Evaluation of Techniques for Visualizing Mathematical Expression Recognition Results

Joseph J. LaViola Jr.*
University of Central Florida
School of EECS
Orlando, FL 32816 USA

Anamary Leal†
University of Central Florida
School of EECS
Orlando, FL 32816 USA

Timothy S. Miller‡
Brown University
Dept. of Computer Science
Providence, RI 02912 USA

Robert C. Zeleznik§
Brown University
Dept. of Computer Science
Providence, RI 02912 USA

ABSTRACT

We present an experimental study that evaluates four different techniques for visualizing the machine interpretation of handwritten mathematics. Typeset in Place puts a printed form of the recognized expression in the same location as the handwritten mathematics. Adjusted Ink replaces what was written with scaled-to-fit, cleaned up handwritten characters using an ink font. The Large Offset technique scales a recognized printed form to be just as wide as the handwritten input, and places it below the handwritten mathematical expression. The Small Offset technique is similar to Large Offset but the printed form is set to be a fixed size which is generally small compared to the written expression.

Our experiment explores how effective each technique is with assisting users in identifying and correcting recognition mistakes with different types and quantities of mathematical expressions. Our evaluation is based on task completion time and a comprehensive post-questionnaire used to solicit reactions on each technique. The results of our study indicate that, although each technique has advantages and disadvantages depending on the complexity of the handwritten mathematics, subjects took significantly longer to complete the recognition task with Typeset in Place and generally preferred Adjusted Ink or Small Offset.

Keywords: pen-based user interfaces, mathematical expression recognition results, usability evaluation, typeset, adjusted ink

Index Terms: H.5.2 [Information Interfaces and Presentation]: User Interfaces—Interaction Styles, Evaluation/Methodology;

1 INTRODUCTION

Computer recognition of handwritten mathematics is an old [1] and important [7] field, and many advances have been made in the decades of research on it. However, we posit that, since some handwritten math is ambiguous even to another human, even the best achievable recognition techniques will at times misinterpret the writer's intent. Thus, identifying and correcting recognition errors can be viewed as a fundamental problem. Nonetheless, compared to the core algorithmic problem of recognizing handwritten mathematics, very little attention has focused on UI techniques which allow users to cope with recognition errors. In particular, we have identified two UI tasks which merit investigation: visualization techniques for depicting the machine interpretation of handwritten mathematics and thus identifying recognition errors, and interaction techniques for correcting the machine interpretation.

This paper focuses on evaluating techniques for the initial task of visualizing the machine interpretation of handwritten mathematics. The previously developed techniques we chose [14] were intended to cover the space of design possibilities and common usage scenarios. Our first priority was to present typeset feedback because of its familiarity and expressiveness. In pilot studies we explored various options for this feedback and concluded that three of those options could represent this space. We also included a fourth option that addressed a specific limitation of typeset feedback when given in place of the user's ink. None of the other techniques we considered seemed to warrant the effort required to expand this evaluation, although not all those techniques were fully implemented.

For this evaluation, we developed a set of tasks intended to be representative of a set of common usage scenarios, for example, transcription vs. problem solving or large open writing spaces vs. small dialogs. The objective was to address quantitatively the question of whether there were significant relative benefits to the different techniques in individual scenarios, and whether there was a generally acceptable consensus technique. The experimental study evaluated each technique in terms of speed, readability, use of white space, distraction, and overall preference. To determine when one technique might be more appropriate than another, subjects evaluated each one under four conditions representing different mathematical expression types of varying complexity. To the best of our knowledge, this is the first study to systematically explore the usability involved with different approaches for presenting mathematical expression recognition results to the user.

In the next section, we discuss work related to evaluating recognition feedback techniques. We then present a discussion of the techniques to provide context for our usability study. Next, we present the details of our experiment and discuss the results. Finally, we present conclusions.

2 RELATED WORK

Although there has been a significant amount of work on mathematical expression recognition over the years[2], there has been little on how to effectively present recognition results to the user. Zanibbi, et al. [13] experimentally evaluated a version of offset typeset feedback and a dynamic morph of the user's ink to fit a cleaned-up version of the input characters' bounding boxes, based on the recognition and parse results. Our evaluation focuses on real-time feedback where theirs used solely batch feedback, and we also compare somewhat different techniques: we don't have an animated morphing technique, but the technique that cleans up the user's ink does so to a fixed set of ink characters (an "ink font"), thus additionally providing feedback on the character recognition.

LaViola [6] evaluated an entire math sketching system, including a recognition feedback technique that changed the user's input to cleaned-up handwritten ink the same way our adjusted technique does. However, the contribution of the recognition feedback technique was not isolated, nor were any alternatives compared. In addition, our adjusted technique additionally shows parse recognition results by coloring the characters.

Smithies, et al. [10] presented a math recognition system that provided an interactive display of symbol recognition by shading the bounding box of each group of strokes corresponding to one symbol and then locating a typeset display of the recognized sym-

*email: jjl@cs.ucf.edu
†email: leal@cs.ucf.edu
‡email: tsm@cs.brown.edu
§email: bcz@cs.brown.edu

bol in the upper-left corner of the bounding box. Thus, this technique is loosely related to our small offset technique; however, it was not a complete interactive recognition display since it could not depict the parse structure. Instead, the user would invoke the equation parser explicitly and see the result in a separate window.

CueTIP [8] presents a mixed-initiative interface for correcting linear handwritten text in which the system continues to assist the user even as corrections are made. The primary focus of this work is on the correction interface and not on the visualization mechanism for identifying a recognition error. The visualization technique used is similar to our small offset typeset display technique, but they offer no analysis of it.

Wais, et al. [12] evaluated recognition feedback in the context of sketch recognition. There feedback consisted of displaying recognized symbols in different colors and drawing text labels next to recognized symbols. Although these techniques are similar to some of the approaches we are testing, this work was focused on recognition feedback for circuit diagrams instead of mathematics.

