Adapting Handwriting Recognition for Applications in Algebra Learning

Lisa Anthony, Jie Yang, Kenneth R. Koedinger
Human-Computer Interaction Institute
Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA 15213
+1 (412) 268-8856
{lanthony, yang+, koedinger}@cs.cmu.edu

ABSTRACT
In this paper we report the progress of our ongoing project exploring the adaptation of handwriting recognition-based interfaces for applications in intelligent tutoring systems for students learning algebra equation-solving. The research is motivated by the hypothesis that handwriting as an input modality may be able to provide significant advantages over typing in the mathematics learning domain. We review the literature of existing handwriting systems for mathematic applications and evaluations of handwriting recognition accuracy. We describe our approach and report results to date in exploring the use of handwriting recognition in interfaces for math learning, from both a technical and a pedagogical perspective. We have found that handwriting input can provide benefits to students learning math, and continue to pursue further technical and pedagogical enhancements.

Categories and Subject Descriptors
H.5.m [**Information Interfaces and Presentation, e.g., HCI**]: Miscellaneous.

General Terms
Algorithms, Human Factors.

Keywords
Handwriting input, handwriting recognition, mathematics learning, recognition accuracy evaluation.

1. INTRODUCTION
This paper reports the progress of our ongoing project in exploring the adaptation of handwriting recognition technologies to improve interfaces of intelligent tutoring systems for students learning algebra equation-solving. Many schools throughout the United States now incorporate computers as a regular part of classroom instruction [37] and use intelligent tutoring systems as supplements to traditional classroom instruction. An intelligent tutoring system is educational software that can monitor the student as he/she works at his/her own pace, and tailor feedback, step-by-step hints, and even the curriculum to address the student's particular needs. This self-pacing provides an opportunity for teachers to give more individual attention to students that need it most. Intelligent tutoring systems provide a unique opportunity to enhance the learning experience, by providing an online environment in which students can work at their own pace and at a time convenient to them.

The best *human* tutors can achieve a two-standard deviation improvement versus standard classroom instruction alone, effectively turning *C* students into *A* students; Cognitive Tutors, one type of intelligent tutoring system, have been shown to raise student achievement *one* standard deviation over traditional classroom instruction [11]. Our goal is to improve mathematics learning in Cognitive Tutors even further, narrowing that two-sigma gap via use of multimodal and multimedia interface technologies.

Although intelligent tutors for math have been improved with respect to pedagogical style and overall effectiveness over the last 15 years (*e.g.*, [10]), their interfaces have remained more or less the same: keyboard-and-mouse windows-icons-menus-pointing (WIMP) interfaces. *Output* modality contrasts have been studied with respect to learning, including the use of animations, diagrams and talking heads (*e.g.*, [15], [26]), but the literature has been silent on the effects of *input* modality on learning[1]. We believe that the input modality is extraneous to the problem-solving process and learning. It may interfere with problem-solving, but the input modality is not itself relevant to the mathematic concept being practiced. WIMP interfaces may impose extraneous cognitive load on the student, because representing and manipulating mathematics equations can be cumbersome in a typing interface. An interface that can more directly support the standard notations for the mathematics that the student is learning would reduce extraneous cognitive load and lead to increased learning (*c.f.*, [34]).

This paper reports progress of our ongoing project exploring the adaptation of handwriting recognition-based interfaces for applications in intelligent tutoring systems for students learning algebra equation-solving progress, as well as evidence in favor of handwriting-based interfaces with respect to learning in the

EMME'07, September 28, 2007, Augsburg, Bavaria, Germany.

[1] Note that input modality here refers to the modality of generation by the student, and the output modality is the modality presented to the student by the system.

domain of algebra equation solving. Our results to date show that handwriting input has benefits both for general usability and for learning. While this work is in the domain of high school algebra learning, it is likely to generalize to other types of math and to other levels of students. The rest of the paper is organized as follows: Section 2 introduces our motivation and approach. Section 3 describes the background and related work to this research. Section 4 discusses evaluation of the handwriting component used in our system. Section 5 presents the handwriting–based intelligent tutor and user study results. Section 6 summarizes the paper and proposes the future work.

2. MOTIVATION AND APPROACH

2.1 Math Learning Challenges

Mathematics is a key building block for high-performing careers in science, engineering, and information technologies. In a recent survey, it was found that American high school students have a poorer mastery of basic math concepts than their counterparts in most other leading industrialized nations [30]. There are many theories on why U.S. students lag behind their peers abroad in math. One of the major reasons is a shortage of teachers, which may be addressable by supplementing classroom instruction with one-on-one tutoring. Bloom found that the best human tutors can raise the grade of a '*C*' student to an '*A*,' known as the "two-sigma effect" [7]. However, it is clearly not feasible from either a financial or human resources perspective to provide every student in America with an expert human tutor. A potential solution to the lack of human teachers and tutors is to use intelligent software math tutors, as many classrooms now are doing. By collecting information on a particular student's performance as s/he problem-solves, the software can make inferences about his/her strengths and weaknesses, tailor the curriculum to address his/her needs, and let them work at their own pace and practice specific concepts. Intelligent tutors for math have been shown to improve student performance one standard-deviation above traditional classroom instruction [11]. We aim to further improve this effect via the exploration of alternative input modalities, specifically handwriting input, and its effect on learning in the domain of high school algebra equation solving.

2.2 Enhancing Intelligent Math Tutors Using Handwriting

Our previous work has shown that handwriting provides *usability* benefits for math input in that the speed of entry increases, user error decreases, and user satisfaction increases. There is evidence that the use of handwriting interfaces could also have *pedagogical* advantages in the domain of math learning environments. In a prior study we conducted, students solving the same problems by handwriting as others who were typing experienced similar learning gains in half the time [5]. We hypothesize that one or both of two specific factors may be responsible for this advantage of using handwriting. The first is that the affordance of handwriting for more direct manipulation may result in a reduction in extraneous cognitive load. Further, students practice in the classroom and on homework and take tests on paper using handwriting; this modality becomes more *fluent* for students when solving algebra equations. An interface that takes this into account will therefore impose less extraneous cognitive load.

