Phrasing Techniques for Multi-Stroke Selection Gestures

Ken Hinckley[1], Francois Guimbretiere[2], Maneesh Agrawala[1], Georg Apitz[2], Nicholas Chen[2]

[1]Microsoft Research, One Microsoft Way, Redmond, WA 98052
[2]University of Maryland, A.V. Williams Building, College Park, MD 20742
kenh@microsoft.com, francois@cs.umd.edu, maneesh@cs.berkeley.edu, geapi@cs.umd.edu, nchen@cs.umd.edu

Abstract

Pen gesture interfaces have difficulty supporting arbitrary multiple-stroke selections because lifting the pen introduces ambiguity as to whether the next stroke should add to the existing selection, or begin a new one. We explore and evaluate techniques that use a non-preferred-hand button or touchpad to phrase together one or more independent pen strokes into a unitary multi-stroke gesture. We then illustrate how such phrasing techniques can support multiple-stroke selection gestures with tapping, crossing, lassoing, disjoint selection, circles of exclusion, selection decorations, and implicit grouping operations. These capabilities extend the expressiveness of pen gesture interfaces and suggest new directions for multiple-stroke pen input techniques.

CR Categories: H.5.2 [Information Interfaces and Presentation]: Input

Keywords: phrasing, tablets, gestures, pen input, multiple strokes

1 Introduction

A number of pen gesture interfaces reported in the literature have sought to phrase together multiple subtasks within a single pen stroke [2,5,6,8,9,11,22]. For example, a single pen gesture can integrate the verb, direct object, and indirect object for a *Move* command by lassoing the objects to move, extending the pen stroke outside of the lasso, and then lifting the pen to indicate where to move the selected objects [11]. This unistroke strategy is extremely tempting for designers of pen-operated devices. Pen contact with the screen phrases multiple subtasks together into a single cognitive chunk [5], and also may help avoid the introduction of persistent modes. The unistroke strategy also amortizes any overhead cost required to initiate a gesture across multiple subtasks.

However, recent results suggest a muliple-stroke strategy can offer advantages. Zhao & Balakrishnan [26] have shown that "simple marking menus" that use multiple straight pen strokes to navigate through a hierarchical marking menu are fast and result in lower error rates than traditional compound marking menus [13] that require a single stroke (with pauses or inflection points delimiting the hierarchy). Guimbretiere and Apitz [2] propose crossing-based widgets in CrossY. The user can draw a single stroke that crosses multiple widgets, but Guimbretiere and Apitz also report that "novice users will perform one command at a time," so CrossY permits the user to draw multiple strokes that individually cross each widget in a dialog box. If CrossY required everything to be done in a single continuous stroke, then a mistake late in the process would force the user to start over. The user exits a CrossY dialog box by crossing the border, which serves as a cue to the system that the multiple-stroke input is complete.

Thus, while it is sometimes desirable (particularly for experts) to provide the option to articulate complex gestures in a single stroke, pen interfaces should not necessarily *require* users to do so. In our *Move* example, if the selection is complex, or if the target for the *Move* operation is on another page, a unistroke syntax may be cumbersome or infeasible. Furthermore, it is difficult to support a mix of selection techniques including tapping, crossing, and lassoing disjoint sets of objects within a unistroke syntax. Another problem with a strictly unistroke strategy is that users sometimes want to pull the pen away, for example while forming a complex selection, so that they can view the screen to verify progress without their hand occluding it.

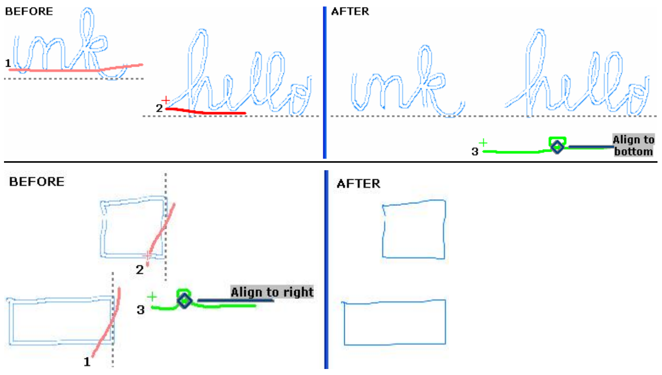

Fig. 1. Multiple-stroke alignment. **Top:** The user crosses the bottom of each word, then a third stroke with a pigtail activates a marking menu, which contains the *Align* command. The numbers indicate the starting position of each subsequent stroke. **Bottom:** Vertical crossing strokes allow the user to employ the same command gesture to perform a different type of alignment.

We advocate multiple-stroke selection gestures and demonstrate how some operations that are difficult to express with traditional graphical interfaces, such as alignment of specific edges of multiple objects (Fig. 1), become quite natural to articulate with pen gestures. However, previous gestural interfaces have had difficulty supporting arbitrary multiple-stroke selections because lifting the pen introduces ambiguity as to whether the next stroke should add to the existing selection, or begin a new one. For example, Kurtenbach & Buxton's GEdit prototype [11] supports circles of exclusion by allowing the user to draw a lasso-within-a-lasso, but does not allow the user to add to the selection nor issue a single command that acts on two or more disjoint lassos. GEdit also cannot support crossing as a selection gesture because a straight-line stroke always triggers a marking menu command. Tivoli [20] supports bump and bite gestures that add or remove objects from a lasso selection, but these gestures "must begin and end on the loop" [20], which makes them difficult to draw and precludes disjoint selection regions. Thus, Tivoli and GEdit can only support multiple-stroke selection gestures in special cases where a pen stroke starts on or within a prior lasso selection.

We first explore techniques that use a nonpreferred-hand button or touchpad to phrase together one or more independent pen strokes into a unitary multi-stroke gesture. Our experimental results suggest that users prefer a button and that a design that requires muscle tension from the nonpreferred hand for the full course of a phrase results in a very low error rate (2.5%). We then illustrate how this design can support multiple-stroke selection gestures with tapping, crossing, lassoing, disjoint selection, and circles of exclusion. We also propose novel selection techniques including *selection decorations* and *implicit groups*. Together, these contributions extend the expressiveness of pen gesture interfaces and may suggest new directions for multiple-stroke pen interaction techniques.