Igarashi, et al. [5] and Goldberg and Goodman [3] present techniques for visualizing recognition errors as alternates in situ of the original drawing context. Thus, in some sense they are equivalent to our adjusted ink technique, but they additionally display recognition alternates. However, the two problems with applying these approaches to handwritten mathematics are that they force errors to be seen and corrected as they occur, and they cannot be easily extended to clearly depict both symbol and parsing alternates in situ.

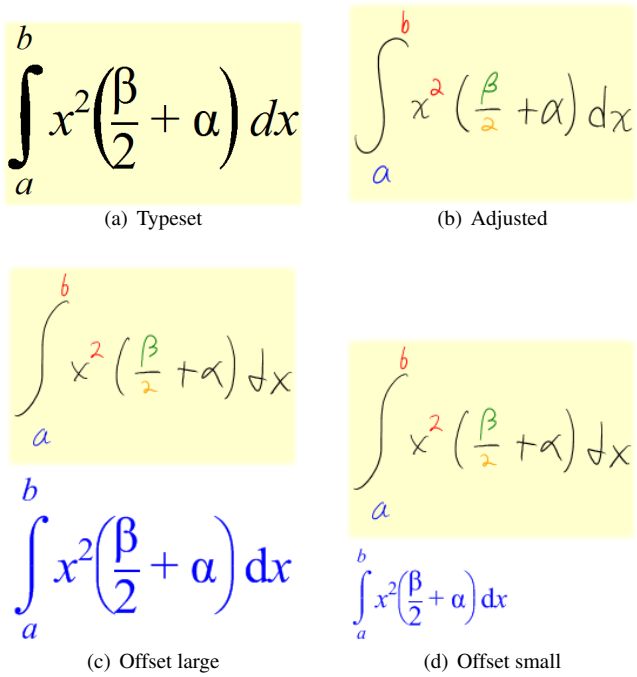
3 Visualization Techniques

When designing our recognition visualization techniques, we established several guiding design criteria. First, we wanted the visualization techniques to require minimal extra space on the display outside of the handwritten math. Second, we wanted to express unambiguously the complete machine interpretation of the recognized mathematics. Third, we wanted to present the recognition interpretation to the user as they were writing so they could identify errors as early as possible. Fourth, we did not want to disrupt or distract the user from their mathematical task. Last, we wanted it to be easy for users to locate recognition errors at their convenience (i.e., ranging from instantly to even minutes later).

We surveyed a number of techniques with overlapping benefits and, in pilot testing, narrowed this set to four techniques which we believe have distinct benefits. We briefly present these four alternative recognition visualizations, along with their hypothesized tradeoffs (Figure 1). The techniques include: *Typeset in Place*, which replaces handwritten ink strokes with typeset mathematics on the fly; *Adjusted Ink*, which replaces handwritten ink strokes with characters from a clarified and colorized ink font; *Large Offset*, which displays typeset recognition results below and scaled to the same width as the handwritten ink; and *Small Offset*, which also displays typeset results below the handwritten ink but at a constant, relatively small font size. More details on these techniques and their development can be found in [14].

3.1 Typeset in Place

The Typeset in Place technique provides a clear, unambiguous display of the full semantics of the machine's interpretation of the handwritten mathematics and typically uses no more (often less) space than the user's own handwriting. This technique replaces the user's ink with appropriately typeset math, at the same approximate size as the ink, during sufficiently long pauses (.5 secs) in the writer's input (see Figure 1(a)). To avoid having to adjust previous typeset characters, the typeset font size is best-fit only to the initial handwritten character. This means that subsequent characters will not match exactly or sometimes even closely to their handwritten counterparts. Since subsequent ink is interpreted relative to the displayed typeset notation, the shift in character location and size



(a) Typeset (b) Adjusted

(c) Offset large (d) Offset small

Figure 1: The four techniques for visualizing recognition and parse results: Typeset in Place (a), Adjusted Ink (b), Large Offset (c), and Small Offset (d). For those reading this in black and white, in the last three sub-figures all the ink is black, except that the a is blue, the b and the 2 in x^2 are red, the β is green, and the 2 below it is orange.

can cause parse recognition errors for users who fail to adapt their writing to the emerging typeset notation. In addition, since the cost of letting an error go is so high in terms of cascading errors to future input, users are essentially forced with this technique to resolve mistakes as they occur.

3.2 Coloring

To avoid symbol adjustment complexities of the typeset replacement technique, symbol coloring can be used to indicate geometrical parse relationships. This colorized display is initially less clear than typeset notation since it requires users to understand a novel mapping between colors and mathematical semantics. In addition, colorization alone does not indicate which symbols have been recognized, so it must be paired with some other technique to present a complete semantic description of the mathematics. We have explored a set of different mappings between color and mathematical semantics, but only present the one approach we believe is most general: we color each character according to the typeset baseline to which that character belongs. The idea is that the user determines the parse structure by considering the relative colors of neighboring characters; characters that have the same color are siblings (i.e. 2x), and characters that have different colors must have different baselines (i.e., 2^x). Since relatively few symbol colors can be readily distinguished for small text, especially on the typical poor Tablet PC screens, we use a palette of five colors (orange, green, brown, red, and blue) and only color the five most recently modified baselines, leaving the rest black.

3.3 Adjusted Ink

The Adjusted Ink technique pairs the Coloring technique with a symbol substitution technique that replaces the user's handwritten symbols with corresponding ink symbols drawn from a pre-defined,

handwritten, but highly legible font (Figure 1(b)).[1] Since the replaced ink occupies exactly the same bounding box as the original ink, this technique requires no more space than the user's original handwriting. Compared to the initial Typeset in Place technique, this technique is less disruptive, just as economical on space, but unfortunately cannot always display mathematical semantics unambiguously.

