5-2021

Introducing Handwriting into a Multimodal LATEX Formula Editor

Yancarlos Diaz
yxd3549@rit.edu

Introducing Handwriting into a Multimodal $\LaTeX$ Formula Editor

by

Yancarlos Diaz

A thesis submitted in partial fulfillment of the
requirements for the degree of
Master of Science
in Computer Science

B. Thomas Golisano College of Computing and
Information Sciences
Rochester Institute of Technology

May 2021

MASTER'S IN COMPUTER SCIENCE
ROCHESTER INSTITUTE OF TECHNOLOGY
ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

MS DEGREE THESIS

The MS degree thesis of Yancarlos Diaz
has been examined and approved by the
thesis committee as satisfactory for the
thesis required for the
MS degree in Computer Science

Dr. Richard Zanibbi, Advisor

Dr. Joe Geigel, Reader

Dr. Reynold Bailey, Observer

Date

Introducing Handwriting into a Multimodal LaTeX Formula Editor

by

Yancarlos Diaz

Submitted to the
B. Thomas Golisano College of Computing and Information Sciences
Department of Computer Science
in partial fulfillment of the requirements for the
Master of Science Degree
at the Rochester Institute of Technology

Abstract

Handwriting has been shown to be a useful input modality for math. However, math recognizers are imperfect, especially when recognizing complex expressions. Instead of improving the recognizer itself, we explore ways to best visualize the recognizer's output to help the user fix recognition mistakes more efficiently. To do this, we propose changes to the visual editing operations in MathDeck, a math-aware search engine and formula editor, as well as the addition of an n-best list of results for each symbol in the recognizer's output. We present two experiments to help us find good ways to help users fix errors in the recognizer, and to test whether these changes help novices input formulas more efficiently than they would if they did not have handwriting as an input modality. In the first experiment, users had the option to fix errors with an in-place drop-down menu of alternate symbols, a side symbol correction panel, or by typing the symbols themselves or dragging them from a symbol palette. In our experiment, most users preferred to fix the errors manually by typing the correct symbols or using the symbol palette. In the second experiment, participants entered formulas using handwriting and/or LaTeX. We found evidence that suggests that novices can input formulas faster when they have access to handwriting, but experts still do better when they can just type LaTeX.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Everyone first learns to write math on a piece of paper. Writing on paper is fast and it feels natural. However, many math-aware search engines use LaTeX or MathML[1](like WolframAlpha and SearchOnMath). This requires people to learn a different language to represent things they know how to represent in writing. Learning these languages takes time and writing formulas using these languages requires a bigger cognitive effort than writing in a language you're comfortable with [27]. We want to introduce handwriting recognition to the MathDeck interface ([19]) to allow novice LaTeX users to write expressions intuitively and turn them into a language that these math-aware search engines can understand.

The current MathDeck interface (shown in Figure 1.1) allows users to input formulas by typing LaTeX, drawing on the canvas, or by uploading an image of the formula they want to recognize.[2] Once a formula is entered, users can change the LaTeX string and the version of the formula on the canvas will change accordingly. Users can also draw additional strokes on the canvas and have the recognizer interpret the whole formula, including the new strokes. However, this approach often results in the formula being misrecognized since the recognizer is not optimized to recognize a mix of handwritten and typeset symbols. Furthermore, like most recognizers, the recognizer used in MathDeck struggles to recognize complex formulas.

[1]MathML is a specification for math content on the web. `https://www.w3.org/Math/`

[2]At the time of this writing, this version of the interface is live at `http://mathdeck.cs.rit.edu/`

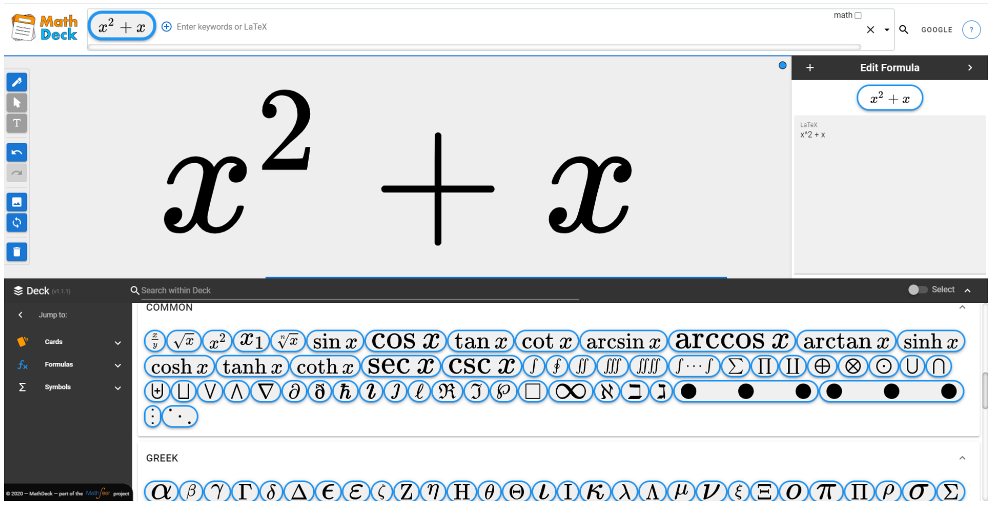


Figure 1.1: A screenshot of the current MathDeck interface. The interface uses formula chips that users can download, edit, reuse, and insert into other formulas. Users can input formulas by typing LaTeX, drawing on the canvas, or by uploading an image of the formula they want to recognize.

Instead of recognizing big formulas directly, one possible solution is to recognize small pieces and put them together. MathDeck allows users to make and reuse formula chips throughout the interface. For example, users can highlight the second x in the LaTeX string and they can click on the chip with $\sqrt{x}$ in the symbol palette at the bottom to replace x and end up with $x^2+\sqrt{x}$. This is relatively easy to do, but it becomes more complex if you want to replace a subscript, or the bounds of an integral, especially if you don't know LaTeX. Nishizawa et al. added visual operations to the MathDeck interface, allowing users to change formulas without having to directly manipulate the LaTeX string [17]. These operations are described in Chapter 3, but it suffices to know that these operations allow the user to simply drag the formula chip with $\sqrt{x}$ directly to the top-right corner of the x to add $\sqrt{x}$ as an exponent and end up with $x^2 + x^{\sqrt{x}}$ instead. Other operations include the ability to select, copy, move, or delete sub-expression on the canvas. Nishizawa et al. did not find that the operations help the user input formulas faster, but they found that it does not make them any slower when compared to text-only input, and users reported finding some of the operations useful. In particular, the delete operation seemed to help users delete sub-expressions faster than if they had to edit the LaTeX string.

The visual operations help users edit formulas even if they don't understand the LaTeX, but they still do not solve the issues we described above regarding handwriting recognition. We propose some changes to the MathDeck interface to make it easier for users to input formulas using handwriting. Before we describe those changes in Chapter 3, we discuss why we think handwriting can be useful in Section 1.1 and discuss the specific questions we want to answer in Section 1.2.

1.1 Why Recognition of Handwritten Math is Useful

Anthony et al. have looked at the benefits of handwriting recognition of math in the context of intelligent tutoring systems for students learning algebra [1] [3] [2]. They have found that handwriting input, although imperfect, can provide benefits to students learning math. One of their studies looked at middle and high school students (ages 11 to 17) solving algebra problems in three modalities: typing, handwriting, and handwriting-plus-speaking. For this last modality, students were asked to speak aloud as they solved problems. The study found that students can solve the same equations and learn the same amount in half the time using handwriting input vs typing input. Many others have looked at the positive effects that pen-based devices can have in learning [9] [20].

Handwriting recognition of math can also be useful for teachers, even if their students do not use it directly. For example, Gross et al. introduced a system to scan and recognize students' work [10]. The system could then automatically grade the students' work and provide the teachers with summaries to assess class performance. Mendes et al. and others present handwriting systems that could be beneficial to professors teaching a lecture [16] [28]. Mendes et al. reported that when asked if their system's features helped them teach mathematics, 62.5% of teachers agreed, and 37.5% strongly agreed. 85% of the students reported that the features helped them understand the mathematical manipulations involved. One reason why handwriting recognition of math is not widely adopted by teachers is that recognizers are not perfect and may slow down teachers if they can't recover quickly.

Handwriting has proven efficient in other fields too. Subramonyam et al. showed that handwritten annotations in digital systems can help people learn more from complex text [26]. In their system, users can highlight important phrases and create diagrams from the words that the system identifies as key causal terms in the text. Similarly, Rodriguez et al. showed that digital annotations can also be useful for real-time document editing where users can, for example, draw a line between two words to merge them or in the middle of a word to split it into two words [21]. Handwriting recognition has been used even in course-of-action military diagrams as a way to allow commanders to draw symbols that are specific to this domain on a map shown in a tablet display to depict battle scenarios [12] [13].

1.2 Questions and Contributions

We would like to answer the following questions regarding handwriting recognition of math:

1. What are good ways to allow users to easily fix errors in a recognizer's output?
2. Can novice LaTeX users input formulas faster when they have a handwriting recognizer available?

Our work will contribute to the field of math input in a few ways. We're looking to discover ways to make math handwriting recognizers useful in real-life applications by acknowledging their weaknesses and working with them. **Once we've done that, we hope to find evidence indicating that adding handwriting operations into MathDeck can improve efficiency for novices.** This would be of great help for high school and college students who don't know LaTeX but need to enter and/or search for math content.

To test this hypothesis we run two experiments using a modified version of the MathDeck interface. For the first experiment, we look at effective ways to help users correct issues in the recognizer's output and find that most users prefer to manually fix the errors by typing the correct symbols or dragging them from a symbol palette. The second experiment tries to answer the question of whether handwriting can help novices input formulas faster and/or only require a small number of editing operations to correct the recognizer's errors. We find this to be true for novices, but experts seemed to input formulas faster if they're simply allowed to type LaTeX than if they're forced to use handwriting.

In the next chapters, we describe the work leading up to our experiments and conclusions: To learn from related work, we look at other math interfaces and intelligent systems in other domains in Chapter 2. Based on lessons learned from others' work, we propose changes to the MathDeck interface and describe those changes in Chapter 3. We describe our experiments to test the effectiveness of our changes and to answer the questions posed above in Chapter 4. We conclude by talking about our findings and future work in Chapter 5.

Chapter 2

Background

Others have studied ways to help users write mathematical expressions more easily. Many existing math interfaces use handwriting as an input modality alongside text. Each interface uses a slightly different paradigm; some are closer to writing on paper, while others pursue something different. Naturally, these interfaces have different ways of dealing with recognition errors. This chapter describes and compares these interfaces. Additionally, we look at intelligent interfaces outside of the math domain to learn about things that have not been explored in the math domain. We use our findings to make decisions about how to best present recognition results and help users fix recognition errors.

This chapter is split into 3 sections. Section 2.1 talks about different interfaces that use handwriting as an input modality, including MathDeck, the interface this work will build on. Section 2.2 focuses on how some of those interfaces handle handwriting recognition errors and how recognition errors are handled in other domains. Section 2.3 discusses what others have found about showing recognition confidence to aid users outside of the math domain.

2.1 Interfaces With Handwriting Recognition of Math

Many other interfaces use handwriting as a way to input mathematical expressions. Some interfaces try to make the user feel like they're truly writing on a piece of paper, while others push a completely different analogy that may help users beyond what pen-and-paper can provide.

Zeleznik et al. fully push the pen-and-paper analogy to a screen [36]. Hands-On-Math is a multi-touch, pen-based system that aims to provide the benefits of Computer Algebra Systems while maintaining the flexible and free-form input of pencil and paper. Their results found that users were not particularly receptive of the bi-manual interaction required to operate the system. Additionally, the system often failed to recognize the difference between math notations from diagrams and free-form annotations. Nevertheless, users still reported that the system has "great potential" to help them do math-related work. Similarly, Zhelezniakov et al. [37] presented SMath, a pen-centric user interface with interactive input and automatic calculation of handwritten math. In this application, the user can draw small sub-expressions and assign the result to a variable to allow for quick reuse. In this way, it helps provides the pen-and-paper feel, while also taking advantage of automatic calculations that the system offers.

Other interfaces use very different design metaphors. Taranta et al. introduce Math Boxes, a pen-based user interface with the goal of making the task of writing difficult math expressions simpler [29]. As its name implies, Math Boxes draws bounding boxes around sub-expressions as the system detects relationships between the sub-expressions. MathDeck, on the other hand, stores formulas as "chips" that can be reused and manipulated individually [19] [8]. Users can input formulas by typing LaTeX, drawing on the canvas, or by uploading an image of the formula they want to recognize. The interface has a deck that provides the user with a collection of commonly used expressions as chips, as well as a list of the formulas the user has created in the past. The deck also contains a list of 'cards' that are made up of a formula chip, a title, and a description. As the user inputs formulas, the interface searches for related concepts and returns them in the form of cards. Users can also make their own cards.

MathDeck's main limitation regarding handwriting is that it's hard for users to combine handwriting with typeset formulas. For example, if the user has the expression $\sqrt{x}$ and wants to change it to $\sqrt{x^2}$ by drawing a two on top of the x, they will be disappointed to find that the recognizer fails to recognize the two as an exponent, and actually throws off the whole expression (See Figure 2.1 for sample result). Chapter 3 describes the steps we're taking to try to address this issue.

MathDeck's chips can be seen as a combination of the variables created

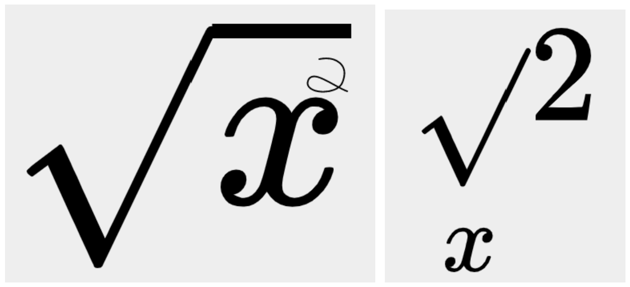


Figure 2.1: One limitation of MathDeck's handwriting recognizer is that it is hard for the recognizer to combine typeset formulas with handwriting. The image on the left shows $\sqrt{x}$ in typeset and a 2 drawn on top of the x. When the recognizer is triggered, it returns the result of the right, which is not what would be expected.

in [37] as well as the boxes in [29]; they're automatically generated and saved in the user's history, and they can be reused as sub-expressions to build bigger expressions. The use of small and reusable mathematical expressions is so common because allowing users to work in small pieces is likely to help them reduce the number of recognition mistakes they have to deal with. Some interfaces provide the user with common symbols to utilize *instead* of handwriting [7]. MathDeck provides the same common symbols but it also allows the user to create new, reusable expressions using text or handwriting.

One thing that all these interfaces have in common is that their handwriting uses in-place ink that displays strokes right where the user draws them on the screen. The alternative to in-place ink is indirect writing, which makes the user draw in a separate window from where their drawing is shown. Gu et al. showed that users are more effective when using in-place ink [11], and we see that most interfaces follow this mode of input.

Something to note about these interfaces is that they mainly target non-disabled users. In fact, many of these interfaces are very hard to use for blind users or users with motor skill disabilities in particular. Attanayake et al. present TalkMaths, an interface that uses speech-to-text to create mathematical expressions as a replacement for typed input [5]. They also introduced SWIMS, a system that assists users by predicting what will appear next and identifying errors as the user interacts with the interface. While these are fea-

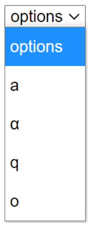


Figure 2.2: One common way to quickly fix symbol misrecognition is to give the user a drop-down of options to choose from based on the recognizer's output

tures that could improve MathDeck's power, we choose to not focus on them as they would increase the complexity of the interface tremendously.

A common theme found in most of these interfaces is that they encourage users to build large expressions one piece at a time [29] [19] [37] [5]. We propose to also add a tab system analogous to web browser tabs to MathDeck to encourage users to work on different sub-expressions in parallel. Another important feature in many of these interfaces is the ability to reuse components, whether it's common symbols such as the Greek letters and relational symbols ([5]) or symbols and expressions created by the user ([19] [7]). However, many of these interfaces do not handle input misrecognition very well as seen in MathDeck and as reported from Hands-on-Math. The next section looks closely at ways others have chosen to handle these recognition errors.

2.2 Fixing Recognition Errors

Handwriting recognizers are imperfect, and they tend to be less effective as the complexity of our input increases. Many have looked at ways to help users spot and fix recognition mistakes easily. When the top result is incorrect, interfaces tend to help users fix the errors in one of two ways. One way is to simply give the user the freedom to change anything they want to change. Another approach is to use the recognizer's n-best results as suggestions for what the user actually meant to write.

Smithies et al. propose an interface that allows users to delete, group, or separate symbols with gestures after their handwritten input has been recognized [24]. For every symbol, it also provides an n-best list for individual symbols that users can use to replace the recognition output. Tran Minh Khuong et al. propose an interface that also allows for correcting symbols by replacing them from a list of n-best results [31]. This is a common approach that assumes that providing the user with a list of likely results will help them fix errors quicker. Figure 2.2 shows how this list of n-best results is often displayed to the user.

Another obvious solution to this problem is to improve the recognizers themselves, but that is easier said than done. Many have even tried to use multimodal input to increase the accuracy of handwriting recognizers [4] [34]. In most of these studies, users have to speak out loud as they draw to give the recognizer another way to guess what the user is trying to input. While their experiments show promising results, relying on a second modality like sound is not always possible or practical, especially in loud places or in situations where users don't have access to a microphone. Schrapel et al. took a slightly different approach to this multimodal input. Instead of using the user's voice, they make use of a special pen that uses the pen's tip motion and sound emission when stroking to identify the digits the user is drawing [22]. The main limitation to this approach is the need for a special type of pen.

We are a long way from speech or handwriting recognizers that are close to perfect. For this reason, assisted error correction is a well-studied topic in other domains such as speech-to-text systems [30] [33]. These studies have shown that providing a list of n-best results helps users arrive at their intended input faster than if they did not have the list.

Stedman et al. showed that having the original input near the recognized output can help users spot and correct errors faster in the context of form filling [25]. Zanibbi et al. looked at a similar problem in the math domain, [35]. They showed that if users are presented with a style-preserving morph of their original strokes instead of the typeset equivalent of the expression, users identify errors in their input more easily and they can enter expressions just as quickly as if the strokes were turned into their typeset equivalent instead.

In most domains, showing a list of n-best recognition results has proven to help users. In speech-to-text, it is common to show a list of n-best results for every word as well as for the whole sentence [30] [33]. Some math interfaces take the same approach by providing a list of n-best symbols as well as different relationships between the symbols [24] [31]. Our interface will provide the user

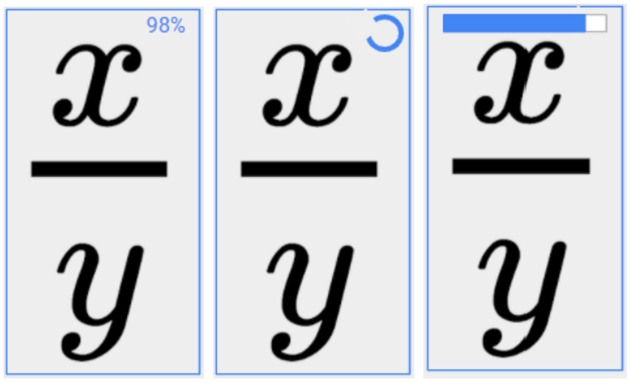


Figure 2.3: Three ways to show recognition confidence.

with an n-best list of recognition results for their whole expression as well as for every individual symbol. If individual symbols are misrecognized, users can choose from the list of n-best symbols, or they can type the correct symbol.

2.3 Showing Recognition Result Confidence

Some believe that one possible way to help users fix mistakes quicker is to show them how confident the system is in its output. Outside of the math domain, researchers have looked at whether showing confidence actually helps users. Some studies have found that the way in which confidence is displayed plays an important role in whether it's useful or simply distracting. Furthermore, others have found that in order for this approach to be useful, the confidence measure must be accurate. In other words, low confidence must actually indicate that the results are likely wrong and high confidence must indicate that results are likely correct.

In the context of search engines, Shani et al. studied the effect of showing confidence values in search results [23]. They found that when users were given a graphical representation of the confidence (like a loading bar), users were more likely to explore more results. Figure 2.3 shows different ways in which we could display recognition confidence in our interface.