2 Related Work

Buxton [5] discusses phrasing as a way to organize subtasks into cognitive subroutines. He suggests that phrasing accelerates the proceduralization of knowledge into routine skills. Buxton raises the example of a pull-down menu, which uses tension on the mouse button to "glue together" the subtasks of invoking the menu, navigating to an item, and validating the command selection. This perspective motivated Sellen et al. [25] to conduct a study which shows that switching modes by pressing and holding a footpedal reduces mode errors in a text editor.

Li et al. [15] demonstrate that a nonpreferred-hand button offers a quick and effective mechanism for mode switching between plain ink strokes and unistroke gestures. Scriboli [9] uses such a button primarily to support a unistroke syntax that delimits the command from the selection by utilizing a self-crossing pigtail within the stroke. In this paper, we adopt the pigtail, but seek to use it as a way to transition to command selection in multiple-stroke phrases rather than as a delimiter within unistroke gestures. Other techniques, such as the selection handle proposed by Kurtenbach [11], could play a similar role but we chose not to use them because handles on multiple selection strokes become confusing and clutter the display. Scriboli [9] and PapierCraft [16] suggest that users might hold the gesture mode across multiple strokes, but neither evaluates this approach.

Some pen systems rely on recognition-based approaches to avoid explicit ink/gesture switching [14,23]. Even if we assume recognition is always correct, which of course it is not, a fundamental problem with automatic approaches is that the system cannot classify a set of strokes as a gesture or as ink until after the user has finished drawing the entire command phrase. This makes it difficult to provide incremental feedback or to prompt the user with available commands before the user commits to an operation. Our study establishes that a simple button can be used for robust articulation of multi-stroke phrases that may be difficult to support with purely recognition-based approaches.

3 Phrasing Techniques

A phrasing technique is the interaction technique used by a system to define the boundaries of a single command phrase in terms of elemental input actions. For example, if a system supports disjoint selections, then how does the user indicate that an individual selection region should be added to a selection, or start a new selection? A phrasing technique must consider both the timing of the input (when can the user start or end the phrase?) and the nature of the input (what does the user do to start, maintain, or end a phrase?). The design also must consider that the user might change his mind or make a mistake at any point in the process, so the technique must be flexible and allow the user to abandon an incomplete phrase.

Previous work [5,25] suggests that the degree of tension required of the user may be an important aspect of phrasing. Buxton [5] argues that "we can use tension and closure to develop a phrase structure [that] reinforces the chunking that we are trying to establish," but we are not aware of any study that has investigated the effectiveness of phrasing techniques for multiple-stroke pen gestures. We use the term *phrase tension* to mean the use of physical tension to 'chunk' actions, and we adopt this as a guiding principle to devise several possible phrasing techniques that combine a nonpreferred hand button with a pen held in the preferred hand, as follows:

Full-tension: The user must maintain muscular tension throughout the phrase. The user presses and holds the nonpreferred hand button at the beginning of the phrase, and must continue holding it until the end of the last pen stroke (Fig. 2). That is, the user must lift the button to end the command phrase.

Half-tension: The user presses and holds the mode switch button while perfoming one or more strokes to indicate the selection, and must continue to do so until a pigtail terminates the selection, and activates a marking menu. The user applies muscular tension at the start of a phrase, and maintains it through the climax of a phrase, but can relax after that (Fig. 3). This is the technique proposed by Scriboli: "the user can combine various scopes into a single phrase that is terminated by a special operator (the pigtail)" [9]. One virtue of the half-tension technique is that it allows the user to issue several commands in a row while continuing to hold the button, since a pigtail (rather than a button-up event) separates commands from one another. However, our experiment does not investigate this task scenario.

Low-tension: Muscular tension of the nonpreferred hand introduces a new phrase, and the phrase is maintined by holding the pen in proximity of the screen. That is, the user clicks (presses and releases) the nonpreferred-hand button to start the phrase, keeps the pen on or near the screen to maintain the phrase, and finally pulls the pen away to end the phrase. We found that it was necessary to ignore brief unintentional gaps where the pen goes out-of-range before returning to proximity; we used a 600ms time-out to bridge these gaps (Fig. 4). We explored this technique for two reasons: (1) it was unclear if users were willing to hold down the button for extended periods; (2) if proximity of the pen could be used to phrase strokes together, this would permit an initial click of either a graphical button or the pen barrel button as alternatives to a nonpreferred-hand button (which may not always be available). For our experiment, however, to make a rigorous comparison between techniques, we evaluated a nonpreferred hand mechanism for all three phrasing techniques.

Button vs. Touchpad: Our phrasing techniques can be implemented with a physical button, or with a touch-sensitive surface such as a touchpad. A touch-sensitive surface may offer advantages since it could easily provide a large area or long touch-sensitive strip for mode switching, thus reducing the need for a small button at a specific location. It also may require less strain than holding down a button, and potentially could support secondary tasks such as bimanual scrolling [7].

Holding a pen's barrel button might offer another way to phrase strokes together, but previous work has demonstrated that a nonpreferred-hand button outperforms a barrel button [15]. Also, current tablets cannot sense a press or release of the barrel button while the pen is out-of-range, making the technique impractical.

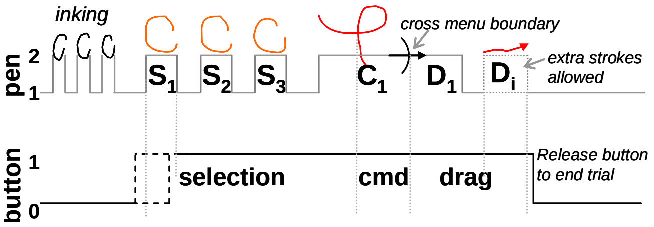

Fig. 2. Full-tension technique timing diagram. The user draws three ink circles, then three selections (S_1, S_2, S_3). The user issues a Move command (C_1) and drags (D_1) the selected objects to a desired location. The user may lift the pen while dragging (D_i). Lifting the button ends the phrase.

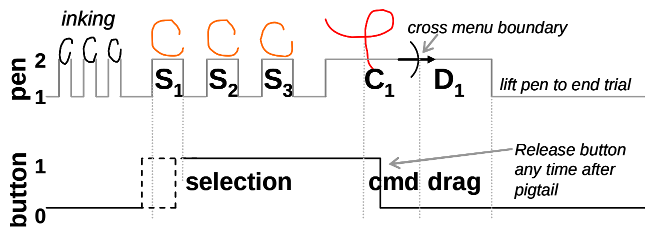

Fig. 3. Half-tension technique timing diagram.