3.4 Large Offset

The Large Offset technique displays recognition results in a typeset form that is below and uniformly scaled to the width of the handwritten ink. This technique also provides a redundant, but in-place, display of the structural parse by using the Coloring technique on the user's own handwritten ink (Figure 1(c)). Since the user's ink is not geometrically adjusted, the typeset notation and the ink colors are updated instantly after each stroke is drawn without causing a necessary disruption in the user's input. Since the typeset notation is fit to the width of the handwritten ink, this technique roughly doubles the display space requirements over just the user's handwriting. However, by matching the size of the typeset to the ink, handwritten characters tend to line up closely with their typeset counterparts, facilitating reasoning about recognition errors. The relationship between ink and typeset can be further explored by hovering the pen over a typeset character to see a red outline around its corresponding ink stroke(s). We chose to put typeset results below rather than above the user's ink because our experimentation showed that putting it above tended to overlap with already-written expressions, making it difficult to refer to previous expressions in a derivation without requiring extra vertical spacing.

3.5 Small Offset

Small Offset behaves the same as Large Offset except that instead of matching the typeset display to the width of the handwritten ink, a fixed, relatively small, font size is used (Figure 1(d)). Small Offset is thus quite similar to approaches taken by other math [13] and plain text (e.g., Microsoft's Tablet PC text input panel) systems which draw unscaled typeset notations near the input ink—the principal difference being that Small Offset is paired with our Coloring technique. Small Offset requires relatively little additional display space, but can be somewhat hard to read for complex expressions and establishes no structured visual relationship between corresponding ink and typeset characters.

4 Technique Conjectures

To understand how these techniques affect a user's ability to detect and subsequently correct mathematical expression recognition errors, we first conducted a small pilot study. In the pilot, we asked three subjects to try out the different techniques when writing several polynomial expressions. From this pilot and from our technique design, we came up with a number of conjectures about each one.

Typeset in Place. The Typeset in Place technique is expected to be desirable because it provides a highly readable, and complete display of the machine interpretation that is effortlessly unified with the handwritten input and thus displayed directly in the user's field-of-view. We believe these advantages would stand out in situations where recognition accuracy can be expected to be high, such as for simple expressions with at most one super- or sub-script, and in situations where display space is at a premium such as when writing densely-packed expressions. However, in practice, achieving truly high recognition accuracy in general is so difficult that we hypothesize that this technique will be the least desirable since it exacerbates recognition errors as it adjusts symbol locations.

Adjusted Ink. We expect that, compared to Typeset in Place, Adjusted Ink will not be considered disruptive. However, we also suspect that users will be less sensitive to detecting certain errors, both because of the the Coloring technique's fundamental inability to express the complete recognition semantics, and because Coloring de-emphasizes certain errors, such as when things that are supposed to be on different baselines are recognized to be on the same baseline (i.e., it's easy to overlook two things having the same color when they should have different colors). Nonetheless, because of its economical use of space and non-disruptive display, we hypothesize that Adjusted Ink will be the fastest to use, the least distracting, and one of the most preferred overall.

Large Offset. The Large Offset technique was designed to provide a clearly readable display, at the expense of using extra display space. Thus, we expect that users will find it most effective for single complex expressions, and least effective when writing multiple expressions. We also suspect that people may find the large size of the typeset display to be distracting. Thus with this technique, we hypothesize that the user experience will be the most varied across different types of input, with the technique faring quite well for isolated complex expressions, and perhaps quite poorly when used to write multiple expressions.

Small Offset. We believe that Small Offset balances trade-offs in space, legibility, expressiveness and disruptiveness well, such that it will have the least variance between its worst and best case scenarios. In particular, we suspect that it will perform worst for long or complex expressions which may be hard to read and associate with handwritten ink, and that it will perform best when writing multiple expressions. Overall, we hypothesize that this technique will be rated highly, on par with Adjusted Ink, and perhaps better because of its ability to completely express the machine interpretation.

5 Usability Evaluation

To explore the effective of our recognition visualization techniques and the conjectures described in the last section, we conducted a formal user evaluation to quantify user preferences across the different techniques and mathematical expression complexities.

5.1 Subjects and Apparatus

Twenty-four subjects (15 male, 9 female) were recruited from the undergraduate population at the University of Central Florida with ages ranging from 18-28. Of the 24 subjects, 11 had used some form of a pen-based interface (e.g., Palm Pilot, Tablet PC), but their interactions with them were minimal. We chose subjects who had taken mathematics classes with a minimum understanding of Calculus since we wanted our subject population to exemplify students that could have used our software in their work. We also wanted to ensure that subjects understood all of the mathematical symbols used in the experiment. The experiment took each subject approximately 60 minutes to complete and all subjects were paid 10 dollars for their time.

The experimental setup (see Figure 2) consisted of a HP TC4400 Tablet PC with 1.83 GHz dual core processor and 2 GB of memory. The Tablet PC was attached to an LCD monitor so the experimenter could see what subjects were writing on the screen without having to look over a subject's shoulder. A video camera was also used so we could analyze each experimental session after it was completed. The software used in the experiment was our custom built mathematical expression recognition engine [14].

5.2 Experimental Task

The task subjects had to perform in the experiment was to enter mathematical expressions using the Tablet PC, and, given a recognition visualization technique, determine whether the recognizer

[1] We considered using an actual typeset font, but found that it produced an ugly "ransom note" look that, interestingly, does not seem to be generated by the handwritten font. We also considered letting users create their own handwriting font, but based on pilot tests found that unnecessary for this evaluation.

Figure 2: A subject participating in our experiment. He works on a Tablet PC while the experiment moderator watches an LCD screen. The screen is also recorded with a video camera for later analysis.