The second factor is the improved support for the two-dimensional spatial information which is inherently meaningful in mathematics (*i.e.*, vertical fraction notation). For example, the placement of the *x* in the following two expressions significantly changes the meaning of the expression: "$2x$" *vs.* "2^x". Handwriting is a much more flexible and robust modality for representing and manipulating such spatial relationships, which become more prevalent as students advance in math training to calculus and beyond.

2.3 Getting Handwriting Input to Work

One concern with the use of handwriting in intelligent tutoring systems, however, is that the recognition technology is not perfect. To the extent that the system cannot be confident of correctly recognizing what the student is writing, it cannot provide detailed, step-targeted feedback (as Cognitive Tutors currently do). Therefore, a trade-off is clear between difficulties with improving recognition accuracy and the need to support step-targeted feedback. One strategy to address this trade-off is modifying the instructional paradigm to include time for students to study worked examples, which may provide a sort of *feed-forward* to guide learners. A second strategy is to investigate technical approaches to improving handwriting recognition accuracy. We are exploring two methods: performing in-advance training of the recognition engine on a data corpus of student writing; and adapting machine learning techniques similar to *co-training* [8]. Co-training allows recognition to be enhanced by using alternate sources of information and learning from unlabeled examples, which are easier to obtain than labeled ones. In this paper we report our progress in using the first strategy.

We have developed a prototype of an intelligent tutoring system (ITS) that allows students to solve algebraic equations via handwriting input. This paper describes the system architecture and some ways we have already explored to improve the handwriting recognition accuracy such that it is usable by students. We reduce, if not entirely eliminate, the need for user repair of recognition errors by collecting a large corpus of student handwriting samples in the target domain and training the engine in advance in a variety of ways.

Training the engine in advance requires a large corpus of domain-specific samples of handwriting that resemble the input the system will see in actual use of the application. Collecting data of this type involves gathering data under certain conditions and labeling the data to provide "ground truth" labels. This process is costly in terms of user hours. Therefore, as part of our contribution, we explored alternative ways of training the recognition engine on our corpus in order to discover how much data is sufficient for this type of application. To this end, we have reported previously results of data-driven training experiments with a number of off-the-shelf handwriting recognition engines [6], in which we found that varying the number of *samples per symbol* in the alphabet *per user* leads to the best possible *a priori* recognition accuracy. Only a few samples per symbol per user are needed in this method, which leads to the least overall required data collection resulting in the highest accuracy. Future application developers can use these guidelines to help determine how much data to collect for her own application.

In our application domain, we are also exploring the possibility that, due to the special nature of a learning task and the needs of a

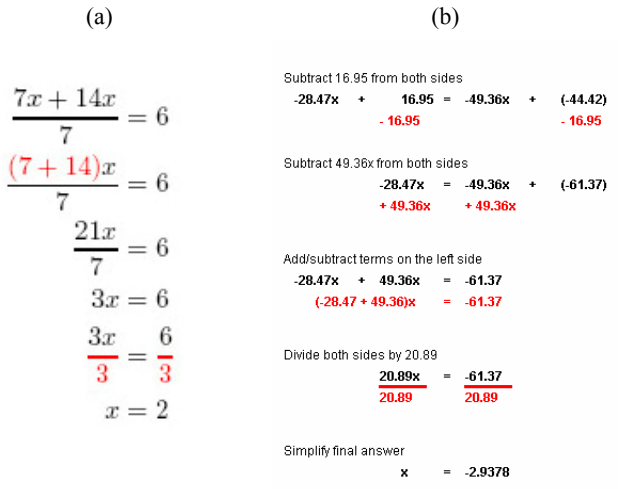

Figure 1. Sample worked examples as they would be displayed to students: (a) no annotations, (b) annotations.

learner, students may not require the system to immediately output a recognition hypothesis. We are investigating how the design of the instructional paradigm used in the ITS can mitigate the need for immediate recognition feedback. Our current paradigm is based on worked examples instruction, in which students copy and study example problems whose solutions are already worked out for them before solving their own problems. This provides a sort of *feed-forward* rather than *feedback* to students. Figure 1 shows two sample worked examples as they would be shown to students, one with annotations and one without. The use of worked example-based problem-solving is a common approach in educational literature, beginning with [34].

2.4 Testing Human Learning

A fundamental goal of this project is to determine the ways in which the use of handwriting input will create a more natural and unconstrained modality for students to input their problem-solving process, thereby leading to increased learning gains. Therefore, user studies measuring students engaged in learning are critical to our approach. We have conducted a number of studies in this line of research to date. The early studies designed to motivate further research were conducted in the laboratory. As part of our affiliation with the Pittsburgh Science of Learning Center (PSLC), *in vivo* classroom studies are emphasized. These user studies take place in a real-world classroom setting in which the experimental system is compared to an authentic control condition in which students use the tools they normally would. The classrooms in which these studies take place use the Cognitive Tutor Algebra curriculum all year long, and only some classes or students switch to the experimental system during the study. This is an emerging trend in educational research, which has often in the past been confined to laboratory studies, which are not accurate with respect to classroom culture, interactions, and motivational elements, and have sometimes has their real-world validity called into question. Being part of the PSLC provides us with this unique opportunity to conduct more real-world applied studies and see the experimental software and technology in use in the field.

The types of measures we can collect during these classroom studies include pre-test to post-test gains, errors during training (use of the ITS), skill mastery, time on task, requests for help, qualitative observations of student behavior, and subjective questionnaire responses.

3. BACKGROUND

3.1 Handwriting Recognition Research

Handwriting recognition has been an active area of research since the 1960s, even for mathematics (*e.g.*, [2]). Techniques for the recognition of handwritten mathematics range from the recognition of a page of notes after it has already been written (offline, OCR), to the recognition of a user's handwriting even while he/she is in the process of writing (online). Many different algorithms have been explored, from neural networks, to Support Vector Machines (SVMs), to Hidden Markov Models (HMMs), and more. Because the focus of this research is not to create new algorithms or techniques for handwriting recognition, interested readers are referred to [9] for an excellent survey of the field.

