Fluid Inking: Using Punctuation to Allow Modeless Combination of Marking and Gesturing

Robert Zeleznik, Timothy Miller, Loring Holden,
Joseph J. LaViola, Jr.

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-04-11
July 2004

Fluid Inking: Using Punctuation to Allow Modeless Combination of Marking and Gesturing

Robert Zeleznik, Timothy Miller, Loring Holden, Joseph J. LaViola, Jr.
Computer Science Department
Box 1910
Brown University
Providence, RI 02912
{bcz,tsm,lsh,jjl}@cs.brown.edu

ABSTRACT

We present a set of principles, called Fluid Inking, for modelessly and fluidly combining gestures with free-form note-taking. The hallmark of these principles is that a gesture is a short sequence of drawn ink marks and terminal punctuation that is collectively and contextually distinct from regular notes even though individual marks and punctuation may be identical to conventional notational idioms, such as scribbling and tapping. Thus, after punctuation is drawn, the preceding ink strokes are compared to gesture templates and if they match they are interpreted as a gestural action. This approach allows gestures to be easy to perform, simple and memorable without restricting the domain of regular notes despite the modeless nature of gesturing and drawing. Further, by omitting earlier punctuation lower-level gestures can be cognitively chunked into higher-level gestures. To illustrate and evaluate our principles, we designed and implemented a full gesture set that handles the core functionality of popular note-taking applications, in addition to making available a special selection mode to support complex disjoint selections. We discuss informal user feedback about our full gesture set and a user study designed to gauge relative user preference for punctuated gestures.

KEYWORDS: gestures, modeless, terminal punctuation, tablet computing

1 INTRODUCTION

The theoretical potential of pen-based computers stems from the notion that stylus-based interactions can be more closely tailored, in many cases, to human capabilities than their computationally equivalent (or even more powerful) mouse-based counterparts. Informally, when human and computing abilities are closely matched the resulting interfaces feel fluid—users can focus on their problem and not on extrinsic user interface activities. To explore this theory, we considered the prevalent activity of free-form note-taking. Such note-taking is important because of its general applicability and its dependence on rapid, natural notational entry and manipulation. Thus the fundamental research question that this paper studies is "how can gestural commands be designed to blend fluidly into the note-taking workflow?"

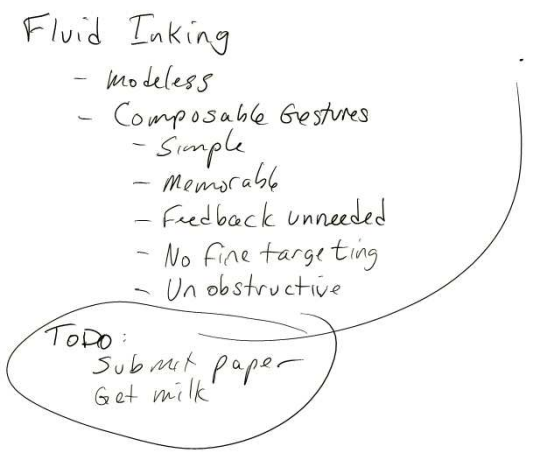

Figure 1: Moving ink strokes may be done by circling them, drawing a stroke to where they should go, and tapping, without the use of any modes.

We approached this question by first identifying the general tasks required in a note-taking application based on an analysis of existing note-taking software, such as Microsoft Journal, Microsoft OneNote, and MathPad^2 [8]. In addition, we noted a set of general characteristics of traditional paper and pencil media interaction to respect and leverage such as pen grasp, prevalence of existing idioms, lack of input modes (except for an eraser), tolerance of targeting errors, and grammatical use of terminal punctuation. Although individually many of these characteristics are honored in current pen-based systems, to our knowledge, there is no note-taking system that leverages all of these characteristics to support a fluid, non-disruptive interface for the range of interactions including: drawing, deleting, moving, resizing, copying, pasting, zooming, inserting space and undoing. Standard GUIs

are inappropriate because of their reliance on targeting buttons to change modes or perform an operation. We also dismissed for our purposes the use of stylus-mounted buttons for performing gestural actions since the buttons are not intrinsically necessary and they raise numerous problems ranging from increased tension during interaction, to restrictions on pen grasp, to accidental button triggering during writing.