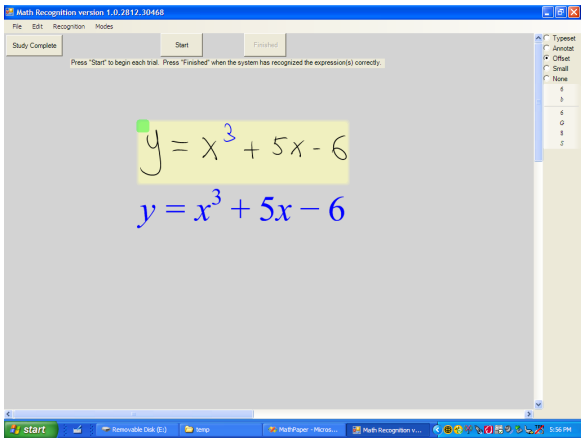


Figure 3: A screen shot of the interface used in our experiment.

correctly interpreted their expression. If an expression had errors, subjects would correct any mistakes until recognition was correct. For each trial, subjects would press the "Start" button to begin writing (starting a timer) and press the "Finished" button (stopping the timer) when recognition was correct (Figure 3).

Subjects wrote four different types of mathematical expressions (simple, multiple, compact, long) during the course of the experiment. Expression types varied in complexity so we could better determine how each recognition visualization technique affected a subject's ability to see and correct recognition errors.

The expression

$$3x^2 - 4y^2 = 7$$

is an example of a simple expression. These types of expressions were simple polynomials designed to be fairly easy to recognize and not take up too much space. They are characteristic of what an algebra student might be working with.

For multiple expressions, users wrote a series of four equations with four unknown variables, such as

$$\begin{aligned} 5x + 3y - 2z + w &= 0 \\ 3x - 5y + z - 4w &= 5 \\ 7x - y + 3z - 2w &= -6 \\ x - 2y + 7z - 2w &= 2. \end{aligned}$$

These expressions were designed so subjects had to write a sequence of expressions vertically (as if working with simultaneous equations), to specifically judge the effectiveness of the "in place" techniques versus the "offset" techniques.[2]

For compact expressions, users wrote mathematics that utilized more complicated structure such as

$$\int_a^b \frac{x^{2+t}\sin x}{\alpha^q \tan(x^3) + c^x}\, dx$$

where superscripts are featured so there would be more symbols closer together. Division lines are also used in these expressions to make them more compact. These expressions were designed to help test user frustration and distraction due to abrupt changes in converting the mathematical symbols into either Typeset or Adjusted Ink in place.

Finally, long expressions were designed to test user distraction as they wrote across the screen and had to utilize the horizontal scroll bar to make room for the entire expression. The expression

$$\frac{3x^5 + 4x^4 - 7x^3 + 5x^2 - 6x + 1}{9x^y - x^p + 2x^d - 2x^y + x - 4}$$

is an example of a long expression.

Subjects had three different methods for correcting recognition errors. First, they could erase parts of the handwritten expression using a scribble gesture and rewrite the appropriate symbols until a correct recognition was made. Second, for incorrectly recognized symbols, subjects could go to the alternate list to find the correct recognition. Third, subjects could circle a symbol or group of symbols and move them on the screen to correct parsing errors.

5.3 Experimental Design and Procedure

We used a 4 x 4 within-subjects factorial design where the independent variables were visualization type and mathematical expression type. Visualization type varied between Typeset in Place ("Typeset"), Adjusted Ink in place ("Adjusted"), Large Offset ("Large"), and Small Offset ("Small"). Mathematical expression type varied between simple expressions, multiple expressions, compact expressions, and long expressions.

The dependent variable was task completion time, defined as the time it took for subjects to press the 'Start' button, write down a mathematical expression, have the system recognize it, make any corrections to the expression (if needed), and press the 'Finished' button when the expression was correctly recognized. In addition to task completion time, we also measured user preferences using a post-questionnaire which asked a series of questions about each recognition visualization technique (see the next section on Usability Metrics for details).

After an initial analysis of the task completion times for each condition, we discovered that the time spent on correcting errors could be a significant confound in the overall task completion time. Thus, we collected more data on task completion time by examining the video recordings from each subject. We recorded the amount of time each subject spent correcting both symbol and parsing errors for each condition in the experiment. In addition, we also collected how many symbol and parsing errors subjects corrected and how many times they used the scribble erase gesture, the alternate list, or the circle gesture in doing so. Note that we found one of the video sessions was corrupted and so we could only gather this data from 23 subjects.

It is important to note that we did not attempt to control for recognition accuracy in our experimental design other than through pilot testing a strategic choice of expressions (i.e., expressions that were not "reasonably" recognizable were avoided) that were likely to produce a representative set of errors across users. We were not

[2]Recall that the offset techniques place the recognition result directly below the handwritten mathematics.

able to devise an experimental procedure that rigorously controlled for recognition accuracy in a valid way, since arbitrarily inducing errors would likely have resulted in introducing "unreasonable" errors that were not reflective of how real recognizers work.

The experiments began with a pre-questionnaire asking each subject their age, gender, and if they had any experience with pen-based computers. Each subject was assigned a 4-digit randomized number that was used to label the questionnaires, timing files, and video files relevant to an individual subject.

Subjects were then given an explanation of how to use the Tablet PC, the experimental task and procedure, and the techniques involved in accomplishing the task. The experiment moderator also highlighted important interaction guidelines for using tablets such as what minimum pressure was needed on the screen for an ink stroke to be rendered, that subjects could use the pen as if using pencil and paper, and that it was okay for the them to rest their hands on the screen.

