Designing a Sketch Recognition Front-End: User Perception of Interface Elements

P. Wais, A. Wolin, M. Weiner, S. Sheehan, S. Harris and C. Alvarado

Department of Computer Science, Harvey Mudd College, Claremont, CA

Abstract

Programs that can recognize students' hand-drawn diagrams have the potential to revolutionize education by breaking down the barriers between diagram creation and simulation. Much recent work focuses on building robust recognition engines, but understanding how to support this new interaction paradigm from a user's perspective is an equally important and less well understood problem. We present a user study that investigates four critical sketch recognition user interface issues: how users integrate the process of triggering recognition into their work, when users prefer to indicate which portions of the diagram should be recognized, how users prefer to receive recognition feedback, and how users perceive recognition errors. We find that user preferences emphasize the importance of system reliability, the minimization of distractions, and the maximization of predictability. Based on the results of this study, we developed a sketch recognition interface to a sketch-based circuit design tool. Further user studies with this complete interface confirm the results of our original study and suggest new areas to explore.

Key words: User interface evaluation, pen-based interfaces, user-centered design, Wizard of Oz

1 Introduction

Many engineering classes rely on simulation technologies to help students understand the systems they design. Unfortunately, mouse and keyboard interfaces to these programs are cumbersome. Students in these courses draw countless diagrams on paper (or on a Tablet PC) because sketch-based diagram creation is quicker and more natural. In fact, many instructors require

Email address:
{pwais,awolin,mgweiner,ssheehan,sharris,calvarado}@hmc.edu (P. Wais, A. Wolin, M. Weiner, S. Sheehan, S. Harris and C. Alvarado).

students to draw diagrams on paper before entering designs into simulation software so that students focus on their design, not on the software interface.

Systems that can recognize and simulate students' hand-drawn sketches have the potential to lower the cognitive barrier between students and simulation software. However, these systems face their own interface challenges.

One challenge is how to allow users to trigger recognition and how to display recognition feedback. Feedback can be distracting in the early stages of design [13], but it also can aid recognition as it can help users adapt their drawing styles to match the system's expectations. Researchers have evaluated the usability of various recognition triggers and feedback mechanisms in isolation (e.g., [3, 23, 20]), but have not compared different techniques directly.

A second challenge is how to allow users to indicate which pieces of the diagram the system should attempt to recognize. Students' homework often consists of a mix of text, equations, and diagrams. Despite advances in parsing heterogeneous notes [26], a recognition system must receive only a single type of input to be practical. Most recognition systems allow the user to draw only one type of input (e.g., electrical circuits [10]), while a few systems allow the user to manually select pieces of their drawing to be recognized after they have finished drawing [20]. Again, little is known about which interface users prefer.

A third challenge is to minimize the impact of recognition errors on usability. Many systems reduce errors by placing constraints on the way users can draw symbols, for example enforcing that users pause between symbols or that strokes do not span more than one symbol (e.g., [25, 10, 14, 4]). Others focus on intuitive error correction mechanisms as a way of reducing the impact of recognition errors [21]. We believe that understanding users' tolerance for different types of recognition errors can help guide sketch recognition research by allowing researchers to focus on eliminating errors that have the biggest impact on usability.

To address the above challenges, we present the first direct comparison of critical free-sketch recognition user interface (UI) elements. Unlike prior research, which usually involves a qualitative analysis of a complete solution, we compare interface elements directly using a novel application of a Wizard of Oz evaluation methodology. Specifically, we evaluate UI mechanisms for triggering recognition, providing feedback, and separating recognizable from unrecognized data, and we examine users' perception of different types of recognition errors. Our results indicate that:

- Users prefer to trigger recognition after they are done drawing, even when the system produces errors.
- Users prefer to segregate pieces of the drawing (e.g., diagram vs. annotation)

at creation time rather than recognition time.
- Users want recognition feedback to transform and clutter their sketch as little as possible (even when they have completely finished sketching).
- Users prefer errors that are predictable and that allow them to understand how to modify their drawing style to prevent future errors.

One important user interface question that we do not address is how gesture-based or menu-based interfaces compare to free-sketch recognition interfaces. For example, users might prefer a reliable gesture-based system to an error-prone free-sketch recognition system. Although this question is important, we focus only on free-sketch recognition interfaces for two reasons. First, the students we talked to expressed a strong desire for a system that could simply transform diagrams they already produce for coursework into recognized circuit schematics; they did not want to have to learn a whole new language for interacting with the simulation software. Second, we cannot compare free-sketch recognition systems to gesture or menu-based systems until we better understand how to design effective user interfaces for these systems.

We used the results of our study to develop a novel interface to the sketch-based circuit design tool that our research group is developing. We describe the interface and present the results of an initial, informal user study that confirms many of the behaviors we observed in our initial study as well as suggests additional areas to explore with future studies.

2 User Interface Elements

This section describes the interface elements and error types we compared. We chose a subset of elements used in existing sketch recognition applications or suggested by six participants in a prestudy we performed. This set provides a restricted yet representative range of options.

We compared three different methods for triggering recognition: button, gesture and pause.

- **Button Trigger.** The user triggers recognition by tapping on a large interface button. This option is reminiscent of traditional recognition systems.
- **Gesture Trigger.** The user triggers recognition using a "check-tap" gesture (a check mark followed by a dot). We selected this element because many researchers posit the efficiency and naturalness of gestures over GUI widgets. Furthermore, this particular gesture seemed to be reliably recognized in informal tests, is not easily confused with diagram-relevant symbols, and is similar to the "circle-tap" gesture used in MathPad^2 [20].
- **Automatic (Pause) Trigger.** The system triggers recognition automati-

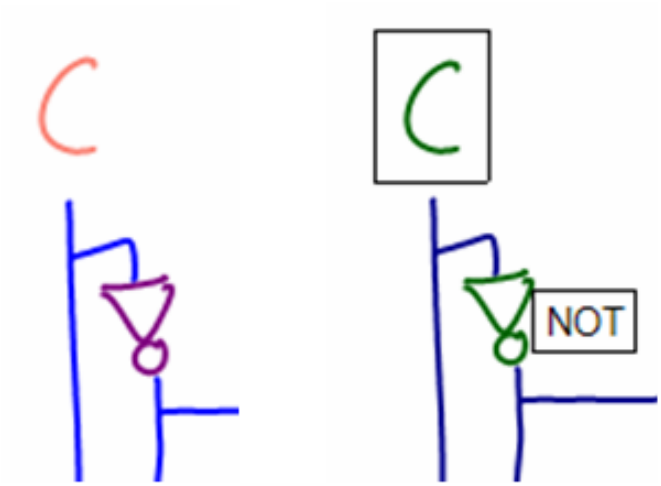


Fig. 1. Text and color feedback comparison.

cally after a brief pause (four seconds) in sketching. We chose a four-second pause to accommodate different student work paces.

The second issue we explored is how to allow the user to indicate which strokes the system should recognize. It is possible to provide separate sketching panels for notes and diagrams, but when users want to make annotations directly on the diagram, this setup is infeasible. We examined two techniques for allowing users to segregate recognizable strokes from unrecognized strokes:

- **Pre-separation: "Color" Tool.** This version of the interface requires users to separate domain strokes from annotation strokes as they draw by toggling between stylus modes using a button on the GUI. The system uses ink color to indicate the current mode: in sketch mode, the stylus draws black ink; in annotation mode, the stylus draws gray ink (hence we call this option the "color" tool).
- **Post-separation: Lasso Tool.** The user draws sketches and annotations freely, but then must lasso-select strokes to be recognized (similar to the interaction mechanism in MathPad^2 [20]).

The third issue we explored is how to display recognition feedback. We compared two different methods: color feedback and text labels (Figure 1). We rejected the idea of replacing the user's strokes with symbols based on lack of common interest in this idea during pre-studies and the results of previous work [13].

- **Color Feedback.** The system displays each recognized symbol in a unique color. In addition, users can see a text label by hovering the stylus over the strokes in the symbol.
- **Text-Label Feedback.** The system draws text labels next to recognized symbols.

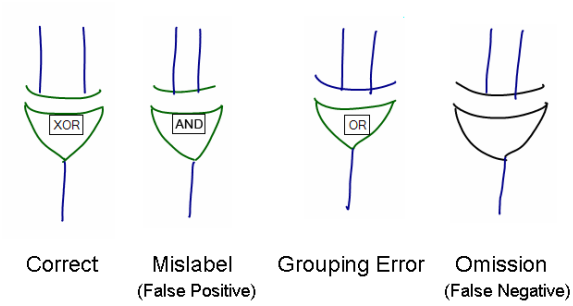


Fig. 2. Illustration of different error types.

