b}{4a}, \tag{A.4}$$

$$a^2 + b^2 = (a + bc)(a - bc), \tag{A.5}$$

$$a^2 - b^2 = (a - b)(a + b), \tag{A.6}$$

$$\frac{2x}{x^2 - 1} = \frac{1}{x - 1} + \frac{1}{x + 1}, \tag{A.7}$$

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}, \tag{A.8}$$

$$ax^4 + bx^3 + cx^2 + dx + e = 0, \tag{A.9}$$

$$a^4 + b^4 = (a^2 + \sqrt{2}ab + b^2)(a^2 - \sqrt{2}ab + b^2), \tag{A.10}$$

$$(a + b + c)^2 = a^2 + b^2 + 2ab + 2ac + 2bc, \tag{A.11}$$

$$r_x = \frac{16a^2bc + 256a^3e - 3b^4 - 64a^2bd}{256a^5}, \tag{A.12}$$

$$y^2 + \sqrt{u - p}\left(y - \frac{q}{2gx}\right) + \frac{2u}{w} = 0, \tag{A.13}$$

$$\sin(t) = -4\sin^3(t) + 3\sin(t), \tag{A.14}$$

$$\tan^2(x) = \frac{1-\cos(2x)}{1+\cos(2x)}, \tag{A.15}$$

$$\cos\left(\frac{a}{2}\right) = \sqrt{\frac{s(s-a)}{bc}}, \tag{A.16}$$

$$a_1 + a_2 = b_3 + c_4 + d^2, \tag{A.17}$$

$$\tan(3a) = \frac{-\tan^3(a) + 3\ \tan(a)}{-3\tan^2(a)+1}, \tag{A.18}$$

$$\sin^4(h) = \frac{1}{8}(3 - 4\cos(2h) + \cos(4h)), \tag{A.19}$$

$$\log(e^x) = x, \tag{A.20}$$

$$\tan\left(\frac{x-y}{2}\right) = \frac{x-y}{x+y}\tan\left(\frac{b+c}{2}\right), \tag{A.21}$$

$$p_x(t) = \frac{\sin(t-a)\sin(t-b)\sin(t-c)}{\sin(t)}, \tag{A.22}$$

$$p_y(t) = \frac{1}{(t-m)^2} + \frac{1}{(t+n)^2}, \tag{A.23}$$

$$\frac{x}{x_0} + \frac{y}{y_0} = 1, \tag{A.24}$$

$$(g^2+d^2)^2 = u^2(g^2-d^2), \tag{A.25}$$

$$\frac{2ab}{a+b} = \sqrt{ab\left(1 - \frac{c^2}{(a+b)^2}\right)}, \tag{A.26}$$

$$r^2 = (x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2, \tag{A.27}$$

$$\int e^{-x}\,dx = e^{-x}, \tag{A.28}$$

$$\int_0^t x^2\cos(x)\,dx, \tag{A.29}$$

$$\int_2^4 y^2 - x^2\,dx, \tag{A.30}$$

$$\iint xy^2 + x^2y\,dx\,dy, \tag{A.31}$$

$$g = 9.8, \tag{A.32}$$

$$\int \frac{e^{ax}}{b + ce^a x}\,dx = \frac{1}{ac}\log(b + ce^{ax}), \tag{A.33}$$

$$\int (a+bx)^n\,dx = \frac{(a+bx)^{n+1}}{(n+1)b}, \tag{A.34}$$

$$\sum_{i=0}^{n-1}(i-1)^2, \tag{A.35}$$

$$y = \begin{cases} t: x<6 \quad \text{and} \quad x>0, \\ t^2: x>8, \\ t^3: \text{else}. \end{cases} \tag{A.36}$$

References

[1] LaViola J, Zeleznik R. MathPad^2: a system for the creation and exploration of mathematical sketches. ACM Trans Graphics (Proceed SIGGRAPH 2004) 2004;23(3):432–40.

[2] LaViola, J. Mathematical sketching: a new approach to creating and exploring dynamic illustrations, PhD dissertation, Brown University, Department of Computer Science; May 2005.

[3] Chin JP, Diehl VA, Norman KL. Development of an instrument measuring user satisfaction of the human–computer interface. In: Proceedings of the ACM conference on human factors and computing systems (CHI'88). New York: ACM Press; 1988. p. 213–8.

[4] Davis FD. Perceived usefulness, perceived ease of use and user acceptance of information technology. MIS Quarterly 1989;13(3): 319–40.

[5] Borning A. ThingLab: a constraint-oriented simulation laboratory. PhD dissertation, Stanford University; 1979.

[6] Shneiderman B. Designing the user interface: strategies for effective human–computer interaction. 3rd ed. Reading, MA: Addison-Wesley; 1998.