Subjects then went through a training session. The first part of the training session was to collect a mathematical expression that is used by the recognizer to help improve recognition accuracy. This expression contained symbols such as '2', 'z', '5', and 's' which have a number of different styles in which they can be written. Collecting how the user wrote these symbols ensured that the recognizer would accurately deal with a particular subject's writing for those symbols. After this step, the experiment moderator showed each subject how to use the techniques for correcting recognition errors, how to switch between different recognition visualization techniques, and how to start and stop each trial. For the second part of the training session, subject were instructed to switch to a technique chosen by the moderator, press the 'Start' button on the screen (starting the timer), write the expression shown by the moderator using an index card, and press 'Finished' once the recognition was correct (after making any corrections). For training, each subject wrote eight expressions, using each recognition visualization technique twice. The order of the training trials was pre-determined and fixed for all subjects. Also note that the eight expressions used in the training session contained all of the symbols that were used in the expressions written in the experiment trials. This approach ensured that if subjects had problems with any particular symbol they could practice getting the recognizer to recognize their handwriting correctly before moving on the the actual trials.

After the training session, subjects were asked if they were comfortable with the task they had to perform. In each case, they said yes, and the experiment was started. For each trial, subjects had to write down one mathematical expression.[3] As in the training session, the experiment moderator showed subjects an index card with the mathematical expression they were supposed to write. There were four expression types and four recognition visualization techniques for a total of 16 trials. To control for order effects, the ordering of the trials was randomized for each of the 24 subjects.

During the trials, if a subject had trouble writing the expression after several attempts, the experiment moderator provided guidance on how to get the expression to be recognized correctly, such as how they might write a particular symbol, and by making suggestions on using the error correction user interface. Note that if a subject accidentally clicked on the "Finished" button before an expression was correctly recognized, the trial was discarded and redone.

Once the trials were finished, subjects were given a post-questionnaire which asked questions about their use of the four different recognition visualization techniques. Subjects could also go back to the Tablet PC and revisit the techniques if they wished.

[3]Even though the multiple expression type has subjects write four expressions, we treat this as one large expression.

5.4 Usability Metrics

In addition, to the task completion time, we felt it was important to gather feedback from each subject about their use of the four recognition visualization techniques in a systematic way. Thus, in the first part of the post-questionnaire, we had subjects respond to a total of four sets of eight identical statements (eight statements for each recognition visualization type). These statements used a seven point Likert scale (1=strongly agree, 7=strongly disagree) and they asked subjects to comment on each technique's

- readability
- ability to detect recognition errors
- ability to avoid and correct recognition errors
- level of frustration
- effective use of screen space
- level of distraction
- overall experience.

In the second part of the post-questionnaire, we asked subjects to write down which technique they preferred the most and the least as well as comment on their experiences with each technique. Finally, we asked subjects to comment on the real-time feedback provided by the techniques.

5.5 Results

5.5.1 Task Completion Time

A repeated measures two way analysis of variance (ANOVA) was performed on task completion time (dependent variable) with expression type (ET) and visualization technique type (VT) as the independent variables. Table 1 summarizes the main effects of the independent variables as well as their interaction effect. These results show there were significant differences in task completion times for both VT and ET and their interaction.

Effect	Task Completion Time
ET	$F_{3,21} = 57.84$ $p < 0.05$
VT	$F_{3,21} = 23.14$ $p < 0.05$
ET × VT	$F_{9,15} = 3.79$ $p < 0.05$

Table 1: The main and interaction effects for expression type (ET) and recognition visualization type (VT) for task completion time.

To gain a better understanding of how the different conditions affected task completion time, we conducted a post-hoc analysis, performing pairwise comparisons on the four VT (six comparisons), and on the interaction between ET and VT (24 comparisons). To control for the chance of Type I errors, we used Holm's sequential Bonferroni adjustment [4] with 30 comparisons at $\alpha = 0.05$.

	Typeset	Adjusted	Large	Small
Mean:	151.89	81.42	78.72	81.39
SD:	53.29	28.24	28.06	31.5

Table 2: Mean completion times (in seconds) for the four recognition visualization techniques when expression type is collapsed.

For VT (see Table 2), there were significant differences between Typeset and Adjusted ($t_{23} = 5.1, p < 0.00178$), Typeset and Large ($t_{23} = 7.66, p < 0.00167$), and Typeset and Small ($t_{23} = 5.23, p < 0.00172$). These results indicate that it took subjects significantly longer to complete the recognition task using Typeset.

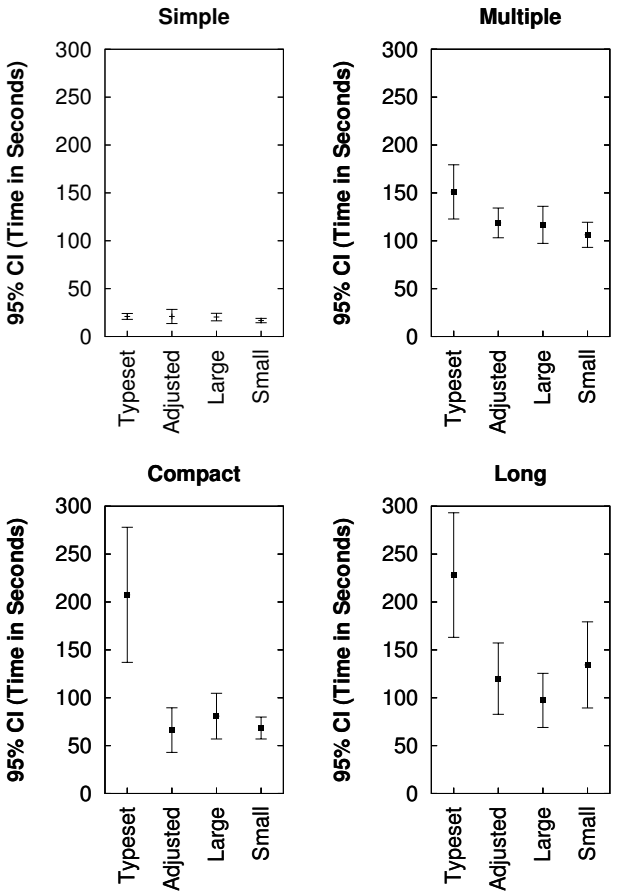


Figure 4: Mean completion times (in seconds) for each condition in the experiment.

