Design and Analysis of Delimiters for Selection-Action Pen Gesture Phrases in *Scriboli*

Ken Hinckley, Patrick Baudisch, Gonzalo Ramos, Francois Guimbretiere
Microsoft Research, One Microsoft Way, Redmond, WA 98052
{kenh, baudisch}@microsoft.com, bonzo@dgp.toronto.edu, francois@cs.umd.edu

ABSTRACT

We present a quantitative analysis of delimiters for pen gestures. A delimiter is “something different” in the input stream that a computer can use to determine the structure of input phrases. We study four techniques for delimiting a *selection-action gesture phrase* consisting of lasso selection plus marking-menu-based command activation. *Pigtail* is a new technique that uses a small loop to delimit lasso selection from marking (Fig. 1). *Handle* adds a box to the end of the lasso, from which the user makes a second stroke for marking. *Timeout* uses dwelling with the pen to delimit the lasso from the mark. *Button* uses a button press to signal when to delimit the gesture. We describe the role of delimiters in our *Scriboli* pen interaction testbed, and show how Pigtail supports scope selection, command activation, and direct manipulation all in a single fluid pen gesture.

Author Keywords
Pen input, marking, delimiters, tablets, gestures

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: Input

INTRODUCTION

In graphical user interfaces, selecting a group of objects and immediately selecting the command to apply to them is a ubiquitous pattern of interaction. For example users might click and drag to sweep out a selection region, and then click on a tool palette or menu to choose a command. This typically requires a *round trip* [7] between the work area and the tool palette or menu bar. While experts sometimes can avoid round trips by learning keyboard shortcuts, this approach only works for desktop configurations. Tablet computers typically have no keyboard, making round trips with the pen unavoidable and tedious.

While the *selection-action* pattern occurs with high frequency in most pen interfaces, the literature lacks a careful study of how the *selection* and *action* subtasks can be combined efficiently. We contribute an experimental analysis and new hybrid approaches that allow designers of pen interfaces to support transitions between and effectively link together selection-action phrases [4]. Our goal is to research new building-blocks for pen interfaces that are (1) *rapid*, with no dwelling or repetitive prompting, but instead using fast, repeatable actions that, with expert use, make minimal demands on visual attention; (2) *unambiguous*, with no recognition unless the user explicitly calls for it; and (3) *expressive*, supporting a variety of commands and using general mechanisms that are not tailored to a specific application domain.

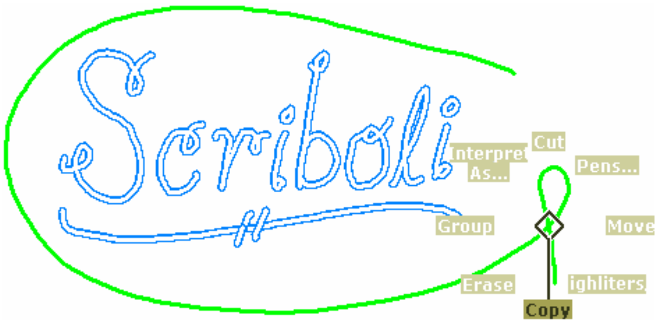

Fig. 1. Pigtail splits the user’s gesture into a lasso, which here selects the ink, and a mark, which here chooses the *Copy* command from 8 possibilities.

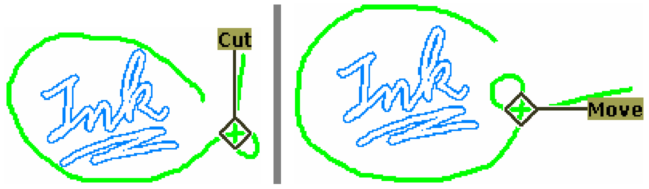

Fig. 2. Drawing the tail in different directions chooses other commands. *Left*: Cut is north; *Right*: Move is east.

One of these key building blocks is the delimiter. Delimiters allow interactive systems to determine the lexical structure of an input phrase [3]. They play a dual role of *connecting* tokens while also *separating* tokens from one another. We focus on delimiters in selection-action phrases, that is, how to merge object selection and command activation in a single fluid interaction. In this paper, we analyze four delimiters for selection-action tasks:

Pigtail: drawing a small loop at the end of the lasso, with the “tail” interpreted as the mark (see Fig. 1, Fig. 2).

Timeout: pausing with the pen at the end of the lasso;

Button: pressing a button to explicitly indicate when the computer should stop the lasso and start the mark;

Handle: lifting the pen after drawing the lasso, then making a mark starting from a handle the system adds to the end of the lasso (as proposed in [12]; see Fig. 3).

We study how these delimiters perform when used to segment a single gesture combining lasso selection of a group of objects with marking menu command selection. *Lasso selection* is the act of circling one or more objects with the pen. Most current Tablet PC applications use tapping for selection, but this just reflects the desktop's legacy. Lassoing is often favored in the pen literature [6, 9, 12, 17, 18] because it is very well suited to selecting handwritten notes or diagrams, which typically contain many small ink strokes that would be tedious to tap on individually. *Marking menus* use a straight pen stroke to select a command from one of 8 compass directions. We focus on marking menus in our experiment because they have been shown to be a rapid and effective means for command activation. While marking menus used in isolation have been rigorously studied [13, 15, 24], techniques to combine scope selection with marking have received less attention [12].

Our results show that statistically the Handle, Button, and Pigtail techniques are equally fast. Handle was preferred by the most users and had a 3% error rate (of selecting an incorrect marking direction) compared to 5% for Button and 6% for Pigtail. Some users preferred Pigtail, but others found it took practice to master. Although Timeout was significantly slower than the other techniques, several users preferred it, as it was easy to learn and had less than 0.5% errors. Nobody liked the Button, as it proved difficult to correctly time the button press. We also report a preliminary follow-up study of a design iteration of Pigtail that suggests it can yield performance similar to Handle, thus making it a promising new design alternative.

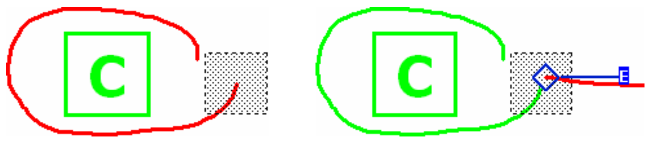

Fig. 3. The Handle. *Left*: The first stroke is the lasso. *Right*: A 2nd stroke starting on the handle is the mark.