Our prior work has found that for the domain of math, handwriting is better than typing and menus in terms of speed and user satisfaction [3]. This result helps motivate continued effort to develop handwriting-based intelligent math tutors. Several research and commercial systems exist that allow users to input and/or edit mathematical expressions via handwriting input. MathPad^2 [21] is among the most robust and complex. In MathPad^2, users can write out mathematics equations and the system animates the physical relationships given by these equations, for example, an oscillating sine curve. Other systems such as xThink's MathJournal [38] allow the sketching and writing of mathematics, but rely on in-context menus to allow users to perform manipulations. Even traditional keyboard-based math software such as Microsoft's Equation Editor and Maple 10 are now offering handwriting-based input, although limited in the amount of the equation that can be written or in what can be done as far as manipulation of the equation once it is input. InftyEditor [19], Natural Log [25], the Freehand Formula Entry System [33], and JMathNotes (*e.g.*, [35]) are simple equation entry/editing programs without the added benefit of sketching or graphing.

The added-value of our prototype over these simple input systems is that we are focusing on *learning* mathematics. Most other systems focus only on letting users input mathematics; they do not provide a structured approach to learning to perform mathematical operations, they assume their users already know how. There is at least one system that does consider education: Jumping Minds' Practice series [18]. The Jumping Minds series is a simple interface in the style of first-generation Computer-Aided Instruction (CAI), in which problems are provided to the students. However, there is no tailored feedback or model of student learning, both of which have been shown to contribute significantly to the advantage of Cognitive Tutors and other third-generation CAI (*c.f.*, [1]).

A technical limitation of recognition technologies such as handwriting is that recognition accuracies are not perfect. It has been shown that humans will tolerate accuracy rates in handwriting recognition for a variety of tasks only as low as 97% [20] (note that human recognition rates are around 96.8% [32]). This is difficult to achieve in current systems. While higher

recognition accuracies may manifest in the domain of beginning algebra equation solving due to the limited symbol set and grammar used, the fact that middle and high school students are the target audience may be another challenge. Children's handwriting may be less legible and/or consistent than the adults on which the systems have been trained in the past. Additionally, children may be less likely to tolerate system recognition errors and cooperate in repairing them. We attempt to address this via in-advance training on labeled datasets of handwritten character samples from the target population of middle and high school math learners.

3.2 Evaluation of Handwriting Recognition

Few rigorous evaluations have been done from a user perspective on handwriting recognizers for any domain, a weakness identified early on in the literature [14] but never pursued. Typically developers report accuracy numbers without much context or detail. Many of the evaluations that do exist are now out-of-date ([23], [32]), as recognition technology has continued to advance over the past 10-15 years. Handwriting recognition systems for math are especially lacking in formal evaluations. MathPad^2 is one of the few recent systems to perform a complete user study designed to gauge both user performance and satisfaction and interface ease-of-use and learnability along with recognition engine performance [21]. The study reported in that paper involved only 7 users and did not report statistical significance of findings because it had only one condition, although other, more rigorous studies are planned. Other recent studies have similarly small numbers of users or report system accuracy as a foot-note in the context of a larger discussion of a new algorithm or technique [33].

User-centered design requires that both halves of the equation be considered when developing an application to use handwriting input: both the accuracy of the system itself and how a user reacts to and interacts with the system. [13] explored the relationship between recognition accuracy and user's satisfaction and found that it was highly task-dependent: some tasks (such as a form-filling task) were rated as very suitable for pen-based input no matter what the recognition accuracy level was, whereas others (such as a diary task) were only rated highly when accuracy was also high. The type of recognition supported may have also impacted these results; the system only accepted isolated characters printed within boundary boxes. As newer, more natural methods of handwriting input become available, it is important to re-evaluate them from a user perspective.

Some handwriting recognition researchers have implied that some of the burden is on the user to "adapt" his/her handwriting style as he/she learns its idiosyncrasies (*c.f.*, [13]). In contrast, the focus of our work is on adapting the *recognizer* to individual writing styles over time. In fact, several researchers have pointed out that errors in handwriting input tend to be constant over time [23], implying that users will not actually adapt to specific recognizers.

The Lipi Toolkit is a project to provide tools to allow data processing and annotation, and adapting recognizer engines to use in applications [24]. It is in the early stages of development and therefore can currently only support a limited set of recognition algorithms and isolated characters (with bounding boxes, for instance). A formal set of evaluations on a Hidden Markov Model recognizer is reported in [22], but the domain is off-line handwriting recognition of cursive handwriting. Its methodology can be informative but must be extended in order to apply to online (real-time) recognition applications.

3.3 Intelligent Tutoring Systems

Intelligent tutoring environments for problem solving have proven to be highly effective learning tools ([1], [36]). Many of these environments present complex, multi-step problems and provide the individualized support that students need to complete them: step-by-step accuracy feedback and context-specific problem-solving advice. They are two or three times as effective as typical human tutors, but only half as effective as the best human tutors [11], which can improve student learning by two standard deviations [7]. This means there is still room for improvement, which I hope to accomplish by making the interfaces more suitable and effective for learning in certain domains.

Cognitive Tutors are a class of intelligent tutoring systems that are designed based on cognitive psychology theory and methods that pose authentic problems to students (*learn-by-doing*) [1]. In Cognitive Tutor Algebra, students represent the situation algebraically in the worksheet, graph the functions, and solve equations with a symbol manipulation tool. Each Cognitive Tutor is constructed around a cognitive model of the knowledge students are acquiring, and can provide step-by-step accuracy feedback and help. They have been created for a variety of learning domains, including algebra, geometry, foreign languages, chemistry, computer programming and more. Cognitive Tutors for mathematics are in use in about 2000 schools in the United States, and have been shown to raise student achievement one standard deviation over traditional classroom instruction [12]. I will use Cognitive Tutors as the intelligent tutor foundation of my system, and will add a handwriting interface to already-existing lessons that have been previously developed and field-tested extensively.

Although Cognitive Tutors, and other intelligent tutoring systems, have begun to explore other interface styles, most systems still currently rely on standard WIMP interfaces. This is due in part to the fact that the technology available to most students in the classroom is limited to keyboard-and-mouse—this situation is changing however, as students receive PDAs or TabletPCs in the classroom instead ([17], [37]). However, while advantages of pen-based input have been explored for the math domain in terms of usability measures such as speed and user satisfaction [3], very little work has been done analyzing the effect of modality on learning. One study has reported results comparing a variety of pen-based interfaces for solving geometry problems with students [28], but it assumes that handwriting is beneficial and does not provide a current practice (typing) control condition for comparison.