Finally, we investigated how three types of common recognition errors impact the user experience: false positives, false negatives and stroke grouping errors (Figure 2). False positives occur when the system incorrectly identifies a symbol (e.g., labels an AND gate as an OR gate); false negatives (or *omissions*) occur when the system fails to identify a symbol; and grouping errors occur when the system incorrectly adds or removes adjacent strokes to or from a symbol.

3 Experimental Design

For this study, we limited our scope to digital circuit design in order to keep tasks consistent across all users so as to better understand how interface decisions (as opposed to domain variability) affect usability. Although focusing on a single domain limits the generality of our results somewhat, the style of the tasks we explored is representative of structured educational design tasks in many fields, such as the sciences and engineering.

3.1 Tasks and Participants

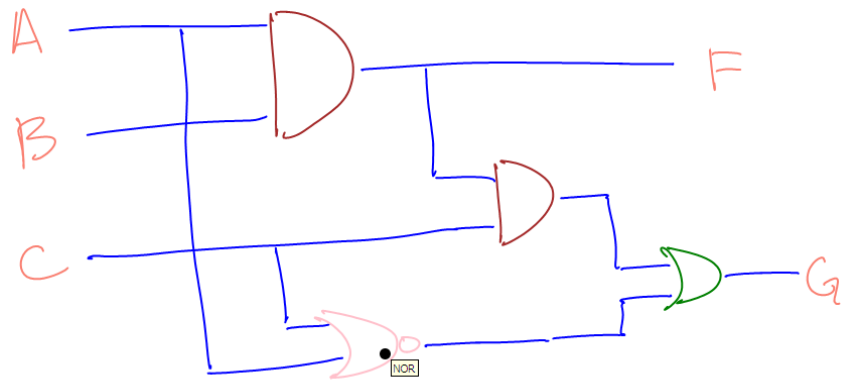
Users in our study designed circuits based on truth table function representations—a common task in an introductory digital design class. Figure 3(a) gives an example of one of these tasks. We allowed users to use any method to create the circuit, and we did not require them to simplify their circuit (i.e., implement it using the fewest number of gates), although they could if they wanted to. In some cases (described in Section 3.3) we also asked users to annotate the shortest and longest path through the circuit they designed.

We used a pre-study to design a set of uniformly difficult tasks. We asked participants to rate the difficulty of several tasks and selected for our study only those tasks that users rated as similarly difficult. Participants reported that these tasks were similar in difficulty to typical homework problems in an

Please draw a circuit diagram for the following truth table:

Input			Output	
A	B	C	F	G
0	0	0	0	1
0	0	1	0	0
0	1	0	0	1
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	1	0
1	1	1	1	1

(a) Truth table task



(b) A possible solution (with color feedback). The black dot denotes the user's stylus.

Fig. 3. A truth table task from our user study

introductory digital design course.

In total, nine people (six male and three female) participated in our formal tests (not including the pre-studies). All participants were Harvey Mudd College students who had taken an introductory digital circuit design class. All participants had used digital circuit simulation software (Xilinx) in their coursework. Six of these students had previous experience (more than an hour) with a Tablet PC, and five had taken notes on a Tablet PC during their digital design course (using Windows Journal or Microsoft OneNote).

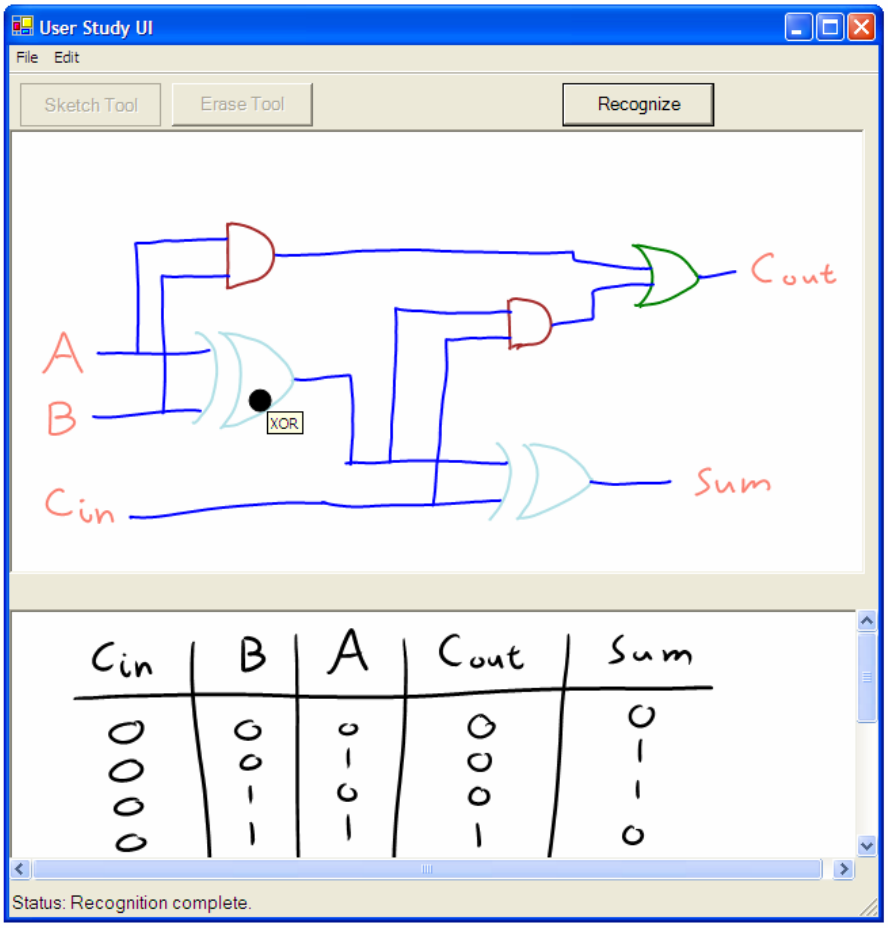


Fig. 4. Complete sketch recognition interface.

3.2 Basic Interface Design

We first designed our prototype interface using iterative design techniques. Figure 4 shows a basic overview of our final interface, although we modified pieces of this interface to explore different interface elements. In this version, the user sketches (unrecognized) notes in the bottom panel, sketches the circuit (to be recognized) in the top panel and presses the "Recognize" button to trigger recognition.

Users could correct recognition errors only by erasing and redrawing their strokes. We did not aim to explore error correction mechanisms, and this method was adequate for users to complete their tasks. However, error correction mechanisms deserve attention from future studies as they are an important element of sketch recognition interfaces.

3.3 User Studies

We conducted two separate studies to investigate the four interface elements described above. In the first study we investigated recognition triggers and diagram separation tools. In the second study we examined recognition feedback and error types. We divided our investigation into two studies in order to minimize the mental burden and scheduling commitment on users.

During both user studies we invited all participants to contribute informal feedback and asked them to fill out questionnaires inspired by the previous work of Chin *et al.* [6] and Landay [17]. Our questionnaires aim to measure user responses to individual interface elements based on relevant characteristics we inferred from our pre-study interviews and related work in sketch recognition user interfaces.

3.3.1 User Study #1: Recognition Triggers and Diagram Separation

Our first study examined triggers and diagram separation preferences across five participants. Each participant completed one truth table task with each recognition trigger using the multi-paneled interface depicted in Figure 4; we balanced the order of triggers tested across participants. Next, each participant completed one truth table task with each diagram separation tool in a single-paneled version of our interface. We prompted participants to label the shortest and longest path in their circuits and to have the system recognize only their diagrams (i.e., not the annotations).

After each task, the experimenter prompted users for qualitative feedback and gave questionnaires asking users to rank the reliability, efficiency, convenience and overall quality of the interface. After completing all tasks, users ranked their preferred feedback and diagram separation mechanisms. Users completed study sessions in 30-45 minutes.

This study used two methods to "recognize" the user's diagrams and gestures. First, our system recognized users' check-tap gestures using the built-in Microsoft Tablet SDK gesture recognizer. Some users' gestures were recognized reliably, but others were not (we discuss the implications below). Second, the system simulated recognition of the users' diagrams by coloring most of the strokes blue, indicating "correct recognition," but displaying approximately 10% of the strokes in red, indicating a "recognition error." No recognition actually occurred, but most users expressed that they genuinely presumed the recognition results and simulated errors were real.

3.3.2 User Study #2: Feedback Mechanisms and Error Types

Our second study examined recognition feedback and recognition errors across six participants (including two participants from our first study). Each participant again completed one truth table task per feedback method and per error type, and we balanced the order of feedback mechanisms and error types, respectively, across users. Feedback tasks always preceded error tasks. When completing the error tasks, users chose the feedback mechanism they preferred. In all tasks, users triggered recognition with a button.