In this paper, we discuss our alternative Fluid Inking gestural design approach that is characterized by the modeless co-existence of gestural input with note-taking inking activities—that is, all gestural actions are inferred from punctuated patterns of ink notations that are otherwise indistinguishable from notes. We also discuss informal user experiences as well as the results of a pilot user study concerning user response to representative fundamental design choices.

2 RELATED WORK

We presented the preliminary notions of a punctuated gesture interface as part of our MathPad^2 system [8]. MathPad^2's gestures focused on domain-specific operations and thus omitted much of the functionality of our system. We are currently redesigning MathPad^2's gestures to adhere to Fluid Inking principles to make it more powerful and extensible.

A popular approach for incorporating gestural input is to use radial menus and their derivatives, such as marking menus [6] and FlowMenus [3]. Although this regularization can aid in learnability, it comes at the cost of making many interactions less natural or fluid than would specialized gestural counterparts. Thus our approach is to combine specialized gestures for common actions with FlowMenu interaction for infrequent or less important operations.

Moran, et al [11] presented a suite of gesture-based interaction techniques targeting collaborative electronic whiteboard interaction. Many of their techniques and principles are closely related in spirit to ours, especially their *openness principle* which does not tolerate restrictions on the kinds of ink marks that can be drawn, for instance by pre-defining certain types of ink marks as gestures. However, their approach of using pen modes to satisfy this principle fundamentally differs from ours and, as they report, comes at the "cost of users having to be vigilant." Our use of punctuated gestures does not in practice appear to violate the openness principle and requires significantly less vigilance. We also alleviate problems they noted with the learnability of a large gesture set by using FlowMenu-based gestures for some actions.

Alvarado, et al [1] presented a modeless gestural user interface as part of their ASSIST system for creating and understanding mechanical sketches. Their interface does not require punctuation because their domain-specific drawing restrictions (i.e., mechanical objects) allow gestures to co-exist with inking. Yet even with such restrictions, they require user assistance to override interpretation errors.

Many other systems combine gestural interpretation with free-form annotation, such as Landay's SILK [7] and Mynatt's Flatland [12]; however, they use explicit prior actions (e.g., button press) to disambiguate gestures from notes.

Saund and Lank [13], however, present a technique for avoiding prior selection of mode when editing stylus input, that is closely related to the general notion of suggestive interfaces developed by Igarashi [4]. However, their technique only avoided prior mode selection for forming the selection, as compared with our more general approach that allows the treatment of an entire simple operation as a single chunk, with no mode selection at any time. In addition, their technique either requires fine-grained targeting of a "select" button, or interferes with drawing if that button is made large.

Mankoff, et al [9] consider general techniques for resolving ambiguity in recognition-based interfaces. However, our desire for modeless drawing and gesturing guarantees that theoretically all gestures will be ambiguous with possible notations. Thus, an explicit mediation-based resolution system would not be appropriate because it would necessarily disturb interaction flow. Instead, our work focuses on the complementary problem of how to design gestures that will not, in practice, require ambiguity resolution.

Buxton [2] noted the value of making a gesture match the mental model of its action to unify a command and arguments in a single "chunk" that avoids errors of syntax. Although this has been one of many important criteria in our design, we believe strict adherence to his uni-stroke "phrasing" notions paradoxically can lead to under-utilization of existing skills (e.g., drawing pairs of crop marks) and overly complex interaction (e.g., difficulty specifying the reference point of a select and move operation). Uni-stroke interaction, as presented by Kurtenbach [5], does not directly assist in eliminating modes when coupled with free-form notations, and is harder to robustly interpret than multi-stroke gestures which are naturally segmented.