4. EVALUATION OF THE HANDWRITING COMPONENT

4.1 Recognition Engine Description

Our prototype uses an off-the-shelf recognizer called FFES/DRACULAE because it relieves us of the technical burden of developing a robust recognizer from scratch and blazes a trail for using handwriting engines in more real-world applications. We actually explored two other possible engines and chose FFES

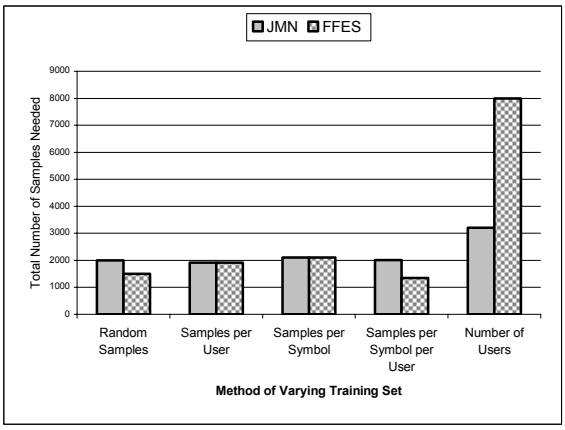

Figure 2. Amount of data needed for both engines tested using each tested training style to reach best accuracy.

because it had the highest accuracy rates in our data-driven experiments, discussed in section 4.2.2. FFES has reported character recognition accuracy rates of about 77%, for both expert and novice users who had not trained the system to their style of writing [33]. With training, FFES can yield accuracy rates as high as 95%.

The Freehand Formula Entry System [33] is based on the CIT character recognizer written by Jim Arvo. It uses a nearest-neighbor classification based on a 48-dimensional feature space and is implemented in C++ under the Gnu Public License. In total there are almost 45,000 lines of code in 191 source files. Its dependencies include a Unix-like environment (or cygwin on Windows) and ActiveTcl. As it is typical of many research systems, its documentation is minimal and there is no usable API.

FFES recognizes mathematical equations via two components: character recognition (CIT) [33], and mathematical expression parsing (DRACULAE) [39]. The main advantage of this handwriting recognizer is that its features are designed to be effective for mathematical symbols and numbers. Stroke grouping (character segmentation) is performed via an algorithm that finds the highest confidence grouping of a set of *m* recently drawn strokes, where *m* is the maximum number of strokes in any symbol in the recognizer's symbol set (*m*=4).

4.2 User-Independent Accuracy

In recognition technology, *user-independent* typically means that the system or engine has been trained on samples from a number of different users, in contrast to *user-dependent*, which means the training samples all come from the same user who will be using or testing the system. User-dependent accuracy rates tend to be higher than user-independent rates in most recognition technologies because differences in handwriting (or speech, etc.) vary more widely *across* users than *within* users. A particular user normally has a particular style of writing which does not vary much over time. For example, someone might write a "4" with one connected stroke vs. with two separate strokes, but is unlikely to vary their dominant style. *User-dependent* recognition represents the "ideal" case for the recognizer: data consists of a set of similarly constructed symbols rather than differing styles.

However, in certain domains, such as learning environments in classrooms, it is not feasible for the user (*i.e.*, student) to spend time training the system with no learning objectives. Training handwriting recognition engines to be *user-dependent* usually involves a large upfront time commitment during which the user inputs many (20+) examples of each character the recognizer is to understand (called *enrollment*). On the other hand, it is difficult to embed the handwriting training task into other, more learning-oriented tasks because the system cannot provide adequate feedback on the learning aspects without good *a priori* recognition accuracy. Therefore, it is important to attempt to improve recognition without upfront training for each particular student. *User-independent* training results in a more *walk-up-and-use* interaction style.

Baseline recognition accuracy also differed depending on the type of samples on which the engine is tested. Usually recognizers are tested on either *letters* or *words*. In our domain, the distinction is better characterized by *symbols* or *equations*. The problem of symbol segmentation, that is, telling which strokes belong to what character, is a challenge that real-time recognition applications must address. It is not a realistic estimate of accuracy rate to test the engine on single characters one at a time. It is important to consider accuracy on full equations (streams of characters) when judging how well a recognition engine will perform for real users. To do this, the engine is given a set of strokes that make up a full equation, and is iteratively asked to identify subsets of them which make up individual characters.

4.2.1 Corpus Used

We have collected a corpus of data from over 40 high school and middle school algebra learners copying out equations. The data have been hand-segmented and hand-labeled. They are grouped by equation as originally written by the users in order to allow real-world equation testing; they can also be separated into individual symbols.

The corpus contains 16,191 characters grouped into 1,738 equations. The symbol set includes 21 symbols: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, x, y, +, -, __, (,), =}. Each user wrote on average 404 samples (min=227, max=471) and 17 samples per symbol (min=0 ("__"), max=44 ("=")). There are on average 770 samples per symbol (min=569 ("9"), max=1581 ("=")).

We chose this population of users based on our target application domain: intelligent tutoring systems for algebra equation solving. It is likely that character form and legibility will be quite different from the adult corpora on which handwriting engines may have been trained by their developers. Depending on the domain, other data may be required. Our work aims to provide methodological recommendations to future application developers.

4.2.2 Results of Data-Driven Experiments

We ran these experiments on two engines: FFES (based on the nearest-neighbor algorithm) and JMathNotes (based on the multi-class SVM algorithm). In papers published on FFES, accuracy measures on simple experiments involving 5 users were reported. *Symbol*-level accuracies of 77% were achieved for users to whom the system had not been specifically trained (*user-independent*), and rates as high as 95% were achieved for users to whom the system *had* been trained (*user-dependent*) [33].

The theoretically optimal way to vary the training set of controlling two factors: the number of samples per *symbol* and the

number of samples per *user* or writing style. Our data-driven experiments confirmed this for FFES, and although JMN reached best accuracy at similar levels of total data for multiple training styles, following the strategy of controlling the number of samples per symbol per user/style will still converge to JMN's best accuracy quickly. Figure 2 shows the convergence points for each training style given in number of total samples needed before amount of accuracy gain per additional data sample dropped below a given threshold. Ensuring an equal representation of symbols is important to provide the recognizer with a balanced model. In beginning algebra equation solving, the most common character may be the equals sign, but otherwise most symbols are evenly represented. Also, ensuring an equal representation of each user's handwriting will prevent the system from having a bias toward a particular style of writing. A user-independent system has been exposed to many styles and will use these to classify new users' writing. In cases where different training styles result in similar convergence points, training in terms of samples per symbol per user minimizes demands on users' time, which can be costly and scarce. With FFES, each of 40 users had to write just *two* samples per symbol.