However, in the context of automatic captioning, Berke et al. showed that visually displaying confidence in captions in various forms did not directly help users get more information from the captions [6]. This is perhaps because changing the way captions are displayed based on the system's confidence can be distracting to users. To complement this finding, Shani et al. also found that displaying the confidence as a number did not help users as much as the graphical representation. This seems to indicate that showing confidence can sometimes be helpful, but one has to be careful about the way it's displayed and whether it helps or simply distracts the user.

Vertanen et al. Looked at showing confidence in the result of a speech-to-text system. The system marked low confidence areas by underlining them in red [32]. They found that when the system's confidence is accurate, it can help users fix mistakes quicker. However, when the system's confidence isn't an accurate representation of the actual errors, it actually slows down users when compared to users who didn't have the visual indications at all.

Some studies have found that displaying the recognizer's confidence can help users, but others have found that it doesn't. What we know for sure is that the accuracy of the confidence value plays an important role in how useful it can be. These mixed findings have led us to avoid using recognition confidence as a feature to help users fix recognition mistakes. Instead, we simply list our recognition results from most confident to least confident. This allows users to focus on the top, most likely correct results without getting distracted by how confident our recognizer is.

2.4 Summary

In Section 2.1 we talked about interfaces with handwriting input and features that aim to help users input expressions easily. Some interfaces push a pen-and-paper analogy, while others use different design paradigms such as chips in MathDeck and boxes in MathBoxes. Most of these interfaces have a way to reuse components to prevent the user from having to do the same thing more than once.

Section 2.2 explored ways to help fix recognition errors both in math and in other domains. We find that an n-best list of results can help the user arrive at their intended input faster than they would if they didn't have one. Most of the time, these lists are presented in a drop-down menu.

In Section 2.3 we looked at showing recognition confidence as a way to help users fix their mistakes. Studies in other domains show conflicting results;

some find confidence useful, while others find it distracting.

These findings have helped us make decisions about the new design of the MathDeck interface. The next chapter describes the state of the MathDeck interface after Nishizawa's work in [17] and our proposed changes to the interface.

Chapter 3

Methodology

We hypothesize that novices will be able to input formulas faster in MathDeck if they have access to handwriting. To test this hypothesis, we made considerable changes to the MathDeck interface. Those changes include both small modifications to the structure operations and the interface layout, as well as the re-introduction of handwriting to the interface. In this chapter, we describe our design goals and the changes we made to achieve those goals.

3.1 Design Goals: Editing, Handwriting, and Error Correction

We had four design goals in producing the new formula editor for MathDeck:

1. Building Large Formulas from Smaller Ones: As formulas get larger and more complex, it becomes harder to edit them. Users can become lost in a long LaTeX string when they are perhaps just looking to edit a small part of a formula. A possible solution to this problem is helping users build formulas in pieces. This requires an interface that allows the user to easily define, use, and combine those pieces. We attempt to address this in MathDeck with formula chips and tabs

2. Direct Manipulation of Rendered Formulas: In existing LaTeX formula editors, there is a disconnect between the LaTeX formula in text and the formula appearance, which is presented separately from the LaTeX string.

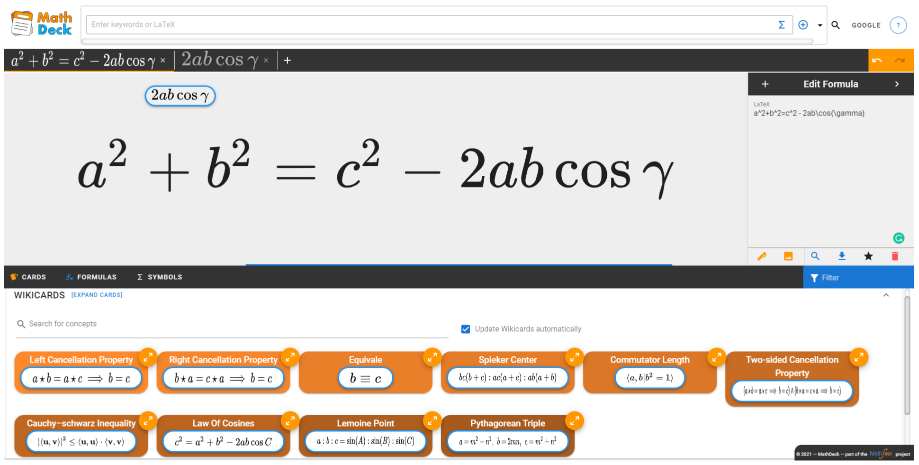


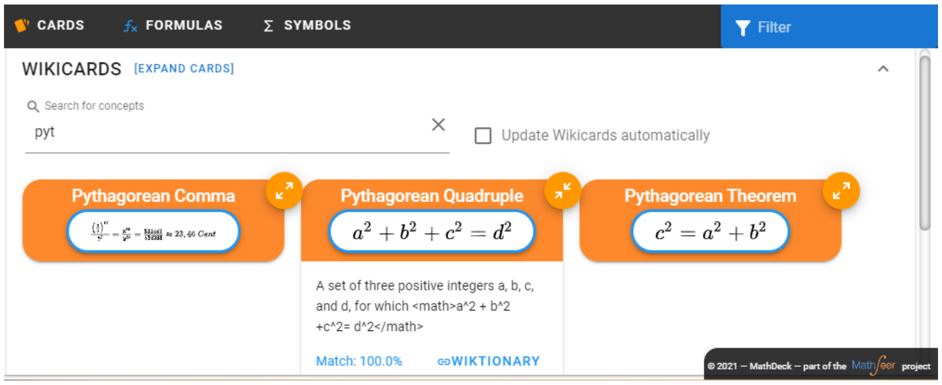
Figure 3.1: The new MathDeck Search Interface.

If users see they have an error in the rendered version of the formula, they must search the LaTeX string for the location of the error. It would be easier if they could change the rendered formula image directly. We use and improve the structure editor introduced by Nishizawa et. al. to alleviate this problem [18].

3. Simple Handwriting in Context: An issue with handwriting in previous versions of the MathDeck interface was that it was not easy for the user to quickly use handwriting to write a sub-expression; they had to juggle between a "draw" mode and a "select" mode. We have reworked the interface to remove the modes and allow users to easily insert handwritten formulas in context.

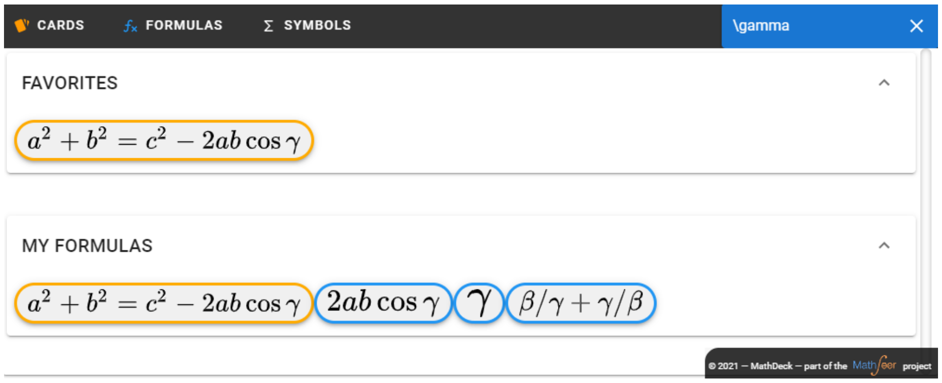
4. Intuitive Error Correction: Since the recognizer will often make mistakes, users must be able to correct errors with ease. We have implemented two new correction mechanisms that use the n-best lists of results from the recognizer to help users get to their desired formulas quickly. Users can fix misrecognized symbols with these new mechanisms and they can correct the structure of the formula using the visual operations.

The next sections outline the existing features and the changes we made to the interface to achieve our design goals.



A set of three positive integers a, b, c, and d, for which <math>a^2 + b^2 +c^2= d^2</math>

(a) Wikicards tab, showing titles starting with 'pyt'


CARDS
FORMULAS
SYMBOLS
\gamma
FAVORITES
$a^2 + b^2 = c^2 - 2ab\cos\gamma$
MY FORMULAS
$a^2 + b^2 = c^2 - 2ab\cos\gamma$
$2ab\cos\gamma$
γ
$\beta/\gamma + \gamma/\beta$
© 2021 – MathDeck – part of the MathSeer project

(b) Formulas tab, showing chips containing gamma

Figure 3.2: The Deck. (a) Shows the Wikicards tab, while (b) shows the tab containing formulas entered and favorited by the user. A third tab provides a symbol palette with chips for symbols and common subexpressions. Users may filter tabs: in (a) cards with a title beginning with 'pyt' are shown, and in (b) formulas containing 'γ' are shown.

3.2 Working in Small pieces

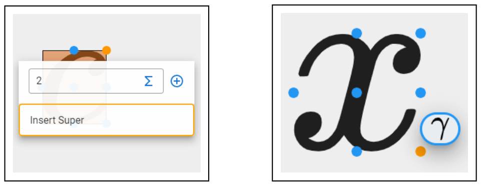
MathDeck provides two ways for users to work in small pieces and then combine them. The first is through formula 'chips.' As the user enters formulas, the interface saves them as chips that can be moved around the interface. Multiple chips may sit on top of the formula editing canvas for later use (see Fig. 3.1), and appear in the deck at the bottom of the interface. Combined with the direct formula image manipulation the interface provides, chips may be used to build big expressions by dragging the chip onto control points located around symbols on the formula canvas, or into the editing panel for the full LaTeX string.

Users can favorite formula chips they use often, to make them more readily accessible (see Fig. 3.2), and chips can be downloaded as jpeg images for later use, and to share with others. Dragging a chip image file onto MathDeck restores the formula chip, as all chip information is serialized in metadata fields of the formula jpeg file, including the associated LaTeX string. All chips have a context menu, allowing them to be duplicated, inserted in the LaTeX panel, copied as a chip onto the canvas, and deleted. Additionally, all formula chips may be searched using the currently selected search engine.

The second way MathDeck helps users work in small pieces is the new tab system. Tabs in MathDeck are analogous to browser tabs - each formula currently being edited has a different tab, and the formula in the currently selected tab appears on the editing canvas (see Fig. 3.1). Users can create as many tabs as they like, and they can move between them to quickly change context and work on a different formula. As the user switches between tabs, the chips on top of the canvas remain in place. Additionally, users can directly drag a tab onto the canvas or onto the editing panel to insert the contents of the tab at the specified location, as they can do with formula chips. Fig. 3.1 shows the MathDeck interface with two tabs. The first tab contains the formula '$a^2 + b^2 = c^2 - 2ab\cos\gamma$' and the second one contains the formula '$2ab\cos\gamma$.' The first tab is actually the result of dragging the second tab onto the canvas when it contained only the formula '$a^2 + b^2 = c^2 -$.'

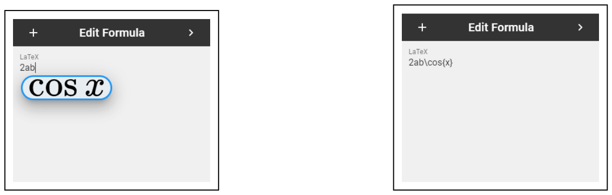
3.3 Structure Operations

Nishizawa et al. created operations to directly edit the rendered formula in place, reducing the disconnect between the LaTeX string for a formula and the formula's appearance [18]. Operations include the ability to select subexpressions of formulas and view, edit, delete, move and copy into a new formula chip ('lift'), or replace the selection with a formula chip or a formula entered as LaTeX.



(a) LaTeX insertion (b) Chip insertion

Figure 3.3: Inserting Formulas with Visual Operations. (a) Inserting '2' as an exponent to 'c' using a LaTeX string. (b) Inserting 'γ' in a chip as subscript of x.



(a) Chip over LaTeX panel (b) After dropping chip in (a)

Figure 3.4: Inserting Formulas in the Editing Panel. (a) A chip containing '$\cos x$' hovering over the LaTeX panel (at right in Fig. 3.1) - the cursor is at the end of the LaTeX string. (b) After dropping '$\cos x$' on the panel, appending '$\cos x$' at the cursor location.

Rendered formula images may be edited directly by using control points around symbols and selections to insert formulas provided as LaTeX strings or in formula chips and tabs (see Fig. 3.3). In Fig. 3.3 part (a) we see LaTeX being entered after clicking a blue dot on the canvas (i.e., in place). As the user types the formula, they can preview the rendered version. If LaTeX is invalid, this is indicated during typing by turning the text box border red. As shown in Fig. 3.3 part (b), users can drag a formula chip to any of the control points on the canvas to insert the formula at the position indicated. They can also drag the chip to the center point to replace a sub-expression. Chips and tabs can also be used to insert LaTeX in the LaTeX panel of the interface (Figure 3.4).

The structure operations introduced by Nishizawa et al. proved to be an effective way to edit rendered formulas. But one clear visual operation that was missing was the ability to replace all instances of a particular symbol. For example, if we have the formula $p^a(\vec{x})p^b(\vec{y})p^c(\vec{x})$, the user may want to

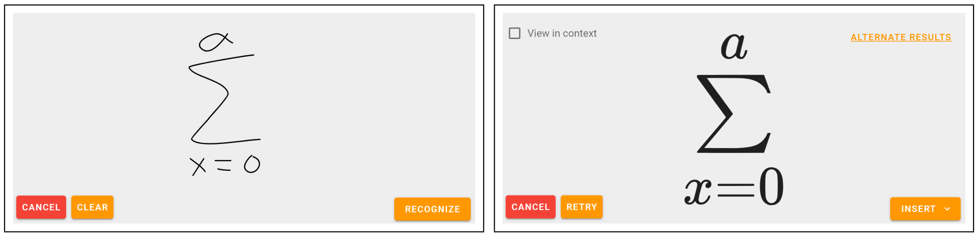


Figure 3.5: The new handwriting recognition window.

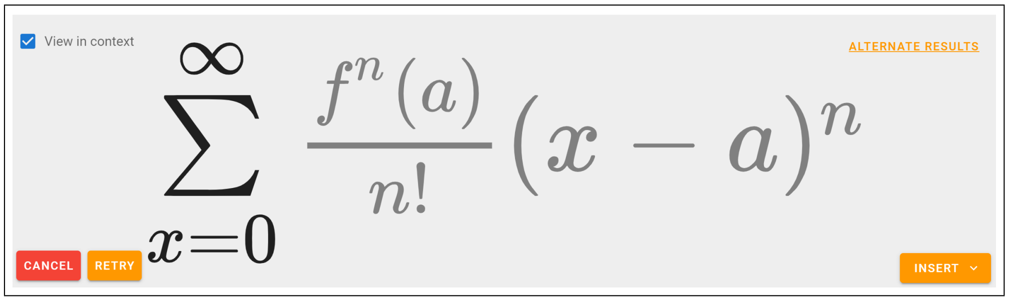


Figure 3.6: Viewing result in-context, before inserting in LaTeX panel. Corrections can be made if needed.

replace all the Ps by the Greek letter ρ. They can now do that by clicking on 'p', typing in \rho and hitting the "Replace All" button, which would result in the formula $\rho^a(\vec{x})\rho^b(\vec{y})\rho^c(\vec{x})$. We have added this operation along with a keyboard shortcut to delete symbols and sub-expressions using the Backspace key.

In the context of handwriting, these operations can be used to quickly replace symbols that the recognizer fails to recognize correctly in multiple parts of the formula. The next section describes the new handwriting interface and how it interacts with the existing features.

3.4 Handwritten Formulas

We have added a new handwriting interface to MathDeck. This new interface uses a new version of the QDGGA system presented by Mahdavi et al. [15]. The recognizer itself has faults, but our goal is to overcome those faults by introducing an intuitive interface that allows users to correct recognizer mistakes easily. The details of the implementation are described in the rest of this section.

To insert a new handwritten expression, users can click the pen button in the LaTeX editing panel at the bottom left. Users can then draw on the canvas, as seen in Fig. 3.5. Upon clicking ‘recognize’, the interface replaces strokes with the recognized formula.

Users do not need to enter the entire formula using handwriting. Instead, they can type whatever they feel comfortable with, and then enter the piece they do not know how to write using handwriting. For example, Most users would be able to enter the $(x - a)^n$ in the Taylor series in Fig. 3.6 even if they do not know LaTeX. However, the fraction and the summation might be harder to write.

Assume the user has entered everything but the summation. The user places their cursor at the beginning of the formula’s LaTeX string and presses the pen button. They then draw the summation with its limits, and the recognizer returns its interpretation. Users can see how the complete formula looks if the formula were to be inserted in the LaTeX panel where their cursor was (Fig. 3.6). In this figure, the context appears in gray, and the recognized formula appears in black. If this is the desired result, they can click “insert” to add the recognized sub-expression. If the user wishes to place the recognized formula somewhere else, they can click on the chevron in the insert button to insert the formula in a chip or tab instead. The user would then be able to drag the chip or tab into any control point or to any point in the LaTeX string.

In addition to inserting a handwritten formula, users can highlight some part of the LaTeX string of a formula and then click on the pen button. This allows them to replace the highlighted string with a handwritten formula. Just like how the user can view the context before inserting (as in Fig. 3.6), they can also view the context before replacing the selection.

Inserting handwritten formulas in context allows users to quickly switch between using text and handwriting instead of committing to one input method. For example, the user could input the fraction in the Taylor series using handwriting, they could type $(x - a)^n$, and then choose to input the summation using handwriting. This also addresses the issue of having a recognizer try to recognize a mix of handwritten and typeset symbols, which was an issue in previous versions of MathDeck.

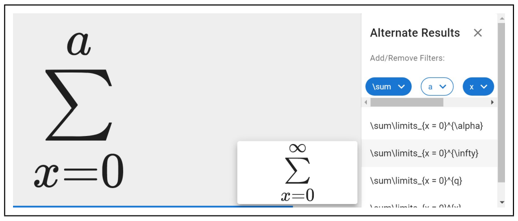


Figure 3.7: The Symbol Correction Panel. Users can filter results by symbol. Alternatives are rendered when the mouse hovers over them.

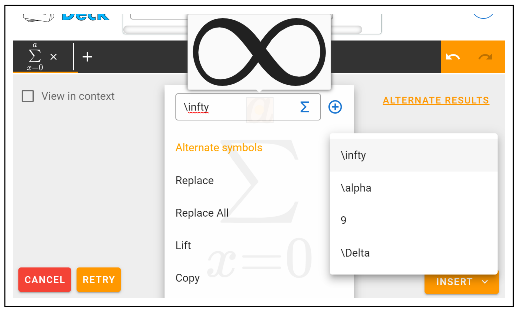


Figure 3.8: Alternate Symbol Drop-down. Alternatives are rendered when the mouse hovers over them.

3.5 Correcting Recognizer Errors

No recognizer is perfect, so users need intuitive ways to correct errors. In previous versions of MathDeck, if the recognizer had errors, users had to correct the errors by either editing symbols and their placement and then attempting to recognize the formula again, or by typing in the correct symbols. This is an issue, as we want to use handwriting to help users who do not know how to write the symbols in LaTeX in the first place.

The new MathDeck interface provides a list of alternative results for the formula, as seen in Fig. 3.7 in the form of what we call the "Symbol Correction Panel". Users can hover over the list of alternatives to see the rendered version of each recognition alternative represented as a LaTeX string. To quickly find the correct result, users can filter the list by the symbols they expect the final formula to have. For example, if the user knows they want a summation symbol, they would click on the \sum to make sure all the results contain that

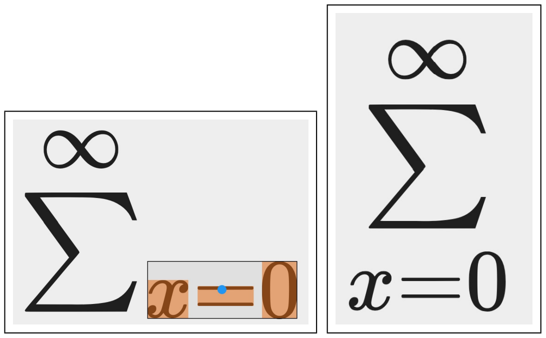


Figure 3.9: Correcting Structure Using Visual Operations. If the recognizer returns the wrong structure like in the left image, users can select sub-expressions and drag them to the correct place

symbol. If they're looking to change the 'a' in the formula, they would leave that filtered unchecked to see all the alternatives to that symbol.