Pigtail offers designers of pen interfaces some unique design properties, such as the ability to signal a state transition in the midst of a pen stroke via self-crossing. To study these we are developing the *Scriboli* testbed. The name is a play on Tivoli [18] that evokes the fast, informal, and uniquely pen-based scribbling nature of the interactions we seek to design. Scriboli is a prototype application with limited functionality, similar in sprit to GEdit [12], but we plan to support note-taking and drawing functionality similar to *Windows Journal* or *OneNote* on the Tablet PC.

Scriboli demonstrates key aspects of the Pigtail delimiter including: a simple animated self-disclosing mechanism; compatibility with various methods of scope selection including disjoint selections; and support for combining scope selection, command activation, and parameter specification via direct manipulation all in a single stroke.

RELATED WORK

GEdit [12] proposes selection-action techniques such as drawing a lasso for selection, and then ending the stroke *inside* the lasso to *Delete* the selected objects, or ending the stroke *outside* the lasso to *Move* a group of objects. To *Copy* a group of objects the user makes the Move gesture but adds a "C" to the end of it. However, these techniques only support three commands (Move, Copy, and Delete) for groups of objects, and the system suffers inconsistencies between "multiple object" and "single object" versions of commands. The delimiter techniques explored in our work generalize and extend GEdit's techniques, providing a consistent mechanism to apply any of 8 different marking menu commands to one or more objects. GEdit even shows a user making a loop similar to our pigtail gesture as the user transitions from lassoing to moving a group of objects ([12], page 141, Figure 5), but GEdit *does not recognize such loops* or use them as part of the interface's design vocabulary. Similarly, FlowMenu [10] shows loops being formed in the course of the user's input, but these are incidental, and do not serve as lexical breaks.

Tivoli [18] uses a pigtail gesture to represent the *Delete* command, but only recognizes the pigtail *after the pen is lifted*. Our system instead uses pigtails as a delimiter: Scriboli looks for pigtails during gesture mode while the pen moves in contact with the display. A self-intersecting gesture meeting our recognition criteria for a pigtail *immediately triggers marking mode, before the pen is lifted*. As Moran et al. note, "Prompting techniques, such as marking menus, can only partially help [users to recall gestures], because many of our gestures must be drawn in a spatial context to indicate their meaning" ([18], p. 53). Pigtail helps to address this fundamental design problem by providing prompted marking commands as a possibility during almost any pen gesture indicating a spatial context.

Fluid Inking [23] explores the use of punctuation as a type of delimiter in pen interfaces. A tap is used as a cue to the system to attempt to recognize the preceding ink stroke(s). Stylus input without prior selection of mode [21] takes a recognition-based approach to classify lassos drawn during ink mode as selection gestures, and to provide options in a menu if it is uncertain.

Many pen interfaces support an *ink mode* for entry of raw ink strokes, and a *gesture mode* for entering commands [18, 19]. Li et al. [16] show that using the nonpreferred hand to perform an explicit press of a button on the Tablet PC's bezel is a robust technique for ink/gesture mode switching, costing only 139ms per mode switch with about a 1% incidence of mode errors. *Our delimiter techniques are intended for use in gesture mode*. Since Li et al. carefully analyze mode switching issues, our experimental task always stays in gesture mode, allowing us to focus fully on delimiters. However, outside of the experiment, Scriboli uses the nonpreferred hand button technique [16].

Crossing interfaces form an area of recent interest for pen interaction [1, 2]. Pigtail suggests self-crossing of a gesture as a new design element for pen interfaces. Although our experiment focuses on lasso selection, we designed all four of our delimiter techniques to be general methods that are applicable to other methods of indicating scope. For example, Fig. 4 illustrates crossing and tapping scopes.

DELIMITER TECHNIQUES

We now discuss the delimiters used in our experiment, so that others can understand our design decisions and reproduce our work. We implemented each technique as closely as possible to the way we expected it would actually be used in practical applications.

Pigtail Delimiter

Pigtail uses the self-intersection point of the loop to terminate the scope (lassos are closed via automatic completion [17]). The self-intersection point also defines the origin of the marking menu. Any additional self-intersections are ignored[1]. Our system starts marking mode (and makes the audio cue) as soon as it identifies a self-intersection in the pen trace that meets our recognition criteria for a pigtail. We pop up the menu if the user has not completed a mark more than 20 pixels long within 333ms.

We only look for pigtails in gesture mode, as natural handwriting contains many closed loops. Pigtail attempts to leverage existing skills of users for the mechanics of handwriting, which essentially consists of variations on small rhythmic looping motions with the stylus [22].

Our recognition of pigtails is straightforward. Each new sample of a gesture forms a line segment. We search the current pen stroke backwards over a 2000ms time window looking for an intersecting line segment. If one is found, we compute the area A and perimeter P of the resulting closed polygon. We ignore degenerate loops (A<5 or P<15 pixels) that tend to occur at inflection points in the pen stroke. Large loops (A>5600 pixels) typically represent self-intersection of a lasso. Everything else is treated as a pigtail. False positive recognition of pigtails is possible, e.g. a small self-intersecting lasso is interpreted as a pigtail. This is not a problem, as automatic completion allows the same selection using a non-self-intersecting lasso.

Handle Delimiter

The Handle delimiter is based on GEdit's technique of attaching a small box to the end of the selection lasso [12]. The purpose of the handle is to provide an activation point for the scope so that the user can defer the *action* portion of a selection-action command phrase to a later point in time, i.e. a subsequent gesture stroke. Kurtenbach & Buxton focus on continuous pen gestures, but point out that splitting such operations into two steps can sometimes be advantageous, such as when selecting a very complex scope, or when re-using a scope for multiple commands.

[1] In our design iteration, we found this design decision may have been a mistake (see the *Design Iteration: Pigtail-2* section).

In our implementation, a 48x48 pixel handle (Fig. 3) appears as soon as the system receives the *Pen Up* event for a valid lasso. Upon a *Pen Down* event inside the handle, marking mode begins and we provide the user with a brief audio cue to signal this. If the user has not completed the mark within 333ms, we pop up the menu [12].

We considered using the entire lasso region as a handle, but decided against this. If the user draws too large of a selection region by mistake, it becomes cumbersome to fix this because starting a new lasso within the mistaken one would be interpreted as a mark. And if all pen events inside the lasso are interpreted by the marking menu, this prevents applications from supporting other gestures (e.g., tap, tap-and-hold, drag) within a lasso. Finally, it also prevents the use of nested circles of exclusion [12] to except objects from the scope. Hence the handle is a more general technique that still provides very quick access to marking.