3 GESTURE PRINCIPLES

To investigate our goal of fluid gestural interactions during note-taking on a tablet computer, we developed a set of gestural principles from a series of observations about traditional paper and pencil note-taking activities. Specifically, we noted the prevalence of:

- existing idioms to "delete", "select" and "move" notes.
- tolerance for poor targeting (e.g., dotting an 'i').
- uni-mode interaction (e.g., X'ing-out instead of erasing).
- no feedback other than pencil marks themselves.
- multi-part terminally punctuated notations (e.g, sentences).

We distilled these collective observations into a set of principles for winnowing from the myriad possible pen-based user interface alternatives a scalable design for a broad range of gestures. These Fluid Inking principles include:

- **Exploit common paper-and-pencil idioms** closely to insure ease of performance, memorability and expressivity.
- **Avoid precise targeting** since it disrupts interaction unless it is an intrinsic task requirement.
- **Support chunked actions** by allowing gestures to be easily combined into higher-level gestures.
- **Suppress intermediate feedback** for all but novice users to prevent distraction during non-interactive operations.
- **Use terminal punctuation** because it is more fluid than prior mode selection.

These principles lead to gesture sets which have similar characteristics to traditional note-taking, while at the same time introducing no exceptional activities and thus requiring no explicit mode changes from regular drawing. Each gesture is a short sequence of drawn ink marks that rarely or never occurs during actual note-taking even though individual marks may be quite common. In general, gesture sequences can be disambiguated from intended notations by making the sequences long or unusual; however, it is desirable and frequently possible to adapt a familiar notation sequence into a semantically related gesture merely by appending to the sequence very simple punctuation, such as a tap. This notational adaptation enables users to draw without disruption what they would have intuitively drawn to *indicate* that action and then draw terminal punctuation to actually *perform* the action. Furthermore, regular use of terminal punctuation in a gesture set allows for a simple, natural approach to conjoining gestures such that two gestures are drawn but only the second gesture is punctuated.

There are a range of theoretical benefits to punctuated gestures. First, by basing gestures on notations that are already common for indicating an action, gestures are likely to be easy to perform and memorable—in fact, distinguishing gestures from note-taking becomes the fundamental challenge. Second, since the gesture set does not incorporate any artificial targets, such as the fixed-size buttons common to standard GUIs and some gestural interfaces [13], targeting tolerance is essentially dictated by the user and how they choose to draw their gestures. Third, a regular gestural composition strategy results in chunked [2] gestures that represent commands at the higher-level expected by expert users, thus making gesture sets for broad functionality more tractable—more memorable, easier to perform and possibly more sophisticated since the combined context of a series of gestures is available for interpretation. Fourth, since no feedback is presented during a composite gesture, the user will not be distracted by the appearance of an interface widget or highlight. Last, since punctuation is drawn only at the terminus of a gesture, there is no need to be aware of modes and there is effectively no disruptive delay between deciding to perform an operation and specifying the intrinsic details of the operation. We performed a simple user study, reported on in Section 5, that is consistent with these theoretical benefits.

To more completely explore our Fluid Inking principles, we applied them to the design of a gesture set for note-taking (described in detail in Section 4). Note-taking commands frequently involve the specification of a context (e.g., a region or set of objects) and a parameterized action (e.g., move to a location). Formally, different gestures are used for each of these specifications. A context selection gesture defines a context and triggers a special local selection mode in which gestures are available for making complicated disjoint selections or gesturally performing actions on the selection. However, for the common case of simple selections, this mode can be completely elided by only punctuating the action gesture and thus composing context selection and action into a cognitively chunked composite gesture. In addition, regular use of different types of terminal punctuation (e.g., tap vs. pause) can parameterize an action, such as to distinguish between interactive (e.g., start drag) and non-interactive (e.g., move to) variants.