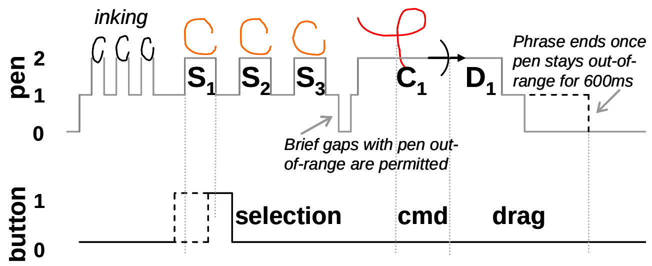
inking
pen
2
1
0
S1
S2
S3
C1
D1
Phrase ends once pen stays out-of-range for 600ms
Brief gaps with pen out-of-range are permitted
button
1
0
selection
cmd
drag

Fig. 4. Low-tension technique timing diagram.

4 Experimental Task

We conducted an experiment to assess the speed and error rates of the full-tension, half-tension, and low-tension phrasing techniques as realized using both a nonpreferred-hand button and a nonpreffered-hand touchpad as the mode switch control (Fig. 5).

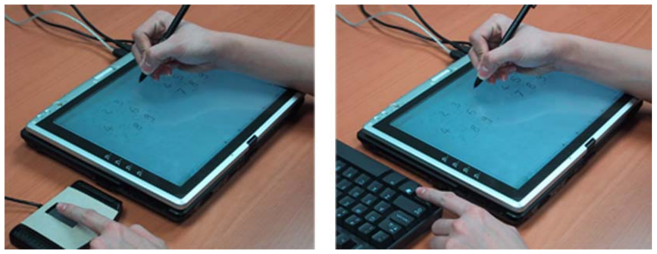

Fig. 5. Set-up. **Left**: touchpad. **Right**: Ctrl key device.

We decided to investigate the effectiveness of the above phrasing techniques for a *Move* command similar to that discussed in the introduction. To assess the phrasing techniques, however, our experimental task employs a *Move* command with multiple direct objects (i.e. multiple disjoint selections). Thus, our task had users switch to gesture mode and form a disjoint selection. Users then chose the verb (the *Move* command) from a 4-way marking menu activated by making a pigtail (a cursive *e*) gesture [9]. Finally, users dragged the pen to indicate the location to *Move* the objects to (i.e. the indirect object), and lifted the pen. Since the nonpreferred-hand device serves as a mode switch between inking and gesturing, our experimental task requires users to enter a mix of ink strokes and gesture strokes. The inking subtask consists of circling objects, rather than free-form writing, to reduce variablity. This task is suitable for experimental study while also being representative of a realistic task.

The task presented two sets numbers arranged in a 3x3 grid, with one set on each side of the screen (Fig. 6). To start, three numbers in the left set were highlighted in blue. The user circled the blue numbers in ink mode; each number turned dark red when it was successfully circled. Three different numbers in the grid then turned orange, and the user switched to gesture mode and selected these numbers using three separate lasso selections with auto-completion [18].

The user's task was to drag these selected numbers to the corresponding empty positions in the right grid (Fig. 6) using the *Move* command. The trials alternated between selecting and dragging the items from left-to-right, and then right-to-left.

The user issued a *Move* command by drawing a pigtail and heading south in the marking menu (Fig. 2). The user dragged the numbers to their final position by extending the marking menu stroke beyond the outer boundary of the menu [8,9,22] and lifting the pen, which ended the trial for the half-tension and low-tension techniques. In the full-tension condition, lifting the button ended the trial, so it was possible for users to lift the pen while dragging; however users rarely employed this capability since the task did not *require* multiple dragging strokes. This is a property of the full-tension technique that should be further explored in future experiments.

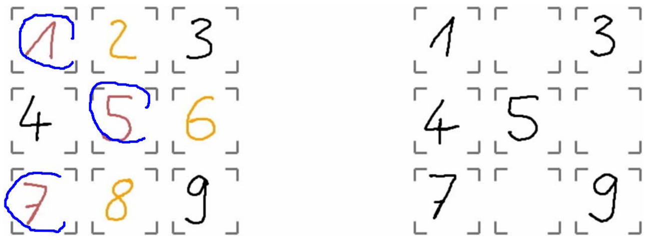

Fig. 6. Task screen: The user has just circled (1,5,7) with ink (left grid). The user then switches to gesture mode, selects the numbers highlighted in orange (2,6,8), and uses the Move command to drag them to the empty slots (at right).

For each trial, the starting 3x3 grid alternated between the left and right sides of the screen. The experiment provided auditory cues [19] when the user initiated gesture mode or selected the *Move* command. Any errors committed by the user produced an error sound. Moving the wrong numbers, moving the numbers to the wrong position, or drawing ink during the selection phase (indicating failure to switch to gesture mode) were all recorded as errors. Selecting the wrong command from the pigtail-activated marking menu also was counted as an error, but users never actually committed this error during any experimental trial.

Exiting gesture mode at the wrong time also was counted as an error. For the low-tension condition, this error occurred if the pen moved out-of-range for longer than 600ms before the *Move* command was issued. For the half-tension condition, this error occurred if the user released the button prior to drawing the pigtail to issue the command. For the full-tension condition, releasing the button too early sometimes could not be distinguished from a move-wrong-position error. Because it was sometimes difficult to determine the type of error, and because a single mistake often led to a cascade of multiple errors within a trial, we treated errors as a pass/fail outcome per trial, rather than summing the total number of errors.

4.1 Mode Switching Devices

For our nonpreferred-hand mode switching button we used the Ctrl key on a keyboard placed next to the Tablet, because our Tablet PC lacked a suitable bezel button. For the touchpad, we used the contact area sensed by a Synaptics touchpad to estimate the pressure exerted on the pad. Our system enters gesture mode when the contact area exceeds a threshold. At the start of the experiment, we ran a brief calibration sequence to set the threshold for each user to provide a good compromise between requiring heavy pressure, versus accidental activation on light touches.