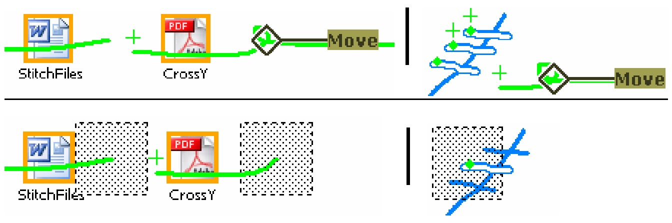

Fig. 4. Pigtail and Handle with crossing and tapping. *Top*: The user crosses one icon, crosses a second icon, and then makes a *Pigtail* to move both icons. The user can also tap objects and then draw a pigtail. *Bottom*: In these cases handles may get in the way or be ambiguous.

However, Handle has its limitations, particularly when applied to very small scopes or disjoint scopes. For example, the situations depicted in Fig. 4 represent tough cases for the Handle delimiter for two main reasons:

Inaccessible space. Much of the empty space between the icons becomes inaccessible for crossing due to occlusion by the handle, making it difficult to successfully draw the second crossing stroke (Fig 4, bottom left). Tapping on one object may make other small objects nearby unreachable (Fig 4, bottom right). Reducing the size of the handle would make it slower and more difficult to hit with the pen.

Ambiguity. It is not clear how to operate on a collective scope since there may be more than one handle. Does each handle affect just the scope it is attached to, or all of them? How does the user disambiguate this confusing situation? As Kurtenbach & Buxton [12] note, “We have established the convention that [a] command must be initiated [from the handle] if it is to affect the encircled objects. The dilemma is, […] should an operation on one simultaneously affect the other? In GEdit, the answer is no.”

Pigtail’s design eliminates both of these problems.

Timeout Delimiter

Many pen-operated devices use a pause without moving the pen as a way to synthesize an extra input state [5]. Our Timeout delimiter is a drag-and-hold gesture, which requires holding the pen still for 500ms while dragging (drawing the lasso). We evaluated this technique with test users in a previous project [11], where we found a 500ms delay between cessation of pen movement and popping up the menu represents a good tradeoff between popping up the menu as rapidly as possible, while not being so short a delay that it leads to excessive accidental activations.

When the 500ms timeout expires, we provide the same audio cue as used for the other delimiters. However, we decided to pop up the menu immediately instead of waiting an additional 333ms. As seen in Fig. 5, our menu has a minimal visual footprint, so we felt that it would not distract users. We did try using the extra 333ms delay in pilot studies, but some users always waited for the menu to appear, which might bias results against the Timeout technique, so we decided to use only the 500ms timeout.

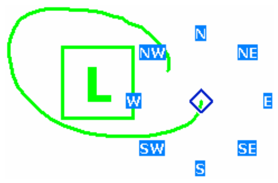

Fig. 5. Timeout technique. As shown here, the menu pops up as soon as the 500ms timeout expires.

Our implementation allows for *pen travel* (incidental movement of the pen) during the timeout. The pen is considered to be stationary as long as it remains within ±5 pixels of the position at which the time-out started. Motion beyond this threshold restarts the timeout.

Button Delimiter

Since Li et al. [16] found using a button in the nonpreferred hand very effective for ink/gesture mode switching, we wondered if a similar button might be effective as a delimiter. The Button Delimiter uses the *timing* of a button press as a cue to indicate when to delimit an ink stroke. Participants in our study used their nonpreferred hand to press the CTRL key. When our system observes the *Key Down* event, we start marking mode (again producing an audio cue), and use the current pen position as the origin of the marking menu. The menu pops up 333ms later.

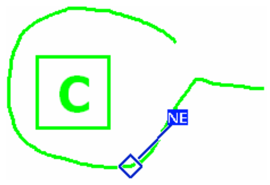

Fig. 6. Button delimiter. Here the user hits the button too early and selects NE instead of E as intended.

Pilot testing led us to suspect that the Button technique might suffer from synchronization errors: it seems to be difficult to hit the button at exactly the right moment. Fig. 6 shows an example where the user intended to select E, but hit the button too early and instead selected NE by mistake. On the other hand, if one can time the button press correctly, or if one can plan ahead such that the lasso is already heading in the desired direction at the time of the button press, the technique seems very fast.

We considered implementing a "rollback mechanism" that, upon the Pen Up event, searches backwards in the pen trace looking for an inflection point to determine the marking direction. However, for single-level marking menus, this is prone to accidentally recognizing curves in the lasso as inflection points. For two-level marking menus, this introduces an ambiguity; e.g. in Fig. 6, the recognizer cannot determine if the user intended to select NE$\rightarrow$E, or if the user intended only to mark to the E but hit the button too early. Since there is no clear solution to this problem, we decided to evaluate the Button delimiter as-is to see if synchronization errors are really an issue in practice.

EXPERIMENT

Our experiment focused on delimiters for pen gestures. The goal was to evaluate the time efficiency and error rates of the Pigtail, Handle, Timeout, and Button delimiters.

Experimental Task

The experimental task prompted users with a selection region and a marking direction. The user's task was to lasso the items in the selection region and apply the correct marking direction to those objects. The marking direction to choose appeared at the top of the screen, using a filled pie wedge labeled with the compass direction (Fig. 7).

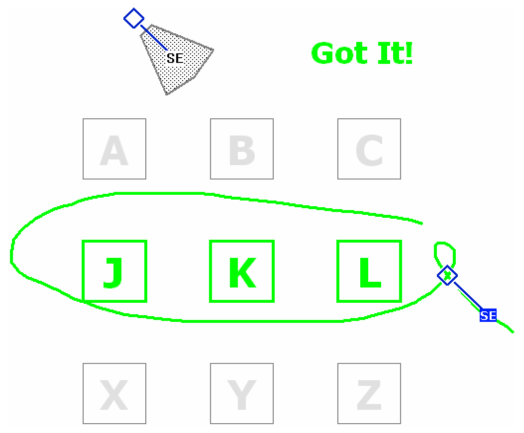

Fig. 7. Example screen from the experiment.

Users were prompted with 9 square targets, each 64x64 pixels, forming a 3x3 grid centered on the screen, with 64 pixels of empty space between targets. Each target contained a single letter; the purpose of the letter was to make the targets distinct and to make it clear that the square represented an "object" and not just an outline. The targets to be selected were shown bolded in black, while other "distractor" targets were shown in gray. The targets were considered to be inside the lasso as long as the center point of the target fell within the lasso. Subjects were instructed that lassos are automatically completed and that lassos can be a bit "sloppy" and cut off the corners of the squares.