5. APPLICATION OF TUTORING SYSTEMS FOR MATH LEARNING

5.1 System Details

We are in the process of developing a prototype system to allow students to solve mathematical equations via handwriting input. As mentioned, to increase the possibility of success, we have built our system with a foundation of state-of-the-art intelligent tutoring system and handwriting recognition components. Cognitive Tutors have been an active area of research at Carnegie Mellon University since the 1980s [12]. They have been created for a variety of domains, including LISP programming, algebra, geometry, foreign language learning, and genetics. Cognitive Tutors for mathematics are in use in over 2000 schools in the United States. In the algebra tutoring lessons, students represent the situation algebraically in a spreadsheet-like worksheet and solve equations in a separate space with a symbol manipulation tool involving typing and menu-based operator selection. Each Cognitive Tutor is constructed around a cognitive model of the knowledge students are acquiring, and can provide step-by-step accuracy feedback and help as students solve problems. Carnegie Learning's Cognitive Tutors are implemented in Java. In our prototype, we replaced part of the Java interface with a handwriting input space that has the handwriting recognizer behind it. Figure 3 shows a screenshot of the prototype system in which the student is solving the problem "*4069.64 + 434.17y = 262.47y + (-3804.02)*" by referring to the worked example on the lefthand side of the screen. The student enters his/her solution process in handwriting and types in the final answer in the text field at the bottom of the screen.

The Cognitive Tutor Algebra I curriculum has over 25 units, at least 5 of which are pure equation-solving units. We can deploy our prototype for any of these units. Some of the problem types we have used in our studies include *ax+b=c*, *x/a + b=c*, *a/x=c*, *ax+bx=c*, *ax+bx+c=d*, etc.

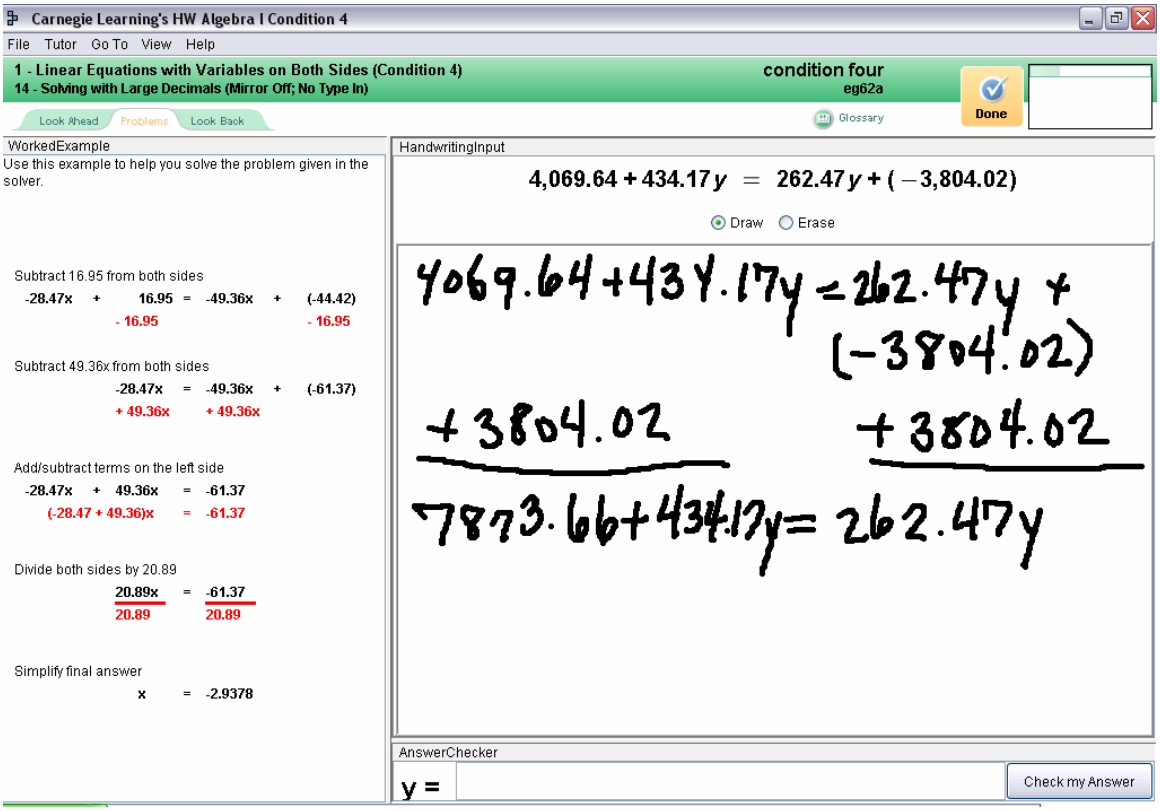

Figure 3. A screenshot of our current prototype system.

The goal of this research is to use multimodal and multimedia technologies to improve mathematics learning for high school students. Besides handwriting, we consider using speech as an additional modality for error repair. Systems actually perform better when the modality of repair is different than the modality of entry, in part because users tend to over-enunciate (in speech) or trace heavily (in writing) and these patterns do not match the system's model (*c.f.*, [27]). Speech seems logical because it can be highly accurate when the symbol vocabulary is small (as in simple algebraic expressions), and because it does not require a return to the keyboard. We also consider using multimedia output. Our system may present to the student animated diagrams helping to explain the problem-solving process when the student needs a hint. Besides visual output, synthesized voice will be used when needed. Mayer's work exploring the principles of multimedia learning [26] will serve as design guidelines for when to include multimedia output and what this output should contain.

5.2 User Studies

As mentioned, we have the opportunity to conduct real-world classroom studies as part of our affiliation with the PSLC. Our first few studies took place in the laboratory, to help us lay the foundation for motivating further research into this line of inquiry. Once we had a theoretical basis and good preliminary evidence showing that our interventions would most likely help students, we moved to the classroom.