	Gesture		Effect
Context	Action	Tap Location*	
		Anywhere for lasso; else in crop region	Select
		Anywhere	Zoom
or		Anywhere	FlowMenu†
+		Anywhere	Move
or		Anywhere	Delete
		Inside ⊃	Local undo
		In context	Resize‡
	· pause	Not punctuated	FlowMenu
		Anywhere	Unzoom
		Anywhere	Insert space
Stand-alone		Anywhere	Delete
		In crop mark	Paste‡
		Inside ⊃	Global undo
	·	On tap	Select object

*All actions are terminated with punctuation (tap or pause) unless noted. †The hooked line ⌉ may be drawn in any direction. ‡The single ⌜ crop marks may be drawn in any of the four orientations: ⌜ ⌞ ⌝ ⌟ to specify a different corner.

Table 1: Illustrative punctuated gestures.

Despite our general principles, we recognize that sometimes no existing existing idiomatic expression is suitable for a particular action. In these cases, it may be appropriate to create a new gestural primitive that is unique during regular note-taking. However, such primitives must be treated with great care to insure that they can be remembered and performed. Our set of gestural primitives has been chosen so that all but two are taken from existing note-taking practice. One novel primitive, "strokehook", used for FlowMenus [3] as described later, models the initial gesture done in a FlowMenu and thus becomes mnemonic similarly to a marking menu [6]. The other, "⊃", used for undo, may have taken as much time to settle on as the rest of the primitives combined, and it's still not clear it's the best possible choice. A discussion of the history of this gesture is included in Section 6.

4 GESTURE SET

4.1 Overview

We have designed a gesture set to illustrate and check our principles. The gestures that perform actual commands are illustrated briefly in Table 1. Our gestures have two or three parts. Every gesture has an action and punctuation, but some actions require a context to be specified first.

The context, when required, is either two crop marks or a lasso. The crop marks indicate a rectangular region, while the lasso indicates a region of more arbitrary shape. Both of these correspond to similar conventional uses. Feedback is specifically not provided when the context is drawn because we do not wish to distract the user when drawing similar strokes in normal notes, and because the intent is that these gestures are drawn as a single chunked operation where the user does not have time to respond to any feedback. Feedback could be provided for novice users, although we are not

Gesture*	Action	Selection bounds:
⌓ *or* ⌜ ⌟	Add	Overlaps.
⌓ *or* ⌜ ⌟ + ·	Add	Contains tap.
· ·	Add	Contains only 2nd tap.
⌞ ⌞	Resizing add	Contains only 1st crop.
⩘	Deselect	Contains start.
·	Toggle	Contains tap.

Table 2: Illustration of selection refinement. *The double crop marks may be drawn either as shown or in the other orientation: ⌞ ⌝; the ⌞ ⌞ crop marks may be drawn in any of the four orientations: ⌜ ⌞ ⌝ ⌟, so long as the orientations match.

sure how helpful this would be.

Some gestures require context. In general, the actions in those gestures must start within the context, except for delete which must contain the context, and resize which may be drawn anywhere. The action may interpret the context to refer to the region itself, or it may operate on those objects contained in the region. There are also "standalone" gestures that generally don't require context, as shown in Table 1. The exceptions are the delete and insert space gestures, which get their context from the shape of the gestures themselves.

The punctuation is either a single tap of the stylus, or pausing with the stylus held down at the end of the previous primitive. Interpretation of the difference in punctuation is completely uniform: tap indicates a non-interactive command, while pause indicates an interactive command. Pause punctuation always indicates the interactive version of the same command that tapping would have indicated. We chose tapping because it is simple, fast, and intuitively corresponds to periods in sentences. We discuss the implications of this choice later in Section 6. Tapping punctuation also provides disambiguation in some cases such as resize vs paste.

Because it is difficult to describe a complex context for an action, we also support a selection mode for specifying such contexts for an action. This mode draws a bounding box around the objects in the selection. Within that selection mode, all the action parts of the gestures in the table still apply, but generally don't require tapping for their noninteractive version. This mode supports a set of operations for refining the selection, as shown in Table 2.