A correct response by the user was rewarded with a happy sound and a "Got It!" message; after 500ms, the experiment advanced to the next trial. An incorrect selection and/or incorrect marking direction played a "miss" sound and indicated the error (e.g. "NE is WRONG: Select N"). This remained on the screen for 1750ms so that users could see their error before the next trial started.

The *Selection Type* was either a *single* target or *multiple* targets. For *multiple* target selections, the selection always consisted of 3 contiguous squares, randomly selected as one of the rows or columns of the grid. We considered using more complex selections, but decided not to for two reasons. First, the locality of reference principle implies that users will most frequently select a set of objects that are spatially proximal, such as a line of handwriting or a column of a table. Irregular selections are less common. Second, our experimental design includes all possible marking directions for each selection type, so adding more selection types would have made the experiment too long. We plan to explore irregular selections as well as scope selection gestures other than lassos in future experiments.

Experimental Design

Independent variables included Delimiter Technique (Pigtail, Handle, Timeout, Button), Marking Direction (N, NE, E, SE, S, SW, W, or NW), and Selection Type (Single or Multiple). Dependent variables included Completion Time (the time between *Pen Down* at the start of the lasso to *Pen Up* at the end of the mark) and Direction Errors (i.e. the rate of incorrect marking direction selection).

For each technique tested, the experimenter explained the technique to be used. Participants performed 32 practice trials to familiarize themselves with the technique. Then they performed 80 main experimental trials. The trials were clustered into sets of 5 repeated trials prompting the user with the same objects to select and the same marking direction to choose. We structured the experiment with clusters of repeated prompts so that we could simulate prolonged practice with each technique in a short time.

Participants then completed a *repeated invocation block* consisting of 24 repetitions of an identical trial. These trials repeatedly prompted the user with the same square to lasso, and the same mark (East). We chose East for repeated invocation as this seems to be a "sweet spot" in the menu that we expect would be assigned to a frequently used command. The repeated invocation block allowed us to assess the performance limits of each technique.

We employed a 4x4 Latin square to minimize order effects, with 2 participants each in 8 different orders. One participant did not show up, leaving the Pigtail 1st orders with just three users instead of four. We were not able to replace this participant, but our analysis showed no order effects. Thus the experiment included 15 users, each with:

4 Delimiter Techniques x
 32 practice trials +
 8 Marking Directions x
 2 Selection Types x
 5 trials = 80 main experimental trials
 + Repeated Invocation block of 24 trials
 = 136 trials per condition

=8160 total trials (including 1920 practice trials, 4800 main experimental trials, and 1440 repeated invocation trials).

Apparatus and Tablet Configuration

Each participant ran the experiment on a Toshiba Portege 3500 TabletPC, running Windows XP SP1 Tablet Edition, with a 24.5 x 18.5 cm (1024 x 768 pixel) display. Since the Button technique uses the CTRL key, we had subjects use the TabletPC in the "clamshell" configuration, with the screen open and angled upwards to make the keyboard accessible. We supported the angled tablet screen from behind with several heavy books so that it would provide a solid, stationary writing surface during the experiment. All subjects used the tablet on a desk. Subjects were encouraged to angle the tablet and screen as they preferred.

Participants

Fifteen persons participated in the study. We recruited users with pen computing experience, as Scriboli's targeted users are current TabletPC users (8/15 participants) or likely future tablet users. All participants were male as it was difficult to find female participants. One was left-handed.

RESULTS

Subjects took approximately 20 minutes to complete each technique, including practice trials, the main experimental trials, and the repeated invocation block.

Completion Time: Main Experimental Block

We conducted a 4 (Delimiter Technique) x 8 (Marking Direction) x 2 (Selection Type) within-subjects ANOVA on the median completion time within each cell of the experimental design for the main experimental trials. We used the median completion time to correct for the typical skewing common to reaction time data; this also removed the influence of any outliers in the data. Order of presentation was included as a between-subjects factor, but yielded no main effect or significant interaction effects.

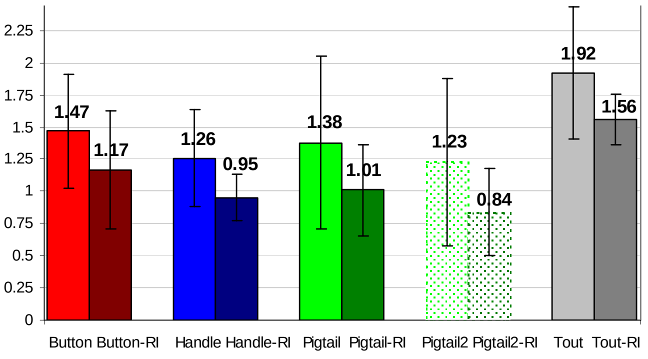

Fig. 8. Average completion time. For each Delimiter Technique, the left bar is the main block; the right bar is the repeated invocation (RI) block. The Pigtail2 bars show preliminary results for our design iteration.

We analyzed completion time (Fig. 8) only for correct responses. While running the experiment, we observed that on the first trial of a new set of 5 repetitions of the same selection-action prompt, subjects often started to respond using the prompt they had just experienced in the *preceding* set of 5 trials. Thus we decided to remove the first trial of each set of 5 repeated trials from our analysis. Our analysis revealed a significant effect for *Delimiter Technique*, $F_{(3,21)}$=18, p<.001. Post-hoc (Bonferroni) pairwise comparisons revealed that Timeout was significantly slower than all other delimiters (p<.01), but completion times for Pigtail, Handle, and Button did not differ significantly from one another. As expected, *Selection Type* $F_{(1,7)}$=151, p<.001 was also significant, as the multiple-target lassos took more time to draw. *Marking Direction* did not yield a significant main effect. There were no significant interactions.

Completion Time: Repeated Invocation Block

A 4-way ANOVA on the median completion time for each subject's repeated invocation block (excluding all error trials) revealed a significant main effect of Delimiter Technique ($F_{(3,42)}$=13.7, p<.001). Post-hoc pairwise (Bonferroni) comparisons revealed that the Timeout was again significantly slower than all other techniques (p<.01), but that the other 3 techniques did not differ significantly.

Inspection of the mean completion time over the 24 trials revealed some interesting trends (Fig. 9). This graph includes error trials. Subjects clearly experienced problems with the Button technique: unlike the other three delimiters, completion time shows an unstable trend. This suggests that users could not effectively compensate for synchronization errors with practice: the problem instead became worse.