The hardware we have used for our classroom studies is the EZ-Canvas by Navisis. It is a device that can attach to any monitor and allow users to write with the provided stylus directly on the interfaces they are using. It uses ultrasonic sensing technology to triangulate the stylus coordinates between two sensors at the top corners of the monitor. The stylus then controls the mouse pointer, so any software expecting mouse input, drawing, or writing, can be used with the EZ-Canvas device. Despite some technical limitations of the devices (*e.g.*, they cannot be placed too close to each other due to interference), the price of about US$125 makes this a low-cost solution to getting handwriting input into classrooms, since it can attach to the computers already in the schools, rather than requiring the purchase of a lab full of expensive TabletPCs.

We have run several studies in this line of research; two were preliminary laboratory studies establishing a grounds for further exploration of these issues within the classroom itself, and one was an *in vivo* classroom study. Table 1 shows an overview of the studies we have run to date and their primary conclusions.

The first study, the Math Input Study, was a motivating study designed to explore what advantages, if any, handwriting-based input has for mathematics entry on the computer. Learning in a classroom setting is characterized by constraints on time, varying student motivation and engagement, and other factors not directly related to test scores. Our study showed that students who entered math equations via handwriting input were three times faster, were less prone to errors in input, and enjoyed their experience more. In the classroom, this can translate to increased depth or breadth of coverage by virtue of the extra time afforded, and to improve student motivation by virtue of their increased engagement. See [3] for more details.

The second study, the Preliminary Learning Study, took the first study's results one step further and applied handwriting-based input to a learning situation. In this study we compared students solving problems in a simple type-in interface with a handwriting input space; instruction was in the form of worked examples interspersed with problem solving, and feedback was answer-only ("Correct"/"Incorrect"). This was a laboratory study to determine whether or not novice math students engaged in a learning task would experience the same positive effects of using a handwriting interface over a typing interface. The typing interface looked just like the handwriting interface: a worked example and a free-form input space into which students could type their solution (see Figure 3) on a standard keyboard. No special-purpose math menus were provided in either modality.

Results from this study showed that students in the handwriting condition finished the learning session in half the time of their typing counterparts ($F_{2,35}$=11.05, $p<0.0005$). Yet there was no significant difference in their pre- to post-test score gains between conditions ($F_{2,35}$=0.293, *n.s.*). Students appear to have learned just as much in about half the time! In a classroom situation, this would allow teachers to give students more practice or move on to more advanced material in the curriculum sooner. There was also a significant interaction between modality and the appearance of fractions in a problem ($F_{2,36}$=5.25, $p<0.01$), shown in Figure 4, which implies that the advantages we've seen for handwriting only improve as the math gets more complex. In their own words, students commented that handwriting "made it easier" and "takes a shorter time"—statements that lend support to the hypothesis

Table 1. Overview of the studies we have conducted to date.

	Study 1	**Study 2**	**Study 3**
Name	Math Input Study	Preliminary Learning Study	Worked Examples Paradigm Study
Research Goal	Establish usability characteristics of math input modalities	Compare impact of math input modalities on human learning	Compare instruction with worked examples vs pure problem solving
Type	Lab	Lab	Classroom
Modalities	Typing, Handwriting, Handwriting+ speaking	Typing, Handwriting , Handwriting +speaking	Typing, Handwriting
Measures	Speed, User preferences, Errors	Speed, Learning gains	Speed, Errors during Training, Learning gains
Conclusion	Handwriting faster and better liked than other modalities.	Handwriting students learned just as much in half the time.	*Results in progress.*

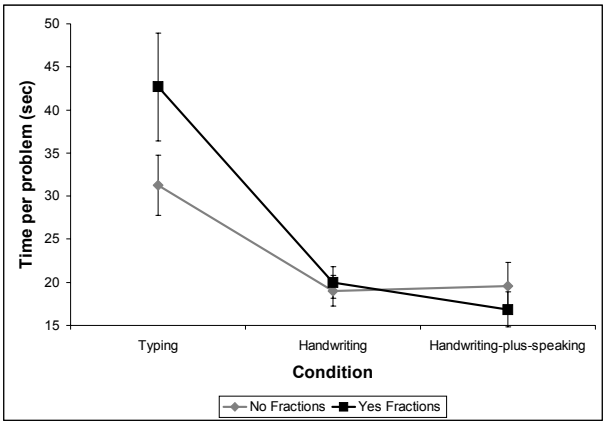

Figure 4. Average time per problem by condition crossed with appearance of fractions in the *learning* phase for both copying examples and solving problems. Error bars show 95% confidence interval.

that handwriting involves less extraneous cognitive load. While this is only a preliminary result, we plan to explore this further in later studies by including a structured self-report of student-perceived cognitive load, modeled after [29], in which they asked students to rate their perceived amount of mental effort during various instructional paradigms.

All students were exposed to all three conditions during the copying phase. Students showed a strong preference for handwriting. Out of 38 total students, only 21% said keyboard/typing was their favorite method, while over 78% preferred one of the methods with handwriting. A variety of typical qualitative comments from some of the students are included in Table 2.

Despite taking about half the time during the learning phase, the handwriting students learned just as much as the typing students. There was no significant difference among the conditions with respect to the learning gain from pre-test to post-test ($F_{2,35}$=0.293, *n.s.*). This means that, even though students solved the same amount of problems and took less time in handwriting than in typing, their learning as measured by performance improved about the same amount (mean=11.75%, stdev=17.34). This measure of learning is relatively coarse; in future studies we intend to analyze in more detail the concepts students mastered rather than purely raw gain scores which do not reflect *how* the learning may have differed among conditions. Although learning gains appeared to be of the same magnitude based on pre- to post-test scores, the fact that the time spent per condition was so different suggests that perhaps handwriting was a more *efficient* learning modality than typing. The concept of *learning efficiency* has been used in, for example, [31], to explore how students may be able to achieve similar levels of mastery but do fewer problems. This is an area we plan to pursue in future work.