To add to the selection, the user uses essentially the subgestures used to specify context, above. The user may also toggle the selection state of objects inside the bounding box by tapping them, and deselect objects by scribbling over them. An additional gesture may be used to grow the bounding box, selecting the newly included objects.

4.2 Formal Description

The feel and usability of gestural user interfaces depends on a broad range of subtle detail. Even small differences between gestures can have significant impact not only on that gesture itself but on impressions of the overall system. Therefore, we include as much specific detail in this section as possible for the purpose of analysis and reproducibility, not for end-user reference.

$\mathrm{strokehook\ tap}$	FlowMenu.
$\mathrm{tap}_{\mathrm{inside}}\ \mathrm{pause}$	FlowMenu.
$\mathrm{stroke\ tap}$	Move.
$\mathrm{crop\ tap}_{\mathrm{inside\ context}}$	Scale contents.
"⊃"$_{\mathrm{inside}}\ \mathrm{tap}_{\mathrm{inside}\ \supset}$	Local undo.
$\mathrm{Z}_{\mathrm{inside}}\ \mathrm{tap}$	Zoom in.
$\mathrm{scribble}_{\mathrm{over\ whole}}\ \mathrm{tap}$	Delete.

Table 3: Actions using context.

4.2.1 *Primitives*

Our gesture set currently has eight basic primitives: stroke (⌒), strokehook (ſ), lasso (⌓), tap (·), crop (⌝), scribble (⩘), "⊃" (⊃), and Z (⩙). Every input stroke is recognized as a "stroke" unless it matches one of the other primitives. A strokehook is a stroke which partly doubles back on itself at the end. A lasso is a closed or nearly closed stroke of sufficient size. A tap is a quick tap of the stylus. A crop is any of the four axis-aligned crop marks ("⌜", "⌝", "⌞", or "⌟"). A scribble is a "scratch-out" gesture: a back and forth scribbling action that should match as many different styles of scratching out as possible. The "⊃" primitive must be drawn from bottom to top. In addition, the pause, as punctuation, simply means to pause in place with the stylus down, at the end of the previous gesture primitive. When describing a gesture which has constraints on the location of its primitives, these are notated with subscripts on the primitive names. Thus, a tap which must be inside the previously specified context would be denoted by "$\mathrm{tap}_{\mathrm{inside\ context}}$".

4.2.2 *Context Specification*

As mentioned above, context for an action may either be a rectangular region, specified with two oppositely-oriented crop marks, or a region of more arbitrary shape, specified with a lasso. These subgestures need to be combined with an action subgesture to create a full gesture. To prevent conflicts with note-taking, a full gesture will not be recognized unless the action either does not operate on objects enclosed by the context (e.g., zoom), or the context encloses at least one object.

4.2.3 *Actions Using Context*

Our system has five operations requiring context that are important enough to have a specialized subgesture: move, resize, local undo, zoom in, and delete. Other operations are accessed through the general FlowMenu [3] gesture. See Table 3. All of these gestures may be punctuated either with a tap (as shown) or a "pause" to invoke respectively non-interactive and interactive variants of the action. The notable exception is "$\mathrm{tap}_{\mathrm{inside}}\ \mathrm{pause}$"—the pause punctuation for this gesture has essentially been merged into the gesture, which exists only in its interactive form. Section 4.2.5 describes the related operation of creating a selection essentially by using a null action.

A "strokehook tap" invokes an entry from the first level of a FlowMenu [3]. In the case of the non-interactive variant, no feedback is displayed except for the stroke itself, using the FlowMenu essentially as a marking menu [6]. The interactive variant will display the menu, centered at the start of the stroke. Tapping and pausing will just display the FlowMenu centered on the tap. We do not allow users to invoke a multilevel FlowMenu selection noninteractively.