The Pigtail and Handle follow performance curves that appear very similar to one another. The Timeout technique appears to quickly reach a plateau, suggesting a possible floor effect on performance due to the timeout.

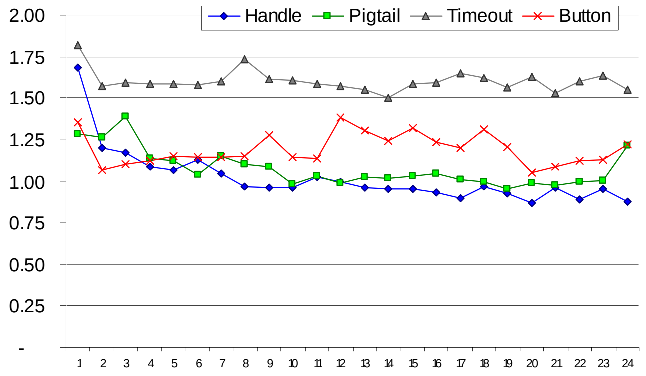

Fig. 9. Learning effects for repeated invocation block.

Errors: Main and Repeated Invocation Blocks

A Kruskal-Wallis nonparametric test on Direction Errors in the main block (Fig. 10) revealed a significant effect of Delimiter Technique, $\chi^2_{(3,\ \underline{N}=64)}$ = 35.4, p<.001. Additional tests for Marking Direction and Selection Type showed no significant effects. In the repeated invocation block, A Kruskal-Wallis test again revealed a significant effect for Delimiter Technique, $\chi^2_{(3,\ \underline{N}=60)}$=18.3, p<.001. The Button Delimiter Technique error rate increased from 4.6% in the main experimental block to 8.1% in the repeated invocation block, suggesting that synchronization errors are exacerbated when the user tries to work quickly, and that practice does not seem to reduce such errors.

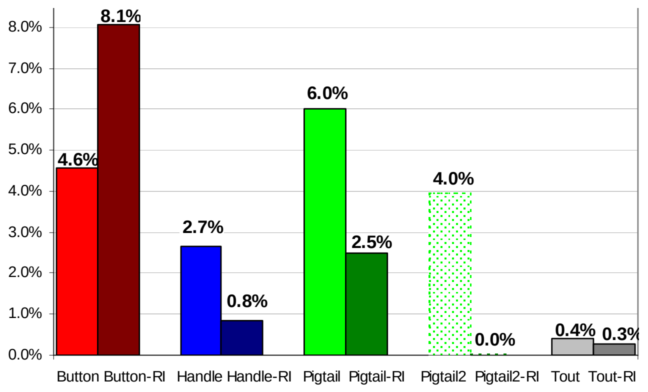

Fig. 10. Marking Direction error rates. Pigtail2 shows the tentative error rate of our design iteration.

Qualitative Results

After the experiment, we asked participants to rank-order each of the four Delimiter Techniques. Seven participants chose Handle as their favorite technique, 4 chose Pigtail, and 4 chose Timeout. Nine participants ranked the Button as their least favorite technique; 3 chose Pigtail, 3 chose Timeout, and nobody ranked Handle as their least favorite.

Clearly, Handle was a predictable technique that provided fairly fast performance for most users. But response to Pigtail was mixed; some users loved it, and others felt that it did not work well for them. For example, one user commented that the Pigtail "felt just like handwriting… it was very natural and once I learned it I could open up and fly!" while another wrote "Pigtail was trouble. I got a little mad at it." Participants felt it took some time to learn, and may need more than 15-20 minutes to become proficient.

The Button clearly was not successful. Using the timing of the button press seems to be a poor choice from a human performance standpoint as users cannot overcome problems with synchronization errors; even though two of our participants recorded their fastest average times with the Button, it was uniformly disliked. There is not any obvious technical solution to its problems, and repeated use only results in higher error rates as the user tries to go faster.

The Timeout technique was slower than the other techniques, but also had by far the fewest errors (<0.5%), suggesting it would be the best technique to use in contexts where errors might have a very high cost. User reaction to Timeout was mixed. All users felt the technique was easy to learn. Some loved its dependability, while others detested having to dwell. Many users felt it is a "good technique for beginners" but "hate waiting for the computer" and want a way to go faster.

DESIGN ITERATION: PIGTAIL-2

Although Handle had the best overall performance in our experimental study, we felt that Pigtail was worth further investigation due to its novelty and unique design properties. If small improvements can reduce the error rate of Pigtail, it may offer a compelling design alternative to the Handle delimiter.

Limitations of Initial Pigtail Implementation

While observing our experimental participants, it became clear that it was not satisfactory to simply define a pigtail as the first closed loop with area 4<A<5600. In the practice session, some participants naturally seemed to start by drawing loops larger than this threshold, resulting in an unrecognized input. This often led to a task strategy of circling multiple times to ensure an intersection; we believe this strategy may have slowed performance and led to more error-prone marking menu selections. However, this seems a problem that may be ameliorated with a better recognizer.

Some users produced *pretzels* (Fig. 11) for certain marking directions. Although these were recognized as pigtails, using the first self-intersection to define the origin of the menu may result in marking menu direction errors.

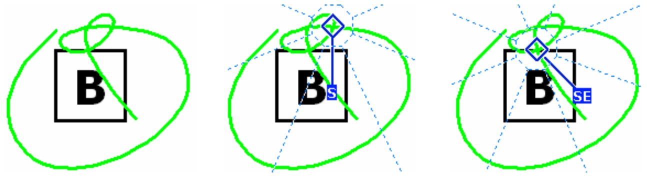

Fig. 11. *Left*: Pretzel drawn for prompt of SE. *Middle (incorrect)*: Using the first intersection leads the system to mistakenly view S as the mark. *Right (correct)*: Our new design correctly interprets the trace to select SE.

We also observed that since the user's hand is moving in a circular motion to form the pigtail, the resulting mark often makes a curve rather than a straight line. Some direction errors occur when the user "curves" a bit too much.

An inherent limitation of the Pigtail is that for new users, some marking directions seem harder to choose than others. In particular, marking in the direction opposite of the current direction of pen motion requires some practice.

Pigtail2 Design Changes to Address Limitations

In response to these problems, we eliminated the upper bound on the pigtail size: our *Pigtail-2* design iteration allows any self-intersection of the pen trace above a minimum area as a pigtail. As a result we must consider multiple self-intersection points as candidates (Figs 11, 12).