Alternatively, the user could click on one of the symbols that were recognized incorrectly on the canvas to find a list of alternate symbols. In the example shown in Fig. 3.8, the user is trying to replace the 'a' with an infinity symbol. This means that even if the symbol returned is not the correct symbol, there is a high chance the user will be able to find the correct symbol in the list of alternate results, even if they do not know how to type it, provided it is a symbol in the recognizer's set of 101 classes [15]. In the future, we will expand this to a larger set to a larger set like the one used un Detexify[1], a well-known tool for symbol search using handwriting.

If none of the alternatives provided contain the correct result but the user knows how to type it, they can always choose to type it themselves. They can also use any of the structure operations available in MathDeck to correct the structure in the recognizer output. For example, if the recognizer returned the 'x = 0' as a subscript as seen in Fig 3.9, the user could select the subexpression and drag it to the correct control point. They could do this before or after correcting the symbols. The changes in structure will be reflected in the Symbol Correction Panel and they will still be able to use the Alternate Results drop-down for each symbol even after changing the structure of the formula.

[1] `https://detexify.kirelabs.org/classify.html`

Table 3.1: Changes to the editing operations in the MathDeck interface. The left column lists the existing operations. Rows without a left cell indicate new editing features.

PreviousOperations	New Operations
Replace expression visually via LaTeX or formula chip	Replace all instances of a sub-expression
Insert Expression in LaTeX string	Insert the LaTeX string of a handwritten formula.
Replace selected substring	Add ability to replace a LaTeX substring from handwritten formula
	N-best results from the recognizer when inserting handwritten formulas. See Figure 3.8 and Figure 3.7
	Use different tabs to work on formulas separately

This means that users may change both symbols and formula structure flexibly in the correction interface.

3.6 Summary

Table 3.6 shows the summary of changes made to the interface. We add the Replace All visual operations and re-introduced handwriting into the interface. Along with the new handwriting interface, we added the Symbol Correction Panel and the Alternate Symbols drop-down as mechanisms to quickly fix errors in the recognizer. In the next chapter, we describe the experiments we ran to test these new features and how they support our hypotheses.

Chapter 4

Experiments

We ran two experiments to test our hypothesis that novice users will be able to input formulas faster if they have access to handwriting. The first experiment asked users to use handwriting to input small formulas. We looked at how users corrected errors from the recognizer when given different correction mechanisms. The second experiment looked at how participants completed tasks when they had access to handwriting vs when they did not. This chapter describes each experiment's design, protocol, participant pool, and results.

4.1 Experiment 1: Correction Mechanisms

Before we looked at how handwriting can help users input formulas when compared to instances when they don't have access to handwriting, we wanted to make sure our handwriting interface was easy to use, even if the recognizer was imperfect. More specifically, we wanted to study different ways to allow users to fix errors from the recognizer.

We looked at two possible ways to correct errors: using the new Alternate Symbols drop-down shown in Fig 3.8 and using the Symbol Correction Panel that uses faceted search, shown in Fig. 3.7.

4.1.1 Experiment Tasks

To measure which mechanism worked best, we presented users with 18 tasks where they had to enter formulas using handwriting (shown in 4.1). These

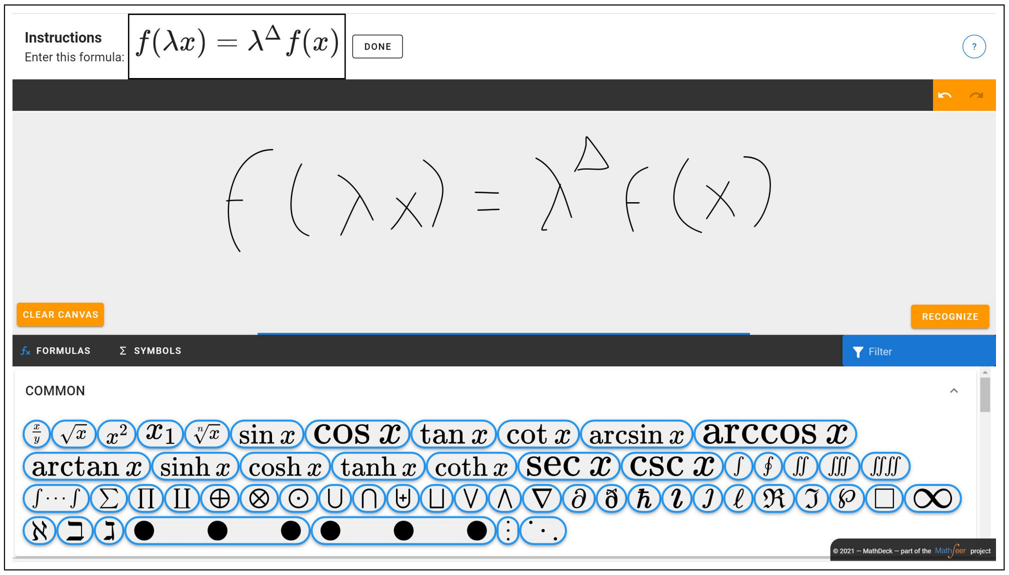

Instructions
Enter this formula:
$f(\lambda x) = \lambda^{\Delta} f(x)$
DONE
CLEAR CANVAS
RECOGNIZE
FORMULAS
SYMBOLS
Filter
COMMON
© 2021 — MathDeck — part of the MathSeer project

Figure 4.1: The interface for the Correction Mechanisms experiments. Users have access to the handwriting window and the Deck.

formulas, which came from Nishizawa's experiment ([18]), contain many special symbols and structures that non-expert mathematicians might have a hard time typing using LaTeX. We chose these tasks because we hoped to see how participants would perform when presented with unfamiliar symbols and structures.

To complete these tasks, users had access to a modified version of the new MathDeck interface (See Fig 4.1.) This version of the interface hides the LaTeX Panel, which encourages users to use handwriting to input the formula. Once they get a result from the recognizer, they have different ways to make corrections which include the Alternate Symbols drop-down, the Symbol Correction panel, and the use of visual operations in combination with the symbol palette in the deck.

Each participant was exposed to 3 different conditions:

1. Symbol only: Participants had access to the Alternate Symbols drop-down but did not have access to the Faceted Symbol Correction panel.

Table 4.1: Correction Mechanisms Experiment Tasks.

Task #	Target Formula
1	$\phi = \frac{1+\sqrt{5}}{2}$
2	$A^{+}AA^{+} = A^{+}$
3	$\rho^{a}(\overrightarrow{x})\rho^{b}(\overrightarrow{y})$
4	$\gamma_1 x_1 + \gamma_2 x_2 + \cdots$
5	$(a \triangleleft b) \triangleleft (a \triangleleft c)$
6	$Q = -k\frac{dT}{dz}$
7	$x = b_0 + \frac{a_1}{b_1}$
8	$F = \frac{B^2 A}{2\mu_0}$
9	$f(\lambda x) = \lambda^{\Delta} f(x)$
10	$\int_{-1}^{1}(1-x)^{\alpha}dx$
11	$R^{i}F(X) = 0 \forall i > 0$
12	$c_0 + c_1 x + \cdots$
13	$a^2 b^2 z^2 - x^4 yc$
14	$\nabla^2 A + k^2 A = 0$
15	$\frac{\partial u}{\partial t} - \frac{\partial u}{\partial x}$
16	$\frac{c}{2\pi\sigma}\nabla U$
17	$\int_{\Omega} f_r(x)$
18	$\frac{x^2}{a^2} = z + \frac{y^2}{b^2}$

2. Faceted only: Participants had access to the Faceted Symbol Correction panel, but did not have access to the Alternate Symbols drop-down

3. Both: Participants had access to both the Alternate Symbols drop-down and the Faceted Symbol Correction panel.

4.1.2 Participants and Blocking Design

We hoped to have at least 30 participants so we invited 40 participants to complete the experiment. Half of those participants are what we considered experts (self-identified as being "Moderately familiar", "Very familiar" or "Extremely familiar" with LaTeX), and the other half were novices (self-identified as being "Not familiar at all" or "Slightly familiar" with LaTeX.) Within those two groups, half of the participants were from RIT's College of Science and the other half were part of the Golisano College of Computing.

To minimize order effects across participants, we changed the order in which participants were presented with the different experiment conditions (Symbol-only, Faceted-only, and Both). The first 6 participants saw the Symbol-only condition, then the Faceted-only, and finally the Both condition. The next 6 participants saw Faceted-only before Symbol-only, but still saw the Both condition last. This pattern is repeated every 6 participants.

We also changed the order in which formulas are presented. We split the 18 tasks into 3 blocks of 6 formulas each. The first block contained formulas 7, 8, 9, 10, 11, and 12, the second block had formulas 13, 14, 15, 16, 17, and 18, and the third block had formulas 1, 2, 3, 4, 5, and 22. We tried to place formulas into blocks so that every block would have about 3 formulas that contained more than one special symbol, and 3 formulas that did not.

We then changed the order of tasks in each of these blocks. That is, the first block for participant 1 may contain tasks 1, 2, 3, 4, 5, and 6, in that order. The second participant's first block would then consist of tasks 2, 3, 4, 5, 6, and 1, in that order. Additionally, we changed the order in which the blocks are presented to the user. The block order for participants 1-6 would be A-B-C, the order for participants 7-12 would be B-C-A, and the order for participants 13-28 would be C-A-B. The block order then repeats starting with participant 19.

Participant Demographics

Table 4.2 shows the level of education, current college at RIT, gender, and age range of the participants who completed the study. We invited 40 participants and tried to have an equal distribution of novices and experts as well as students from the College of Science and students from the College of Computing. 30 participants completed the experiment and completed the exit questionnaire. 3 participants did not complete the experiment but completed the questionnaire; their responses are not considered here. Out of those 30

Table 4.2: Participant Demographics for the Correction Experiment. We aimed to have an even distribution of novices and experts.

Highest Level of School Completed			
	Novices	Experts	Total
High School	6	8	14
Associate's	2	1	3
Bachelor's	5	4	9
Master's	1	2	3
PhD	0	1	1
College			
	Novices	Experts	Total
College of Science	7	9	16
College of Computing	7	7	14
Gender			
	Novices	Experts	Total
Female	6	6	12
Male	6	10	16
Non-binary	1	0	1
Prefer not to say	1	0	1
Age			
	Novices	Experts	Total
18-24	10	10	20
25-34	4	6	10
Total	14	16	30

participants, 14 (47%) were novices and 16 (53%) were experts, which is very close to our goal. From those 14 novices, 7 reported being “Not familiar at all” with LaTeX and the other 7 reported being “Slightly familiar.” 4 experts reported being “Moderately familiar,” 8 reported being “Very familiar,” and 4 reported being “Extremely familiar” with LaTeX.

4.1.3 Experiment Protocol

We emailed students of all levels (both undergraduate and graduate) in the College of Science and the Golisano College of Computing and Information Sciences to recruit participants. We asked them to fill out a questionnaire asking about their experience with LaTeX and mathematics.

If selected, participants received an email with a URL and instructions to begin the experiment on their personal computers. They were presented with an overview of the system and instructions to record their screen if they chose to.

After the overview, participants completed 5 practice tasks to familiarize themselves with the interface. The practice tasks focused on the visual operations, the handwriting input, and the correction mechanisms. For each of these practice tasks, the user was presented with a video showing how to complete the task.

The first practice tasks had participants replace two numbers in a formula using the visual operations. The second task asked participants to move a sub-expression, copy and sub-expression, and place it somewhere else using the visual operations. The third practice task taught participants how to drag the chips in the Deck to the formula on the canvas to insert a symbol and to replace an existing symbol. The fourth practice task introduced the recognizer and the Symbol Correction Panel to teach participants how to make corrections using the panel. The last practice task made participants use the recognizer and fix misrecognized symbols using the Alternate Symbol drop-down.

After the 5 practice tasks, participants began completing the real 18 tasks. Before each task, they were told whether the Alternate Symbol drop-down and/or the Symbol Correction Panel were available.

Data Collection. The interface logged whenever the formula on the canvas changed. This allowed us to calculate the number of changes, or operations, the user performed in each task. Operations that require multiple clicks (i.e drawing and recognizing, dragging a chip to a control point, selecting an alternate symbol, etc), are counted as one operation.

Additionally, we log specific, relevant actions regarding the correction mechanisms. We log when users:

1. Make a recognition request
2. Select a symbol from the Alternate Symbols drop-down
3. Select a symbol in the Symbol Correction panel
4. Select a formula in the Symbol Correction panel
5. Use a visual Operation (Replace, Replace All, Delete, Copy, etc.)

Each logged event is accompanied by the user id that triggered it, a timestamp, the target formula, and the current experiment condition. This allows us to easily distinguish and analyze how participants act under different conditions. These actions still count as just one operation; they are not double-counted.

We calculate how long it takes a participant to complete a task by getting the difference between the time the user starts the tasks and when they hit "Done" in the interface. The number of operations is calculated by counting the number of times the formula on the canvas changed. By this definition, handwriting counts as one operation when the user hits the recognize button.

Post Questionnaire

Figure 4.2 shows a summary of participants' answers to the following questions:

1. How difficult were the provided formula editing tasks?
2. How easy to use was the handwriting interface?
3. How easy was it to correct the formula if it didn't display what you expected?
4. How easy was it to correct individual symbols in the formula?

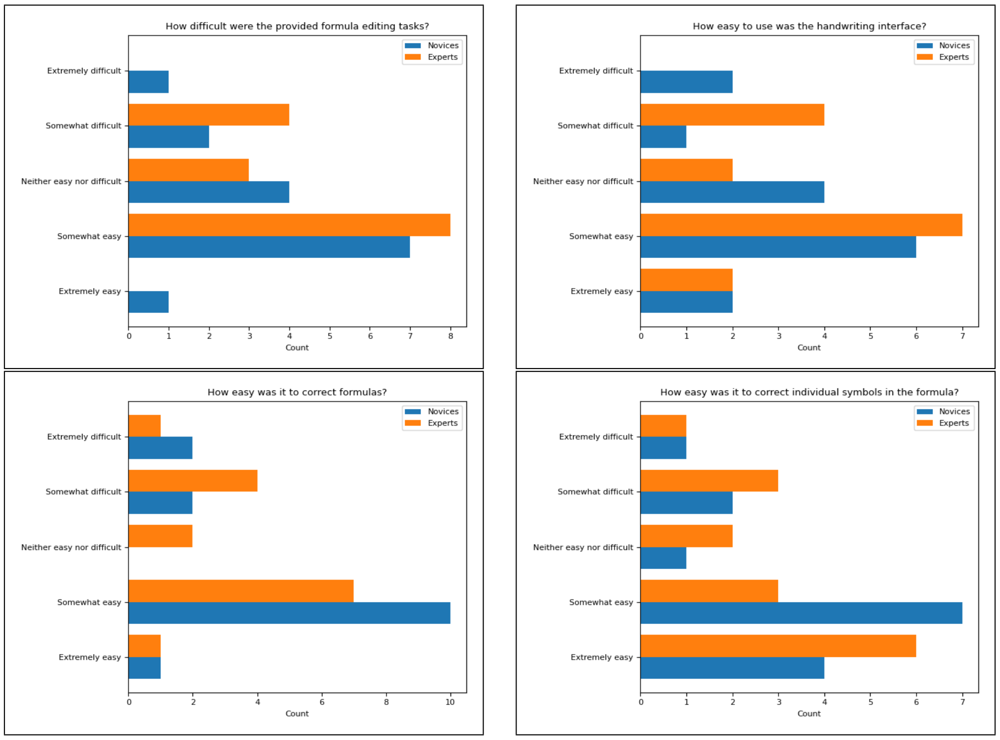


Figure 4.2: Participants' answers when asked how easy were the tasks (Top Left), how easy to use was the interface (Top Right), and how easy it was to correct entire formulas (Bottom Left) and individual symbols (Bottom Right). For most questions, the most frequent response for "Somewhat easy" among novices and experts alike. Results not broken by expertise can be found in Appendix A.

For the first 3 questions, the most frequent answer among novices and experts alike was "Somewhat easy," which indicates that many participants found the interface was easy to use. However, a fair number of experts (4) answered that the tasks, using the interface, and correcting symbols was "Somewhat difficult." While this does not constitute a majority, it's worth noting that some experts and a similar number of novices found the experiment difficult.

Reading the open-ended responses we learned that some participants were frustrated because the recognizer rarely recognized formulas correctly. The frustration was not helped by the fact that sometimes they could not find the correct symbols in the Alternate Symbols drop-down or the Symbol Correction panel. One user shared that the recognizer "never detected the correct formula and correcting it is so difficult."

Other users seemed to have trouble with the visual operations. One user shared that they wished "you could drag the symbol and place it in the center dot to replace what was there," which was one of the features we presented in the practice tasks. A handful of users asked for shortcuts such as being able to hit the enter key instead of clicking "Replace" in the visual operations drop-down or being able to delete symbols more easily by selecting a symbol and pressing the backspace key. The interface already has both of those features, but they were hard to convey during the videos in the practice tasks where users cannot see the keys being pressed in the demonstration videos.

Other users reported being positively surprised with the recognizer. One user reported that "even with a mouse, the handwriting was pretty accurate" and another reported that "Even using a mouse to draw, the system was surprisingly painless compared to similar drawing recognition tools I've used in the past," while others reported that drawing with a track-pad was difficult.

4.1.4 Results

Rather than rely on questionnaire data alone, we look at two metrics to measure performance across conditions: completion time and the number of operations. The ideal condition would be a condition with the lowest completion time and number of operations.

For the results discussed in the section, we're considering 29 out of the 30 participants described in the previous sections. We removed the data from one of the participants because their completion times seemed to indicate they left their computer in the middle of the experiment, which would throw off the averages reported here.

Not all participants were able to successfully complete every task; some participants clicked "Done" without matching the goal formula. 97% (28) of participants completed tasks 6 and 13. 93% (27) completed tasks 2, 3, 7, 8 and 11, 89% (26) completed tasks 1, 5, 14, 16, 17, and 18. 86% (25) completed tasks 9, 10, and 15. The two tasks with the lowest completion rate were tasks 4 and 12 with a completion rate of 48% (14) and 66% (19) respectively. The reason why so few participants finished tasks 4 and 12 is because many of them could not differentiate between the center dots $(\cdots)$ in the goal tasks and the plain dots $(\ldots)$ the recognizer returned.

Looking at the average completion time and number of operations shown in table 4.3 we can observe a few things. The first important observation we can make is that, for most formulas, users were able to complete the tasks faster in the "Symbol Alternative only" condition when compared to the "Alternative Results Panel (Faceted) only" condition. Participants in the "Both" condition did better than participants in the Symbol and Faceted conditions overall, as seen in Figure 4.4), where the Both condition has the lowest median in both Completion Time and Number of Operations for both novices and experts. As seen in table 4.3, participants in the "Both" condition completed tasks 1, 2, 4, 5, 7, 8, 9, 10, 14, 17, and 18 faster than participants who completed those tasks in the "Symbol" or "Faceted" conditions. This makes sense considering the "Both" condition always came last and participants would be more used to the interface at that point.

Furthermore, we cannot say that the differences in performance across conditions are solely due to the condition. In fact, we found that users in the "Symbol Correction Panel (Faceted)" condition often did not use the panel at all. Figure 4.5 shows the number of users that used the Symbol Correction panel and the Alternate Symbols drop-down. We can see, for example, that even though the Faceted condition has the best average performance in task 6 (as seen in table 4.3), only 3 users actually used the panel in that task (as seen in Figure 4.5). This means that the difference in performance across these conditions cannot be solely based on the conditions themselves.

Figure 4.5 also shows that participants used the Alternate Symbol drop-down more than the Symbol Correction panel. The task where most users (7) used the Alternate Symbols drop-down were tasks 10, 16, and 17. Those tasks also happened to have a lower average completion time in the Symbol condition than in the Faceted condition. The average completion time for task 10 was 48.5 in the symbol condition, and 75.9 in the Faceted condition. The average times for task 16 were 99.1 seconds (Symbol) and 136.6 seconds (Faceted.) The average times for task 17 were 74.8 (Symbol) and 111.3 (Faceted.)