The “stroke” in the move gesture may not be degenerate; it must have a minimum length to avoid being recognized as a “tap”. To avoid conflicts with note-taking, the stroke must also intersect the bounds of the context drawn. Thus, attempting to move the contents a small distance interactively with the basic move gesture may disrupt the user by first requiring the contents to be moved further away. To support small distance moves more cleanly, one of the entries in our FlowMenu is “Move”, which activates only when the user pauses at the end of the gesture, allowing precise, jump-free manipulation corresponding to the relative motion of the stylus once the stylus begins to move again. If the FlowMenu had already displayed and was thus interactive, then the contents start to move as soon as the stylus crosses back into the rest area.

The “crop $\text{tap}_{\text{inside context}}$” scale gesture, in our system, scales the objects enclosed by the context so that the bounding box corner corresponding to the orientation of the crop becomes located at the apex of that crop. In the interactive variants of the scale gesture, the stylus may seem to end up at the “wrong” place because the gesture endpoint is not the corner of the crop. We actually view this as a feature because the physical stylus can obscure the screen, and the user can control the tradeoff between lack of obscuration and directness of control by changing the size of the crop mark. Thus, the contents are first scaled to make the indicated corner coincide with the cusp of the crop mark, and interactivity is obtained by scaling the contents to correspond with the relative motion of the stylus after the “pause”.

The local undo action applies only to changes made to objects in the context. Each change made to each object is ordered by time and undone or redone separately. The actual undo gesture is somewhat unusual in our gesture set. Once the non-interactive variant is invoked, the original “⊃” stroke behaves more like a widget: “short” right-to-left strokes starting inside the “⊃” undo further back in history while “short” left-to-right strokes starting inside perform a redo. Taps inside undo act like right-to-left strokes. Any marking outside the “⊃” terminates the gesture and undraws the original “⊃” gesture stroke. As a special exception, the undo gesture may be punctuated with a “short” horizontal stroke from inside the “⊃”; this invokes the appropriate undo or redo, depending on the direction of the stroke, as well as leaving the undo gesture widget up for further undoes and redoes. The interactive version of undo undoes successively further back in history the further to the left the cursor moves, and redoes further forward in history as the cursor moves to the right. This provides a virtual timeline of actions in history that the user can scroll through.

The user may zoom in on a specified region by indicating the region as context and then writing a “Z” inside it. The system will zoom so that the bounding box of the context (*not* that of any objects contained) fills the available space as much as possible. A “pause” may be used to interactively fine-tune the zoom scale.

Scribbling over the context deletes everything contained in it. There are a number of possibilities for what “over the context” could mean; one very simple variant would be that the scribble’s bounding-box must include the context’s bounding box, but bounding boxes in general can be perceptually hard to gauge or physically tedious to specify, especially with the irregular shapes often used for deletion. Instead we choose to look for a scribble that merely starts outside the context and goes inside it at some point. The interactive variant of this gesture first deletes the objects in the context and then deletes any object the cursor touches until the stylus is lifted. One possibility for a future version is to delete with an eraser the same approximate size and shape as the context.

scribble tap	Delete.
crop $\text{tap}_{\text{inside}}$	Paste (see text).
“⊃” $\text{tap}_{\text{inside}}$	Global undo.
stroke $\text{stroke}_{\text{back}}$ $\text{stroke}_{\text{crossing}}$ tap	Insert/remove space.
lasso $\text{Z}_{\text{enclosing lasso}}$ tap	Zoom back.

Table 4: Standalone gestures.

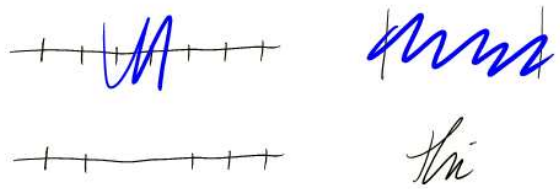

Figure 2: *Narrow delete*: by tapping inside the scribble on the top-left, only the tick-marks contained in the scribble are deleted. *Wide delete*: both lines on the top-right would be deleted because the start and end of the scribble were not empty. The word ’This’ is not deleted even though it looks like a scribble and a tap (dot over the ’i’) since the scribble is mostly empty.