For the interaction between VT and ET (see Figure 4), Table 3 summarizes the pairwise comparisons that are either significant or close to being significant.[4] These results show it took significantly longer for subjects to complete the recognition task using Typeset for Compact or Large.

In an effort to isolate any potential confounding effect if different expression or visualization types resulted in different proportions of error correction technique usage, we isolated the amount of time the users spent entering expressions other than when they were correcting errors. Analysis otherwise identical to the analysis for the total entry time was performed on this "non-error" time. For VT (see Table 4), there were significant differences between Typeset and Adjusted ($t_{22} = 3.985, p < 0.00172$), and Typeset and Large ($t_{22} = 4.731, p < 0.00167$), and Typeset and Small ($t_{22} = 3.617, p < 0.00179$).

For the interaction between VT and ET (see Figure 5), Table 5 summarizes the pairwise comparisons that were close to significant. Due to the Bonferroni adjustment, no pairwise comparisons were significant.

5.5.2 Post-Questionnaire Results

In the first part of the post-questionnaire, we asked each subject to respond using a Likert scale to eight statements (on readability, ease of error detection, helpfulness for error avoidance and correction, frustration, efficiency of screen space usage, distraction, and overall experience) for each of the four recognition visualization techniques. To analyze their responses, we conducted Friedman tests on each set of statements (eight tests in total), and to further analyze the data, we ran a post hoc analysis performing pairwise comparisons using Wilcoxon Signed Rank tests. For the post-hoc analysis, we also used Holm's sequential Bonferroni adjustment [4] with 6 comparisons at $\alpha = 0.05$ for each test.

Significant differences were found for all eight statements except for the one on error detection. For all statements where significant differences were found, the results of post-hoc analysis are summarized in Figure 6.

In the second part of the post-questionnaire, subjects were asked to choose which technique they preferred the most and the least (see Figure 7). Subjects preferred either Adjusted or Small the most and disliked Typeset and Large the most which confirms the results from subject responses to the Likert scale statements.

Subjects were also asked to give comments on their experiences with each technique and their responses are in line with their overall preferences. For Typeset in Place, approximately 25% of the comments were positive with particular attention to ease of use and readability. Of the remaining 75% of the comments, many subjects

[4]Note that the pairwise comparisons marked with a * in Table 3 are not significant due to due the Bonferroni adjustment.

Comparison	Test Statistic	P Value
Simple		
Typeset - Small*	$t_{23} = 2.62$	$p = 0.015$
Multiple		
Typeset - Small*	$t_{23} = 3.16$	$p = 0.0044$
Compact		
Typeset - Small	$t_{23} = 4.04$	$p < 0.00185$
Typeset - Large	$t_{23} = 3.78$	$p < 0.00192$
Typeset - Adjusted*	$t_{23} = 3.5$	$p = 0.002$
Long		
Typeset - Large*	$t_{23} = 3.38$	$p = 0.0025$
Typeset - Adjusted*	$t_{23} = 2.58$	$p = 0.0168$

Table 3: Relevant interaction effects between VT and ET for task completion time.

	Typeset	Adjusted	Large	Small
Mean:	68.87	54.10	54.09	56.41
SD:	87.72	68.90	69.25	73.32

Table 4: Mean completion times (in seconds) for the four recognition visualization techniques when expression type is collapsed.

Comparison	Test Statistic	P Value
Simple		
Typeset - Small*	$t_{22} = 2.352$	$p = 0.028$
Multiple		
Typeset - Small*	$t_{22} = 2.469$	$p = 0.022$
Compact		
Typeset - Large*	$t_{22} = 2.672$	$p = 0.014$
Typeset - Adjusted*	$t_{22} = 3.090$	$p = 0.005$
Long		
Typeset - Large*	$t_{22} = 2.914$	$p = 0.008$
Large - Small*	$t_{22} = -2.615$	$p = 0.016$

Table 5: Relevant interaction effects between VT and ET for task completion time, excluding time spent correcting errors.

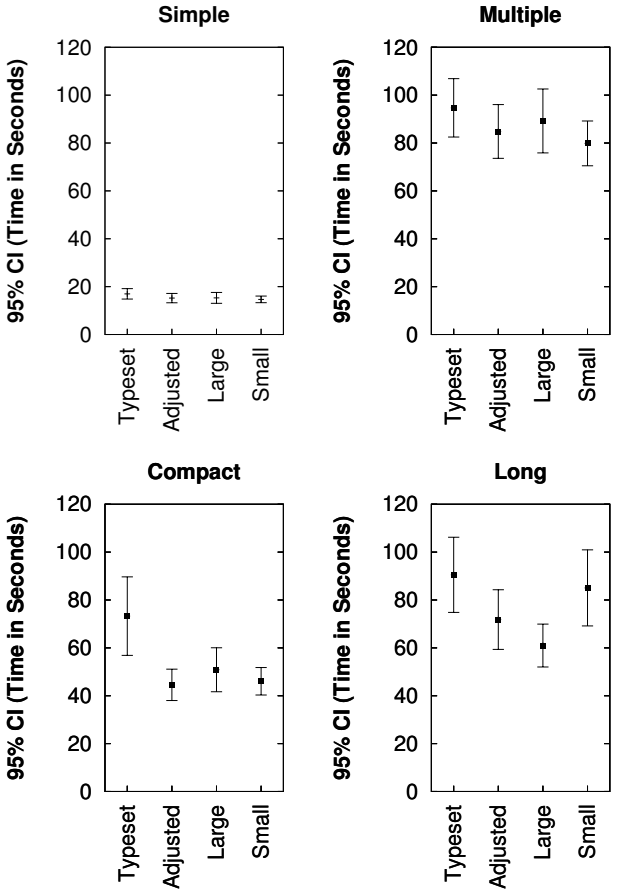


Figure 5: Mean completion times (in seconds), excluding error correction, for each condition in the experiment.