4.2 Participants and Apparatus

We recruited 12 undergraduate and graduate university students as participants in the experiment. Of the 12 participants, 11 were right-handed and 7 were male. Each participant received $20 for their participation in the experiment. None of the participants had previously used a Tablet PC.

Each participant used a Toshiba Protégé Tablet PC (1.5 GHz, 512 MB) at 1024x768 (307mm diagonal) a desk in slate mode. The tablet had an external keyboard and a Synaptics touchpad to serve as the mode switching devices (Fig. 5).

4.3 Experiment Design

We used a 3x2 within-subjects design, so each user saw all conditions. Independent variables were *Phrasing Technique* (low-tension, half-tension, full-tension) and *Device* (touchpad, button). Dependent variables were time to complete the entire gesture phrase, and error rate. Order of presentation of the phrasing techniques was fully counterbalanced. For each of the 6 possible orders, one participant used the Button (Ctrl key) first and another used the Touchpad first. The 12 participants each performed:

3 Phrasing Techniques (Low, Half, Full) x
2 Devices (Button, Touchpad) x
40 practice trials + 30 timed trials
= 420 trials per subject (180 timed trials)
= 5,040 total trials (2880 practice, 2160 timed)

5 Results

We discarded the first two experimental trials to avoid any start-up effects, and used trials 3-30 for analysis. All outliers more than 3 standard deviations from the mean were removed (a total of 1.63% of all trials).

A 3 x 2 within-subject ANOVA on **error rate** for the factors of *Phrasing Technique* and *Device* (Fig. 7) revealed a significant main effect of *Phrasing Technique* ($F_{2,22}$,=4.914, $p<0.05$), but no significant main effect of *Device,* and no significant interactions. Bonferroni-corrected post-hoc comparisons revealed that the full-tension technique had a significantly lower error rate (2.5%, p=.013) than the low-tension technique (5.2%). Half-tension had a 4.3% error rate. The most common errors were drawing while in the select phase, which means that users forgot to invoke the gesture mode and tried to select without it. Moving the selection to the wrong position or moving the wrong selection were other common source of mistakes. None of these errors were significantly more common in any condition.

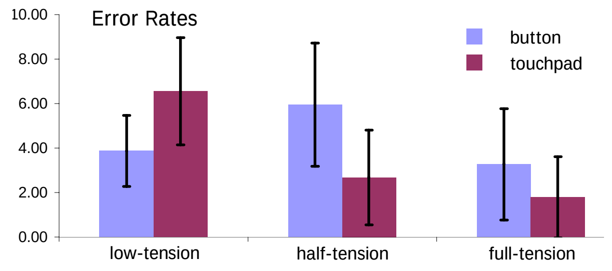

Fig. 7. Error rate for each Phrasing Technique.

A 3 x 2 within-subject ANOVA showed no significant effect on **completion time** for either *Device* or *Phrasing Technique*. Participants averaged approximately 5 seconds per trial for all techniques and devices (Fig. 8). However, we did not expect to observe large time differences because the multi-step task takes several seconds to complete, whereas initiating or terminating a gesture mode requires only a few tenths of a second [15]. In future work, a more complicated experimental design might be able to tease apart mode switching times from other user actions (e.g. [10]), but our focus in this study was to determine which techniques allowed users to effectively articulate multiple-stroke phrases. Thus, for our present experiment, error rate appears to be the more telling performance metric; this has also been the case in previous work on mode controls, e.g. [25].

The *Device* x *Phrasing Technique* interaction was not significant for completion time, but we did observe a significant *Order* x *Phrasing Technique* interaction ($F_{4,63}$=5.542, $p<0.001$). Inspection of the means revealed this is probably a floor effect: the low-tension and half-tension techniques tended to benefit more than the full-tension condition from exposure to previous conditions (Fig. 9). Thus, our experiment may slightly overestimate the average time for full-tension.

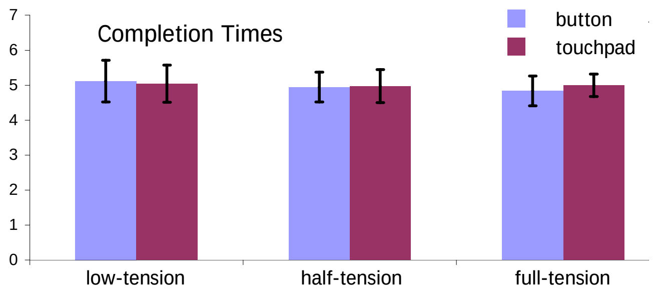

Fig. 8. Completion time for each device and technique.

As shown Fig. 9, while both the low-tension and half-tension conditions exhibit a normal learning pattern, the full-tension condition exhibits a flat learning curve. This suggests the possibility of asymmetrical skill transfer and more precisely that our experimental setting over-estimates the total completion time for the full-tension condition. For error rate, Order X Phrasing Technique was not significant ($F_{4,63}$=.72, p=0.58).

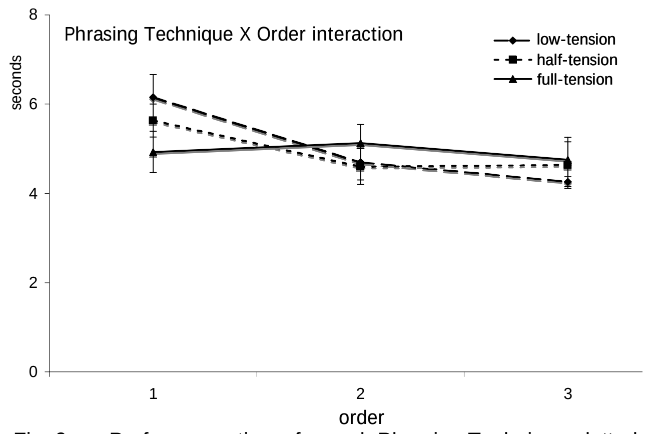

Fig. 9. Performance times for each Phrasing Technique plotted by order (block) of trying each technique.

5.1 Qualitative Results and Full-Tension Iteration

Only 1 of the 12 participants preferred the Touchpad over the Button. The lack of tactile feedback resulted in comments that “the [Ctrl] key feels more reliable.” Several users reported that even though they only had to maintain light contact with the touchpad, it felt more secure to maintain solid pressure on it during the entire phrase.