Again after each task, the experimenter prompted users for qualitative feedback and gave them questionnaires asking them to assess the reliability, efficiency, convenience and overall quality of the interface. After completing all tasks, users ranked which feedback mechanism they preferred and which errors were most confusing or difficult. Users completed study sessions in 45-75 minutes.

This study simulated diagram recognition through a novel application of a Wizard of Oz technique. Users worked with a realistic Tablet PC application while a human "Wizard" actively labeled user-drawn symbols. Wizard of Oz studies have proven effective for developing speech recognition interfaces [7,15], but this is the first application of Wizard of Oz studies to the design of sketch recognition systems of which we are aware. Davis has developed a Wizard of Oz system to support sketch recognition user interface development [8], but this system was not ready in time for our study.

Our experimental Wizard of Oz system consists of two Tablet PCs directly networked over an Ethernet connection. As the user sketches on one tablet, the Wizard actively receives copies of user strokes and labels relevant symbols on a second tablet. Once the user triggers recognition, the Wizard sends a labeled result to the user tablet. For this stage of our study, our human subjects committee required us to inform participants that a human was recognizing their strokes.

The Wizard simulated perfect recognition while participants tested feedback mechanisms. To test error types, we simulated a 15% (approximate) error rate. We chose this rate because it is a realistic target for sketch recognition systems in the near future. We simulated each error type as follows:

- **False Positives.** The user end of our Wizard of Oz system filtered recognition results from the Wizard and randomly applied incorrect labels to approximately 15% of the recognition results.
- **False Negatives.** The system again filtered the Wizard's simulated results, randomly deleting the labels from approximately 15% of symbols.
- **Grouping Errors.** The Wizard incorrectly grouped the strokes in approximately 15% of symbols drawn, usually giving about 3 to 4 grouping errors

per sketch.

4 Results and Discussion

In this section we present quantitative and qualitative results of our study. In part due to our small sample size, even when users agreed, many of our survey responses do not show statistically significant differences. In many of these cases, however, qualitative feedback supports patterns in quantitative results. When users had conflicting opinions, we summarize these different beliefs so as to inform interface designers about the range of user preferences.

4.1 Student Workflow

Students unanimously reported that they prefer to design their circuits on a Tablet PC or whiteboard rather than enter them directly into a simulation tool. Furthermore, during our study, all of our participants used almost identical workflows. When designing a circuit, each student would write notes or equations, sketch a circuit, trigger recognition, and then correct recognition errors. Even when the experimenter reminded users that they could trigger recognition intermittently during the design process, participants continued to trigger recognition only after sketching a complete or significant portion of a circuit. User comments reveal two reasons for this workflow: users prefer to focus on the design and sketching portions of their tasks without interruption, and they prefer to correct recognition errors in a batch instead of individually.

4.2 Recognition Triggers

Figure 5 summarizes users' responses to different recognition triggers. User reactions suggest that high reliability is the most important characteristic of a recognition trigger. Most users were able to trigger recognition successfully using check-tap only once for every five failures. Though many users admit the gesture trigger offers a desirable convenience, the majority of users (n=3) ranked the button trigger higher than the other two triggers, and users unanimously rated it as highly reliable (Figure 6).

The button trigger's high reliability minimizes users' mental effort and allows them to devise efficient workflows. One user commented that the button trigger helped her make her "approach more systematic" and "figure[] out [the] most efficient way to" use the interface. A second user commented that "[he could] always get faster with a tool that works every time."

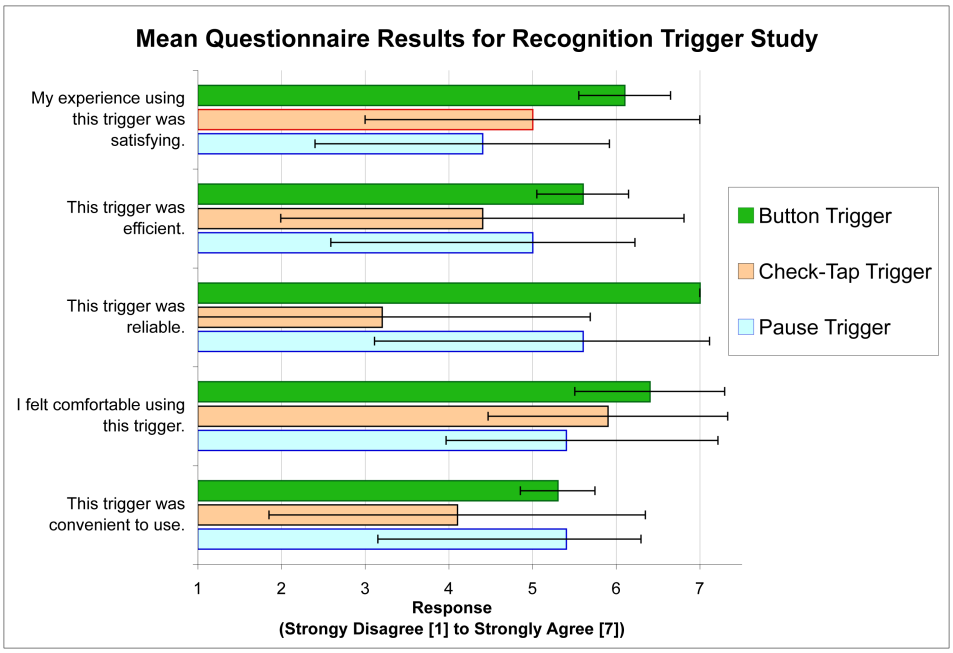


Fig. 5. Questionnaire results for Recognition Triggers. Error bars show one standard deviation from the mean response.

The relatively high error rate of the check-tap gesture colored many users' reactions to this trigger. The one user that ranked the gesture trigger as the most desirable was able to trigger recognition successfully on his first six attempts. Even still, this user admitted to seeing only a "marginal difference" between the gesture and button triggers.

User reactions to the pause trigger suggest that students may find the pause trigger acceptable only if it matches the speed of their thought processes. Two users commented that a four second pause was too long, while one user found the pause trigger quite "distracting" to her mental flow and suggested a longer pause. The one user who ranked the pause trigger as his favorite trigger commented that it allowed him to "check as [he] move[d] through creating the diagram."

4.3 Separation/Annotation Tools

Figure 7 summarizes user response to survey questions about separation techniques. Users unanimously preferred the pre-separation (color) tool to the post-separation (lasso) tool, and all users found the pre-separation tool more satisfying to use than the post-separation tool (Figure 8). Many modal interfaces traditionally suffer from the problem that users often forget to toggle to the appropriate mode before performing a task. However, in our study, only one of five users forgot to toggle the color tool into annotation mode before

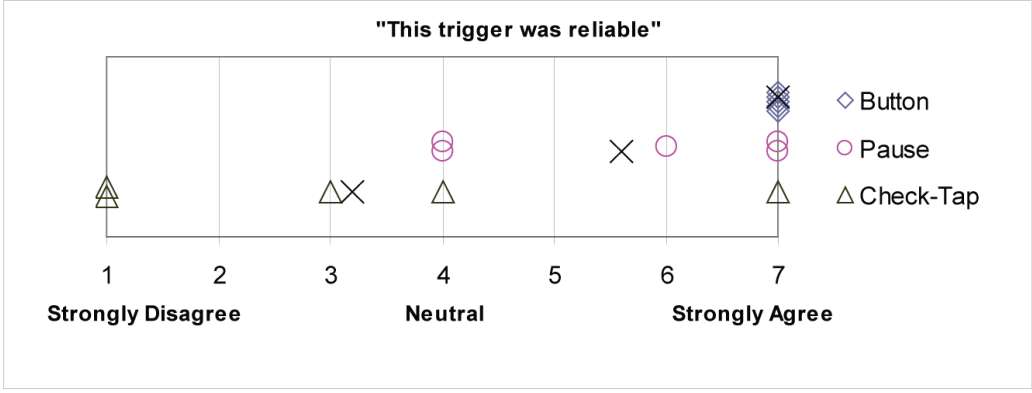


Fig. 6. Individual user responses to recognition trigger reliability. Each symbol represents a single user's response. "X"s show mean values.