Many participants shared that they used the symbol palette in the Deck to find symbols that they did not know. This would explain why the Alternate Symbol drop-down and the Symbol Correction panel were not used as often; participants felt that the deck was reliable and a good-enough solution. However, the numbers shared above tend to indicate that when users actually used the Alternate Symbol drop-down the average completion time was lower. Perhaps if the recognizer could provide the correct symbols more reliably, more users would be able to use the Alternate Symbol drop-down as a quick solution instead of having to scroll through the symbol palette.

Some participants were able to complete the experiment relatively quickly, but others took much longer than we anticipated. The shortest time for completing all experiment tasks was 16 minutes. The longest time to complete all experiment tasks was 42 minutes. These times do not include the training tasks or time spent on the exit questionnaire, which means some participants spent over an hour in the entire experiment, even though we expected them to spend 30 minutes.

The reason some tasks took so much longer than other tasks is that participants came across unexpected problems. One such problem was the problem in tasks 4 and 12 where participants could not differentiate between the center dots and the plain dots. This caused a lot of confusion in Task 4, as easily seen in Figure 4.6. Other issues stemmed from the fact that the recognizer would often not return the expected symbols, even if the user drew them perfectly.

One interesting pattern we found was that sometimes experts were performing considerably worse on average than novices in terms of completion time. For example, the completion time for tasks 10, and 18 in the Symbol condition is more than twice as large for experts, as seen in Fig 4.6. Upon further investigation, we found that this is due to a few outliers spending more than 3 minutes on those tasks. Typically, we found that those outliers made more than one recognition request, which means they had to draw the formula multiple times.

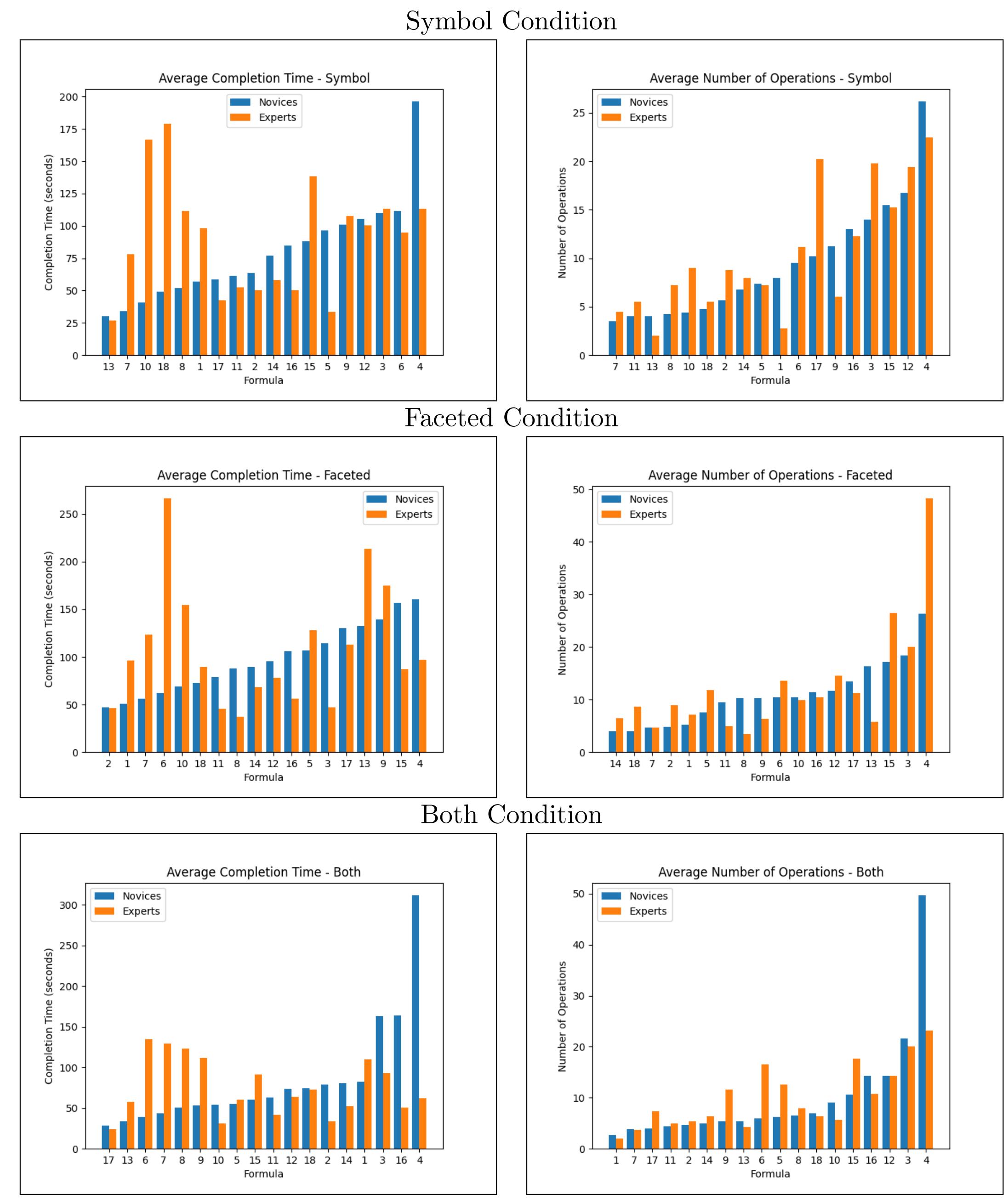


Figure 4.3: Average Completion Time and Average Number of Operations across Conditions and Tasks in Experiment 1

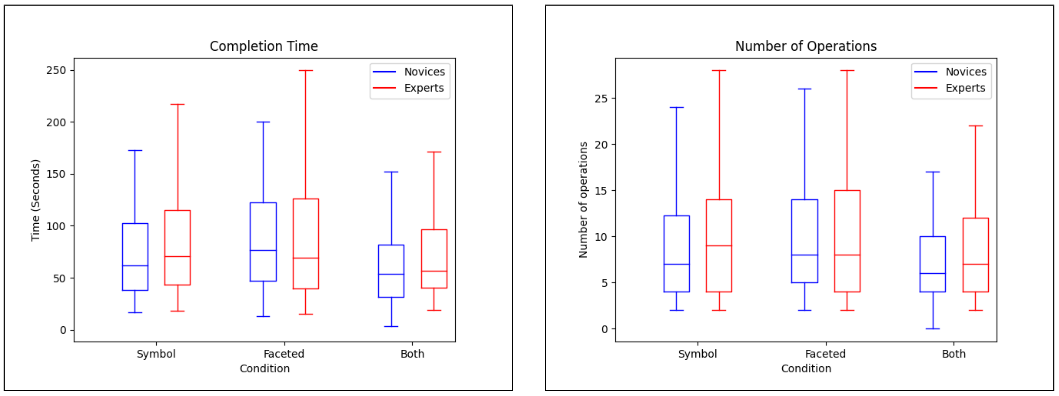


Figure 4.4: Box plot of the Completion Time and Number of Operations for the entire experiment.

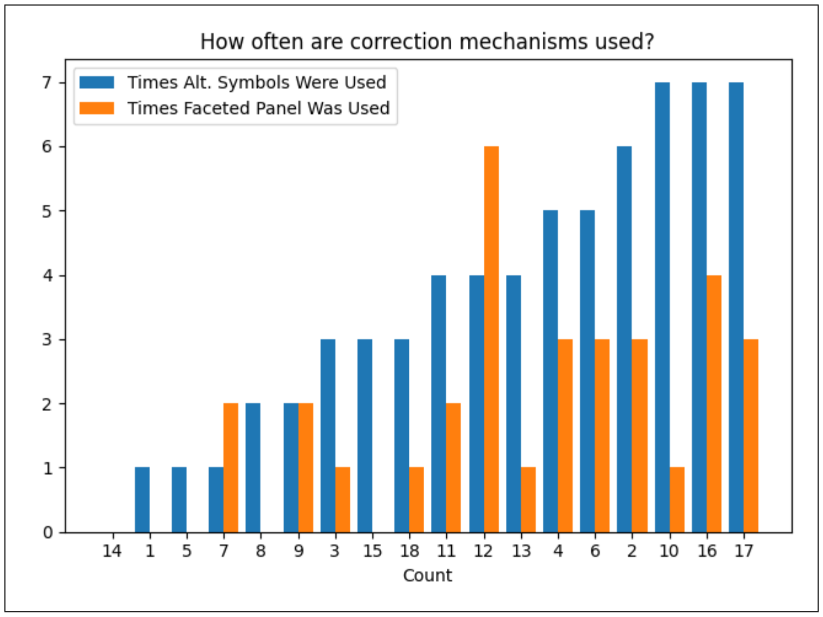


Figure 4.5: Summary of Correction Mechanisms used in each tasks. The Alternate Results drop-down was used more often that the Symbol Correction Panel in most tasks. These numbers represent the total number of times each correction mechanism was used across all conditions combined.

Table 4.3: Summary of Average Completion Time and Average Number of Operations across tasks and condition.

	Average Completion Time			Average Number of Operations		
Task	Symbol	Faceted	Both	Symbol	Faceted	Both
1	42.1 (30.0)	48.7 (47.6)	50.0 (80.5)	5.4 (6.8)	6.2 (7.0)	2.3 (1.0)
2	70.3 (41.1)	71.6 (64.2)	65.7 (31.1)	7.1 (4.4)	6.9 (7.3)	5.1 (2.7)
3	135.5 (65.7)	119.2 (27.4)	145.4 (34.2)	16.6 (7.6)	19.2 (5.7)	20.6 (3.6)
4	189.2 (157.4)	207.3 (138.1)	197.7 (136.4)	24.7 (17.4)	36.1 (30.3)	33.1 (27.1)
5	103.9 (36.9)	128.0 (59.2)	92.7 (50.7)	7.3 (2.5)	9.4 (4.3)	9.8 (7.8)
6	105.6 (78.8)	75.9 (41.9)	84.3 (64.0)	10.3 (4.5)	12 (8.6)	12.6 (9.9)
7	38.3 (17.6)	49.8 (27.8)	39.3 (23.3)	4 (2.3)	4.7 (2.2)	3.8 (1.2)
8	52.1 (22.3)	54.2 (45.0)	54.2 (20.9)	5.9 (3.5)	5.8 (5.5)	7 (3.4)
9	72.8 (35.6)	91.9 (69.5)	66.2 (54.3)	8.3 (5.3)	7.7 (3.0)	7.4 (8.5)
10	48.5 (28.7)	75.9 (72.5)	49.8 (21.3)	6.4 (5.0)	10 (8.7)	7.9 (5.1)
11	55.8 (27.4)	65.2 (32.8)	63.1 (35.8)	4.8 (4.0)	6.8 (5.1)	4.6 (2.1)
12	123.6 (71.8)	117.4 (49.2)	73.2 (27.2)	18.2 (9.1)	13.6 (5.8)	14.3 (11.8)
13	32.0 (6.8)	98.3 (126.0)	34.0 (11.8)	3 (1.5)	12.1 (18.8)	4.6 (1.3)
14	92.2 (56.7)	99.8 (69.3)	63.0 (36.3)	7.4 (4.0)	5.1 (2.5)	5.9 (3.2)
15	94.3 (38.4)	179.6 (149.1)	94.8 (56.1)	15.4 (6.9)	20.9 (16.3)	15.6 (9.0)
16	99.1 (52.6)	136.6 (88.9)	118.5 (100.3)	12.6 (4.9)	11 (3.6)	12 (6.4)
17	74.8 (47.2)	111.3 (100.9)	44.0 (27.4)	14.7 (11.8)	12.4 (8.9)	6.4 (2.8)
18	81.2 (82.9)	82.1 (64.1)	66.8 (58.2)	5.1 (3.2)	5.8 (6.1)	6.6 (6.0)

4.1.5 Summary

The goal of this experiment was to learn about good ways to make corrections to the formulas returned by the recognizer. Our results indicate that most participants used the symbols in the symbol palette or simply typed their correction. Many users also used the alternate symbols drop-down, but only a few used the alternate results panel. One possible reason for this could be that the recognizer could not recognize many of the symbols in some of the formulas because some of the symbols in the formulas were not part of the recognizer's vocabulary. This means that users would often not find their desired output in the list of alternate symbols.

Since we did not see strong evidence suggesting that participants found the Alternate Results Panel useful, we decided to remove it for the second experiment. It is possible that the results may have been different if we had only used symbols that the recognizer could handle, making users more likely to use the new correction mechanisms to more quickly fix issues without looking through the deck to find a symbol. But for the sake of time, we decided to move on without this feature and reconsider the tasks for the next experiment with the recognizer's limitations in mind.

4.2 Experiment 2: Handwriting vs No Handwriting

The second experiment was designed to test whether having handwriting would help users complete more tasks than if they did not have handwriting. We also wanted to see what impact it would have on the speed at which they completed tasks. Would using handwriting slow things down? Or would it make it faster for novices? Additionally, we wanted to look at the number of operations it would take for participants to complete tasks. Would handwriting help reduce the number of operations required to complete a task? Or would it require more operations because of the errors users would have to correct?

For this experiment, we counted the number of operations the same way we did for the first experiment. That is, any action that changes the formula displayed on the canvas counts as an operation. This includes any of the visual operations including dragging chips from the canvas and using a symbol from the alternate symbols drop-down. Entering a handwritten formula counts as one operation.

4.2.1 Experiment Tasks

Having learned how long it took participants to complete the first experiment, we reduced the number of tasks for the second experiment to 9. These tasks, shown in table 4.2.1, were designed so that the recognizer could recognize all the symbols and structures required to complete the tasks. This would allow novices to use handwriting if they were not familiar with special structures like fractions or special symbols like the Greek letters.

Like in the first experiment, users had to complete these tasks in a modified version of the interface, shown in Fig 4.2.1. This version of the interface brings the LaTeX Panel back and now users have to click the Handwriting button in the LaTeX Panel to enter a handwritten formula. The Alternate Results button in the handwriting recognition window is hidden as well as the option to insert a formula in context as shown in Figure 3.6.

Table 4.4: Handwriting vs. No Handwriting Experiment Tasks.

Task #	Target Formula
1	$F = \frac{B^2 A}{2\mu_0}$
2	$f(\lambda x) = \lambda^{\Delta} f(x)$
3	$\int_{-1}^{1} (1+x)^{\beta} dx$
4	$R^i F(X) = 0 \forall i > 0$
5	$\frac{du}{dt} - \frac{du}{dx}$
6	$\frac{x^2}{a^2} = z + \frac{y^2}{b^2}$
7	$\phi = \frac{1+\sqrt{5}}{2}$
8	$F = -\frac{c}{12\pi\sigma}$
9	$X = -k\frac{dT}{dz}$

Each participant was exposed to 3 conditions:

1. Handwriting Only: Participants had access to the handwriting button but they could not edit the formula through the LaTeX Panel
2. LaTeX Only: Participants had access to the LaTeX Panel but could not insert formulas using Handwriting.
3. Both: Participants had access to both the LaTeX Panel and the Handwriting button.

4.2.2 Participants and Blocking Design

We again aimed to have at least 30 participants. After going through the first experiment, we decided to invite 50 participants to get enough responses in a timely manner. Just like in the first experiment, half of those participants are what we considered experts and the other half were novices. We tried to recruit an equal number of participants from the College of Science and the College of Computing.

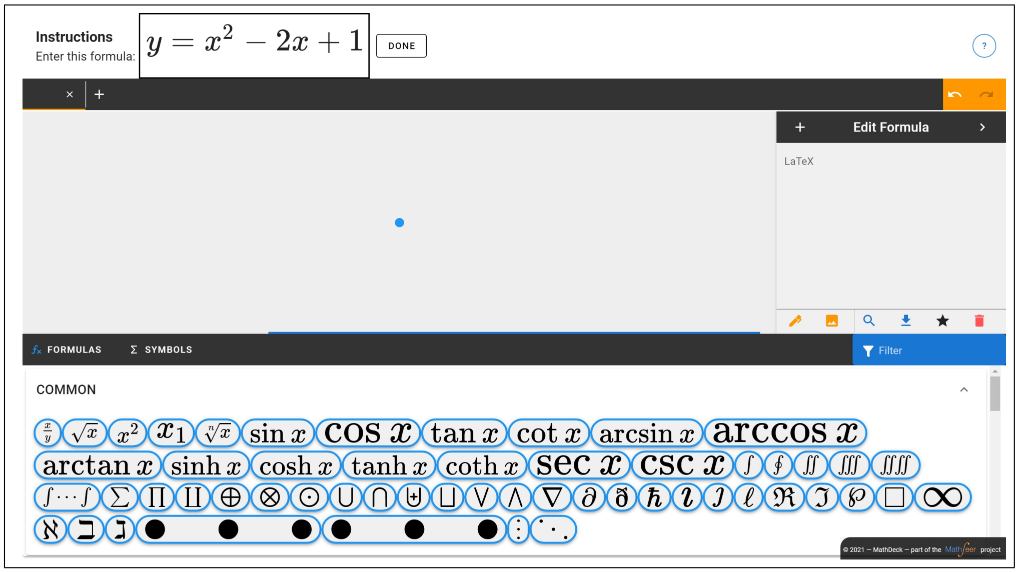


Figure 4.6: The interface for the Handwriting Interface experiments.

We replicated the blocking design from the first experiment: The first 3 participants saw the following order of conditions: Handwriting only, LaTeX only, and Both. The next 3 participants saw LaTeX only before they saw Handwriting. We split the 9 tasks into 3 blocks of 3 formulas each. The first block contained formulas 1-3, the second block had formulas 4-6, and the third block had formulas 7-9. We tried to place formulas into blocks such that every block had at least one formula with more than one special symbol and at least one formula with at most one special symbol.

We also changed the order in which they saw the formula blocks and changed the order in which the blocks were presented. That is, if the first block for participant 1 contains tasks 1, 2, and 3, in that order, then the second participant's first block would have tasks 2, 3, and 1, in that order. The third participant would see task 3, then 2, and then 1. The block order for participants 1-3 would be A-B-C, the order for participants 4-6 would be B-C-A, and the order for participants 7-9 would be C-A-B. The pattern repeats starting with participant 10.

Participant Demographics

Highest Level of School Completed			
	Novices	Experts	Total
High School	16	10	26
Associate's	1	0	1
Bachelor's	0	7	7
Master's	0	3	3
College			
	Novices	Experts	Total
College of Science	6	8	14
College of Computing	11	12	23
Gender			
	Novices	Experts	Total
Female	3	10	13
Male	11	8	19
Non-binary	2	1	3
Prefer not to say	1	1	2
Age			
	Novices	Experts	Total
18-24	16	17	33
25-34	1	3	4
Total	17	20	37

Table 4.2.2 shows the highest level of education achieved by participants who completed the experiment, their current college, gender, and their age range. Out of the 50 participants we invited, 37 completed the experiment and the exit questionnaire. 17 (46%) of those participants were novices and 20 (54%)) were experts. Of those 17 novices, 14 reported being "Not familiar at all" with LaTeX and the other 3 reported being "Slightly familiar." Out of the 20 experts, 6 experts reported being "Moderately familiar," 11 reported being "Very familiar," and 3 reported being "Extremely familiar" with LaTeX. The split between colleges was not as even as we would have liked it to be because there were not as many participants who identified as novices in the College of Science.

4.2.3 Experiment Protocol

Like in the first experiment, selected participants received an email with a URL and instructions to begin the experiment. They were given an updated overview of the system and instructions to record their screen if they wanted to share their experience.

This time participants had 7 practice tasks to familiarize themselves with the interface. For each practice task, they could watch a video to learn how to complete the task. Each practice task covers a different feature in the interface. The features covered by each task are the following:

1. Using the LaTeX panel to edit formulas
2. Using visual operations to replace and delete symbols
3. Copying and moving sub-expressions with visual operations
4. Dragging chips from the symbol palette to insert and replace symbols
5. Using the recognizer and the Alternate Symbol drop-down
6. Using the recognizer to create a new chip
7. Using the recognizer to create a new tab

After the practice tasks, they completed the 9 real tasks. Before each task, they were told which condition they were in. At the end of the experiment, they completed an exit questionnaire about their experience in the experiment.

Post Questionnaire

Figure 4.7 shows a summary of participants answers to the following questions:

1. How difficult were the provided formula editing tasks?
2. When using handwriting, how easy to use was the handwriting interface?
3. When using handwriting, how easy was it to correct the formula if it didn't display what you expected?
4. When using handwriting, how easy was it to correct individual symbols in the formula?