We start by assuming the first intersection point is the correct origin for the menu. This may in fact be the user's intent, and we cannot wait until the pen is lifted to decide. We then recenter the marking menu at a later self-intersection point if our algorithm determines that it is a better candidate for a pigtail. Recentering the menu in this manner did not seem to cause any significant problems for test users in a preliminary pilot study of *Pigtail-2*, although one user felt it was sometimes a little distracting. But recentering allows us to accept pigtails of any size and to correctly interpret traces such as those shown in Fig. 11 and Fig. 12. A pigtail is deemed better than the previous candidate if (1) their time intervals overlap and the ratio of perimeters is less than 1.75, or (2) if the centroid of the new pigtail falls within the lasso and the ratio of perimeters is less than 0.6.

Note that with this technique, any self-intersecting lasso is treated as a "pigtail" that selects the contained items and invokes a marking menu. To select items without acting on them, the user must draw a non-self-intersecting lasso.

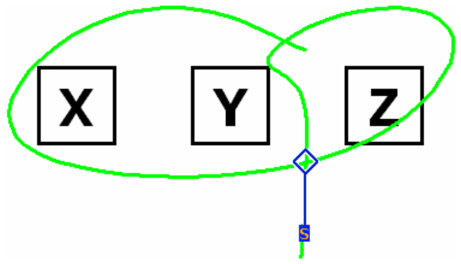

Fig. 12. Example of a very large pigtail with two intersection points. Here the user correctly selects S.

We also dynamically adjust the marking menu sector boundaries based on the stroke path to accommodate users' natural curving motions. We slightly expand the marking menu sector that currently contains the pen stroke. We expand the boundary by 3°, but only in the direction of the pigtail loop (typically counterclockwise). Once the trace passes this 3° expansion, the adjacent sector becomes selected, and the boundary is adjusted back to its original position (with no expansion).

Pigtail-2 Preliminary Results

We ran a pilot study, using 6 colleagues (5 male, 1 female) to reproduce the Pigtail condition of our experiment. The enhancements noted above do seem to improve performance of Pigtail, particularly with respect to errors: Pigtail-2 resulted in 4% errors in the main block of the experiment, and we observed no errors in the repeated invocation block. The average completion times also dropped slightly to 1.23 seconds (main block) and 0.84 seconds (repeated invocation block), but these have high standard deviations due to the small number of pilot subjects. These are shown as tentative results in Figs. 8, 10.

In some implementations, designers may not wish to allow arbitrarily large self-intersecting loops to be interpreted as pigtails. Inspection of the frequency distribution of pigtail size (including practice trials) suggested the presence of a bimodal distribution, with 93.5% of pigtails having an area of 10000 pixels or less. The remaining very large loops mostly represent self-intersecting lassos that lack a true "pigtail." Our pilot users infrequently made their marking menu selections in this manner; most subjects experimented with it a few times, but abandoned it in favor of a more consistent strategy of always forming a pigtail loop and then heading in the desired direction. This suggests that using an upper bound of approximately A<10000 pixels would likely be acceptable to most users and have little impact on performance of the technique.

DISCUSSION OF THE SCRIBOLI TESTBED

We now discuss our Scriboli pen interaction testbed and show how we currently use Pigtail in an application fragment. The intent of this discussion is to give the reader a better understanding of how we envision using delimiters in an application (as opposed to our experimental task), and also to raise a number of design issues that we considered.

Scriboli includes support for freeform ink input, pen gesture input, and structured objects (e.g. pictures and icons). We plan to develop Scriboli into a “scrapbook” application supporting note-taking and ideation activities surrounding personal photographs, drawings, or clippings from the Web, for example. In its present form Scriboli is not intended to be a complete application, but is a testbed and tool for us to experiment with design alternatives and conduct quantitative studies on pen input techniques. Scriboli currently implements Cut, Copy, Paste, and Move operations on all objects, plus a few other commands. Our figures and accompanying video show a number of other commands in Scriboli’s menus; these are unimplemented placeholders that give a feel for how a more fully featured application could be structured using our techniques.

Preserving the Design Space of Gestures

A primary design goal for Scriboli was to devise techniques that could be applied to a wide range of pen-operated interfaces. For this reason, we tried to be very careful not to consume any more of the design space of pen interaction states [5] and possible gestures than absolutely necessary.

In the course of demonstrating Scriboli to colleagues, some have suggested that we could just use the final direction of a lasso selection (without any delimiter) as the marking direction. This might work (although the user would have to plan how to draw the lasso to end in the correct direction), but it is not a very general technique because it would *impose a command activation (marking) phase upon all gestures*. This would make it impossible to draw a gesture that is not immediately interpreted as a command.

For example, when the user starts drawing a lasso in Scribioli, the user may safely lift the pen and start over at any time if he or she makes a mistake. But this type of failsafe operation would not be possible if the system interpreted all gestures as ending with a mark. The presence of a delimiter leaves the user in control of whether or not a transition to marking mode occurs. Furthermore, without a delimiter, it would not be clear when to pop-up the menu for prompted selection. This would prevent self-revelation, which is widely considered to be one of the major benefits of the marking menu approach. One could use a pause to pop up the menu, but then this essentially becomes the equivalent of our *Timeout* delimiter technique (i.e. it introduces a delimiter).

Self Revelation of Pigtail via Stroke Extension

After watching our experimental participants learning to use Pigtail, it was clear it could be improved by making it more easily discoverable and by somehow providing the user with examples of how to draw the gesture. Since our participants found Timeout simple to learn, but tedious to wait for with repeated use, we felt that we could improve on both techniques by combining them in a new hybrid approach. Some devices that support tap-and-hold already provide speculative feedback that shows a button press animation or a circling clock before the timeout expires.

Like the Timeout technique, *Stroke Extension* (Fig. 13) uses a pause while drawing a gesture to initiate a timeout. The system *draws an animated extension of the current stroke to form a pigtail for the user*. If the user continues dwelling, the system pops up the menu (like the Timeout technique). If the user instead draws a pigtail, this short-circuits the timeout so that the user need not wait for the computer.

We are also experimenting with stroke extension after a *Pen Up* event for a scope that lacks a pigtail. In this case, the menu does not activate, but the user must instead trace the dotted line, thus drawing a pigtail and bringing up the menu. Unlike the Handle technique, this stroke extension disappears after a couple of seconds, and does not make any pixels of the screen unreachable.