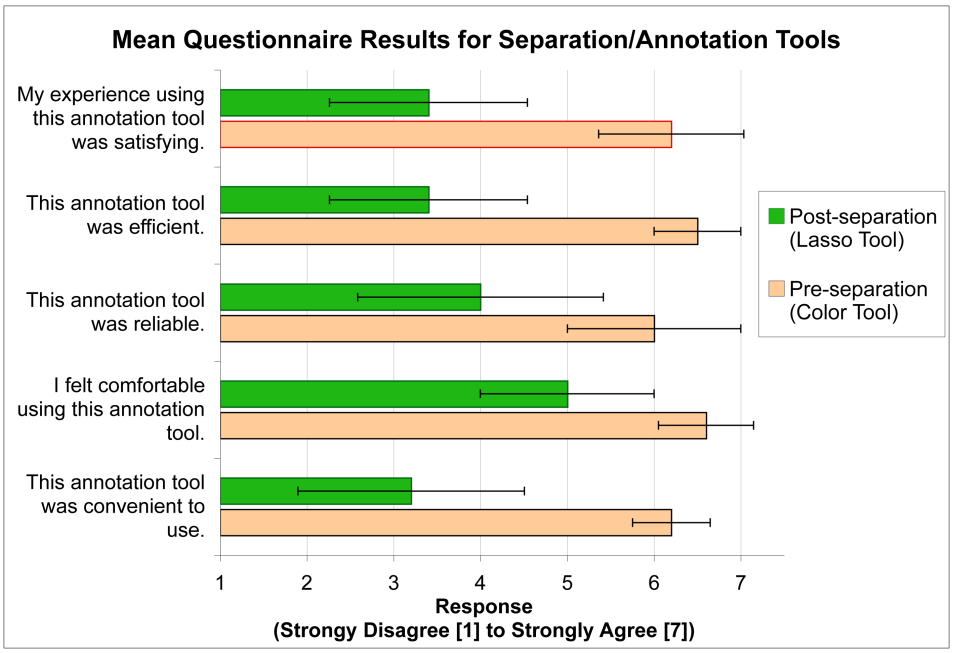


Fig. 7. Questionnaire results for Separation/Annotation Tools.

writing notes; furthermore, this user corrected her mistake immediately.

Two participants commented that the lasso tool constrained the way they could create their diagrams. One user described that the color tool "required less planning about where and when to write so as not to screw up [the] lasso[] [gesture]." Another user commented that she "like[d] the [color tool's] ability to draw annotations wherever [she wanted]."

One issue that may have influenced user responses to the separation mechanism is that the lasso tool required two check-tap gestures while the color tool required only one. Nevertheless, users responded more negatively to the burden of circling the symbols and the physical segregating of annotation and

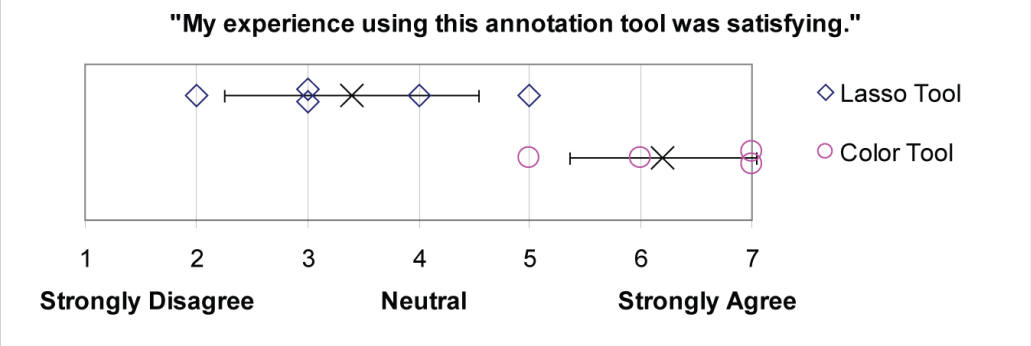


Fig. 8. Individual user responses to sketch separation method satisfaction. Error bars are shown for one standard deviation.

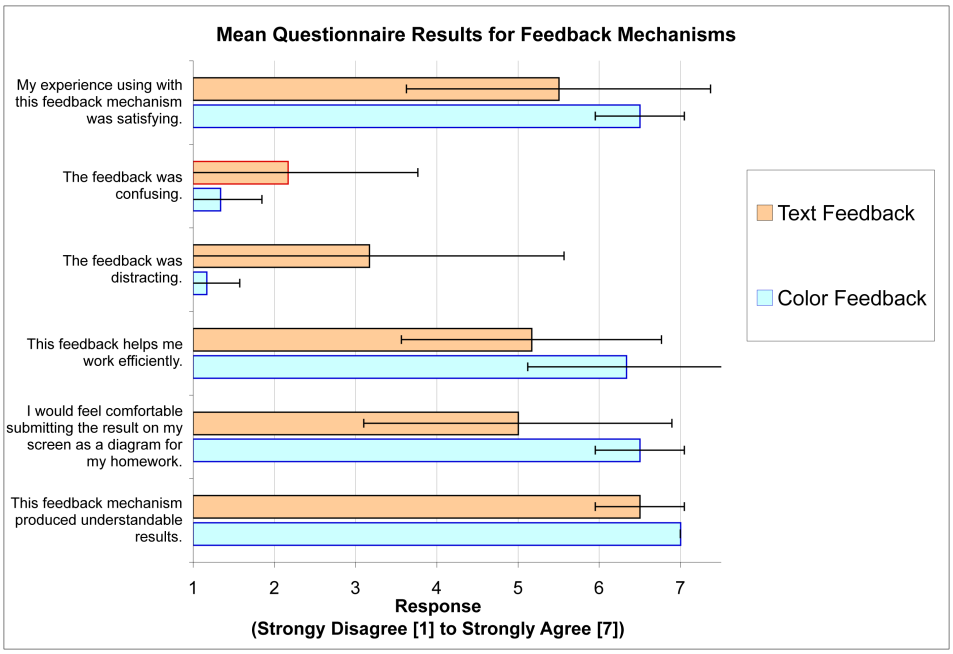


Fig. 9. Questionnaire results for Feedback Mechanisms.

schematic strokes than to executing the gesture. One user even modified her questionnaire to express her preferences for annotation tools that hypothetically used the button trigger rather than check-tap; this user still preferred the color tool.

4.4 Feedback Mechanisms

Figure 9 summarizes user survey responses to feedback mechanisms. Users unanimously agreed or strongly agreed that color feedback "helped them work efficiently" and "produced understandable results"; most users preferred color feedback to text-label feedback. Users found text feedback distracting (Figure

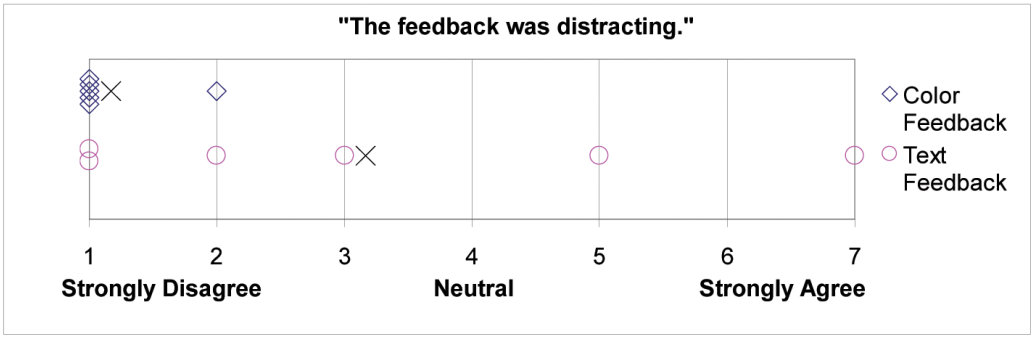


Fig. 10. Individual user responses to feedback distraction.

10) and that it added an unnecessary mental burden to the design process. One user commented that "The text gets very cluttered [and] covers up the [sketch's] points of interest quickly," thus interrupting his ability to reason about the diagram. Another user commented that text labels "can get a little distracting with four or five [instances] of the same gate" and finds the color feedback "more elegant" and "less redundant." Although our specific text placement may have influenced user perception of the text labeling feedback mechanism (i.e., sometimes the text obscured part of the sketch), automatic text placement is a difficult problem, and any automatic text placement method likely will obscure some portion of the user's sketch.

Despite the distracting nature of text feedback, color feedback alone is likely not informative enough. During the color feedback tests, each of the six participants immediately hovered her or his stylus over individual gates to verify the labels. In addition, one user commented that she "like[d] both" methods of feedback and found that "a button to go back and forth [between feedback mechanisms] would be nice."