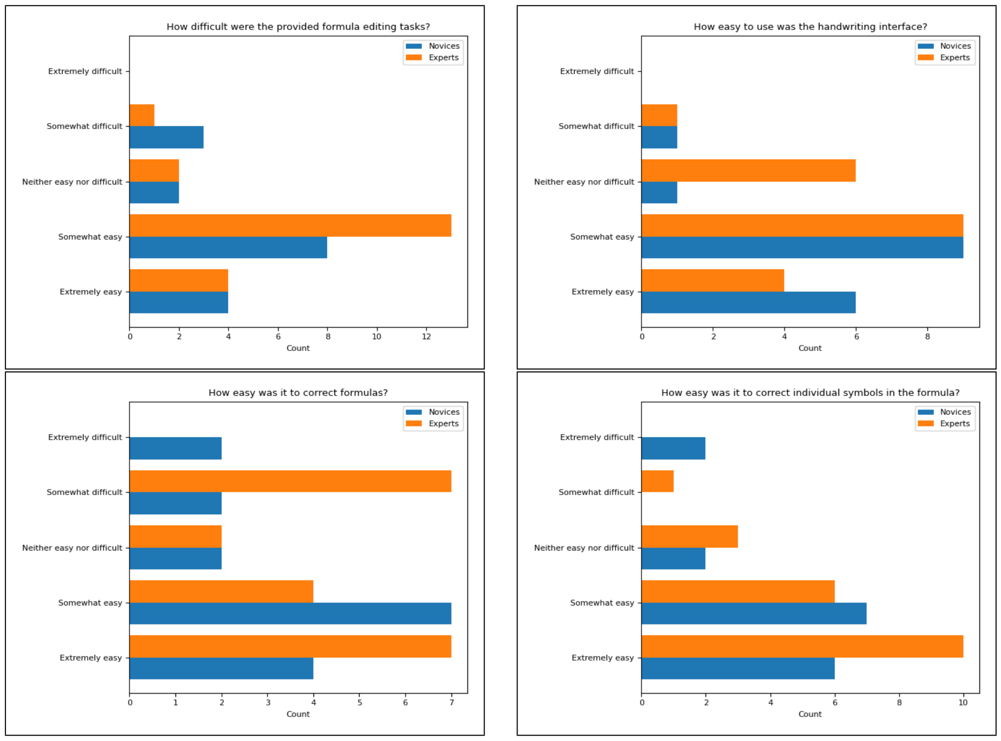


Figure 4.7: Participants answers when asked how easy were the tasks (Top Left), how easy to use was the handwriting interface (Top Right), and how easy it was to correct entire formulas (Bottom Left) and individual symbols (Bottom Right). For most questions, the most frequent responses were "Somewhat easy" and "Extremely easy" among novices and experts alike. Results not broken by participants can be found in Appendix B

For the first 3 questions, the most frequent answer among novices and experts alike was "Somewhat easy," which indicates that many participants found the interface was easy to use (See Fig. 4.7.) For the last question, the most frequent answer was "Extremely easy," which suggests participants thought changing individual symbols was much easier than perhaps changing the structure of the formula.

We also see that 7 experts reported that correcting formulas was "Somewhat difficult" but only expert 1 said that correcting symbols was "Somewhat difficult." This is likely the case because, for experts, it's often easier to change the LaTeX directly rather than have to use the visual operations. Since they did not have access to the LaTeX panel in the Handwriting condition, they may have felt restricted in what they could do.

Some participants shared that they thought the recognizer worked well. One participant reported that "Even when handwriting did not get the right symbol, it always had the right one in the replacement options which made it very easy to change out any wrong symbols." Another participant shared that "Sometimes, the handwriting editor didn't recognize what I had written, but it just happened 2-3 times. Otherwise, the experience was smooth."

Many participants thought that drawing using their computer's track-pad was not easy. Some experts shared that they preferred writing LaTeX directly rather than draw the formulas because it was "much faster," which is to be expected. This was probably one of the reasons why so many experts reported that correcting formulas was "Somewhat difficult."

One participant was quick to appreciate the multi-modal nature of the tool and shared the following: "I really like that you can enter a formula via handwriting, LaTeX, and by dragging symbols and notation. I feel like it caters to a very wide audience. It's nice to have regular LaTeX editing in there for people who are acquainted LaTeX. Having the "deck" of symbols to drag makes it accessible and useful for people who aren't familiar with LaTeX."

4.2.4 Results

We again looked at completion time and number of operations to measure performance. Table 4.5 shows a summary of these metrics across conditions. As expected, participants performed best in the Both condition because it's the last condition, so they have more experience with the interface. In the

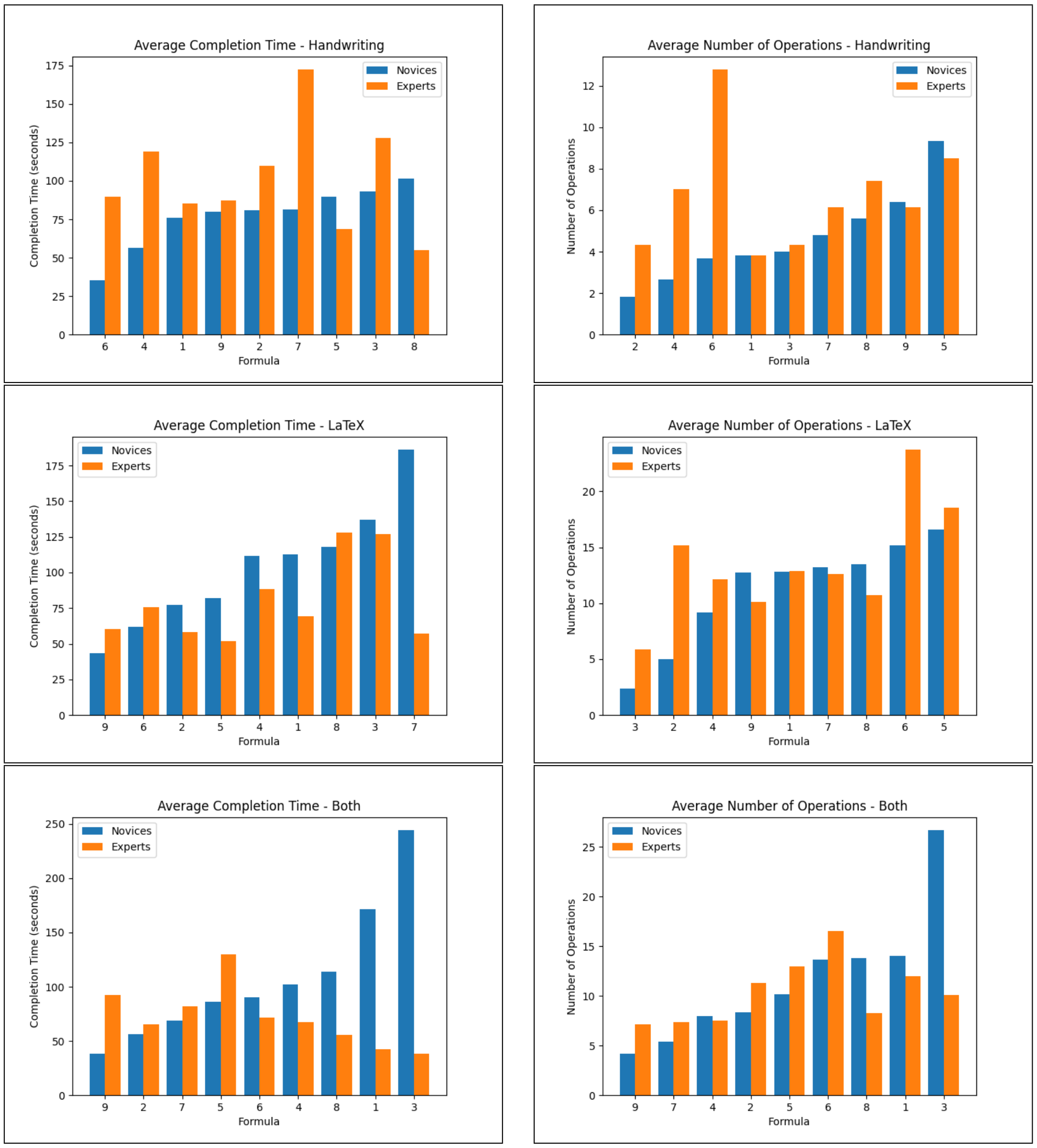


Figure 4.8: Average Completion Time and Average Number of Operations across conditions and tasks in the Handwriting vs. No Handwriting experiment.

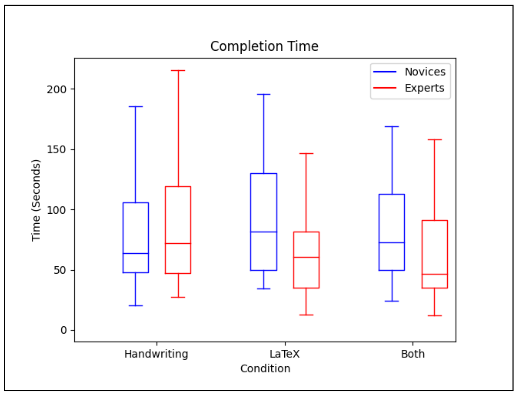
Table 4.5: Summary of Average Completion Time and Average Number of Operations across tasks and condition.

	Average Completion Time			Average Number of Operations		
Task	Handwriting	LaTeX	Both	Handwriting	LaTeX	Both
1	83.0 (50.0)	81.9 (52.8)	112.1 (118.7)	3.8 (2.1)	12.8 (7.1)	12.5 (7.9)
2	99.9 (69.3)	76.5 (37.2)	63.2 (31.9)	3.1 (2.5)	10.9 (11.4)	10.6 (10.7)
3	89.1 (55.6)	91.1 (66.5)	122.6 (149.9)	4.2 (3.0)	4.4 (5.4)	14.2 (15.8)
4	79.4 (43.4)	76.9 (68.5)	115.9 (88.7)	5.9 (7.5)	10.9 (6.5)	7.8 (6.0)
5	104.1 (47.3)	85.8 (119.3)	78.7 (47.7)	8.7 (5.2)	17.8 (14.1)	11.6 (7.3)
6	138.1 (168.5)	66.2 (37.0)	78.9 (42.0)	10.5 (9.9)	20.2 (10.7)	15.1 (11.0)
7	74.1 (48.1)	147.3 (86.3)	60.5 (34.8)	5.6 (3.4)	12.8 (10.1)	6.6 (5.7)
8	116.8 (135.4)	123.9 (147.0)	70.1 (71.8)	6.7 (4.8)	11.7 (4.6)	10.4 (5.6)
9	65.4 (37.7)	52.6 (30.8)	38.5 (23.7)	6.2 (3.4)	11 (6.8)	6 (3.1)

case of novices, it will also mean that they will have learned more about how to write formulas in LaTeX.

In general, participants successfully completed more tasks in this experiment than they did in the first experiment. All 37 participants completed task 7. 97% (36) participants completed task 8. 95% (35) completed tasks 2, 4, and 9. 92% (34) completed tasks 3 and 6. 90% (33) of participants completed task 1. The task with the lowest completion rate was task 5 with a completion rate of 84% (31). It's hard to say why this was the case. Only half the cases can be attributed to errors from the recognizer; for the other half, users were able to type the formula using LaTeX, which should have been easy to do.

What we care about the most is the difference in completion time and number of operations between the Handwriting-only and the LaTeX-only conditions because this would indicate if handwriting can be a more efficient way of inputting formulas when compared to typing LaTeX. Figure 4.9 shows the overall comparison between the three conditions. We see that novices spent slightly less time in the Handwriting and LaTeX condition. The median for novices in the Handwriting condition was approximately 60 seconds while the median in the LaTeX condition was closer to 80 seconds. The longest time for novices in both conditions was about 3 minutes. Experts, on the other hand, spent more time in the handwriting condition, with some experts taking as long as over 3 minutes per task in the Handwriting condition. This lines up with the comments in the exit questionnaire reporting that LaTeX was "much faster" for experts.



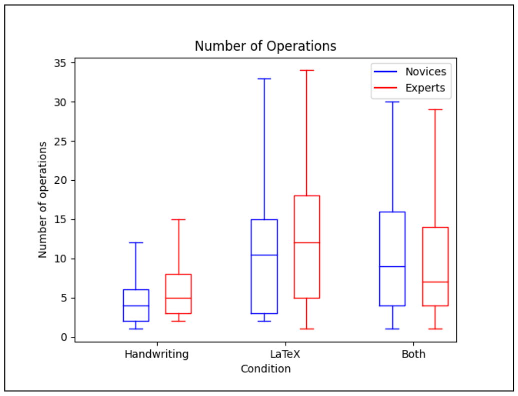


Figure 4.9: Box plots of the Completion Time and Number of Operations for the entire experiment.

Additionally, we see drastic changes in the number of operations. Novices and experts alike completed tasks in fewer operations in the Handwriting condition. This is likely because handwriting counts as one operation. Most users would draw the entire formula and fix small errors in the recognizer's output. Typing on the LaTeX panel, however, can count as multiple operations. When the user stops typing in the LaTeX for 2 seconds, the formula on the canvas re-renders, which counts as an operation. These can stack up pretty quickly if the user constantly stops to see the new rendered formula.

We investigated why some completion times for experts were so much higher than for novices in the Handwriting condition. We found that some of the outlier values come from users who made more than one recognition request. This adds up a lot of time because drawing formulas takes a lot of time, especially if the user is using a track-pad to draw, which was the case for many users, as reported in the exit questionnaire.

Like in the first experiment, we wanted to confirm that the decrease in completion time and number of operations actually came from differences in the conditions. Table 4.6 shows the number of users who used the handwriting feature when it was available. For tasks 3, 4, 5, and 17 the majority of users used handwriting when it was available. All of those tasks yielded the smallest average number of operations in the handwriting condition. The handwriting condition also resulted in the smallest average completion time in task 3.

Table 4.6: Summary of Times Handwriting and the Alternate Symbol Drop-down Was Used Across Tasks. Users used handwriting at least half of the time it was available for some tasks (Task 2) and as many as 78% of the time for other tasks (Task 5). Handwriting was not available equally across tasks because of our blocking design and because not all of the invited participants completed the experiment.

Task #	Used HW	Used Alt. Symbols	Total Available
1	15	2	24
2	12	2	24
3	**18**	3	24
4	17	4	24
5	**18**	5	23
6	14	2	24
7	16	3	25
8	17	**6**	25
9	15	4	25

What's interesting about this task is that it's the only task that contained an integral with limits, which may have resulted in more users using the recognizer if they did not know how to write the limits in LaTeX or using the visual operations.

For all of the other tasks, at least half of the participants used handwriting when it was available. A handful of those participants also used the Alternate Symbols drop-down to correct errors in the recognizer, but, as seen in the first experiment, participants reported that they often chose to write the correct symbol themselves or drag it from the symbol palette. Additionally, the recognizer did a better job at recognizing the symbols correctly in these tasks because we designed the tasks around the classes of symbols we knew the recognizer could handle.

4.2.5 Summary

We found that when handwriting was available, novices took about 10 seconds (based on median) less to complete tasks and they did so using few operations after the handwritten formula was recognized. A reason why handwriting might not result in even shorter completion times could be that drawing strokes takes longer than typing LaTeX, especially if participants are using a track-pad on their laptop as opposed to a computer mouse or a pen.

On the other hand. Experts completed tasks more quickly in the LaTeX condition than they did in the Handwriting condition. However, just like novices, they performed fewer operations in the Handwriting condition because one handwritten formula counts as one operation.

An unexpected result is that participants performed more operations in the Both condition than they did in the Handwriting condition. This is perhaps because they saw that typing the formulas was faster, even if it took more operations like dragging symbols and structures from the symbol palette. We also found that novices sometimes spent more time in the Both condition than they did in the Handwriting condition. A possible explanation for this could be that novices spent some time getting used to the mix of handwritten formulas and being able to edit the LaTeX directly.

4.3 Summary of Findings

Participants in the correction mechanisms experiment preferred to correct errors in the recognizer by either typing the correct symbols themselves, or by using the symbol palette. Their next most used strategy was the alternate symbols drop-down, so we decided to keep this feature for the second experiment.

The second experiment showed that novices spend less time inputting formulas when they use handwriting while experts spend less time if they simply type LaTeX. However, we also found that the number of operations after inputting a handwritten formula is very small, regardless of the level of expertise.

Chapter 5

Conclusion

Typing formulas in LaTeX can be challenging for non-expert mathematicians. Our main hypothesis was that handwriting could allow for entering formulas more efficiently in MathDeck.

We presented changes to the MathDeck interface to allow users to enter handwritten formulas into the system. Most recognizers are imperfect, so we added an Alternate Symbol drop-down and a Symbol Correction panel to easily correct errors. We ran an experiment to test the best ways to correct errors in the recognizer. Users had the option to use the Alternate Symbol drop-down, the Symbol Correction panel, or to correct errors manually by typing the correct symbol or finding it in a symbol palette. We found that participants mostly used the symbol palette or typed the correct symbols to fix misrecognized symbols.

To test our main hypothesis, we ran a second experiment to see how both novice and expert users performed when they had access to handwriting compared to when they could only write LaTeX. We found that novices performed best in terms of completion time when they had access to handwriting and that experts performed better if they could simply type the formulas using LaTeX. We also found that both novices and experts could correct handwritten formulas using very few operations in the new MathDeck interface.

5.1 Future Work

Our results showed evidence that handwriting can help novice users input formulas, but the system could use some improvements to make the process of entering handwritten formulas even faster. For example, improving our recognizer to recognize a larger set of symbols and to more accurately classify symbols can reduce the number of corrections the user has to make. Additionally, the formulas in our second experiment are not very complex. We decided to make them simple enough so that the recognizer would be able to handle them properly. Given an improved recognizer, it would be interesting to look at how users deal with more complex structures such as matrices and nested structures.

With this change in place, we could also add the ability to search for symbols in the symbol palette by drawing the symbol instead of having to scroll through a list of symbols if you don't know the name of the symbol you're looking for. Detexify is an example of a tool that allows users to search for symbols by drawing strokes [14].

The next steps for the MathDeck interface should focus on making the interface more user-friendly and polishing the existing features. Some users still found that some of the visual operations are not intuitive or could be simplified. In the future, we could evaluate better ways to do simple operations like copying or replacing sub-expressions. Additionally, some parts of the interface could be reworked to be faster so that there is less wait time between operations.

For the experiments described here, we removed the interface's search features such as the ability to search for any or multiple formulas using one of 11 different search engines. The full version of the interface also automatically searches for related formulas and concepts as the user inputs formulas, which can serve as a way to auto-complete formulas or find related formulas. Once these search features are brought back into the interface and the issues are fixed, MathDeck will be a tool that novices and experts alike can leverage to work with math formulas efficiently.

Bibliography

[1] Lisa Anthony, Jie Yang, and Kenneth Koedinger. Towards the Application of a Handwriting Interface for Mathematics Learning. In *2006 IEEE International Conference on Multimedia and Expo*, pages 2077–2080, Toronto, ON, Canada, July 2006. IEEE.

[2] Lisa Anthony, Jie Yang, and Kenneth R. Koedinger. Adapting handwriting recognition for applications in algebra learning. In *Proceedings of the international workshop on Educational multimedia and multimedia education - Emme '07*, page 47, Augsburg, Bavaria, Germany, 2007. ACM Press.

[3] Lisa Anthony, Jie Yang, and Kenneth R. Koedinger. Benefits of handwritten input for students learning algebra equation solving. In Rosemary Luckin, Kenneth R. Koedinger, and Jim E. Greer, editors, *Artificial Intelligence in Education, Building Technology Rich Learning Contexts That Work, Proceedings of the 13th International Conference on Artificial Intelligence in Education, AIED 2007, July 9-13, 2007, Los Angeles, California, USA*, volume 158 of *Frontiers in Artificial Intelligence and Applications*, pages 521–523. IOS Press, 2007.

[4] Xiang Ao, Xugang Wang, Feng Tian, Guozhong Dai, and Hongan Wang. Crossmodal error correction of continuous handwriting recognition by speech. In *Proceedings of the 12th International Conference on Intelligent User Interfaces*, IUI '07, page 243–250, New York, NY, USA, 2007. Association for Computing Machinery.