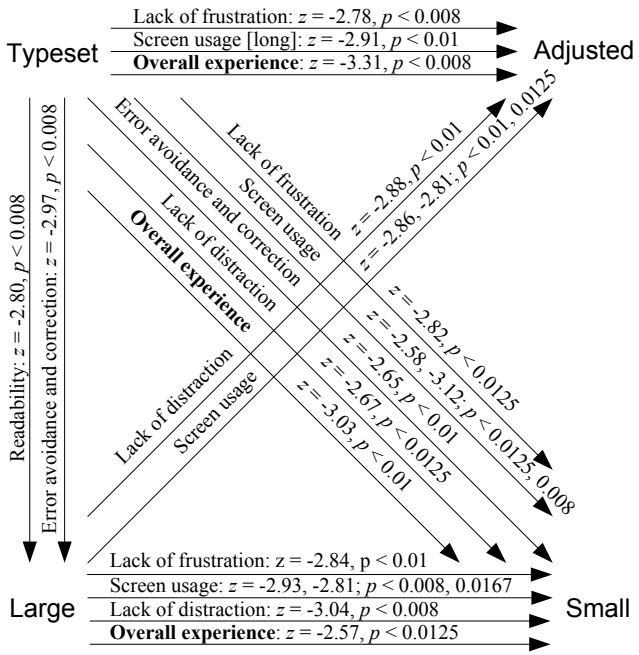


Figure 6: The statistically significant results from post-hoc analysis of the Likert scale parts of our post questionnaire. Arrows point in the direction of the technique that was rated more favorably by users on the labeled statement. The screen usage efficiency attribute was rated separately for the multiple equations and long equations conditions; in this figure the statistics are reported for the multiple condition followed by the long condition except in the one case where only one reached significance (noted in brackets).

mentioned the real-time feedback was slow compared to their writing speed. These subjects said they had to slow down their writing speed to compensate for the delay. Other negative comments included a general frustration with the technique, difficulty dealing with square roots, subscripts, and superscripts, and difficulty in fixing errors when they occurred.

For Adjusted Ink, approximately 75% of the comments were positive. One of the most significant comments was on Adjusted's positive aesthetic value. One subject commented, "This was the easiest to use, and if you made an error the easiest to correct. This looked the cleanest and was actually fun." Subjects also noted that they liked Adjusted because it looked like their own handwriting. One subject commented, "I love it. I felt most comfortable writing fast and reading my handwriting. In other words, I didn't feel judged by my bad handwriting." Subjects also stated that Adjusted made it easier to fix errors and was less distracting than Typeset. Of the 25% of the comments that were negative, the majority of them were about decreased readability with Adjusted and a general dislike of the ink font. Examining the experiment videos showed that those subjects who disliked the ink font had handwriting that differed significantly from the ink font we used.

For Large Offset, 33% of the comments were positive. Subjects noted that they like the offset nature of the technique and that it was easy to fix errors when using it. 12% of the comments on this technique were neutral. Subjects either noted it was "usable" or "okay" indicating there was no real preference for the technique. The remaining comments (55%) were negative toward Large. The majority of these comments focused on the font being too large or the technique taking up too much space. Subject also commented that the large font size made the technique distracting.

For Small Offset, 77.5% of the comments were positive. A majority of the comments noted the efficient use of screen space with this technique and several subjects liked the small font. A smaller font for displaying recognition results uses less screen space, so subjects appear to like the *result* of having a smaller font rather than having the smaller font itself. Several subjects also said the technique was good in general terms. Of the negative comments (22.5%), there were subjects who did not like the small font, saying it was legible, but not as easy to read as Large.

Finally, when asked to comment on the real-time feedback from the techniques, approximately 33% of the subjects felt the feedback was useful, readable, and not distracting. However, the majority of the subjects felt that, in many cases, the feedback was distracting at times and that the real-time approach is useful depending on what technique is employed. Given the distribution of the results, real-time feedback seems to have both advantages and disadvantages.

6 Discussion

The results of our experiment suggest that a number of the conjectures made on the effectiveness and tradeoffs of each recognition visualization technique are correct. For Typeset in Place, the technique was, indeed, the slowest one to use, and subjects did find it to be distracting, which caused high levels of frustration. However, subjects did not find the technique to be any more readable than Large Offset, for example. In addition, contrary to our original conjecture, Typeset in Place was not preferred over Large and Small Offset when dealing with multiple and long expressions. We expect this is due to subjects taking so long to complete recognition

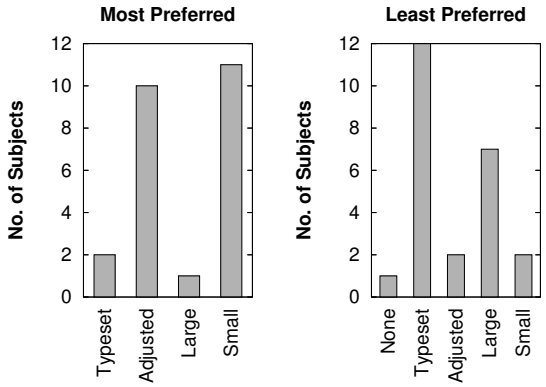


Figure 7: The recognition visualization technique subjects liked and disliked the most. Subjects preferred Adjusted and Small the most and disliked Typeset and Large the most.

tasks using Typeset In Place. Despite all of these problems, our intuition is that there is some creative change that can be made that would lessen the impact of recognition errors.