Unlike the work by Zeleznik *et al.* [29], our study did not examine the utility of giving feedback by displaying beautified versions of the recognized symbols, either in addition to or in place of the user's strokes. Users in our prestudy expressed a general disinterest in this style of feedback, and the results of our study suggest why. Users generally disliked clutter in the diagram, and displaying beautified versions of users' hand-drawn symbols in addition to the users' original strokes would only increase this clutter. On the other hand, removing users' original strokes and replacing them with beautified symbols would make it difficult for users to detect recognition errors, as reported by Zeleznik and Miller [29].

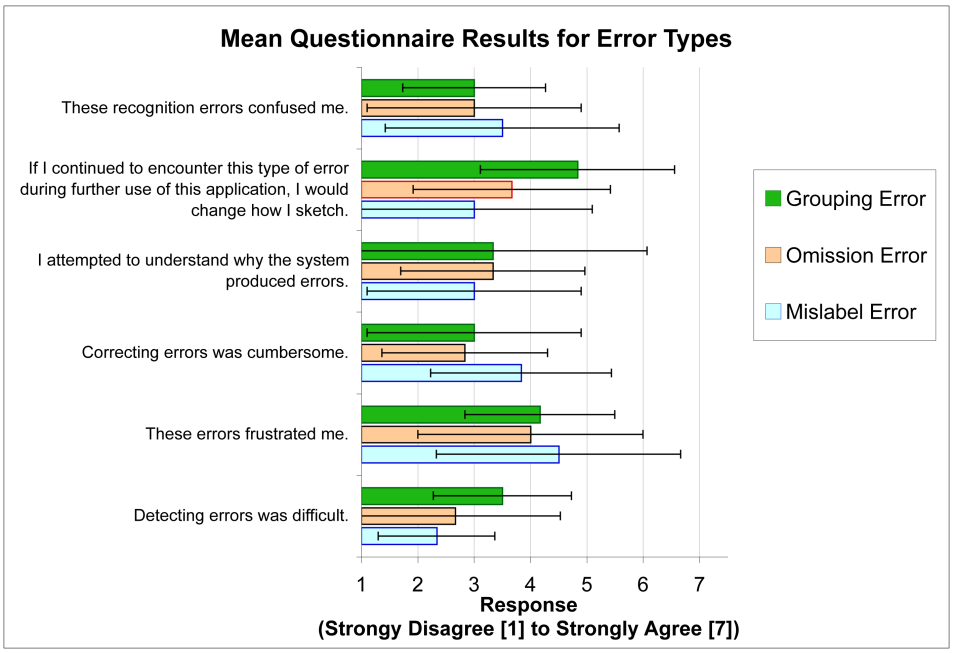


Fig. 11. Questionnaire results for Error Types.

4.5 Error Types

Figure 11 summarizes our survey results to error types. Users had diverse reactions to recognition errors, and our quantitative results show no strong trends. This lack of quantitative trends may be due in part to an unintended effect of the way we generated errors. To keep error rates consistent, the computer system automatically generated omissions and false positives. However, generating grouping errors was too difficult to automate (it would require a deeper understanding of the sketch in order to split and merge adjacent groups), so the human Wizard generated grouping errors. Consequently, false positives and omissions were often more surprising to users (e.g. a wire classified as an AND gate) than grouping errors. This effect limits the direct comparison we can make between user responses to error type, but allows us to better understand the importance of how users perceive errors based on how well they understand them.

Our qualitative results suggest that user acceptance of the system is related not only to absolute error rates but also to how well they understand and trust the system's recognition. Generally, our users initially happily corrected any type of recognition error, but quickly became frustrated when they could not understand why errors occurred. Two users specifically explained that their acceptance of the system would depend not on the error rate as much as on the predictability of errors. One of these users remarked that he would not be willing to accept a system with the error rate exhibited unless errors were more predictable and helped him adapt his drawing style to avoid them.

This user also found false negatives the most confusing because these errors failed to inform him about "what to move away from" (i.e., how to change his sketching style to reduce errors). On the other hand, another user preferred false negatives because it made him trust the system more when it did recognize his strokes. Though related work with speech and handwriting suggests that user acceptance may improve with lower absolute error rates [27, 22], our results illustrate the importance of user perception of error type to acceptance of the system.

We also found that users created dramatically different mental models of errors. Two users considered themselves responsible for grouping errors. Two users blamed the system for mislabel and omission errors. One of these users remarked that mislabel and omission errors seemed "out of context," or unpredictable. Two users specifically remarked that they did not try to form a mental model of errors.

Qualitative user responses related to frustration with errors exhibited a general accordance. Four out of six users responded that omission errors were the easiest to perceive; users likely find that the lack of a label is easier to detect than an incorrect label. Furthermore, four out of six users remarked that the grouping error was the most confusing type of error despite the fact that these errors were human-generated.

Finally, although we did not specifically study error correction here, it clearly impacts users' acceptance of errors and deserves future study. In our study users corrected errors by erasing misrecognized symbols using the eraser end of the stylus and redrawing them. Three users remarked that more effective error correction mechanisms would make dealing with errors less frustrating. Though most users were comfortable erasing and redrawing incorrectly recognized symbols, one user found the erase gesture hard to control because of the size of the stylus' eraser, and another user requested a selection tool for erasing large amounts of ink. In addition to providing more flexible erasers, future studies should explore how error correction mechanisms such as N-best lists, explicit symbol grouping via stroke selection, or multi-modal techniques influence users' perception of different types of errors.

5 Implications for System Design

The results of our study inform the design of educational sketch recognition systems. Here we summarize the major implications for both user interface and recognition engine developers.

Recognition triggers should be user-triggered, efficient, and reliable.

Our study illuminates that, with respect to our tasks, users prefer reliability as much as they do convenience. Gestures and system activated recognition may be an option, but should not be the only option.

The interface should provide users with a way to separate recognized from unrecognized strokes as they draw. Users desire efficiency and are willing to put in small amounts of effort while they work to avoid larger amounts of effort later in the process. Our study participants preferred our pre-separation tool for its relatively low mental and physical overhead.

Recognition feedback should provide minimal clutter and transform users' strokes as little as possible. Stroke color effectively provides contrast between recognition labels with minimal stroke transformation. More dramatic transformation should occur only upon user request.

Users are willing to correct errors after they are done drawing. Our users treated correction as a game or a necessary evil. Users do error correction all in one batch and tend not want to disturb their design process with on-the-fly correction (this result is consistent with [13]).

Errors must be predictable and/or understandable. Automatic recognition engines can potentially make nonsensical errors. Recognition engines that incorporate adjustable confidence levels may help user interface designers strike an optimal balance between false positives and false negatives. A recognition engine that could describe precisely why a given interpretation was missed could also help users modify their drawing styles to raise recognition rates.

6 Sketch-based Circuit Design User Interface

Our research group is developing a complete sketch-based tool for digital circuit design education. Here we present the user interface to this tool that we developed according to the results of our user study. We also present qualitative results from an additional informal user study we conducted to evaluate this interface. The results of this study confirm many of our previous observations and provide further insight into user preferences.

6.1 Interface Overview

Figure 12 shows the user interface to our group's sketch-based digital circuit design tool that we developed. The simulation tool supports typical introductory circuit design tasks similar to the truth table tasks discussed above. Our

tool allows students to sketch circuits freely and then to analyze the behavior of their recognized circuits via existing simulation software (Xilinx ISE).

A typical use case for this interface consists of the following steps. The student begins by writing notes and equations in the "notes" panel on the interface to organize her thoughts. The system will not recognize any ink the in the notes panel.

Next, the student draws the corresponding truth table in the "Truth Table" panel, specifying the input and output behavior of the circuit. After drawing the truth table, the student taps the "Recognizer Ink" button near the top of the interface. The application colors recognized strokes in the truth table window to indicate its recognition, using a separate color for each component in the table. (Components include 0's, 1's, X's, labels, and dividers). The student corrects any recognition errors by erasing and redrawing her strokes and then retapping the "Recognize Ink" button.

After drawing the truth table, the student designs her circuit. First, the student draws the circuit and then taps on the "Recognize Ink" button to invoke symbol recognition on the circuit diagram; the application uses color feedback, as described in Section 2, to convey the result of recognition. (The student may also intermittently tap "Recognize Ink" to invoke recognition as she draws her circuit.) Again, the student corrects recognition errors by erasing and redrawing strokes and then reinvoking recognition.