One of our hypothesized advantages of using the handwriting modality is that handwriting will allow a greater degree of transfer to paper than using typing interfaces. In this study we attempted to assess level of transfer in each condition by correlating the pre-test score and post-test score with performance during training. We hypothesized that the cases in which there was a *modality switch* (*i.e.*, writing on the pre-test to typing in the interface to writing on the post-test) should have a lower correlation in performance during training *vs.* on the tests. Our results confirmed this. We ran bivariate correlations of percent of problems solved on the first try during *training* and the *post-test* score, grouped by condition. The Pearson correlation for the typing condition was not statistically significant (0.320, p=0.310), whereas for the two handwriting conditions, there was a significant correlation (0.708, $p<0.01$ for handwriting; 0.553, p=0.05 for handwriting-plus-speaking). These results show that handwriting does indeed afford students a higher degree of transfer to paper. Performance during testing more closely matches performance during training when the modality of testing is similar to that of training (or vice versa).

Students in the handwriting-plus-speaking condition in this study performed just as well as students in the other two conditions. This multimodal condition may have both pedagogical and technical advantages over pure handwriting. Prior educational literature has demonstrated that students learn better when they self-explain, and even further, that they learn better when their self-explanations are spoken rather than typed [16]. In addition, overall recognition accuracy can be better when the system is provided with input from two modalities, each of which would be too error-prone on their own to be accurate enough [27].

The third study, the Worked Examples Study, is currently ongoing. It is an *in vivo* study taking place in the LearnLab. Intelligent tutoring systems such as Cognitive Tutors have long incorporated directed step-by-step feedback throughout the problem-solving process; while this is considered to be the strength of the method, it has not been shown to be critical to effective learning. If such detailed feedback is not necessary for student success, the instructional paradigm can be altered significantly when using handwriting input to prevent recognition errors from interrupting the student. For example, we could rely more heavily on worked examples as a method of *feed-forward* to

Table 2. Comments made by students on the post-session questionnaire about each modality**.**

Typing	Handwriting	Handwriting-plus-speaking
"It took too long and was hard to get everything where I wanted."	"Yes, because it is how I'm used to doing problems in math class, by writing them out."	"It made it easier to think it out when I said it while doing it."
"It takes me longer to type math problems [as opposed to] to [writing] them."	"It is easier than typing."	"[It's] easier to understand when you talk through the problems."
	"It was better than typing."	"I like talking through the problems it made me focus more."
	"It was a lot easier and I finished quickly."	

help students. The study underway is designed to begin to address this concern by comparing existing Cognitive Tutors that provide detailed feedback to the same systems that also provide worked examples during problem solving. We intend to analyze student learning as well as student hint and help-seeking during use of the tutoring system to determine whether students that are provided with worked examples use the hint facilities of the tutor less frequently. This study will help shed light on how to effectively design instructional paradigms that can take advantage of the benefits of handwriting input for learning.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have presented an approach to adapting, deploying and testing handwriting-based interfaces in the domain of intelligent tutoring systems for algebra equation solving. The studies we have conducted have given the first positive evidence in favor of handwriting interfaces for *learning* applications. Students can learn the same amount in half the time using handwriting input vs typing-based interfaces. This, combined with the expected decrease in extraneous cognitive load, can allow students to focus more directly on the mathematics, achieving farther curricular goals and finishing with a deeper understanding of the material.

Our next steps will involve exploring other strategies for enhancing recognition accuracy, for instance, by adapting machine learning techniques such as co-training to this application domain [8]. In co-training, two independent labelers (for instance, one which reads the text on a website and one that reads the text of links pointing to a website) can each use the other's guesses to boost their own confidence, thus resulting in a classifier with greater overall accuracy than either one alone. We believe that similar techniques could be applied to our application, in which the handwriting engine is treated as one "labeler" and we combine it with various other "labelers" to achieve more accurate results. Potential labelers include the context of the problem we know the student is solving (which means we know what the student *should* be inputting), and also knowledge about common student errors as well as *this student's* current skills (which means we know what the student *might* be inputting instead). Once these engine enhancements are in place, a summative user study will compare the handwriting-based tutor to the standard Cognitive Tutor classroom practice, focusing on differences in learning and cognitive load.

While this work is in the domain of high school algebra learning, it is likely to generalize to other types of math and to other levels of students. Future efforts in this line of work will explore other domains such as calculus and geometry, which rely even more on spatial information in annotations.

7. ACKNOWLEDGMENTS

Work supported by NSF Award #SBE-03554420 and the Pittsburgh Science of Learning Center. The authors are grateful to the following people who have provided valuable time, support and equipment: Chris Atkeson, Shelley Evenson, Sue Fussell, Tom Bolster, Noboru Matsuda, Jiazhi Ou, Datong Chen, Jacob O. Wobbrock, Adam M. Fass, Darren Gergle, Ryan S. Baker, Angela Wagner, Anupriya Ankolekar, Scott Davidoff, Elsa Golden, Amy Hurst, Ido Roll, Cristen Torrey, Aaron O. Bauer, and Sonya Allin.

8. REFERENCES

[1] Anderson, J.R., Corbett, A.T., Koedinger, K.R., and Pelletier, R. (1995) Cognitive Tutors: Lessons Learned. *Journal of the Learning Sciences*, 4, 167-207.

[2] Anderson, R.H. (1968) Syntax-directed recognition of hand-printed two-dimensional mathematics. In *Interactive Systems for Experimental Applied Mathematics* (eds. Klerer, M. & Reinfelds, J.), Academic Press, NY.

[3] Anthony, L., Yang, J., and Koedinger, K. R. (2005) Evaluation of Multimodal Input for Entering Mathematical Equations on the Computer. In *Proceedings of the SIGCHI Conf. on Human Factors in Computing Systems (CHI'05)*, 1184-1187.

[4] Anthony, L., Yang, J., and Koedinger, K. R. (2006) Towards the Application of a Handwriting Interface for Mathematics Learning. In *Proceedings of the IEEE Conf. on Multimedia and Expo (ICME'06)*, 2077-2080.

[5] Anthony, L., Yang, J., and Koedinger, K. R. (2007) Benefits of Handwritten Input for Students Learning Algebra Equation Solving. In *Proceedings of the Int'l Conf. on Artificial Intelligence in Education (AIEd'07)*, to appear.

[6] Anthony, L., Yang, J., and Koedinger, K. R. Data-Driven Methodology for Handwriting Recognition in Mathematics. Under review.

[7] Bloom, B.S. (1984) The Two Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring. *Educational Researcher*, Vol. 13, Nos. 4-6.

[8] Blum, A. and Mitchell, T. (1998). Combining Labeled and Unlabeled Data with Co-Training. *Proceedings of the Workshop on Computational Learning Theory (COLT'98)*, 92-100.