When asked to rank all three phrasing techniques, users split between the low-tension and full-tension techniques (6 users preferred each). When asked to compare only the full-tension and half-tension settings, 8 users preferred full-tension, 2 preferred half-tension, and 2 claimed no preference. With low-tension, many users felt confined by having to keep the pen in range, and sometimes felt uncertain of the mode. Furthermore, canceling an operation after a mistake incurred a 600ms delay (for the pen-proximity timeout to expire and exit gesture mode).

For the half-tension technique, we instructed participants to remove their finger as soon as they felt it was convenient to do so, yet 6/12 participants tended to keep the button pressed until the end of the phrase, thus revealing a strong tendency to hold tension for an entire chunk [5].

The full-tension technique was very reliable: users exhibited no errors at all in 14 of the 24 sessions. But many users disliked the requirement to coordinate both hands at the end of the task. Based on this feedback, we altered our implementation to be forgiving with respect to whether the pen lifts first or the button lifts first at the end of the phrase. The dotted rising and falling edges for the button (Fig. 10) show how our design iteration relaxes the relative timing of the button press/release events with respect to the corresponding pen-up/pen-down events. In informal testing, this appears to foster the same usage model without requiring hand synchronization at the end of a phrase.

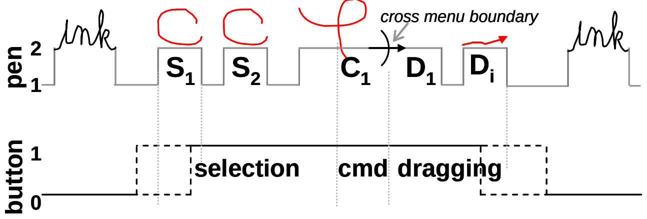

Fig. 10. Final full-tension technique. The dotted rising and falling edges for the button show how relative ordering of pen/button events is flexible in our revised implementation.

6 Multiple-Stroke Selection Techniques

Our study shows that the full-tension technique in particular is a practical and robust way to articulate multiple-stroke gestures. To illustrate the expressive power of multiple-stroke gestures, this section demonstrates how pen gesture commands can benefit from selections specified using multiple strokes.

We implemented a prototype sketching application that supports selection of objects by lassoing, tapping on an object, crossing (drawing a straight line that crosses at least one edge of an object), or drawing a carat that points to the object (Fig. 11). We implemented several commands, including *Align*, *Rotate*, and *Flip*, to illustrate how the spatial properties indicated by these selection operators can be leveraged and composed to create powerful, general-purpose commands.

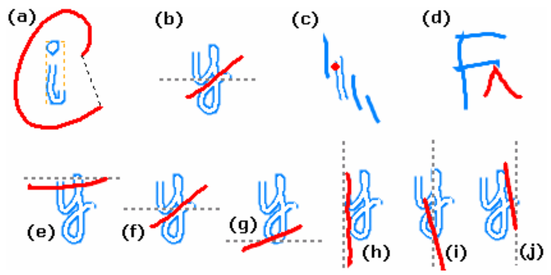

Fig. 11. Selection gestures. (a) lasso (b) crossing (c) tapping (d) carat. Crossing edges: (e) top (f) horizontal center (g) bottom (h) left (i) vertical center (j) right.

As seen in Fig. 12 our prototype implements complex selection features of previous work, namely circles of exclusion [11] and bump and bite gestures [20]. Unlike previous approaches [11,20] the additional strokes are not constrained to start and end within the initial lasso or on the initial lasso stroke, which makes them easier for the user to draw casually. This also makes the techniques easy to implement: any partial or complete lasso toggles the selection bit of the objects it contains; we then add the selected objects to the selection list, and remove any deselected objects. Thus the same code can implement both circles of exclusion and bump/bite.

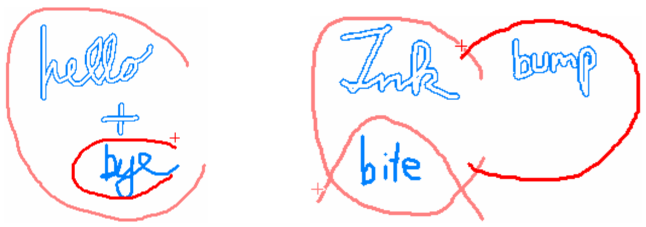

Fig. 12. **Left**: Circle of exclusion; the user lassos *hello+bye* then recircles bye to exclude it from the selection. **Right**: Adding to selection with a bump and removing with a bite.

6.1 Selection Decoration

A selection decoration is a gesture that selects an object *and* indicates a spatial property of the object. Drawing a decoration gesture on or near an object selects some spatial property of the object. As a convenience, if the object is not already selected, it is selected as well. However, a decoration never excludes an object from a selection: adding a decoration to an already selected object does not toggle its selection bit, it just decorates it. This allows multiple decorations to be composed on objects in a selection.

Our prototype currently supports three types of selection decoration:

Crossing. Highlights the closest principle edge of an object with a dotted line, as seen in Fig. 11 e-j.

Carat. Used to specify an optional center of rotation, at the inflection point of the carat, for the *Rotate* command (Fig. 14). Note that we also could have used the tap selection gesture in this way, but since users seem to expect tap to select (or deselect), and nothing else, overloading a tap to also indicate a center of rotation seemed problematic.

Lasso. Instead of using lassos just for circles of exclusion, we have experimented with an implementation that automatically creates a group object, known as an implicit group, when the user lassos multiple objects. This implicit group can then be acted upon by subsequent gestures, but is automatically ungrouped when the phrase ends. Implicit groups are one of the most powerful features in our prototype; we will discuss some uses for implicit groups shortly. Using lassos to form implicit groups is currently an option in our system; we do not support implicit groups and circles of exclusion (Fig. 12, left) at the same time. However, this is a limitation of our implementation, rather than a fundamental conflict in functionality. When the implicit groups option is enabled, currently users can only exclude objects from a selection by tapping on them (which toggles their selection bit).

Decorations add feedback to an object beyond the normal selection feedback. For example, the crossing selection decoration adds a dotted line that identifies the snap edge that is nearest to the crossing stroke (Fig. 11e-j). The carat shows a blue box at the selected point (Fig. 14c). Decorations do not necessarily have to be attached to a specific object. For example, the user can place the carat over white space.