Once the user has completed drawing the circuit, she taps the "Recognize Circuit" button to invoke recognition of the circuit's structure (which includes wire connections, labeled inputs and outputs, and other structural features). The system displays circuit structure feedback in a number of ways (Figure 13). First, the system highlights the endpoints of connected and unconnected wires in the circuit with black squares and yellow circles, respectively. Second, the system alerts the student to any potential structural problems that may be a result of either misrecognition or incorrect circuit construction. Third, when the user selects a wire in the circuit, the system highlights all other wires that it has detected as belonging to the same mesh (*mesh highlighting*). Based on this feedback, the student corrects errors by erasing strokes, drawing new strokes, and then invoking both symbolic and structural recognition on the new strokes.

Finally, the student taps the simulate button. The system generates a benchmark test, tests the circuit's behavior against the behavior specified by the truth table, and alerts the students to any discrepancies. Based on the results of this simulation, the student can modify her original design to fix any conceptual errors.

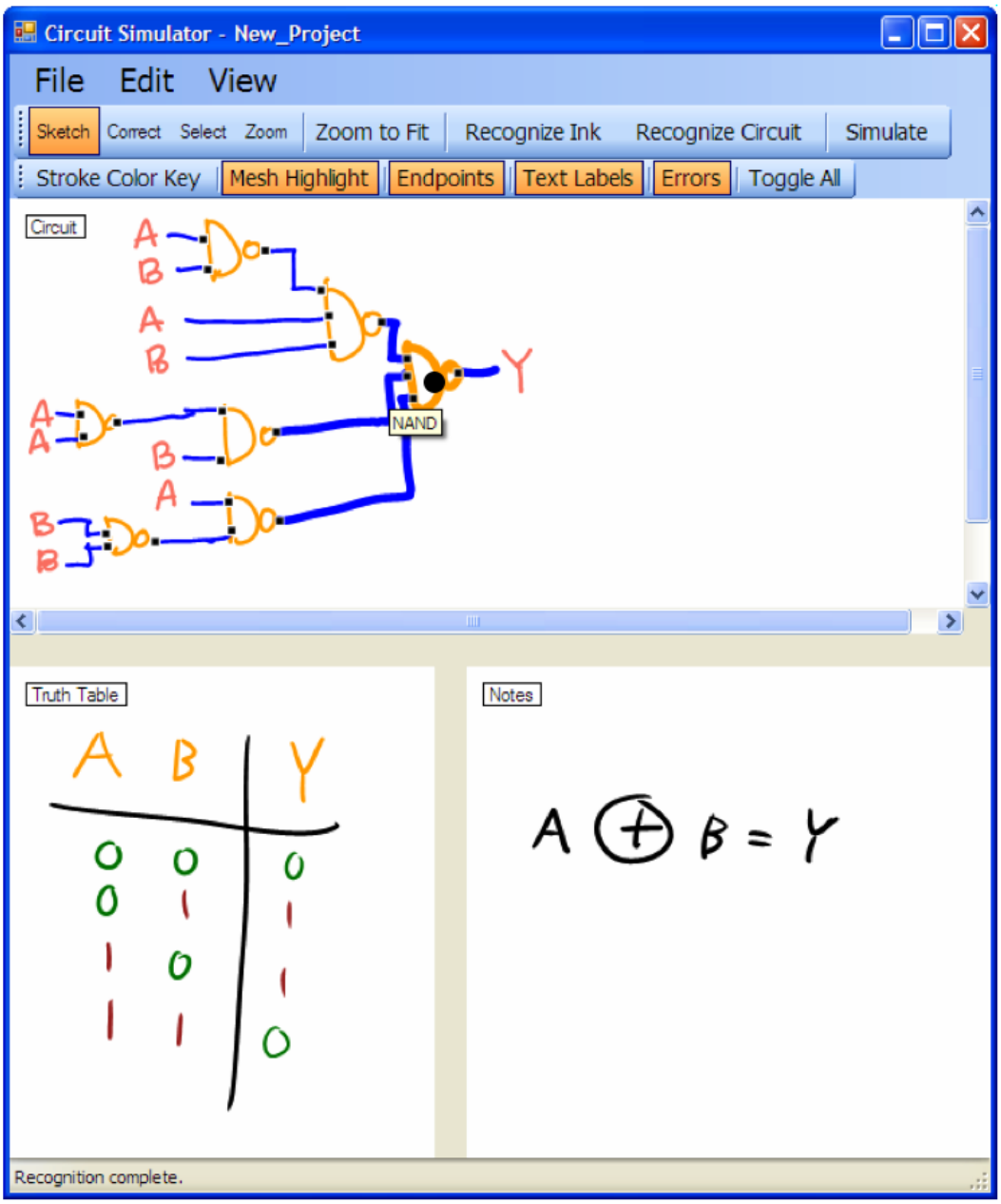


Fig. 12. User interface for sketch-based digital circuit design application. The black dot denotes the current location of the user's pen.

6.2 Design Justification

We constructed our interface to support the work flow pattern we observed in our user studies as well as to make recognition feasible. The three panels in the interface reflect the three major stages we observed in students' work flow when designing a circuit: notes and equations, truth table construction, and circuit design. The user can toggle the visibility of each panel in order to devote more screen (drawing) space to one particular diagram. This design enforces pre-

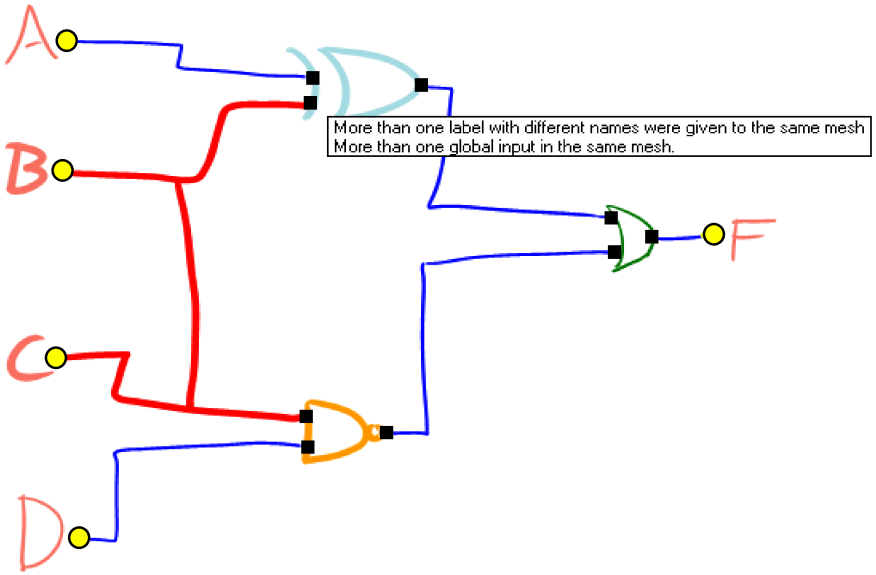


Fig. 13. Demonstration of circuit structure feedback, including an error message for an illegal circuit structure.

separation of different types of drawings, which users in our previous study tended to find a more efficient means of diagram separation. Each panel is linked to a different recognition engine, or no recognition engine in the case of the notes panel, removing the need for the system to separate circuit elements from truth tables from notes, and making recognition feasible.

The recognition feedback in our interface aims to display recognition results clearly while minimizing clutter in the diagram. We use color feedback for symbol recognition (in the circuit and truth table panels) including a text label that the application displays when the user hovers the stylus near a recognized symbol. To further minimize clutter, many of the recognition display options can be toggled on and off, including the mesh and endpoint highlighting, circuit error feedback, and text labels for individual symbols.

We use a two-stage, button-triggered recognition method to help minimize confusion due to recognition errors by minimizing the complexity of the errors the user must deal with. In the first stage, triggered by the "Recognize Ink" button, the system recognizes individual circuit symbols (e.g., wires and gates). In the second stage, triggered by the "Recognize Circuit" button, the system links circuit components together to recognize the structure of the circuit. Separating these recognition processes provides for more comprehensible errors because recognition errors from the former stage will not cascade to and further complicate the errors of the latter stage (assuming the user corrects all errors).

Note that while the user sees a two-stage recognition process, the underlying recognition engine need not make this distinction. Although in our application

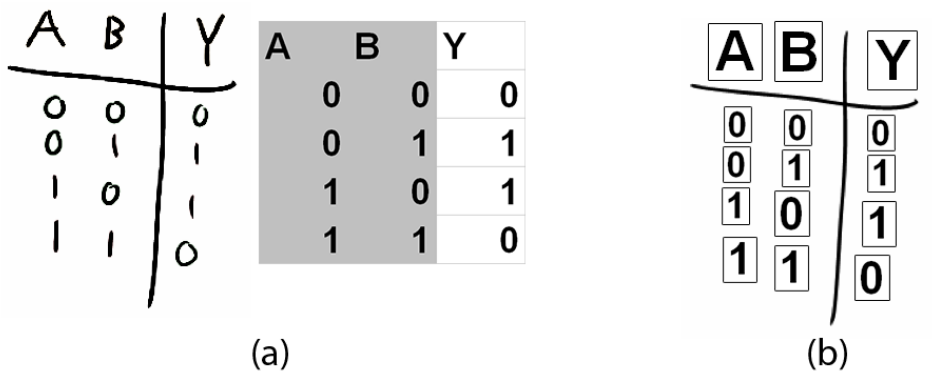


Fig. 14. Examples of (a) data table and (b) text replacement feedback. Figure 12 gives an example of color feedback.

we currently separate symbol recognition from circuit structure recognition, we are exploring ways for the circuit recognition phase to help correct errors in the symbol recognition phase before the recognized symbols are displayed to the user.