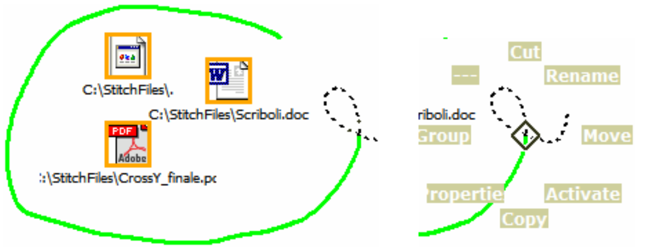

Fig. 13. Animated stroke extension (see also our video). *Left*: Stroke extension shows how to draw a pigtail. *Right*: The menu pops up when the animation finishes.

Kurtenbach et al. [14] explore contextual animation of gestural commands in Tivoli. Tivoli provides animated gestures, but they are not appended to a user’s ongoing pen stroke in real time. Rather they must be explicitly requested by clicking on a “crib sheet” of available gestures. Stroke Extension is a novel technique that allows *Scriboli* to show the user how to draw the pigtail shorthand gesture that can be used to short-circuit the timeout. New users can wait for the timeout, but in so doing can also be led to learn the pigtail as a way to speed their performance.

Types of Scopes

Our experiment focused on delimiters for lasso selection. Scriboli implements several other types of scopes.

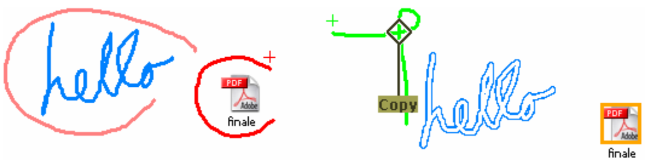

Fig. 14. Disjoint scope via holding the gesture button. The user draws two lassos, then pigtails to *Copy* both.

Scriboli supports *disjoint scopes* (Fig. 14). Scriboli allows the user to continue holding the nonpreferred hand ink/gesture mode button of Li et al. [16], thus using muscle tension to phrase [4] together multiple pen gesture strokes. Reselecting an object toggles its selection bit, allowing scopes with exceptions: for example, a nested lasso deselects the encircled objects [12]. Note that disjoint scopes can be used to effectively work at the edges of the screen; the user can lasso objects near the bezel, and then separately draw the pigtail elsewhere. Disjoint scopes underscore the design advantages of the Pigtail technique, as the user can combine various scopes into a single phrase that is terminated by a special operator (the pigtail). As shown earlier in Fig. 4, the Handle technique suffers in such situations.

A *single item scope* accesses the menu specific to an individual object on the screen. Drawing a pigtail on top of a single item (with no preceding lasso) selects that item and activates its menu. Here, we select the topmost item that is hit by the self-intersection point of the pigtail. This is the equivalent of a click with the mouse, but has the virtue that it leaves the pen's *Tap* event available for other application-specific uses, rather than consuming it for menu activation.

Scriboli does optionally support *tapping scopes*, however. Tapping is the ultimate way to select individual objects in a cluttered scene. Tapping cannot lead directly to a pigtail: the user must tap object(s) to select them, and then separately draw the pigtail, which acts on the selection.

Scriboli also supports crossing [1, 2]. A *crossing scope* is any line that completely crosses one or more objects (e.g. Fig. 4). Crossing is distinguished from lassoing by initially assuming the stroke is a lasso, and then comparing the ratio of the automatic completion line to the stroke length. A value near unity indicates a crossing stroke.

Finally, the *null scope* allows access to a global menu by drawing a pigtail over empty space (with no prior selection). This menu currently includes commands such as Paste, Exit, Next Page, and Previous Page. The Paste command uses the self-intersection of the pigtail to determine where to Paste the clipboard contents.

Continuous Direct Manipulation Phase in Scriboli

Pigtail supports a direct manipulation phase similar to that of FlowMenu [10] or ControlMenu [20] to allow the integration of command selection with direct manipulation. For example, a FlowMenu user can select *Move*, cross the outer boundary of the menu, and continue dragging to interactively reposition an object. FlowMenu also supports a *Select* command that starts a lasso upon crossing the outer menu boundary ([8], video for [9]). But FlowMenu cannot first specify the scope and then the command; it has to be the other way around. Unlike FlowMenu, Pigtail supports all three phases in a single continuous pen stroke (Fig. 15).

FlowMenu always uses crossing of the outer boundary of the menu to activate commands, whereas marking menus exhibit scale independence [24] so that the user can draw a mark of varying sizes in the mark-ahead mode. In practice this means the command activates on *Pen Up*, which prevents traditional marking menus from supporting an integrated direct manipulation phase.

To enable *Scriboli* to support a direct manipulation phase while also allowing for scale independence in mark-ahead mode, we devised a variation on outer-boundary crossing. We observed that the scale independence property only matters in mark-ahead mode: once the visual menu appears, the scale of the marks is determined by the size of the menu. But if a command does include a direct manipulation phase, this will become a visually guided dragging task that continues beyond the 333ms delay between the start of marking mode and popping up the menu. So, we support scale independence for the first 333ms of a mark, but then support direct manipulation after that time-window expires.

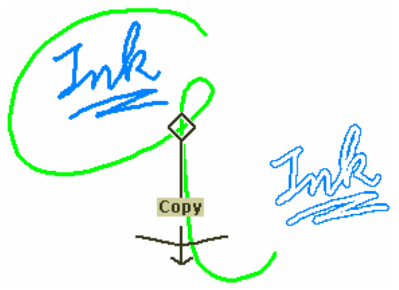

Fig. 15. Pigtail direct manipulation phase. Here the user drags a copy of the "Ink" to the desired location.

Thus, we simply defer crossing detection until after the 333ms menu popup delay expires. We believe this detail is an important extension to radial menu techniques, as it allows our technique to support *both* scale-independent mark-ahead as well as a crossing-based direct manipulation phase. This approach allows designers to unify properties of the previously separate techniques for radial menus with crossing [10, 20], versus radial menus with mark-ahead [12, 24]. In our experience with the technique so far, it seems to work extremely well, but we have not yet conducted usability testing focused on this feature (menu boundary crossing was always turned off during our experiment).

CONCLUSION AND FUTURE WORK

Altogether, Scriboli takes an alternative approach to the design of pen interfaces. It does not place recognition problems that may be intractable (in the general case of unrestricted ink input) in the way of providing new user experiences for currently available pen-operated devices. Scriboli instead seeks to introduce fundamental building-blocks that are rapid, unambiguous, and expressive. The Pigtail delimiter offers one such building- block.