6.2 Example: Multiple Edge Alignment

The user can use the crossing selection operator (Fig. 11e-j) to indicate edges of objects for alignment (Fig. 1). The final edge crossed is used as the reference edge (the edge that is aligned to). If the user misses the intended edge, he can cross the object again to override the previous stroke. This approach offers a nice economy of design, since the system includes just a single general-purpose *Align* command, rather than multiple versions (*Align Left, Align Center, Align Middle*, etc.) as seen in Microsoft PowerPoint, for example. Even with all these commands, some operations are still difficult to express with traditional approaches (such as aligning the left edge of one object to the right edge of another, Fig. 13). During informal tests, users commented that using our crossing selection operators with a single general purpose *Align* command was natural and seemed obvious.

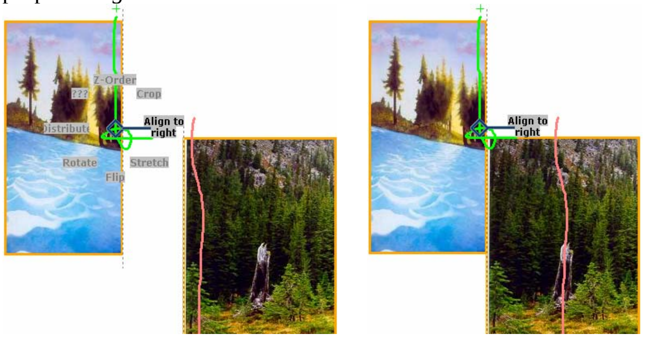

Fig. 13. Aligning opposite edges. **Left**: The user crosses the left edge of one photo, then indicates the right edge of another photo to align to. **Right**: Result after the alignment.

6.3 Example: Rotation Using the Carat

The carat selection operator selects a point on an object or on empty whitespace (Fig. 11d, Fig. 14a-e). The *Rotate* command uses the inflection point of the carat as a center of rotation. The user first draws the carat, then draws a pigtail to activate a marking menu, and then chooses the *Rotate* command. Dragging past the outer boundary of the menu then rotates the object around the center of rotation via direct manipulation. We actually implemented two different versions of our prototype sketching application to more quickly explore various design options; only one version (with a different menu visualization) currently supports the *Rotate* command, and that version is shown in Fig. 14a-e.

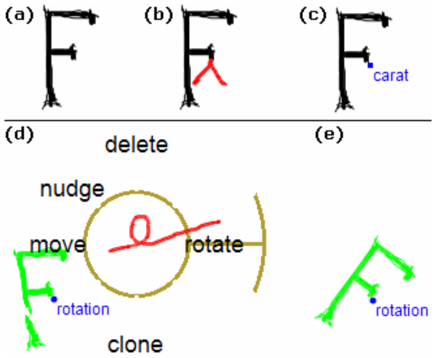

Fig. 14. Rotation using the carat. (a) before, (b) draw carat (c) feedback for selection decoration (d) choose rotate command (e) dragging with the carat as center-of-rotation.

6.4 Example: The Flip Command

The *Flip* command in our prototype is an example of a general purpose command that can span the full range from simple to sophisticated operations. The most elementary way to use *Flip* is to cross a single object, and then pigtail to choose *Flip* from the marking menu; in fact this can be done as a unistroke gesture if desired (Fig. 15). The *Flip* command uses the crossing axis, here the horizontal center of the handwritten word “ink,” as the axis of rotation; the user can draw a vertical cross to instead flip about a vertical axis.

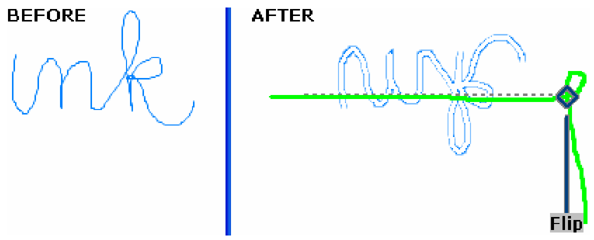

Fig. 15. Simple Flip command on a single object.

The user can also flip several objects in one operation by individually crossing each object along the desired axis of rotation (Fig. 16). Here, the user crosses the horizontal center of each object, and then chooses Flip to rotate each object about its own local axis of rotation.

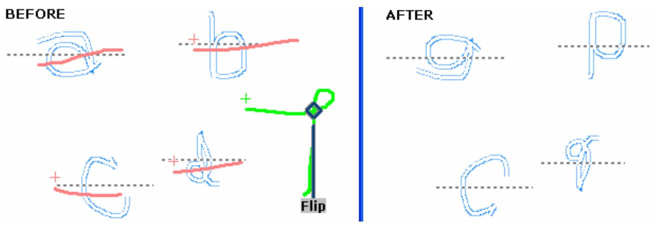

Fig. 16. Flipping objects about their local horizontal center.

But what if the user instead wanted to preserve the relative spatial relationship between the objects (*a, b, c, d*) and instead flip the entire group around one of the objects? Our implicit grouping feature can be used to specify these types of operations.

6.5 Complex Operations via Implicit Grouping

In pen interfaces, drawing a lasso often allows commands to act on objects as a unit [6,11,20]. We take this concept to its logical conclusion by using the lasso as a way to specify an implicit group. With implicit groups, when the user draws a lasso, this acts exactly as if the user had selected the objects and applied the *Group* command. However, our prototype flags implicit groups internally, so that when they become deselected, the *Ungroup* command is automatically applied. Thus the “group” only exists during the articulation of the command, and the user does not have to explicitly invoke the *Group* and *Ungroup* commands. We provide feedback of the implicit group by drawing a dotted orange rectangle around the lassoed objects (Fig. 17).

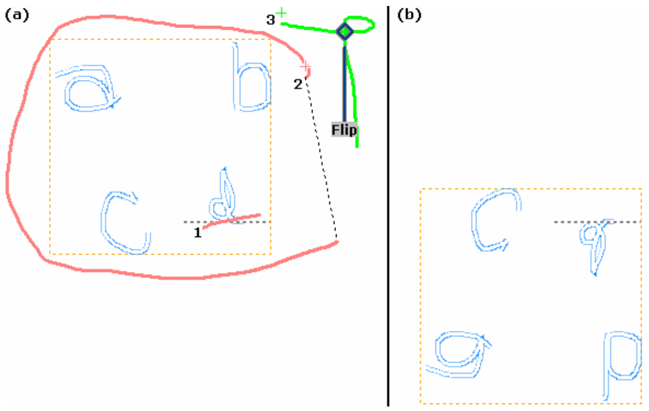

Fig. 17. Flip a, b, c, and d around bottom of d. First the user crosses the bottom of the “d.” Then user lassos a, b, c, and d. The user then applies the Flip command.