6.3 Evaluation of Circuit Design Interface

This section describes an informal user study evaluating several features of our new circuit design interface. We specifically aimed to assess user preferences for truth table feedback mechanisms, the multi-panel organization of the user interface, and the mesh highlighting feature. In addition, we gathered qualitative feedback on the overall interface.

We directly compared three feedback mechanisms for truth table recognition: color feedback, text replacement feedback, and data table feedback (Figure 14). As in our previous study, color feedback involves coloring each recognized symbol with a unique color. Text replacement feedback involves replacing recognized zeros, ones, and column labels with equivalent text. Data table feedback involves drawing a formatted, spreadsheet-like data table adjacent to the the user's handwritten table.

Our participants included twelve Harvey Mudd College engineering students who had all previously completed digital design coursework and used standard digital circuit simulation software (e.g. Xilinx ISE) but had not participated in our first study. We asked each participant to complete a truth table task similar to the task in Figure 3(a). We requested that the user draw the truth table and then tap "Recognize Ink." The system recognized the truth table using our own automatic truth table recognizer and displayed its feedback using one of three methods described above. We then showed the user the same recognition feedback using each of the other two methods and asked him

or her to rank the methods in order of preference. We balanced the order in which we displayed feedback across participants.

We subsequently asked nine of the twelve users to draw the circuit corresponding to the given truth table in the circuit panel. (Due to scheduling conflicts, the first three study participants could not complete the circuit design portion of the task.) Because our circuit recognition engine is still under construction, we again used simulated recognition for this part of the evaluation, and we informed users that the system was using simulated recognition. We simulated symbol recognition by coloring the user's strokes blue or red to indicate correct or incorrect "recognition," as in the user study described in Section 3.3.1. When the user tapped "Recognize Circuit," we replaced his or her diagram with pre-constructed, pre-labeled version of the same circuit (shown in Figure 12) and asked the user to interact with this circuit give us qualitative feedback on the mesh highlighting feature.

The results of this study verify several trends in user preferences observed in our first study. First, most users (n=9) preferred color feedback. The other three preferred data table feedback and ranked color feedback as their second choice. No user ranked text replacement feedback as a first choice. Users expressed a diverse set of reasons for preferring color feedback. One user specifically explained that he "[does not] like to have what [he draws] covered up" and spoke generally against feedback that entails any sort of automatic ink replacement. Two other users explained that they preferred color feedback for the ease with which they could verify recognition results. These users remarked that color feedback allowed them to "immediately check" and "instantly identify" the recognition results as correct or erroneous. Moreover, another user who preferred color feedback over data table feedback explained that she found it too hard to compare each element of the data table with what she wrote. In contrast, the three users who preferred data table feedback all found the data table easier to read and verify for correctness.

Users reacted positively overall to the mesh highlighting feature. Seven of nine users regarded the feature as useful, including one user who rated the feature as "very useful." Three users found the feature slightly distracting due to a display buffering issue that caused flickering whenever highlighting of the diagram changed. Five of nine users requested the ability to toggle this feature for better control of the appearance of the diagram. In response to this feedback, we added the ability to toggle mesh highlighting to our interface.

While users had no problem drawing their notes, truth tables and circuits in different panels, users often had to resize panels to get more screen space for the component they were currently working on. We propose a menu that toggles the interface between standard window configurations to support different phases of the design process. When we asked users for their thoughts on such

an option, they reacted positively, although which configurations should be included in such a menu (and whether these configurations should be user-defined) remains an area for future study.

The results of this study also gave us further evidence of how users integrate the task of triggering recognition into their workflows. Eight of the nine users who completed the circuit design task triggered recognition after drawing the entire circuit. One user experimented with the recognition trigger before completing the circuit; however, we observed no user who frequently triggered recognition while completing the design task. These observations are consistent with the user workflows observed in our original study.

7 Related Work

Previous user studies of sketch-based user interfaces focus mainly on the development or evaluation of complete systems. Researchers rely heavily on interviews and ethnographic studies to identify and understand user preferences of sketch-based computer tools. For example, Landay and Myers designed SILK [18], a sketch-based system for user interface design, based on the results of a survey of professional user interface developers. Newman *et al.* worked closely with designers throughout the design of DENIM [23], a sketch-based system for web page design. Their interaction with users revealed that web page development requires very little sketch recognition: DENIM uses gesture recognition techniques to recognize pages (rectangles) and links (arrows) but leaves the rest of the user's sketch unrecognized. Educational software, on the other hand, requires a deeper understanding of the user's sketches.

Among work that focuses on interaction with recognition-intensive systems, a recent study by Zeleznik *et al.* takes an approach very similar to our own [29]. They present four different feedback methods for recognized mathematics equations. Contrary to our results, they find that providing recognition feedback by changing the colors of users' strokes is insufficient because users find color "insufficient to highlight certain errors" [29]. Instead they argue that the system should provide feedback by displaying a typeset version of the equation offset from the users' original stokes. The difference between their results and ours likely stems from the difference in domain. First, circuit diagrams contain fewer symbols that are more spread out in space than math equations, making stroke color easier to distinguish. Second, unlike circuit diagrams, equations are essentially one-dimensional, making it feasible to display recognition feedback below the original equation. Offsetting recognized symbols in a circuit diagram likely would clutter the diagram to the point where the circuit would become difficult to understand.

Evaluation of many other recognition-intensive systems tends to focus on recognition error rates, and provides only general insight into user interface issues (e.g., "users found recognition errors frustrating") [3, 10, 12]. A few researchers, however, have examined system usability in more detail. LaViola's evaluation of MathPad^2, a sketch-based system that recognizes freely-drawn equations and physical diagrams, reveals specific user preferences: users like MathPad^2's scribble-erase gesture and find recognition errors frustrating but tolerable for this task [20]. Other user studies of recognition-based systems are currently underway [16, 25].

Other researchers have studied the usability of pen-based interaction techniques that are complementary to sketch recognition. The CrossY interface [5] explores several novel interface elements that combine gestures and traditional graphical user interface components. Long *et al.* [1] provide a model for measuring the visual similarity of gestures in an effort to inform effective gesture design. Lank and Saund present a model of users' pen-based selection gestures to inform the design of a faster, more accurate selection mechanism [19]. Finally, Zeleznik and Miller propose a *Fluid Inking* framework that combines several pen-based interaction techniques in order to create a natural and powerful interface [28]. The results of these previous studies are complementary to our results in the creation of a complete sketch recognition system.

Though little is known about how free sketch recognition errors affect usability, researchers have studied user perception of handwriting and speech recognition errors. Rhyne and Wolf present early work in this area [24]. More recently, Frankish *et al.* find that the relationship between error rates and user acceptance is dependent on the perceived cost to benefit ratio of a specific task [9]. We expect that the same trend holds for sketch recognition systems. Wu *et al.* find that handwriting task completion time is most sensitive to error rates above 6% [27]. Munteanu *et al.* find a linear relationship between speech recognition accuracy and the quality of user experience with webcasts [22]. Although we do not examine error rates specifically, our analysis helps inform user perception of sketch recognition errors.

Finally, Wizard of Oz techniques are commonly used in the development of speech-based interfaces, and they are beginning to emerge as a feasible technique for sketch-based interfaces. Researchers have used Wizard of Oz techniques successfully to design and evaluate speech-based interfaces for many years [7, 11]. Noting the success of this technique, Klemmer *et al.* developed the SUEDE toolkit to support Wizard of Oz studies in speech-based interface development [15]. Although we developed our own Wizard of Oz tool for our study, Davis *et al.*'s SketchWizard takes the first stride towards constructing a generic, multi-domain tool for conducting Wizard-of-Oz-based development of sketch-based systems [8].

8 Future Work

Both our initial study and the development of our revised user interface suggest several important areas for future study. In this section we detail the most critical and present suggestions for exploring them.