[7] Laleuf JR, Spalter AM. A component repository for learning objects: a progress report. In: Proceedings of the 1st ACM/IEEE-CS joint conference on digital libraries. New York: ACM Press; 2001. p. 33–40.

[8] Spalter AM, Simpson RM. Integrating interactive computer-based learning experiences into established curricula: a case study. In: Proceedings of the 5th annual SIGCSE/SIGCUE ITiCSE conference on innovation and technology in computer science education. New York: ACM Press; 2000. p. 116–9.

[9] Christian W, Titus A. Developing web-based curricula using java physlets. Comput Phys 1998;12(3):227–32.

[10] Warner S, Catterall S, Lipson E. Java simulations for physics education. Concurr Practice Exper 1997;9(6):447–84.

[11] Baker J, Cruz I, Liotta G, Tamassia R. Algorithm animation over the world wide web. In: Proceedings of the international workshop on advanced visual interfaces (AVI '96), 1996. p. 203–12.

[12] Feiner S, Salesin D, Banchoff T. Dial: a diagrammatic animation language. IEEE Computer Graphics and Applications 1982;2(7): 43–54.

[13] Brown MH, Sedgewick R. A system for algorithm animation. In: Proceedings of the 11th annual conference on computer graphics and interactive techniques (SIGGRAPH'84). New York: ACM Press; 1984. p. 177–86.

[14] Stasko JT. Animating algorithms with XTANGO. SIGACT News 1992;23(2):67–71.

[15] Stasko JT. Using student-built algorithm animatios as learning aids. Technical Report GIT-GVU-96-19, Graphics, Visualization, and Usability Center, Georgia Institute of Technology, Atlanta, GA; August 1996.

[16] Guzdial MJ. Squeak: object-oriented design with multimedia applications. Englewood Cliffs, NJ: Prentice-Hall; 2000.

[17] Carlson P, Burnett M, Cadiz J. A seamless integration of algorithm animation into a visual programming language. In: Proceedings of the workshop on advanced visual interfaces (AVI'96), 1996. p. 194–202.