[9] Chan, K.-F. and Yeung, D.-Y. (2000) Mathematical Expression Recognition: A Survey. *International Journal of Document Analysis & Recognition 3*, 1, 375-384.

[10] Corbett, A.T. and Trask, H. (2000) Instructional Interventions in Computer-based Tutoring: Differential Impact on Learning Time and Accuracy. In *Proceedings of the SIGCHI Conf. on Human Factors in Computing Systems (CHI'00)*, 97-104.

[11] Corbett, A.T. (2001) Cognitive Computer Tutors: Solving the Two-Sigma Problem. *Proceedings of User Modeling*, 137–147.

[12] Corbett, A.T., Koedinger, K.R., Hadley, W.H. (2001) Cognitive Tutors: From the Research Classroom to All Classrooms. In: P. Goodman (ed.): *Technology Enhanced Learning: Opportunities for Change*. L. Erlbaum, Mahwah New Jersey, 235–263.

[13] Frankish, C., Hull, R., and Morgan, P. (1995) Recognition Accuracy and User Acceptance of Pen Interfaces. In *Proceedings of the SIGCHI Conf. on Human Factors in Computing Systems (CHI'95)*, 503-510.

[14] Goldberg, D. and Goodisman, A. (1991) Stylus User Interfaces for Manipulating Text. *Proceedings of UIST 1991*, 127-135.

[15] Graesser, A., Moreno, K.N., Marineau, J.C., Adcock, A.B., Olney, A.M. and Person, N.K. (2003) AutoTutor Improves Deep Learning of Computer Literacy: Is it the Dialog or the Talking Head? *Proceedings of the Int'l Conference on Artificial Intelligence in Education (AIEd 2003)*, 47-54

[16] Hausmann, R.G.M. and Chi, M.T.H. (2002) Can a Computer Interface Support Self-explaining? *Cognitive Technology* 7, 1, 4-14.

[17] Jackson, Lorrie. (2004) Laptops, Handhelds, or Tablet PCs? *Education World Technology* Article: http://www.education-world.com/a_tech/tech/tech198.shtml

[18] Jumping Minds. (2003) Practice Series. http://www.jumpingminds.com.

[19] Kanahori, T., Tabata, K., Cong, W., Tamari, F., and. Suzuki, M. (2000) On-Line Recognition of Mathematical Expressions Using Automatic Rewriting Method. *Proc. ICMI 2000*, 394-401.

[20] LaLomia, M.J. (1994) User acceptance of handwritten recognition accuracy. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'94)*, 107.

[21] LaViola, J.J. (2006) An Initial Evaluation of a Pen-Based Tool for Creating Dynamic Mathematical Illustrations. *Proc. Eurographics Workshop on Sketch-Based Interfaces and Modeling*, EG Workshop Series, 157-164.

[22] Liwicki, M. and Bunke, H. (2005) Enhancing Training Data for Handwriting Recognition of Whiteboard Notes with Samples from a Different Database. *Proceedings of the Int'l Conference on Document Analysis and Recognition (ICDAR'05)*, 550-554.

[23] MacKenzie, I.S., and Chang, L. (1999) A Performance Comparison of Two Handwriting Recognizers. *Interacting with Computers 11*, 283-297.

[24] Madhvanath, S., Vijayasenan, D., and Kadiresan, T.M. (2006) LipiTk: A Generic Toolkit for Online Handwriting Recognition. *10th International Workshop on Frontiers in Handwriting Recognition*, CENPARMI.

[25] Matsakis, N.E. (1999) *Recognition of Handwritten Mathematical Expressions*. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA.

[26] Mayer, R. E. (2001) Multimedia Learning. New York: Cambridge University Press, Boston, MA.

[27] Oviatt, S. (2000) Taming Recognition Errors with a Multimodal Interface. *Communications of the ACM*, 45-51.

[28] Oviatt, S., Arthur, A. and Cohen, J. (2006) Quiet Interfaces that Help Students Think. In *Proeedings of UIST 2006*, 191-200.

[29] Paas, F. and Van Merrienboer, J. (1994) Variability of worked examples and transfer of geometry problem-solving skills: A cognitive-load approach. *Journal of Educational Psychology*, 86, 122-133.

[30] PISA (2005). International Outcomes of Learning in Mathematics Literacy and Problem Solving. http://nces.ed.gov/pubs2005/2005003.pdf

[31] Ringenberg, M.A. and VanLehn. K. (2006) Scaffolding Problem Solving with Annotated, Worked-Out Examples to Promote Deep Learning. *Proceedings of the International Conference on Intelligent Tutoring Systems (ITS 2006)*, 625-634.

[32] Santos, P.J., Baltzer, A.J., Badre, A.N., Henneman, R.L., and Miller, M.S. (1992) On handwriting recognition performance: Some experimental results. *Proceedings of the Human Factors Society 36th Annual Meeting*, 283-287.

[33] Smithies, S., Novins, K. and Arvo, J. (2001) Equation Entry and Editing via Handwriting and Gesture Recognition. *Behaviour and Information Technology*, 20, 53-67.

[34] Sweller, J. (1988) Cognitive Load During Problem Solving: Effects on Learning. *Cognitive Science 12*, 2, 257-285.

[35] Tapia, E. and Rojas. R. (2004) Recognition of On-Line Handwritten Mathematical Expressions using a Minimum Spanning Tree Construction and Symbol Dominance. In *Graphics Recognition: Recent Advances & Perspectives* (eds. Lladós, J. & Kwon, Y.-B.), *Lecture Notes in Computer Science 3088*, 329-340.

[36] VanLehn, K., Lynch, C., Taylor, L., Weinstein, A., Shelby, R., Schulze, K., Treacy, D. and Wintersgill, M. (2002) Minimally Invasive Tutoring of Complex Physics Problem Solving. Proceedings of the Intelligent Tutoring Systems Conference, 367-376.

[37] Wood, C. (2002) Technology and Education. *PC Magazine*: http://www.pcmag.com/article2/0,4149,15154,00.asp.

[38] xThink MathJournal. http://www.xthink.com/MathJournal.html

[39] Zanibbi, R., Blostein, D., and Cordy, J.R. (2002) Recognizing Mathematical Expressions Using Tree Transformation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 11, 1455-1467.