First, in the immediate future we plan to conduct a larger follow-up study using the same design as our first two studies to confirm the trends we observed and to resolve some of the issues with our initial study implementation. For example, in our follow-up study we will allow users to set their own pause time for automatic recognition; we will choose a gesture that is more reliably recognized, and we will eliminate unrealistic false-positives (e.g., where the system "recognizes" a wire as a NAND gate) from our error perception study.

A second critical area for future study is to design an effective error correction interface. Error correction difficulty will surely impact users' perception of a sketch recognition system, including their acceptance of various types of errors. However, how to construct effective sketch recognition error correction interfaces is an open problem because of the two-dimensional nature of the task. One-dimensional error correction interfaces such as the Tablet PC TIP (text-input panel) do not extend easily to a two-dimensional environment.

In our future work we plan to compare a number of error correction interfaces, including:

- **Erase and Redraw** Users use a gesture or the eraser of the stylus to erase incorrectly recognized strokes and then re-draw the misrecognized symbols.
- **N-Best** Users select an alternate interpretation for a set of misrecognized strokes, either from a list, or by tapping the stylus to display alternate interpretations.
- **Trace over** Users trace over misrecognized pieces of the sketch, triggering re-recognition of the traced strokes.
- **Grouping correction** Users correct grouping errors by explicitly grouping incorrectly grouped strokes. The system then re-recognizes the strokes using the revised grouping.

A third related area of future study is to explore how to help users change their drawing styles to help the system recognize their drawing more reliably. None of our users expressed a willingness to learn a gesture-based command language to draw their circuits, but many expressed a willingness to slightly change their drawing style to aid recognition. How to create an interface that suggests to users how to modify their drawing styles, and exactly how much users will be willing to modify their drawing styles remain open questions.

Finally, our initial study design only begins to address users' perception of

different types of errors, and there is plenty of future work to be done in this area. In particular, in our complete interface we propose a two-stage error correction process that separates symbolic from semantic errors, but we do not yet know how users will perceive this process.

9 Conclusion

We have presented the first direct comparison of user interface elements in a sketch-based user interface. We found that users prefer to trigger recognition at the end of a task and to have segregated sections for different tasks (schematic, annotation). Users also prefer a tool that does not clutter or modify their sketch and that has predictable recognition errors. Our study indicates that features common in traditional schematic-entry systems, such as mesh highlighting, are also useful in a sketch-based interface. These results have direct implications for the design of free-sketch interfaces, both in circuit design and a variety of other domains. Based on these results we constructed a free-sketch user interface for a digital circuit design tool.

While our study reveals a number of important results, perhaps its most important outcome was the excitement users showed for a sketch-based user interface to replace the cumbersome interface they currently use. The major advantage they perceived in a sketch-based interface is the lower cognitive load in entering their circuits, and they expressed a willingness to cope with and correct the recognition errors inherent with such a system.

10 Acknowledgements

We would like to thank our study participants and Kris Karr (our Wizard). We would also like to thank Susan Matonosi and Deb Mashek, who provided helpful feedback on our study design, and the rest of the HMC Sketchers group including Anton Bakalov, Andrew Danowitz, Sam Gordon, Ellen Kephart, Raquel Robinson, Devin Smith, and Alice Zhu. This work is supported in part by an NSF CAREER award (IIS-0546809), the Baker Foundation and a Harvey Mudd College Beckman Research Award.

References

[1] J. A. Chris Long, J. A. Landay, L. A. Rowe, J. Michiels, Visual similarity of pen gestures, in: Proc. of CHI, ACM Press, 2000.

[2] C. A. Aaron Wolin, Devin Smith, A pen-based tool for efficient labeling of 2d sketches, in: Eurographics Workshop on Sketch-Based Interfaces and Modeling, 2007.

[3] C. Alvarado, R. Davis, Preserving the freedom of paper in a computer-based sketch tool, in: Human Computer Interaction International Proceedings, 2001.

[4] C. Alvarado, R. Davis, Sketchread: a multi-domain sketch recognition engine, in: Proc. of UIST, 2004.

[5] G. Apitz, F. Guimbretière, CrossY: a crossing-based drawing application, in: Proc. of UIST, ACM Press, 2004.

[6] J. P. Chin, V. A. Diehl, K. L. Norman, Development of an instrument measuring user satisfaction of the human-computer interface, in: Proc. of CHI, ACM Press, 1988.

[7] N. Dahlbäck, A. Jönsson, L. Ahrenberg, Wizard of oz studies: why and how, in: Proc. of IUI, ACM Press, 1993.

[8] R. C. Davis, T. S. Saponas, M. Shilman, J. A. Landay, SketchWizard: Wizard of Oz prototyping of pen-based user interfaces, in: Proc. of UIST, 2007.

[9] C. Frankish, R. Hull, P. Morgan, Recognition accuracy and user acceptance of pen interfaces, in: Proc. of CHI, ACM Press/Addison-Wesley Publishing Co., 1995.

[10] L. Gennari, L. B. Kara, T. F. Stahovich, Combining geometry and domain knowledge to interpret hand-drawn diagrams, Computers and Graphics 29 (4) (2005) 547–562.

[11] J. D. Gould, J. Conti, T. Hovanyecz, Composing letters with a simulated listening typewriter, Communications of the ACM 26 (4) (1983) 295–308.

[12] T. Hammond, R. Davis, Tahuti: A geometrical sketch recognition system for uml class diagrams, in: AAAI Spring Symposium on Sketch Understanding, AAAI Press, 2002.

[13] J. Hong, J. Landay, A. C. Long, J. Mankoff, Sketch recognizers from the end-user's, the designer's, and the programmer's perspective, in: Sketch Understanding, Papers from the 2002 AAAI Spring Symposium, 2002.

[14] H. Hse, A. R. Newton, Recognition and beautification of multi-stroke symbols in digital ink, Computers and Graphics 29 (4).

[15] S. R. Klemmer, A. K. Sinha, J. Chen, J. A. Landay, N. Aboobaker, A. Wang, Suede: a wizard of oz prototyping tool for speech user interfaces, in: UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology, ACM, New York, NY, USA, 2000.

[16] K. Koile, K. Chevalier, M. Rbeiz, A. Rogal, D. Singer, J. Sorensen, A. Smith, K. Tay, K. Wu., Supporting feedback and assessment of digital ink answers to in-class exercises, in: Conference on Innovative Applications of AI (In submission), 2007.

[17] J. A. Landay, Interactive sketching for the early stages of user interface design, Ph.D. thesis, Carnegie Mellon University (1996).

[18] J. A. Landay, B. A. Myers, Interactive sketching for the early stages of user interface design, in: Proc of CHI, 1995.

[19] E. Lank, E. Saund, Sloppy selection: Providing an accurate interpretation of imprecise stylus selection gestures, Computers and Graphics 29 (4) (2005) 490–500.

[20] J. J. LaViola, An initial evaluation of a pen-based tool for creating dynamic mathematical illustrations, in: Proc. of Eurographics Sketch-Based Interfaces and Modeling, 2006.

[21] J. Mankoff, S. E. Hudson, G. D. Abowd, Providing integrated toolkit-level support for ambiguity in recognition-based interfaces, in: Proc. of CHI, 2000.

[22] C. Munteanu, R. Baecker, G. Penn, E. Toms, D. James, The effect of speech recognition accuracy rates on the usefulness and usability of webcast archives, in: Proc. of CHI, ACM Press, 2006.

[23] M. W. Newman, J. Lin, J. I. Hong, J. A. Landay, DENIM: An informal web site design tool inspired by observations of practice, Human-Computer Interaction 18 (3) (2003) 259–324.

[24] J. A. Rhyne, C. Wolf, Recognition based user interfaces, Advances in Human-Computer Interaction (1993) 191–250.

[25] D. Tenneson, Technical report on the design and algorithms of chempad, Tech. rep., Brown University (2005).

[26] X. Wang, M. Shilman, S. Raghupathy, Parsing ink annotations on heterogeneous documents, in: Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBIM), 2006.

[27] C. Wu, K. Zhang, Y. Hu, Human performance modeling in temporary segmentation chinese character handwriting recognizers, Int. J. Hum.-Comput. Stud. 58 (4) (2003) 483–508.

[28] R. Zeleznik, T. Miller, Fluid inking: augmenting the medium of free-form inking with gestures, in: GI '06: Proceedings of Graphics Interface 2006, Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 2006.

[29] R. Zeleznik, T. Miller, C. Li, Designing ui techniques for handwritten mathematics, in: Proceedings of the 4th Eurographics Workshop on Sketch-Based Interfaces and Modeling, 2007.