For Large Offset, the technique was, as expected, ranked as highly readable and highly distracting to subjects. Even with this high level of distraction, subjects did find the technique to be more helpful in avoiding and correcting errors than Typeset in Place. This indicates that distraction level may have contributed more to subjects overall technique preference.

Although Adjusted Ink shows many benefits over Typeset in Place and Large Offset in the figures, it shows no benefits over Small Offset, and is likely to be worse in some cases, due to the ambiguity of its parse feedback. There may be benefits to its aesthetic look and feel, but from a functional perspective we believe Small Offset is a better choice.

Small Offset performed worse with long expressions, consistent with our expectations, which can be partly attributed to the need to scroll those expressions to see the typeset output. However, as you can see in Figure 6, there are many other benefits to this technique which outweigh that problem in general. Small Offset can be improved even further by changing it to always be on the screen, despite scrolling, and using a larger font size for readability.

The trends visible in our Results section are consistent, with one exception. Together, they indicate a clear overall trend that ranks Small Offset and Adjusted Ink as the best techniques and Large Offset and Typeset in Place as the worst techniques. The one exception to this trend is that Large Offset seems to perform disproportionately well and Small Offset disproportionately poorly with Long expressions.

7 Future Work

Since nearly half the subjects preferred Adjusted Ink the most, and because we still observed people missing errors using Small Offset, such as a z substituted for a 2 or a sloppy exponent interpreted as a sibling, we expect improvements are still possible. For instance, combining Small Offset with Adjusted Ink , or further improvement to Adjusted Ink, might result in a consensus best technique.

Rather than just studying the visualization techniques in isolation, we believe it will be valuable to gain a deeper understanding of the interaction between editing and visualization techniques. For instance, many people instinctively try to edit the typeset feedback rather than the ink, even when they already know the system doesn't allow that. In addition, there are further opportunities to enhance visualizations to cover semantic concepts rather than just syntax. There may be ways to make Typeset in Place work better or find contexts where it already works better; Thimbleby [11] presents a calculator using a technique very similar to Typeset in Place but which to us seems to perform acceptably in the limited context the calculator provides.

8 Conclusion

We have presented an experimental study on four different recognition visualization techniques, Typeset in Place, Adjusted Ink, Large Offset, and Small Offset. The goal of this study was to determine how effective each technique is in assisting users in identifying and correcting recognition mistakes under different types and quantities of mathematical expressions. Using task completion time and preference information from a post-questionnaire, we found that subjects generally prefer Adjusted Ink or the Small Offset technique and are significantly faster at viewing and correcting errors with them than with Typeset in Place. However, we believe that there is opportunity to do even better by further enhancing Adjusted Ink or combining it with Small Offset.

9 Acknowledgements

This work was sponsored in part by grants from Microsoft Research and IARPA.

References

[1] Blackwell F. W. and R.H. Anderson. An On-Line Symbolic Mathematics System Using Hand-Printed Two-Dimensional Notation. In *Proceedings of the 1969 24th national conference*, ACM Press, 551–557, 1969.

[2] Chan, Kam-Fai and Dit-Yan Yeung. Mathematical Expression Recognition: A Survey. In *International Journal on Document Analysis and Recognition*, 3(1):3-15, 2000.

[3] Goldberg D. and A. Goodman. Stylus user interfaces for manipulating text. In *Symposium on User Interface Software and Technology*, ACM, 127–135, 1991.

[4] Holm, S. A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics*, 6:60–65, 1979.

[5] Igarashi T., S. Matsuoka, S. Kawachiya, and H. Tanaka. Interactive Beautification: A Technique for Rapid Geometric Design. In *Symposium on User Interface Software and Technology*, ACM, 105–114, 1997.

[6] LaViola Jr., J.J. An Initial evaluation of a Pen-Based Tool for Creating Dynamic Mathematical Illustrations. In *the Proceedings of the Eurographics Workshop on Sketch-Based Interfaces and Modeling*, 157–164, 2006.

[7] Martin W. A. Computer Input/Output of Mathematical Expressions. In *SYMSAC '71: Proceedings of the Second ACM Symposium on Symbolic and Algebraic Manipulation*, ACM Press, 78–89, 1971.

[8] Shilman M., D. S. Tan, and P. Simard. CueTIP: a mixed-initiative interface for correcting handwriting errors. In *Symposium on User Interface Software and Technology*, ACM, 323–332, 2006.

[9] Smirnova E. and S.M. Watt. *A Pen-Based Mathematical Environment Mathink*. Research Report TR-06-05, Ontario Research Centre for Computer Algebra, University of Western Ontario, 2006.

[10] Smithies S., K. Novins, and J. Arvo. A Handwriting-Based Equation Editor. In *Graphics Interface 1999*, 84–91, 1999.

[11] Thimbleby W. A novel pen-based calculator and its evaluation. In *NordiCHI '04: Proceedings of the third Nordic conference on Human-computer interaction*, 445–448, 2004.

[12] Wais P., A. Wolin and C. Alvarado. Designing a Sketch Recognition Front-End: User Perception of Interface Elements. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling*, 2007.

[13] Zanibbi R., K. Novins, and J. Arvo. Aiding Manipulation of Handwritten Mathematical Expressions Through Style-Preserving Morphs. In *Graphics Interface 2001*, Canadian Information Processing Society, 127–134, 2001.

[14] Zeleznik, R.C., T.S. Miller, and C. Li. Designing UI Techniques for Handwritten Mathematics, In *Eurographics Workshop on Sketch-Based Interfaces and Modeling*, Riverside, CA, August 2007.