[5] Dilaksha Attanayake, James Denholm-Price, Gordon Hunter, Eckhard Pfluegel, and Angela Wigmore. Intelligent Assistive Interfaces for Editing Mathematics. pages 286 – 297.

[6] Larwan Berke, Christopher Caulfield, and Matt Huenerfauth. Deaf and hard-of-hearing perspectives on imperfect automatic speech recognition for captioning one-on-one meetings. In *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '17, page 155–164, New York, NY, USA, 2017. Association for Computing Machinery.

[7] Stephen Cummins, Ian Davies, Andrew Rice, and Alastair R. Beresford. Equality: A Tool for Free-form Equation Editing. In *2015 IEEE 15th International Conference on Advanced Learning Technologies*, pages 270–274, Hualien, Taiwan, July 2015. IEEE.

[8] Yancarlos Diaz, Gavin Nishizawa, Behrooz Mansouri, Kenny Davila, and Richard Zanibbi. The mathdeck formula editor: Interace formula entry combining latex, structure editing, and search. In *Proc. ACM CHI*, page to appear, 2021.

[9] Mitsushi Fujimoto. An Interface for Math e-Learning on Pen-Based Mobile Devices.

[10] Eric Gross, Safwan Wshah, Isaiah Simmons, and Gary Skinner. A handwriting recognition system for the classroom. In *Proceedings of the Fifth International Conference on Learning Analytics And Knowledge - LAK '15*, pages 218–222, Poughkeepsie, New York, 2015. ACM Press.

[11] Jiseong Gu and Geehyuk Lee. In-Place-Ink: Toward More Direct Handwriting Interfaces. In *Proceedings of the 2016 ACM on Interactive Surfaces and Spaces - ISS '16*, pages 67–76, Niagara Falls, Ontario, Canada, 2016. ACM Press.

[12] Tracy Hammond, Drew Logsdon, Joshua Peschel, Joshua Johnston, Paul Taele, Aaron Wolin, and Brandon Paulson. A sketch recognition interface that recognizes hundreds of shapes in course-of-action diagrams. In *Proceedings of the 28th of the international conference extended abstracts on*

Human factors in computing systems - CHI EA '10, page 4213, Atlanta, Georgia, USA, 2010. ACM Press.

[13] J. Johnston and T. Hammond. Computing confidence values for geometric constraints for use in sketch recognition. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, SBIM '10, page 71–78, Goslar, DEU, 2010. Eurographics Association.

[14] Daniel Kirsch. Detexify: Erkennung handgemalter LaTeX-Symbole. page 83.

[15] Mahshad Mahdavi, Leilei Sun, and Richard Zanibbi. Visual parsing with query-driven global graph attention (QD-GGA): preliminary results for handwritten math formula recognition. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14-19, 2020*, pages 2429–2438. IEEE, 2020.

[16] Alexandra Mendes, Roland Backhouse, and Joao F. Ferreira. Structure Editing of Handwritten Mathematics: Improving the Computer Support for the Calculational Method. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces - ITS '14*, pages 139–148, Dresden, Germany, 2014. ACM Press.

[17] Gavin Nishizawa. Visual Structure Editing of Math Formulas, 08 2020.

[18] Gavin Nishizawa. Visual structure editing of math formulas. Master's thesis, Rochester Institute of Technology, USA, 2020.

[19] Gavin Nishizawa, Jennifer Liu, Yancarlos Diaz, Abishai Dmello, Wei Zhong, and Richard Zanibbi. Mathseer: A math-aware search interface with intuitive formula editing, reuse, and lookup. In Joemon M. Jose, Emine Yilmaz, João Magalhães, Pablo Castells, Nicola Ferro, Mário J. Silva, and Flávio Martins, editors, *Advances in Information Retrieval*, pages 470–475, Cham, 2020. Springer International Publishing.

[20] Sharon Oviatt, Alexander Arthur, Yaro Brock, and Julia Cohen. Expressive pen-based interfaces for math education. In *Proceedings of the 8th iternational conference on Computer supported collaborative learning - CSCL'07*, pages 573–582, New Brunswick, New Jersey, USA, 2007. Association for Computational Linguistics. ISSN: 15734552.

[21] J. Rodriguez, G. Sanchez, and J. Llados. A Pen-Based Interface for Real-Time Document Edition. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2*, pages 939–943, Curitiba, Parana, Brazil, September 2007. IEEE. ISSN: 1520-5363.

[22] Maximilian Schrapel, Max-Ludwig Stadler, and Michael Rohs. Pentelligence: Combining Pen Tip Motion and Writing Sounds for Handwritten Digit Recognition. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*, pages 1–11, Montreal QC, Canada, 2018. ACM Press.

[23] Guy Shani and Noam Tractinsky. Displaying relevance scores for search results. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval - SIGIR '13*, page 901, Dublin, Ireland, 2013. ACM Press.

[24] Steve Smithies, Kevin Novins, and James Arvo. A Handwriting-Based Equation Editor. *Proceedings of Graphics Interface*, 05 2002.

[25] Ryan Stedman, Michael Terry, and Edward Lank. Aiding human discovery of handwriting recognition errors. In *Proceedings of the 15th ACM on International Conference on Multimodal Interaction*, ICMI '13, page 295–302, New York, NY, USA, 2013. Association for Computing Machinery.

[26] Hariharan Subramonyam, Colleen Seifert, Priti Shah, and Eytan Adar. texSketch: Active Diagramming through Pen-and-Ink Annotations. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13, Honolulu HI USA, April 2020. ACM.

[27] John Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12(2):257 – 285, 1988.

[28] E. Tapia and R. Rojas. Recognition of on-line handwritten mathematical expressions in the E-Chalk system - an extension. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 1206–1210 Vol. 2, August 2005. ISSN: 2379-2140.

[29] Eugene M. Taranta and Joseph J. LaViola. Math Boxes: A Pen-Based User Interface for Writing Difficult Mathematical Expressions. In *Proceedings of the 20th International Conference on Intelligent User Interfaces - IUI '15*, pages 87–96, Atlanta, Georgia, USA, 2015. ACM Press.

[30] Ryotaro Toba, Tsuneo Kato, and Seiichi Yamamoto. Efficient Speech-Recognition Error-Correction Interface for Japanese Text Entry on Smartwatches. In *Proceedings of the 21st International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 1–7, Taipei Taiwan, October 2019. ACM.

[31] Vu Tran Minh Khuong, Minh Khanh Phan, and Masaki Nakagawa. Interactive User Interface for Recognizing Online Handwritten Mathematical Expressions and Correcting Misrecognition. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, pages 26–30, Sydney, Australia, September 2019. IEEE.

[32] Keith Vertanen and Per Ola Kristensson. On the benefits of confidence visualization in speech recognition. In *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, page 1497, Florence, Italy, 2008. ACM Press.

[33] Keith Vertanen and Per Ola Kristensson. Intelligently aiding human-guided correction of speech recognition. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, page 1698–1701. AAAI Press, 2010.

[34] Xugang Wang, Junfeng Li, Xiang Ao, Gang Wang, and Guozhong Dai. Multimodal error correction for continuous handwriting recognition in pen-based user interfaces. In *Proceedings of the 11th international conference on Intelligent user interfaces - IUI '06*, page 324, Sydney, Australia, 2006. ACM Press.

[35] Richard Zanibbi, Kevin Novins, James Arvo, and Katherine Zanibbi. "Aiding Manipulation of Handwritten Mathematical Expressions through Style-Preserving Morphs". page 8.

[36] Robert Zeleznik, Andrew Bragdon, Ferdi Adeputra, and Hsu-Sheng Ko. Hands-on math: a page-based multi-touch and pen desktop for technical

work and problem solving. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology - UIST '10*, page 17, New York, New York, USA, 2010. ACM Press.

[37] Dmytro Zhelezniakov, Anastasiia Cherneha, Viktor Zaytsev, Tetiana Ignatova, Olga Radyvonenko, and Oleg Yakovchuk. Evaluating new requirements to pen-centric intelligent user interface based on end-to-end mathematical expressions recognition. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, pages 212–220, Cagliari Italy, March 2020. ACM.

Chapter 6

Appendices

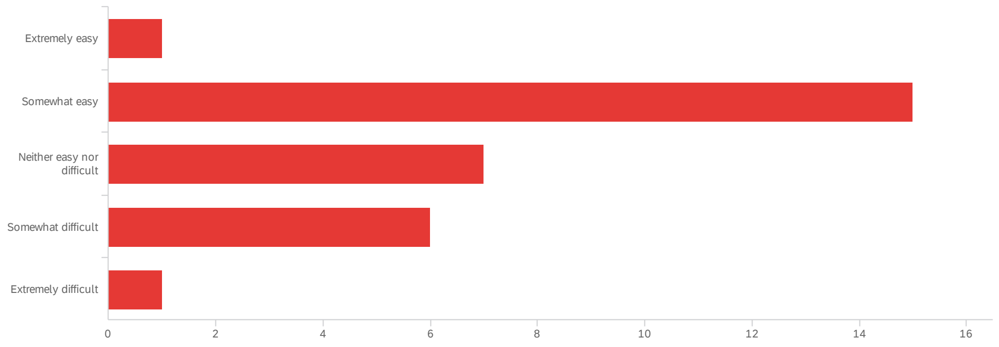
6.1 Appendix A - Correction Experiment Exit Questionnaire

Default Report

Exit Questionnaire

May 2, 2021 2:37 PM MDT

Q2 - How difficult were the provided formula editing tasks?

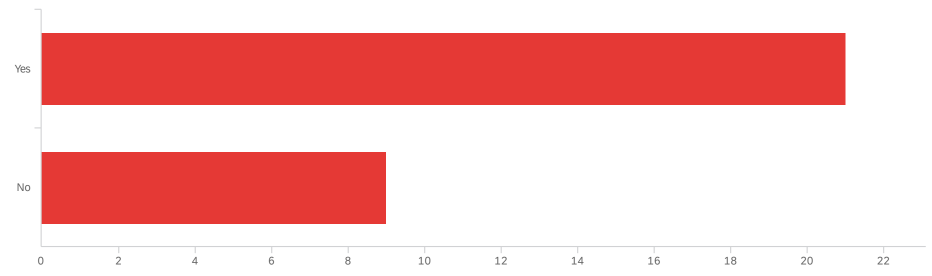


#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	How difficult were the provided formula editing tasks?	1.00	5.00	2.70	0.94	0.88	30

#	Field	Choice Count	
1	Extremely easy	3.33%	1
2	Somewhat easy	50.00%	15
3	Neither easy nor difficult	23.33%	7
4	Somewhat difficult	20.00%	6
5	Extremely difficult	3.33%	1
			30

Showing rows 1 - 6 of 6

Q3 - Have you ever used LaTeX for formulas before this study?

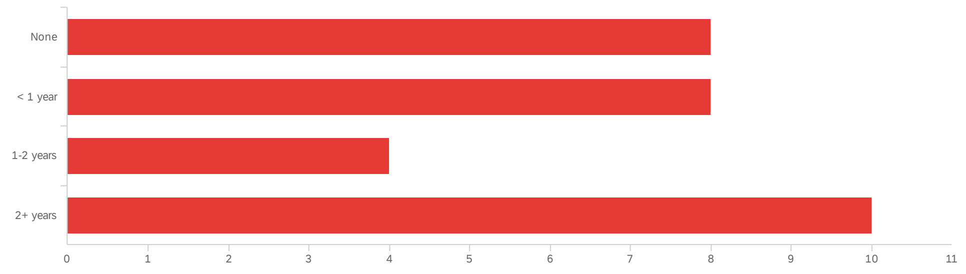


#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	Have you ever used LaTeX for formulas before this study?	1.00	2.00	1.30	0.46	0.21	30

#	Field	Choice Count	
1	Yes	70.00%	21
2	No	30.00%	9
			30

Showing rows 1 - 3 of 3

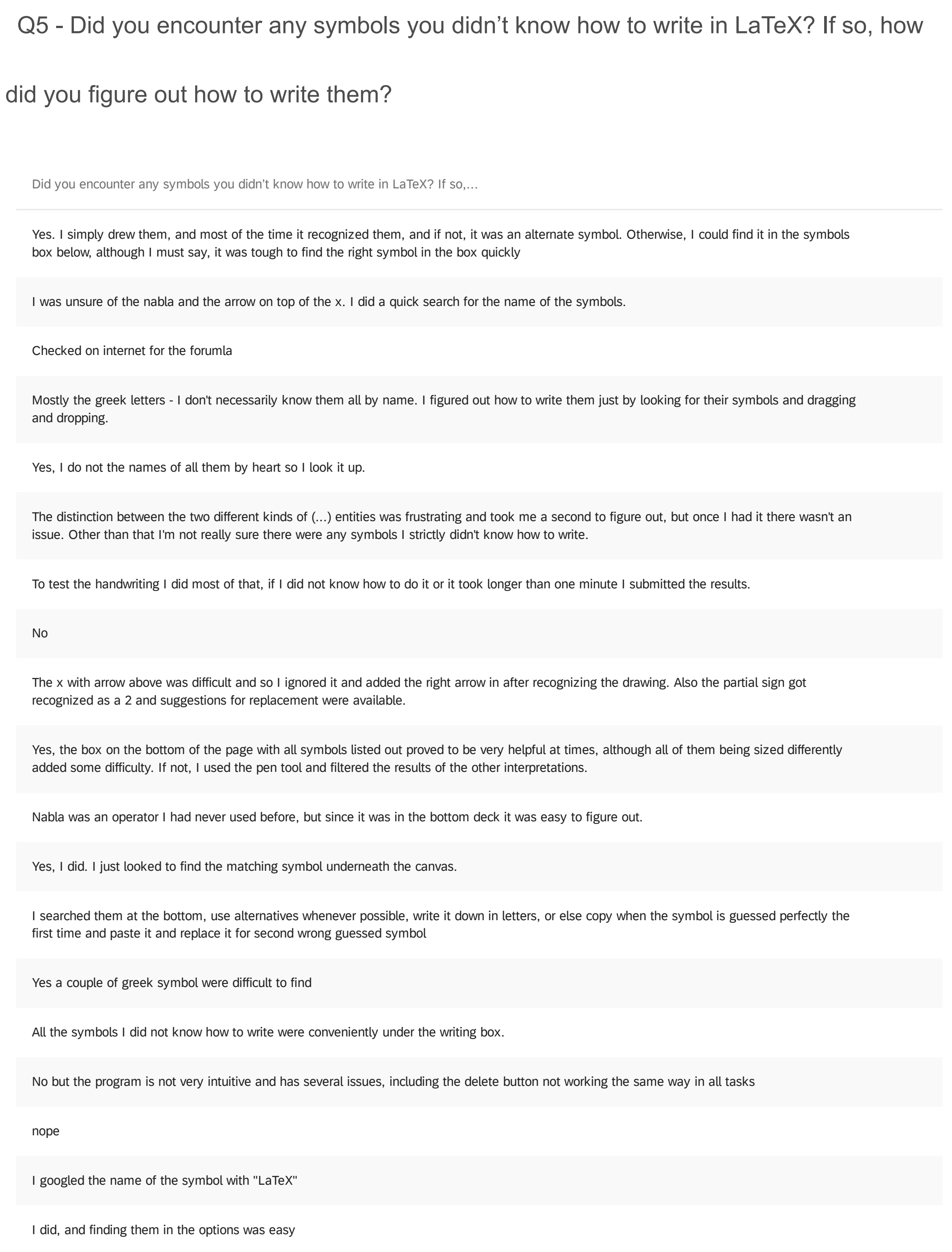
Q4 - Before this study, how much experience did you have using LaTeX?



#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	Before this study, how much experience did you have using LaTeX?	1.00	4.00	2.53	1.20	1.45	30

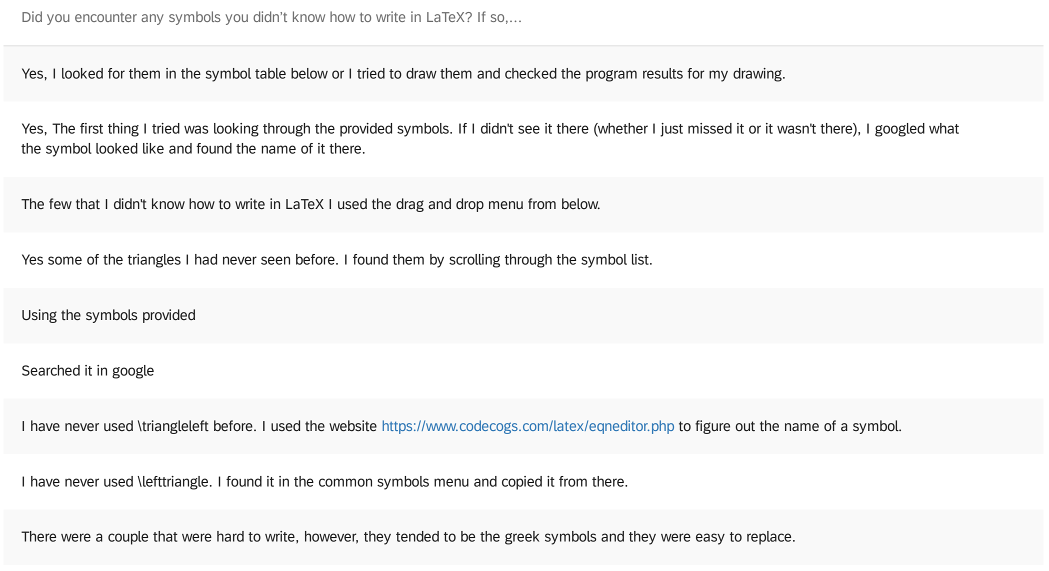
#	Field	Choice Count	
1	None	26.67%	8
2	< 1 year	26.67%	8
3	1-2 years	13.33%	4
4	2+ years	33.33%	10
			30

Showing rows 1 - 5 of 5



Q5 - Did you encounter any symbols you didn't know how to write in LaTeX? If so, how did you figure out how to write them?

Did you encounter any symbols you didn't know how to write in LaTeX? If so,...
Yes. I simply drew them, and most of the time it recognized them, and if not, it was an alternate symbol. Otherwise, I could find it in the symbols box below, although I must say, it was tough to find the right symbol in the box quickly
I was unsure of the nabla and the arrow on top of the x. I did a quick search for the name of the symbols.
Checked on internet for the forumla
Mostly the greek letters - I don't necessarily know them all by name. I figured out how to write them just by looking for their symbols and dragging and dropping.
Yes, I do not the names of all them by heart so I look it up.
The distinction between the two different kinds of (...) entities was frustrating and took me a second to figure out, but once I had it there wasn't an issue. Other than that I'm not really sure there were any symbols I strictly didn't know how to write.
To test the handwriting I did most of that, if I did not know how to do it or it took longer than one minute I submitted the results.
No
The x with arrow above was difficult and so I ignored it and added the right arrow in after recognizing the drawing. Also the partial sign got recognized as a 2 and suggestions for replacement were available.
Yes, the box on the bottom of the page with all symbols listed out proved to be very helpful at times, although all of them being sized differently added some difficulty. If not, I used the pen tool and filtered the results of the other interpretations.
Nabla was an operator I had never used before, but since it was in the bottom deck it was easy to figure out.
Yes, I did. I just looked to find the matching symbol underneath the canvas.
I searched them at the bottom, use alternatives whenever possible, write it down in letters, or else copy when the symbol is guessed perfectly the first time and paste it and replace it for second wrong guessed symbol
Yes a couple of greek symbol were difficult to find
All the symbols I did not know how to write were conveniently under the writing box.
No but the program is not very intuitive and has several issues, including the delete button not working the same way in all tasks
nope
I googled the name of the symbol with "LaTeX"
I did, and finding them in the options was easy



Did you encounter any symbols you didn't know how to write in LaTeX? If so,...

Yes, I looked for them in the symbol table below or I tried to draw them and checked the program results for my drawing.

Yes, The first thing I tried was looking through the provided symbols. If I didn't see it there (whether I just missed it or it wasn't there), I googled what the symbol looked like and found the name of it there.

The few that I didn't know how to write in LaTeX I used the drag and drop menu from below.

Yes some of the triangles I had never seen before. I found them by scrolling through the symbol list.