For most commands, such as lassoing objects and moving them as tested in our experiment, the creation of the implicit group is not important and need not concern the user. However, knowledgeable users can specify more sophisticated operations by using selection decorations and implicit groups together.

Fig. 17 shows the *Flip* command with an implicit group. Here, we return to the example raised above, where a user wants to Flip the objects (*a*,*b*,*c*,*d*) around the *d*. To do this, the user just crosses the *d*, then lassos all four objects, and then pigtails to choose Flip. This causes the *Flip* command to rotate the implicit group about the bottom of *d*. Once the command is finished, the selection is automatically cleared, which results in our software ungrouping the implicit group.

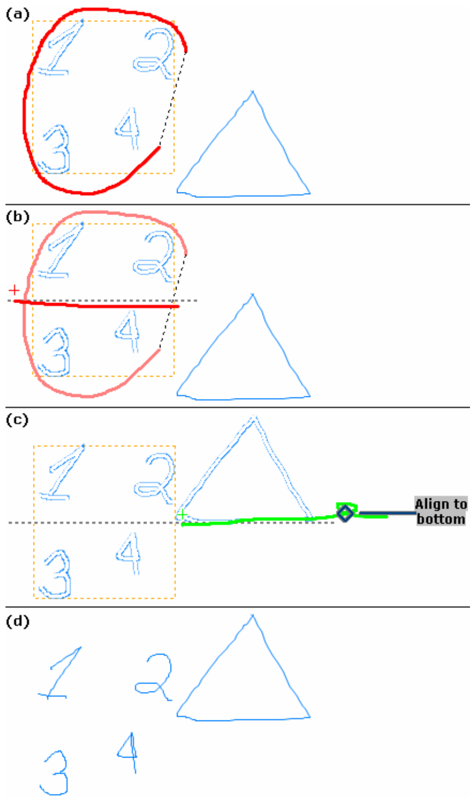

Fig. 18. Alignment to center of implicit group. (a) Lasso items to form implicit group. (b) Cross center of implicit group. (c) The center of the implicit group is aligned with the bottom of the triangle. (d) Final result at completion of command; the implicit group is automatically unpacked.

The user can apply selection decorations to an implicit group, allowing composition of functionality. For example, the user can use this capability to align the center of a group of objects to another object without any explicit use of the "Group" command (Fig. 18). As another example, to instead flip (*a*,*b*,*c*,*d*) around their collective right edge, the user would first lasso the objects (thus creating the implict group), then cross the right edge of the implicit group, and then choose *Flip*. This pivots the objects about the rightmost edge. Implicit groups can also be nested by drawing a lasso that encompasses one or more previous lassos. Both of these interactions are illustrated in the accompanying video. Nested implicit groups obviously are a feature for expert users. Nonetheless, it seems compelling that a basic lasso selection operation in a pen interface, which test users immediately understood, can be used in more sophisticated ways by experienced users.

In practice, we would not expect a novice user to immediately grasp the full complexity that can be expressed with our selection operators. Yet these examples show how composing multiple selection operators allows experienced users to specify complex operations without having to repeatedly *Group*, act on, and then *Ungroup* the objects. Because the commands in our prototype have such general functionality, the command set as experienced by the user can remain simple (*Flip*, *Align*, etc.). The spatial properties needed to give specific meaning to a command come from the multiple strokes that together specify the selection.

6.6 Selection Recall

During early testing, users would become frustrated if they drew one or more selection strokes, activated a command, but then made a mistake or wanted to re-use the selection for another command. It was frustrating for users to have to redraw the same selection again. To our knowledge, existing graphical interfaces do not include mechanisms to undo or redo selections, but it appears that one consequence of making selections more expressive in our system is that it may become necessary to include a feature to "recall" a prior selection.

A naive full-tension design precludes the repeated invocation of different commands that act upon on the same selection. To address this problem, we allow the user to tap twice on the mode switch button to recall the prior selection. The user then holds down the button, and can modify the prior selection (if desired) before proceeding with a new command as usual. This works, but users have to be told to tap the gesture key twice to invoke this feature. Users suggested a "recall selection" icon or a double tap of the pen as alternatives, but we have not yet tried these.

7 Conclusions and Future Work

We have conducted the first experimental study of interaction techniques for phrasing of multiple-stroke gestures. We have demonstrated how our prototype uses multiple-stroke phrasing techniques to elegantly support selection techniques explored by previous systems, including disjoint selection regions, circles of exclusion, and bump/bite gestures. But our approach also enables new techniques, such as selection decorations and implicit groups, which allow for sophisticated functionality from a few general-purpose commands. However, while we have conducted some informal user tests, these advanced features have not yet been subjected to careful experimental studies.

Our experimental results are consistent with the conclusion that the full-tension phrasing technique provides the strongest reinforcement of the chunking of multiple-stroke selection, command, and parameter phrases that our designs are trying to establish. However, the low- and half-tension techniques worked well enough that they could be considered as alternative multiple-stroke phrasing techniques. For example, our full-tension technique might not be practical for an electronic whiteboard that lacks any button for the nonpreferred hand, or in some cases the specific details of an application's interaction techniques might be incompatible with the full-tension technique.

We have described several benefits and extensions to existing techniques that follow from supporting a multiple-stroke syntax for pen gesture interactions:

- The ability to select objects using multiple strokes allows additional functionality, including more precise control over multiple object selections, easier editing of selections, and nesting of multiple operations within a selection. The resulting polymorphic commands also offer a nice economy of design, because the system can offer a single gesture command that acts on a wide variety of selections (e.g. Align to right, Align to bottom, Align to horizontal center, etc).