[18] LaFollette P, Korsh J, Sangwan R. A visual interface for effortless animation of C/C++ programs. Journal of Visual Languages and Computing 2000;11(1):27–48.
[19] Stasko JT. Using direct manipulation to build algorithm animations by demonstration. In: Proceedings of the ACM conference on human factors and computing systems (CHI'91). New York: ACM Press; 1991. p. 307–14.
[20] Damm CH, Hansen KM, Thomsen M. Tool support for cooperative object-oriented design: gesture-based modeling on an electronic whiteboard. In: Proceedings of the 2000 SIGCHI conference on human factors in computing systems. New York: ACM Press; 2000. p. 518–25.
[21] Gross MD, Do EY-L. Ambiguous intentions: a paper-like interface for creative design. In: Proceedings of the 9th annual ACM symposium on user interface software and technology. New York: ACM Press; 1996. p. 183–92.
[22] Gross MD. Sketch-a-sketch: a dynamic diagrammer. In: Proceedings of the IEEE symposium on visual languages. New York: IEEE Press; 1994. p. 232–8.
[23] Zeleznik RC, Herndon KP, Hughes JF. SKETCH: an interface for sketching 3D scenes. In: Proceedings of the 23rd annual conference on computer graphics and interactive techniques. New York: ACM Press; 1996.
[24] Igarashi T, Matsuoka S, Takeo H. Teddy: a sketching interface for 3D freeform design. In: Proceedings of the 26th annual conference on computer graphics and interactive techniques. New York/Reading, MA: ACM Press/Addison-Wesley; 1999. p. 409–16.
[25] Forsberg A, Dieterich M, Zeleznik R. The music notepad. In: Proceedings of the 11th annual ACM symposium on user interface software and technology. New York: ACM Press; 1998. p. 203–10.
[26] Landay JA, Myers BA. Interactive sketching for the early stages of user interface design. In: Proceedings of the 1995 SIGCHI conference on human factors in computing systems. New York: ACM Press; 1995. p. 43–50.
[27] Moran TP, Chui P, Van Melle W. Pen-based interaction techniques for organizing material on an electronic whiteboard. In: Proceedings of the 10th annual ACM symposium on user interface software and technology. New York: ACM Press; 1997. p. 45–54.
[28] Mynatt ED, Igarashi T, Edwards WK, LaMarca A. Flatland: new dimensions in office whiteboards. In: Proceedings of the 1999 SIGCHI conference on human factors in computing systems. New York: ACM Press; 1999. p. 346–53.
[29] Hinckley K, Baudish P, Ramos G, Guimbretière F. Design and analysis of delimiters for selection–action pen gesture phrases in scriboli. In: Proceedings of the ACM conference on human factors in computing systems (CHI 2005). New York: ACM Press; 2005.
[30] Lin J, Newman MW, Hong JI, Landay JA. DENIM: finding a tighter fit between tools and practice for web site design. CHI Lett 2000;2(1):510–7.
[31] Windows Journal, Microsoft Corporation, 1981–2006.
[32] Saund E, Fleet D, Larner D, Mahoney J. Perceptually-supported image editing of text and graphics. In: Proceedings of the ACM symposium on user interface software and technology (UIST 2003). New York: ACM Press; 2003. p. 183–92.
[33] Saund E, Lank E. Stylus input and editing without prior selection of mode. In: Proceedings of the ACM symposium on user interface software and technology (UIST 2003). New York: ACM Press; 2003. p. 213–7.
[34] Igarashi T, Hughes JF. A suggestive interface for 3D drawing. In: Proceedings of the ACM symposium on user interface software and technology (UIST 2001). New York: ACM Press; 2001. p. 173–81.
[35] Alvarado CJ. A natural sketching environment: bringing the computer into early stages of mechanical design. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology; May 2000.
[36] Kara LB, Gennari L, Stahovich TF. A sketch-based interface for the design and analysis of simple vibratory mechanical systems. In: Proceedings of ASME international design engineering technical conferences, 2004.
[37] Pickering J, Bhuphaibool D, LaViola J, Pollard N. The Coach's playbook. Technical Report CS-99-08, Brown University, Department of Computer Science, Providence, RI; May 1999.
[38] Davis RC, Landay JA. Informal animation sketching: requirements and design. In: Proceedings of AAAI 2004 fall symposium on making pen-based interaction intelligent and natural, Washington, DC, October 21–24, 2004.
[39] Davis J, Agrawala M, Chuang E, Popovic Z, Salesin D. A sketching interface for articulated figure animation. In: Proceedings of the eurographic/SIGGRAPH symposium on computer animation; 2003. p. 320–8.
[40] Moscovich T, Hughes JF. Animation sketching: an approach to accessible animation. Technical Report CS-04-03, Computer Science Department, Brown University, Providence, RI; February 2004.
[41] Tall D. Graphical packages for mathematics teaching and learning. In: Johnson DC, Lovis F, editors. Informatics and the teaching of mathematics. Amsterdam: North-Holland; 1987. p. 39–47.
[42] Chan K-F, Yeung D-Y. Mathematical expression recognition: a survey. International Journal of Document Analysis and Recognition 2000;3(1):3–15.
[43] Matsakis NE. Recognition of handwritten mathematical expressions. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology; 1999.
[44] Miller EG, Viola PA. Ambiguity and constraint in mathematical expression recognition. In: Proceedings of the 15th national conference on artificial intelligence, 1998. p. 784–91.
[45] Zanibbi R, Blostein D, Cordy J. Recognizing mathematical expressions using tree transformation. IEEE Transactions on Pattern Analysis and Machine Intelligence 2002;24(11):1–13.
[46] Anderson RH. Syntax-directed recognition of hand-printed two-dimensional mathematics. PhD dissertation, Department of Applied Mathematics, Harvard University; 1968.
[47] Martin WJ. A fast parsing scheme for hand-printed mathematical expressions. artificial intelligence. Memo No. 145, Massachusetts Institute of Technology; 1967.
[48] Chan K-F, Yeung D-Y. PenCalc: a novel application of on-line mathematical expression recognition technology. In: Proceedings of the sixth international conference on document analysis and recognition, September 2001. p. 774–8.
[49] Chan, K-F, Yeung D-Y. A efficient syntactic approach to structural analysis of on-line handwritten mathematical expressions. Technical Report HKUST-CS98-10, Department of Computer Science, The Hong Kong University of Science and Technology, Hong Kong, China; August 1998.
[50] Chan K-F, Yeung D-Y. Error detection, error correction, and performance evaluation in on-line mathematical expression recognition. Pattern Recognition 2001;34(8):1671–84.
[51] Li X, Yeung D-Y. On-line handwritten alphanumeric character recognition using dominant points in strokes. Pattern Recognition 1997;30(1):31–44.
[52] Chan K-F, Yeung D-Y. Elastic structural matching for on-line handwritten alphanumeric character recognition. In: Proceedings of the 14th international conference on pattern recognition 1998; 1508–11.
[53] Connell SD, Jain AK. Template-based on-line character recognition. Pattern Recognition 2000;34(1):1–14.
[54] Scattolin P, Krzyzak A. Weighted elastic matching method for recognition of handwritten numerals. In: Proceedings of vision interface'94, 1994. p. 178–85.
[55] Garain U, Chaudhuri BB. Recognition of online handwritten mathematical expressions. IEEE Trans Syst Man Cybern Part B Cybern 2004;34(6):2366–76.
[56] Fukuda R, Sou I, Tamari F, Ming X, Suzuki M. A technique of mathematical expression structure analysis for the handwriting input system. In: Proceedings of the 5th international conference on document analysis and recognition. New York: IEEE Press; 1999. p. 131–4.