Using the symbols provided

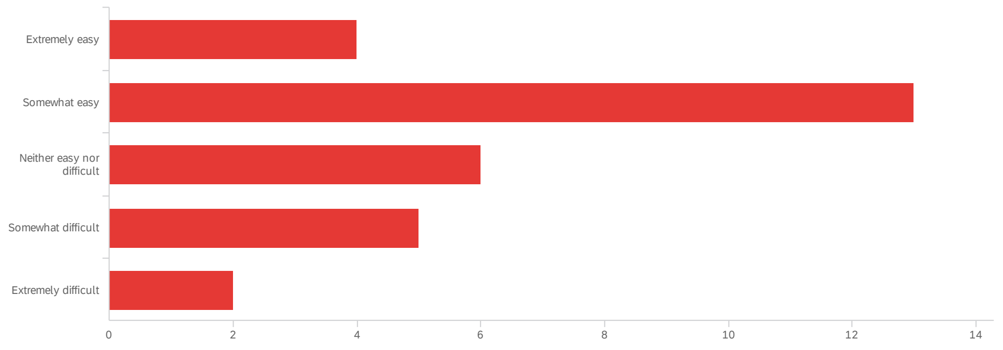
Searched it in google

I have never used \triangleleft before. I used the website https://www.codecogs.com/latex/eqneditor.php to figure out the name of a symbol.

I have never used \lefttriangle. I found it in the common symbols menu and copied it from there.

There were a couple that were hard to write, however, they tended to be the greek symbols and they were easy to replace.

Q6 - How easy to use was the handwriting interface?

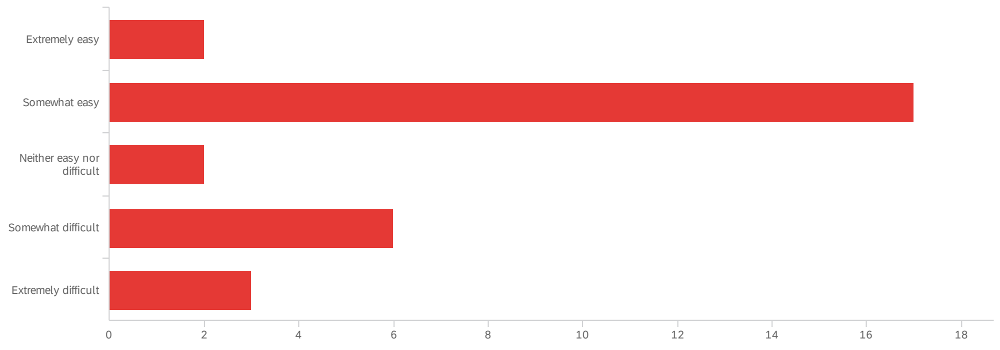


#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	How easy to use was the handwriting interface?	1.00	5.00	2.60	1.11	1.24	30

#	Field	Choice Count	
1	Extremely easy	13.33%	4
2	Somewhat easy	43.33%	13
3	Neither easy nor difficult	20.00%	6
4	Somewhat difficult	16.67%	5
5	Extremely difficult	6.67%	2
			30

Showing rows 1 - 6 of 6

Q7 - How easy was it to correct the formula if it didn't display what you expected?

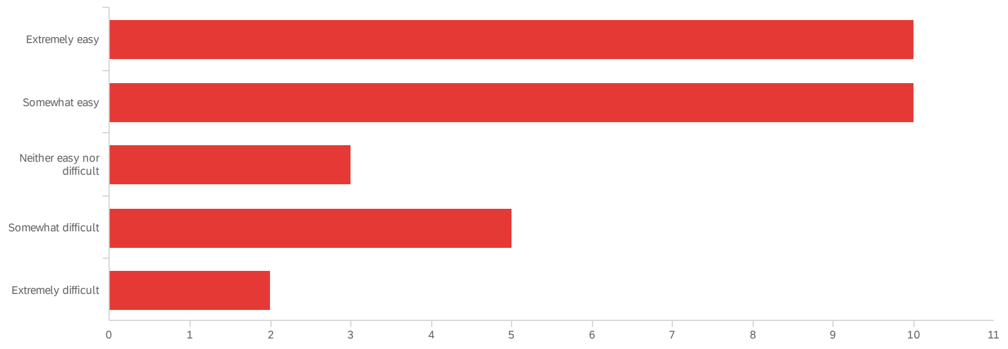


#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	How easy was it to correct the formula if it didn't display what you expected?	1.00	5.00	2.70	1.16	1.34	30

#	Field	Choice Count	
1	Extremely easy	6.67%	2
2	Somewhat easy	56.67%	17
3	Neither easy nor difficult	6.67%	2
4	Somewhat difficult	20.00%	6
5	Extremely difficult	10.00%	3
			30

Showing rows 1 - 6 of 6

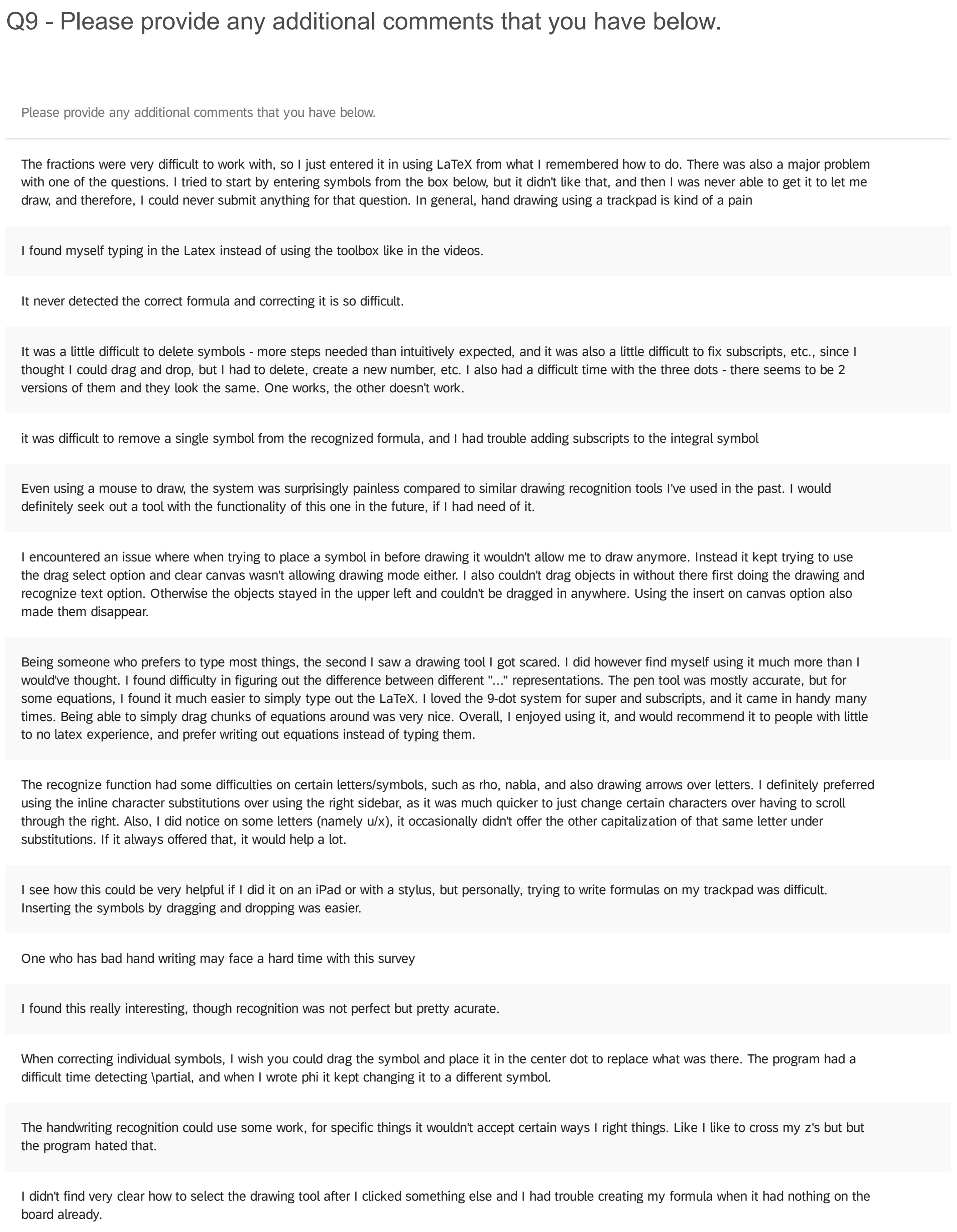
Q8 - How easy was it to correct individual symbols in the formula?



#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	How easy was it to correct individual symbols in the formula?	1.00	5.00	2.30	1.27	1.61	30

#	Field	Choice Count	
1	Extremely easy	33.33%	10
2	Somewhat easy	33.33%	10
3	Neither easy nor difficult	10.00%	3
4	Somewhat difficult	16.67%	5
5	Extremely difficult	6.67%	2
			30

Showing rows 1 - 6 of 6



Q9 - Please provide any additional comments that you have below.

Please provide any additional comments that you have below.

The fractions were very difficult to work with, so I just entered it in using LaTeX from what I remembered how to do. There was also a major problem with one of the questions. I tried to start by entering symbols from the box below, but it didn't like that, and then I was never able to get it to let me draw, and therefore, I could never submit anything for that question. In general, hand drawing using a trackpad is kind of a pain

I found myself typing in the Latex instead of using the toolbox like in the videos.

It never detected the correct formula and correcting it is so difficult.

It was a little difficult to delete symbols - more steps needed than intuitively expected, and it was also a little difficult to fix subscripts, etc., since I thought I could drag and drop, but I had to delete, create a new number, etc. I also had a difficult time with the three dots - there seems to be 2 versions of them and they look the same. One works, the other doesn't work.

it was difficult to remove a single symbol from the recognized formula, and I had trouble adding subscripts to the integral symbol

Even using a mouse to draw, the system was surprisingly painless compared to similar drawing recognition tools I've used in the past. I would definitely seek out a tool with the functionality of this one in the future, if I had need of it.

I encountered an issue where when trying to place a symbol in before drawing it wouldn't allow me to draw anymore. Instead it kept trying to use the drag select option and clear canvas wasn't allowing drawing mode either. I also couldn't drag objects in without there first doing the drawing and recognize text option. Otherwise the objects stayed in the upper left and couldn't be dragged in anywhere. Using the insert on canvas option also made them disappear.

Being someone who prefers to type most things, the second I saw a drawing tool I got scared. I did however find myself using it much more than I would've thought. I found difficulty in figuring out the difference between different "..." representations. The pen tool was mostly accurate, but for some equations, I found it much easier to simply type out the LaTeX. I loved the 9-dot system for super and subscripts, and it came in handy many times. Being able to simply drag chunks of equations around was very nice. Overall, I enjoyed using it, and would recommend it to people with little to no latex experience, and prefer writing out equations instead of typing them.

The recognize function had some difficulties on certain letters/symbols, such as rho, nabla, and also drawing arrows over letters. I definitely preferred using the inline character substitutions over using the right sidebar, as it was much quicker to just change certain characters over having to scroll through the right. Also, I did notice on some letters (namely u/x), it occasionally didn't offer the other capitalization of that same letter under substitutions. If it always offered that, it would help a lot.

I see how this could be very helpful if I did it on an iPad or with a stylus, but personally, trying to write formulas on my trackpad was difficult. Inserting the symbols by dragging and dropping was easier.

One who has bad hand writing may face a hard time with this survey

I found this really interesting, though recognition was not perfect but pretty acurate.

When correcting individual symbols, I wish you could drag the symbol and place it in the center dot to replace what was there. The program had a difficult time detecting \partial, and when I wrote phi it kept changing it to a different symbol.

The handwriting recognition could use some work, for specific things it wouldn't accept certain ways I right things. Like I like to cross my z's but but the program hated that.

I didn't find very clear how to select the drawing tool after I clicked something else and I had trouble creating my formula when it had nothing on the board already.

Please provide any additional comments that you have below.

There were a few symbols that the handwriting really didn't like, they were mostly the math symbols that look like various triangles. Other than that it was pretty easy to get it to recognize what I wanted it to say or there were quick fixes to it

Using touchscreen as opposed to a touchpad made it interesting, as some of the menus did not seem to want to work properly (such as the alternate symbol menu and the drag and drop). Overall, I liked the system. There are many times when being able to hand write the symbol and then correct things that are incorrectly read would be very useful, especially when doing long equations.

I did not realize there were two "..." symbols leading to me having incorrect answers for some of the tasks though I thought they looked the same. Also I feel that it would be faster in many cases to type out the LaTeX rather than use the hand writing interface.

The software works pretty well. Found it fun when it recognized my formula exactly what was expected. Minor bugs I faced: 1)Undo isn't working when I select it. Tapping it multiple times worked. 2) While writing the formula using the pen, I clicked on a symbol. I couldn't write again. I couldn't find the option to select the pen and continue. 3) Couldn't complete 2 formulas that required me to enter dots. The formula seemed to match what was expected.

This took me over a hour and a half to complete. Not “30 minutes” as you guys said it would take.

No comments.

On Task 4, I tried to edit the input by deleting something, but the whole input disappeared. When I cleared the canvas to start again, my cursor was stuck in selection mode, and I could not write anything. There was no restart drawing button either that I could find.

There were a couple times where the website glitched out. There were also a couple of times where, even though the formula I had formed looked identical to the target formula, it wouldn't be recognized as completed (the one with the bunch of A's for instance). There were also a couple times where like a plus sign would be slightly lower than it was in the target but I couldn't figure out how to correct it and match them. Overall the system is pretty intuitive and cool. I kinda wish that, once I had written the formula and got it to convert my handwriting to the symbols, that I could rewrite only a portion of it (for instance if it got the first half of the equation correct but the second half wrong). Also, the conversion of handwriting to symbols was a bit sporadic sometimes. For example, sometimes I would accidentally make a stray dot away from the equation and sometimes it would pick it up and add three dots to the end of the equation and other times it would not pick it up at all.

End of Report

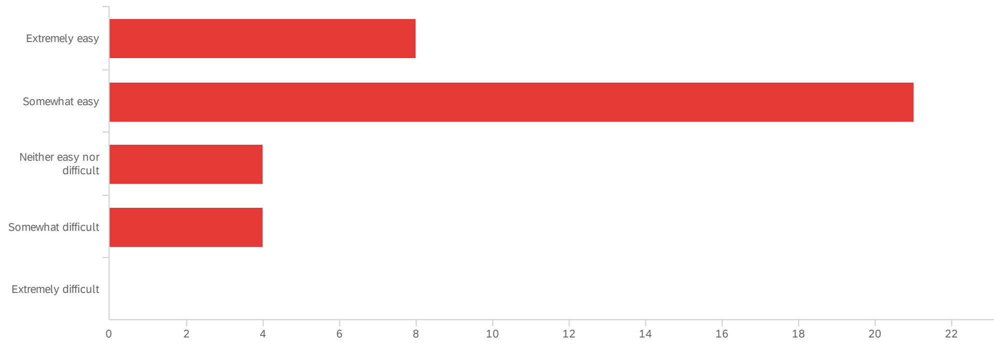
6.1.1 Appendix B - Handwriting vs. No Handwriting Experiment Exit Questionnaire

Default Report

MathDeck Exit Questionnaire

May 2, 2021 2:39 PM MDT

Q2 - How difficult were the provided formula editing tasks?

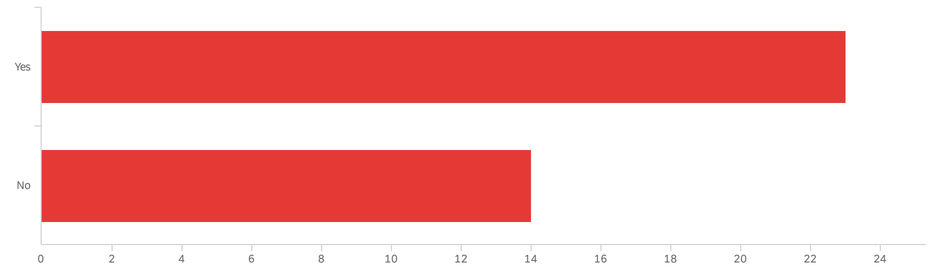


#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	How difficult were the provided formula editing tasks?	1.00	4.00	2.11	0.86	0.75	37

#	Field	Choice Count	
1	Extremely easy	21.62%	8
2	Somewhat easy	56.76%	21
3	Neither easy nor difficult	10.81%	4
4	Somewhat difficult	10.81%	4
5	Extremely difficult	0.00%	0
			37

Showing rows 1 - 6 of 6

Q3 - Have you ever used LaTeX for formulas before this study?

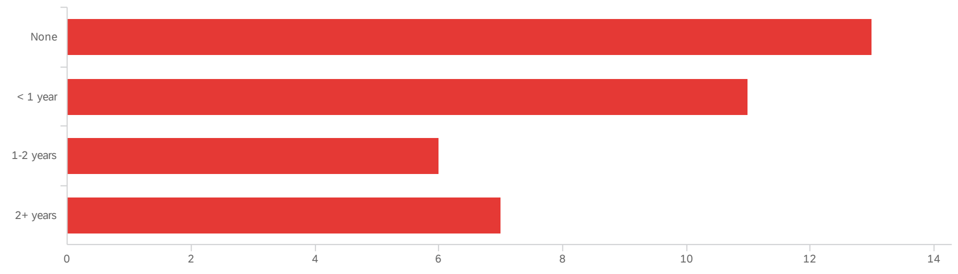


#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	Have you ever used LaTeX for formulas before this study?	1.00	2.00	1.38	0.48	0.24	37

#	Field	Choice Count	
1	Yes	62.16%	23
2	No	37.84%	14
			37

Showing rows 1 - 3 of 3

Q4 - Before this study, how much experience did you have using LaTeX?



#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	Before this study, how much experience did you have using LaTeX?	1.00	4.00	2.19	1.11	1.23	37

#	Field	Choice Count	
1	None	35.14%	13
2	< 1 year	29.73%	11
3	1-2 years	16.22%	6
4	2+ years	18.92%	7
			37

Showing rows 1 - 5 of 5

Q5 - Please describe the strategies you used to edit the starting formula into the target formula when handwriting was available.

Please describe the strategies you used to edit the starting formula into t...
minimally use handwriting. mostly enter everything in the typing panel on the right. for fractions, the x/y predefined function was inserted and further edited as per requirement. similar approach was taken was any terms that were squared (the x^2 function in from the tab at the bottom was inserted. For symbols that could not be typed directly from the keyboard, drag and drop from the bottom tab was used.
things that were intuitive to type I found easier to type, but for more things that weren't so easy to type (fractions, integrals, etc) handwriting was most efficient.
I wrote the whole formula by hand, and then used the editing features to adjust it
I would select the values that needed to be modified and replace them, click on a point on a value to add something in relation to it, or drag in a symbol to include. I did not use handwriting.
I tried to handwrite the formula first, then corrected any mistakes made by the recognition tool. When fractions were involved, it would sometimes pick them up as subscripts, so I would build the formula in pieces and use the fraction bubble.
I used handwriting to copy the equation as I saw it, and changed the letters that were not correctly recognized
I tried writing the formulas by handwriting and then tried to insert them. I also tried to insert into the new tab and then move it to latex column. However, for one particular task, I was not able to do it.
I wrote the equations and hit recognize. I then edited the pre-existing equation using the editing tools to match it to the required formula.
I used handwriting if there were any symbols that I did not know such as the definite integral or when I had to input numbers when they were in fractions.
When handwriting was available, I used handwriting first. It recognized my handwriting most of the time, but even when it didn't it was easy to change it using the suggested alternate symbols. The only time it got it wrong was that it recognized my phi as what I think is \varphi, and my 2 as an 'a'.
Handwriting worked depending on my penmanship working on a laptop, in addition to whether if i am in a rush. Handwriting worked 80% of the time.
I used handwriting to write most of the part after which I corrected minor mistakes using latex. I felt that this approach was much easier
I typed it out when I thought typing would be faster than handwriting. If I didn't know a quick way to type it I would use handwriting
I tried to write as clearly as possible and then clicked individual wrong guesses to edit them. The feature that would provide alternative guesses was helpful.
I would start with handwriting and try a couple times until it was close enough. Then I would replace symbols manually by typing if I could, or by dragging in the appropriate symbol, often Greek letters.
mostly handwriting; sometimes the text editor

Please describe the strategies you used to edit the starting formula into t...
I tried to first use handwriting to draw the formula, then click on the recognized formula to modify the incorrect symbols.
I would use it only for fractions
I tried to split the formulas into "chunks" because sometimes it recognized part of it well and part of it poorly and I didn't want to redraw the whole thing.
First using the handwriting function to write most of the function, and then using the replace and lift buttons to make small edits
The handwriting made longer Latex actions like fractions and greek letters easier.
i used handwriting to write the formula first then if i noticed that it had trouble recognizing certain drawings i would clean it up after inserting it
I never used handwriting. I began each formula by trying to type as much out normally as possible then picked out the letters and equations (particularly the division symbol, because / is not treated the same).
I usually just rewrote the formula myself as that was the easiet.
Most of the time I handwrote the formula then pressed recognize. I replaced any wrong symbols by clicking in the middle of the wrong ones and either manually typing in the right number or letter variable, or I went into the editing panel on the right and edited them there. However, any greek or logic symbol I almost always dragged and dropped into the main formula window even when I could handwrite it.
When it is come to handwriting become available, I usually focus on write to the full formula as I can and less depends on the canvas selection.
When handwriting was available, I still defaulted to using LaTeX. For the one case of the upside-down "A," I would have used the handwriting option if I did not immediately find it on the menu of symbols.
I tried using the handwriting tool, but when it didn't pick up on my handwriting, I simply wrote as much as I could into the editor before looking for symbols in the lower section and moved things around to get the proper notation.
Click and replace a single symbol with with cursor looking button
I almost entirely edited the formula using handwriting and then I would fix any mistakes that would appear when I clicked recognize. One of the main fixes would be "X" to "x".
I did not use the handwriting feature. Due to mouse control, I feel this can be hard to control and easy to mess up. If I had an electronic drawing pad, then I would use that feature more.
I used the handwriting tool as a starting point, then made quick spot corrections where my handwriting wasn't read correctly. In most cases I elected to use this approach, since I type rather slowly.
I preferred to use anything but handwriting, as I don't have a drawing tablet it was very slow to use. I still preferred it over dragging in symbols one-by-one though, as naturally finding each symbol took a while.
Make the handwriting precise
I usually just wrote it all out, changed anything that the program mis-recognized, and then hit submit.
I first chalked down the easy-to-write material and then decided to go with the difficult sections of the formula. One inference I deduced was that it would have been easier and much more convenient to draw if I had a mouse or a stylus. Otherwise, it gets into a lot of re-writing.