- The use of nonpreferred-hand physical tension as a phrasing technique for multiple-stroke interactive dialogs offers a concrete realization of ideas in the literature [5,10,12,15,24] that can be applied to realize new gesture-based interaction techniques such as selection decorations.
- Our implicit group selection mechanism can streamline some operations by removing the necessity to explicitly group and ungroup objects, potentially improving the cognitive and physical workflow of such operations. This would be an interesting issue to explore further in future studies.
- Our multiple-stroke approach to selection extends or subsumes a number of individual techniques (bump, bite, crossing, lasso selection, tapping) that have appeared in long history of gesture-driven systems [1,2,4,6,9,11,18,21], but now can be offered in a single integrated technique that offers these as variants.
- Our experimental validation confirms that multiple-stroke phrases can be supported without introducing any ambiguity of which strokes "belong" to a command. In particular, the 2.5% error rate exhibited by the full-tension phrasing technique suggests that it can offer particularly robust performance. However, a more sophisticated experimental design would be required to determine if there are any subtle differences between the techniques in terms of mode switching times [10,15]. Future studies of phrasing techniques should also investigate alternative sequences of commands [3,10,17]. For example, if the user must issue three successive commands, does the need to "remember" to release the button between commands pose any problem for the full-tension technique?

Many of these capabilities would be difficult or impossible to support within a unistroke command syntax. Thus, holding down a button to explicitly signal a mode transition between inking and gesture input not only offers a fast and reliable means to switch modes [15], it also offers an effective way to phrase together multiple-stroke pen gesture commands. We hope that our discussion of the multi-stroke design strategy will help pen gesture interaction designers to think about and explore new ideas in the design space of multi-stroke gestural techniques, without the limitations and ambiguity that multi-stroke gestures have raised in previous systems.

ACKNOWLEDGEMENTS

This work has been supported in part by the Microsoft Center for Interaction Design and Visualization at the University of Maryland and NSF under grant IIS-0414699. We also would like to thank Corinna Löckenhoff for assistance with the data analysis.

REFERERNCES

[1] Accot, J., Zhai, S. *More than dotting the i's-- Foundations for crossing-based interfaces*. ACM CHI 2002, 73-80.

[2] Apitz, G., Guimbretiere, F. *CrossY: A crossing based drawing application*. UIST 2004, 3-12.

[3] Appert, C., Beaudouin-Lafon, M., Mackay, W. *Context matters: Evaluating interaction techniques with the CIS model*. Proc. HCI 2004, Springer Verlag, 279-295.

[4] Buxton, W. *An Informal Study of Selection-Positioning Tasks*. Proceedings of Graphics Interface '82: 8th Conference of the Canadian Man-Computer Communications Society, 323-328.

[5] Buxton, W. *Chunking and Phrasing and the Design of Human-Computer Dialogues*. Information Processing `86, Proc. IFIP 10th World Computer Congress, Amsterdam: North Holland, 475-480.

[6] Buxton, W., Fiume, E., Hill, R., Lee, A., Woo, C. *Continuous hand-gesture driven input*. Proceedings of Graphics Interface '83, 191-195.

[7] Buxton, W., Myers, B. *A Study in Two-Handed Input*. Proceedings of ACM CHI'86:, 321-326.

[8] Guimbretiere, F., Winograd, T. *FlowMenu: Combining Command, Text, and Data Entry*. ACM UIST 2000, 213-216.

[9] Hinckley, K., Baudisch, P., Ramos, G., Guimbretiere, F. *Design and Analysis of Delimiters for Selection-Action Pen Gesture Phrases in Scriboli*. CHI 2005, 451-460.

[10] Hinckley, K., Guimbretiere, F., Baudisch, P., Sarin, R., Agrawala, M., Cutrell, E. *The Springboard: Multiple Modes in One Spring-Loaded Control*. CHI 2006, to appear.

[11] Kurtenbach, G., Buxton, W. *Issues in Combining Marking and Direct Manipulation Techniques*. UIST'91, 137-144.

[12] Kurtenbach, G., Fitzmaurice, G., Owen, R., Baudel, T. *The Hotbox: Efficient Access to a Large Number of Menu-items*. CHI'99, 231-237.

[13] 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).

[14] LaViola, J. a. Z., R., *MathPad2: A System for the Creation and Exploration of Mathematical Sketches.* ACM Transactions on Graphics, 2004. **23**(3): p. 432-440.

[15] Li, Y., Hinckley, K., Guan, Z., Landay, J. A. *Experimental Analysis of Mode Switching Techniques in Pen-based User Interfaces*. CHI 2005, 461-470.

[16] Liao, C., Guimbretiere, F., Hinckley, K. *PapierCraft: A Command System for Interactive Paper*. UIST 2005, 241-244.

[17] Mackay, W. E. *Which Interaction Technique Works When? Floating Palettes, Marking Menus and Toolglasses Support Different Task Strategies*. Proc. AVI 2002 International Conference on Advanced Visual Interfaces, ACM, 203-208.

[18] Mizobuchi, S., Yasumura, M. *Tapping vs. Circling Selections on Pen-based Devices: Evidence for Different Performance-Shaping Factors*. CHI 2004, 607-614.

[19] Monk, A., Mode Errors: *A User-centered Analysis and some Preventative Measures Using Keying-contingent Sound.* International J. Man-Machine Studies, 1986. **24**: p. 313-327.

[20] Moran, T., Chiu, P., van Melle, W. *Pen-Based Interaction Techniques for Organizing Material on an Electronic Whiteboard*. UIST'97, 45-54.

[21] Pier, K., Landay, J. A., *Issues for Location-Independent Interfaces*. 1992: Technical Report ISTL92-4, Xerox Palo Alto Research Center.

[22] Pook, S., Lecolinet, E., Vaysseix, G., Barillot, E. *Control Menus: Execution and Control in a Single Interactor*. CHI 2000 Extended Abstracts, 263-264.

[23] Saund, E., Lank, E. *Stylus Input and Editing Without Prior Selection of Mode*. UIST'03, 213-216.

[24] Sellen, A., Kurtenbach, G., Buxton, W. *The Role of Visual and Kinesthetic Feedback in the Prevention of Mode Errors*. Proc. IFIP INTERACT'90: Human-Computer Interaction, 667-673.

[25] Sellen, A., Kurtenbach, G., Buxton, W., *The Prevention of Mode Errors through Sensory Feedback*. J. Human Computer Interaction, 1992. **7**(2): p. 141-164.

[26] Zhao, S., Balakrishnan, R. *Simple vs. Compound Mark Hierarchical Marking Menus*. UIST 2004, 33-42.