In our experimental selection-action task, our initial implementation of Pigtail did not perform quite as well as we had hoped, yet several test users still preferred the technique. In our study, Pigtail exhibited an error rate of 6%, but this dropped to 2.5% with repeated invocation. This suggests users can substantially reduce their error rates with practice. Our Pigtail-2 design iteration suggests that minor improvements can reduce the error rate of Pigtail to about 4% with time efficiency that compares favorably to the other techniques. We plan to further explore Pigtail's unique design advantages in future work.

On the other hand, while we initially thought the Button delimiter might offer a fast approach, our study suggests that it suffers significant human performance limitations. Unlike our improvements to Pigtail, we do not see any way to fix the problems we observed with the Button delimiter.

Since the main problem with the Handle in practice is that it adds visual clutter and sometimes gets in the way (Fig. 4), it is tempting to make the handle smaller. However, a small handle would likely make the technique much slower to use. Some subjects in our experiment achieved fast performance with the handle by doing a little hop with their pen at the end of the lasso; the handle was large enough that this hopping movement would always hit the box. Since our experiment has shown that the Handle technique offers good performance and is well accepted by users, we plan to further study the impact of handle size on performance. We are also considering ways of improving Handle or of integrating it with the Pigtail and Timeout techniques, since they do not inherently conflict with one another.

We would like to conduct a longitudinal study of Pigtail, as we expect the technique has more value as users become practiced with it. We are excited to further flesh out Scriboli and test the overall feel of the interactions when most or all of an application is accessed in this way. It should be possible to add support for hierarchical marking menus [13, 15, 24]. Pigtail allows for a direct manipulation phase at the terminus of any command, and we devised a novel implementation that provides this feature while preserving the scale-independence property of blind marking. In future experiments we would like to determine if this integration can yield significant performance benefits. We would also like to determine if our stroke extension techniques could serve as the basis of a self-disclosing gestural interface.

We would also like to explore what new capabilities and command structures can be added to Scriboli using its mechanisms for handling complex scopes consisting of multiple strokes prior to a pigtail for command activation. Finally, although the present design of Scriboli deliberately avoids recognition-based approaches, we would like to explore the use of Scriboli as a correction or direct manipulation layer for use in conjunction with recognition methods. In this way users could stay in control yet also harness the power of sophisticated recognition techniques.

ACKNOWLEDGMENTS
We would like to thank Ed Cutrell, Maneesh Agrawala, Shendong Zhao, and Gina Venolia for helpful comments.

REFERENCES
1. Accot, J., Zhai, S. More than dotting the i's-- Foundations for crossing-based interfaces. CHI 2002, 73-80.
2. Apitz, G., Guimbretiere, F. CrossY: A crossing based drawing application. UIST 2004, 3-12.
3. Buxton, W., Lexical and Pragmatic Considerations of Input Structure. Computer Graphics, 1983. 17(1): p. 31-37.
4. Buxton, W. Chunking and Phrasing and the Design of Human-Computer Dialogues. IFIP Information Processing`86, Amsterdam: North Holland, 475-480.
5. Buxton, W. A three-state model of graphical input. Proc. INTERACT'90, Amsterdam: Elsevier Science, 449-456.
6. Buxton, W., Fiume, E., Hill, R., Lee, A., Woo, C. Continuous hand-gesture driven input. Graphics Interface '83, 191-195.
7. Fitzmaurice, G., Khan, A., Pieke, R., Buxton, B., Kurtenbach, G. Tracking Menus. UIST 2003, 71-79.
8. Guimbretiere, F., Fluid Interaction for High Resolution Wall-size Displays. 2002, Ph.D. Thesis, Stanford University.
9. Guimbretiere, F., Stone, M. C., Winograd, T. Fluid Interaction with High-resolution Wall-size Displays. Proc. UIST 2001, 21-30.
10. Guimbretiere, F., Winograd, T. FlowMenu: Combining Command, Text, and Data Entry. UIST 2000, 213-216.
11. Hinckley, K., Ramos, G., Guimbretiere, F., Baudisch, P., Smith, M. Stitching: Pen Gestures that Span Multiple Displays. Advanced Visual Interfaces (AVI 2004), 23-31.
12. Kurtenbach, G., Buxton, W. Issues in Combining Marking and Direct Manipulation Techniques. UIST'91, 137-144.
13. Kurtenbach, G., Buxton, W. The Limits of Expert Performance Using Hierarchic Marking Menus. INTERCHI'93, 482-487.
14. Kurtenbach, G., Moran, T., Buxton, W. Contextual Animation of Gestural Commands. Proc. Graphics Interface'94, 83-90.
15. Kurtenbach, G., Sellen, A., Buxton, W., An emprical evaluation of some articulatory and cognitive aspects of 'marking menus'. J. Human Computer Interaction, 1993. 8(1).
16. Li, Y., Hinckley, K., Guan, Z., Landay, J. A. Experimental Analysis of Mode Switching Techniques in Pen-based User Interfaces. CHI 2005.
17. Mizobuchi, S., Yasumura, M. Tapping vs. Circling Selections on Pen-based Devices: Evidence for Different Performance-Shaping Factors. CHI 2004, 607-614.
18. Moran, T., Chiu, P., van Melle, W. Pen-Based Interaction Techniques for Organizing Material on an Electronic Whiteboard. UIST'97, 45-54.
19. Mynatt, E. D., Igarashi, T., Edwards, W. K., LaMarca, A. Flatland: New Dimensions in Office Whiteboards. CHI'99, 346-353.
20. Pook, S., Lecolinet, E., Vaysseix, G., Barillot, E. Control Menus: Execution and Control in a Single Interactor. CHI 2000 Extended Abstracts, 263-264.
21. Saund, E., Lank, E. Stylus Input and Editing Without Prior Selection of Mode. UIST'03, 213-216.
22. Wilson, F. R., The Hand: How its use shapes the brain, language, and human culture. 1998, New York: Pantheon.
23. Zeleznik, R., Miller, T., Holden, L., LaViola, J., Fluid Inking: Using Punctuation to Allow Modeless Combination of Marking and Gesturing. 2004, Brown University, TR CS-04-11, ftp.cs.brown.edu/pub/techreports/04/cs04-11.ps.Z.
24. Zhao, S., Balakrishnan, R. Simple vs. Compound Mark Hierarchical Marking Menus. UIST 2004, 33-42.