Please describe the strategies you used to edit the starting formula into t...

When handwriting was available it was a lot easier to just that and then go through and correct the few mistakes It made. There were a few times I didn't use the handwriting feature when it was available but that was only when I thought I knew where all the needed functions were.

Q6 - Please describe the strategies you used to edit the starting formula into the target formula when handwriting was not available.

Please describe the strategies you used to edit the starting formula into t...
similar approach as described above.
the formula sheet at the bottom was my go-to when I couldn't use handwriting. I would use those formulas to get the symbols I needed, and then replace the values with the text editor.
When the LaTeX was available, and there were no fractions/sums, then I prefered to just type the equation out in the LaTeX field. If there were fractions, then I used the editing panel to do most of the editing
I would select the values that needed to be modified and replace them, click on a point on a value to add something in relation to it, or drag in a symbol to include.
When I could remember the LaTeX command, I would type them in; otherwise I would look in the lower panel
I typed the formula into the bar and used the presets for formatting and non-numeric symbols (eg: greek letters)
When the handwriting was not available, I selected the symbols, clicked on the typed letter and tried to replace, insert to right, insert super, insert sub.
I used the symbol selection area and typed the formulas in as well.
I mainly looked for the symbols and dragged them onto the editing screen. If I knew the latex, I would put it in but this is mostly when the symbols were easy like basic arithmetic operations.
When handwriting was no available, I found it quickest to write out the formula in LaTeX since I am very familiar with the notation.
Without the handwriting tool it make task slightly more difficult. I'd have to first drop in fractions then edit the basic fraction, which is easier then figure out how to write in fractions in the text part.
I used latex to write Then I used the symbols available to enter in latex. I felt that using latex was a bit more effort than using the handwriting panel.
I would type whatever I could, but sometimes I had to drag formulas or greek letters in when I didn't know how to type them
I mostly used Latex. It was a pain to drag easy numbers over or edit, so when it was just a symbol or letter, i used LaTex.
I would use the LaTeX editor if I knew what to do, or would drag up symbols from the bottom panel.
text editor and inserting symbols
I prefer to directly write latex code.
Latex
I just typed out the formula when I could

Please describe the strategies you used to edit the starting formula into t...
Drag and drop function operators from the bottom, and then editing the LaTeX to include the necessary numbers
When I couldn't draw, if I knew what the command was I would just type it, but if I didn't remember it I would look through the submenus for 'Logic' or 'Greek'
i wrote everything that didnt require additional symbols first then used the blue dots to add super/sub and dragged the greek letters and symbols not on my keyboard like integral to the blue dots
I wrote the function normally (see handwriting available answer)
I sometimes used LaTeX to type it in other times I dragged in the appropriate character usually when it was a greek letter or just typed in the replacement if it was something I could type.
Any easily typed letter variable, number, parenthese, or equality symbol I would type in the edit panel on the right. I would then drag and drop any greek or logic symbol into the main formula panel. If there was a repeating greek/logic symbol I would select it, copy, and drag it to the other spot where it was supposed to be. Another strategy I used was with exponents, in which if it was an easily typed number or letter I would type it in the editing panel on the right. If it was a greek/logic symbol it would be dragged and dropped onto where it was supposed to be.
I will focus on completing the part where it is more complicated by picking what the canvas selection has and edit it. Then, add a new part to the formula by click the farthest right side of the end. if there is more but nothing left to do, I will use a new tab containing a separate part and plug it into the main tab with it.
As mentioned, I never defaulted to using the handwriting option when I could do otherwise.
When handwriting was not available, I wrote as much as I could into the editor before looking for symbols in the lower section and moving things around to get the appropriate notation.
Straight Latex, write it out
I would type some of the formulae out, specifically anything that I could easily do without prior knowledge of LaTeX. If there was a fraction, greek symbol, or an integral, I would just drag and drop them in rather than typing them. I tried to figure out how to the code for a fraction but I kept on using a forward slash instead of a backslash and once I realized that it wasn't even used anymore. For a quick example of what I mean: for one of the last examples with lambda and delta, I wrote out f(x)=f(x) and then added the greek symbols after.
Depending on the formula structure, I would start in the LaTex editing window. If the formula started to get messy in the LaTex window, I would move to the interactive window to finish the formula.
I used the editor to add the LaTeX that I knew, and dragged appropriate symbols onto the canvas when I did not recognize or know how to write them.
I always preferred to write the TeX formula in directly when I could, and I found the ability to put some TeX into a particular location visually quite nice.
Latex formulation
If there was a symbol not on my keyboard, I tried to select it first from the bottom array of symbols; if it was there, I typed it all out to the best of my ability. If something required a fraction or a square root, I selected it from the array as well.
when the starting formula was not available, I preferred to use latex commands. For those portions that I didn't remember, I preferred to use the drag and drop and then edit it if necessary.

Please describe the strategies you used to edit the starting formula into t...

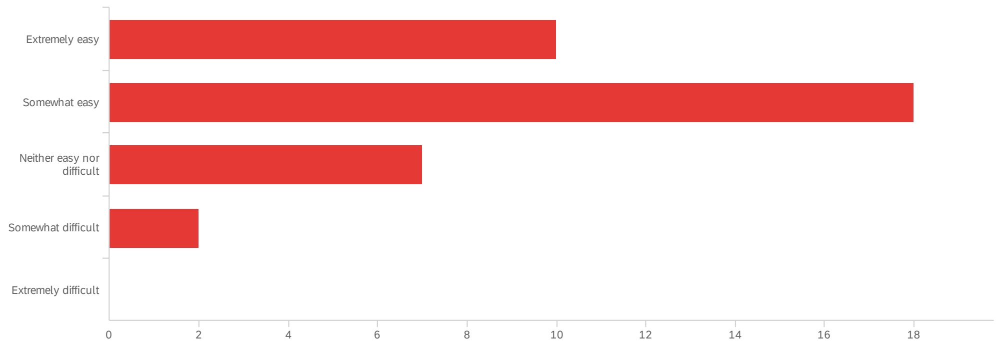
When handwriting wasn't available I just typed in/ inserted as much as I knew how to type or things I could easily find and put placeholders in for the symbols I didn't know how to insert yet. After I had most of the equation set up I went through and replaced or inserted things that were wrong/ missing.

Q7 - Did you encounter any symbols you didn’t know how to write in LaTeX? If so, how did you figure out how to write them?

Did you encounter any symbols you didn't know how to write in LaTeX? If so,...
i simply inserted them into the formula from the functions tab at the bottom. i used the search bar to find any symbols i was unsure of, by typing in the alphabet the symbol most resembled in the search bar.
Yes, I encountered quite a few. I would look to other tools such as the handwriting tool or the formulas & symbols if I didn't know how to type it.
Yes, I located them in the editing panel, and dragged them in, and saw what the LaTeX output was
I encountered some symbols that I didn't know the names of, like the for all operator. I had to look through the available symbols to find it.
I initially couldn't remember the \frac command; using the bubble on the lower panel helped. I also couldn't remember the name of a Greek symbol so I used the lower bubble as well.
I searched through the given symbols until I found the matching one
I could not figure out the 'belong all" symbol.
Yes I googled the symbol and copied and pasted it in the text editor.
Yes. I either handwrote them but if I did not get it, I would look it up and replace it by dragging.
I did not encounter symbols I did not know how to write in LaTeX.
The I did not figure out how to write symbols.
Yes, there were symbols that I didn't know how to write in latex, I searched for them in the symbols section and then dragged and dropped them in the editor box in such cases.
I did encounter those symbols. I didn't figure out how to write them, I just dragged them from the helpful menu below.
There were a few Greek symbols I didn't know. I either had the handwriting guess or I looked in the symbols area.
I wasn't sure how to write for all (upside down A) so I filtered the bottom panel to find it.
yes, either used the symbols given below or handwriting
No
Yes, i used the symbols given beow
I couldn't remember how to write a root symbol but I found it in the keyboard after a few minutes of frustrated trial and error. I also couldn't remember how to write fractions, but I didn't even try with that and went straight to the keyboard.

Did you encounter any symbols you didn't know how to write in LaTeX? If so,...
I was not sure of some of the greek letter names, but hovering over them at the bottom provided their names
I wasn't sure the commands for 'For all' or 'Integral', so I would draw them if I had the option because I know what they look like, but otherwise I searched by type like 'Logic' for the 'For All'
i didnt know how to write anything in latex but i realized that i could hover over the symbols in the bottom panel and it showed how to write them in latex
Yes. I picked them out from the selection below. Also, / should be sufficient for a fraction.
The greek ones couldn't be typed without a code so I dragged them in from the greek letter bank.
There was the upside-down capital A logic symbol I've never seen before. I tried hand drawing it but the LaTeX program wouldn't recognize it after 3 attempts. So I gave up on hand drawing it and went and searched through the symbols pages for a minute or two looking for it and found it eventually.
There is one character I didn't know in LaTex and I was able to solve it by initially using handwriting which is the easiest to access. Once that character is done, I then simply highlight and copy it for re-use on the next need.
I did not know the code for the upside-down "A." After finding it on the menu of symbols, the editing window simply displayed that character and not the code, so I did not learn what the code was.
There were symbols that I did not know how to write, so I figured out how to get them by looking through every symbol in the symbols list at the bottom until I found the right one.
The upside down looking A. I searched for it visually in the symbols given
Square root, greek symbols, fractions I saw that a lot of the greek symbols were just the greek symbols but I would just drag and drop them in rather than googling the symbol for pi and then copying/pasting it into the code. For square roots, I saw the code and continued to drag and drop a square root x in, and I would change the x as needed. For factions, I tried to learn the code because it came up a lot but as I said earlier I did a forward slash instead of a backslash, and by the time I realized that there were no more questions with fractions. So I just dropped x/y in and changed x and y as needed.
I usually never remember any symbol besides the common Greek letters. I would always want a symbol bank for easy of access to the symbols.
I relied solely on the provided symbols when handwriting was not available, but in the cases that the handwriting tool was enabled, I quickly wrote them out and added them.
I found them in the symbol list.
Drag and drop
Most of the symbols I already knew how to draw, and if I didn't I could just guess and correct it later with the symbol array.
yes, there were many. I would either use drag and drop and then edit them, or else I would use the draw option if available.
When I couldn't find the symbol I used the handwriting feature, but most of the time I just spent an extra minute or so looking through the bottom area to insert what I was looking for.

Q8 - When using handwriting, how easy to use was the handwriting interface?

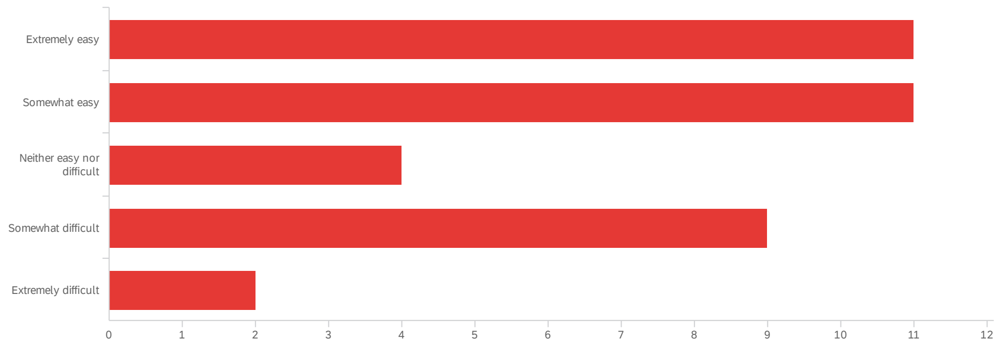


#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	When using handwriting, how easy to use was the handwriting interface?	1.00	4.00	2.03	0.82	0.67	37

#	Field	Choice Count	
1	Extremely easy	27.03%	10
2	Somewhat easy	48.65%	18
3	Neither easy nor difficult	18.92%	7
4	Somewhat difficult	5.41%	2
5	Extremely difficult	0.00%	0
			37

Showing rows 1 - 6 of 6

Q9 - When using handwriting, how easy was it to correct the formula if it didn't display what you expected?

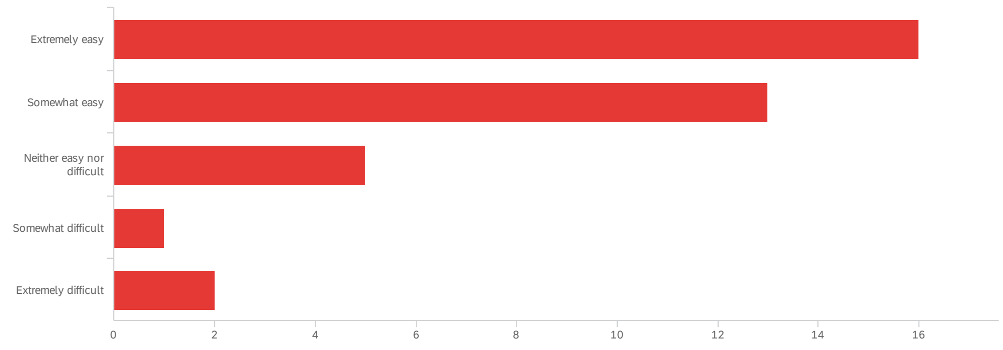


#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	When using handwriting, how easy was it to correct the formula if it didn't display what you expected?	1.00	5.00	2.46	1.29	1.65	37

#	Field		Choice Count
1	Extremely easy	29.73%	11
2	Somewhat easy	29.73%	11
3	Neither easy nor difficult	10.81%	4
4	Somewhat difficult	24.32%	9
5	Extremely difficult	5.41%	2
			37

Showing rows 1 - 6 of 6

Q10 - When using handwriting, how easy was it to correct individual symbols in the formula?



#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	When using handwriting, how easy was it to correct individual symbols in the formula?	1.00	5.00	1.92	1.07	1.16	37

#	Field	Choice Count	
1	Extremely easy	43.24%	16
2	Somewhat easy	35.14%	13
3	Neither easy nor difficult	13.51%	5
4	Somewhat difficult	2.70%	1
5	Extremely difficult	5.41%	2
			37

Showing rows 1 - 6 of 6

Q11 - Please provide any additional comments that you have below.

Please provide any additional comments that you have below.
I did not use handwriting because I was confused as to how it was implemented.
It felt like handwriting mode often struggled to understand fractions. Additionally - when placing a symbol next to a subtraction or fraction bar, the top-right, right, and lower-right bubbles were very close. I attempted to record via zoom meeting but I'm not sure if I did it correctly.
The interface of the software can be better. Some responses are slow.
I really like that you can enter a formula via handwriting, LaTeX, and by dragging symbols and notation. I feel like it caters to a very wide audience. It's nice to have regular LaTeX editing in there for people who are acquainted LaTeX. Having the "deck" of symbols to drag makes it accessible and useful for people who aren't familiar with LaTeX. Great work!!
Sometimes, the handwriting editor didn't recognize what I had written, but it just happened 2-3 times. Otherwise, the experience was smooth.
Sometimes it felt like certain symbols would just disappear or change when I went to the main formula after dragging in a handwritten sample. It can be hard to click on the dot for horizontal lines like minus or the fraction symbol. Also it would be nice to have a quick way to type fractions. This is a pretty awesome program though, I'd love to use it once it's released!
This is an excellent tool! Can't believe I haven't used it before. I might start using it in my coursework and I'll be sure to share it with others.
1) The website loading is too slow. 2) The code showing in Latex box is a bit lagged behind my typing speed, feels like the typing is not fluent and would cause typing error.
prefer latex over handwriting. much faster
The drawing was a little difficult to use because I am on a laptop using a trackpad.
if it got the handwriting wrong the first time it seemed like it would never get it right no matter how clearly i tried to write what i wanted so eventually i stopped trying to rewrite the formula and just instantly clicked insert no matter what it recognized and fixed it afterward. it made the ability to retry drawing feel useless.
You really ought to fix the division symbol. a/b = a above a horizontal line b below the same horizontal line, but it was too hard to type. I might be the same with * instead of an x for multiplication, but there wasn't any questions like that.
I know it was for research purposes but I actually found it quite enjoyable.
Writing integrals is really difficult without the handwriting feature. I had to write the numbers separately and drag/lift them and place them on its spot on the integral symbol. Which, being a brand new LaTeX user was kind of hard/frustrating. I also wanna note that because I was screen recording using zoom my computer was very slow and laggy which made it hard to aim to place anything in its right spot sometimes. You can view it on the recording in general and on certain formulas that had a bunch of numbers/variables/symbols in close proximity.
The handwriting tool was very bad at picking up on my admittedly sloppy handwriting, but it made it hard to use for even simple formulas, and instead of misreading things simply, the misreadings could turn one letter into three. Even for complex formulas, I found it easier to just use the editor than to struggle with the handwriting tool's recognition software.
I don't know if it's just for me but when I was dealing with fractions and didn't have handwriting available. I would change the x/y in the editing section on the coding part, and when I would go to type in whatever the formula had in the denominator and numerator, it would throw me into the search function for the functions (that you could drag and drop in from below) which was annoying.

Please provide any additional comments that you have below.

Adding in standard keyboard controls like ctrl+z for undoing or adding in the ability to use the delete key would be nice.

The dot interface became a little dubious when fractions were involved; it looked like superscript and subscript dots overlapped with the right dot on the fraction bar itself. I did not attempt to try this solution, but in some cases I would have preferred to replace items on the canvas by highlighting and typing out a replacement, like I would in the LaTeX editor, rather than selecting the middle dot and clicking the replace button.

Being able to pick from a searchable pop-up symbol list when inserting through a blue dot on the visual formula would have been nice

I was surprised how easily the LaTeX writing tool worked with my trackpad, since it's usually pretty finicky when I try to draw on it. Maybe it's the program, maybe it's just me getting better at it...

All of the options if available were very useful when it comes to an overall experience. Latex was preferred for equations that were hard to draw, Draw and drag-drop was convenient for confusing latex commands, and the option to add suffix and prefix was excellent.

Even when handwriting did not get the right symbol, it always had the right one in the replacement options which made it very easy to change out any wrong symbols. Handwriting probably would be the fastest way for most equations for me personally, but I always dislike drawing with my mouse so I might not always use it.

End of Report