4.2.4 Standalone Gestures Our gesture set also includes standalone gestures that do not require or use any context (Table 4). These gestures must be executed outside any existing selection, and they may all be punctuated with pause to invoke the interactive variant.

Because the scribble gesture for delete is naturally capable of indicating a context implicitly based on the boundary of the scribble, and since deletion can be a very common operation, we provide a standalone delete gesture. See Figure 2. The non-interactive variant of delete itself has two subvariants: if the punctuating tap is outside the scribble, a “wide delete” is used, while if the tap is inside the scribble, a “narrow delete” is used. The narrow delete is less aggressive and is intended to allow the user to delete small things which overlap larger things without deleting the larger things. See Section 4.3 for more details. Since one author reflexively taps on the scribble while another reflexively taps off the scribble, there will probably need to be a user preference for whether narrow delete is recognized this way or possibly through scribbling twice (using the second scribble as punctuation). The interactive variant of the delete gesture first deletes the indicated object(s) (a wide delete, though there’s no particular reason it couldn’t be a narrow one instead) and then deletes any object the cursor touches until the stylus is lifted. A future version of interactive delete may delete with an eraser the same ap-

proximate size and shape as the scribble.

The paste gesture aligns the matching corner of the bounding box of the pasted contents to the crop mark corner. That is, if the crop mark drawn is the upper-left one, "⌜", the upper-left corner of the bounding box will be aligned. The interactive version of the paste gesture allows the user to interactively position the pasted contents.

The global undo command behaves essentially the same way as the local undo command described above, except the global command requires no context. Also, since multiple objects may have been changed simultaneously (e.g., a group move), a single global undo or redo action may modify more than one object. We do nothing special to handle the interaction between local and global undo timelines.

The insert/remove space gesture extends Microsoft OneNote's [10] notion of shifting contents vertically by allowing space to be added and removed in other directions. Furthermore, noticing how difficult it can be to choose straight lines to separate notes that may slant or overlap, we allow the line marking where space should be inserted to be curved. Thus, our gesture begins with drawing an arbitrary, unclosed, non-self-intersecting path to indicate the dividing line. A second stroke must then be drawn to closely follow the original but in the reverse direction. The criteria we use is that the total area between the strokes (connecting the corresponding endpoints with lines) must be "small" compared to the square of the sum of the lengths of the two lines. Also, the start of the second stroke must be closer to the first stroke's end than its start. The user then draws a "straight" stroke crossing the first two strokes to indicate the direction and quantity of space to be added or removed. The endpoints of the first stroke are extended to infinity along the direction of this final stroke to determine the region affected. Currently, whether space is added or removed is determined by whether the direction of this final stroke is drawn closer to the lower-right or the upper-left, respectively. The gesture is punctuated with either a tap or pause, for the non-interactive and interactive variants, respectively.

A lasso, followed by a "Z" whose bounding box encloses the lasso, followed by a tap, causes the system to zoom out to the previous zoom level in the zoom history for this document. The lasso is not used to denote context in this gesture. We currently do not have an interactive version of "zoom back."

4.2.5 Selection We also support a mode for specifying a complex selection that may later be used as the context for an action. The user encloses objects for selection with a context from Section 4.2.2 and punctuates with a tap, without drawing any action. Also, tapping twice over a single object will select it—tap is not supported for context specification because of potential conflicts with handwriting. If a selection exists, performing these gestures completely outside it will switch the selection, but if the gestures overlap the existing selection they are interpreted differently and no punctuation is needed, as described in Section 4.2.6. Punctuating the crop gesture with a "pause" initiates interactive selection in which the last crop mark is dragged to resize the selection rectangle. We have not fully investigated interactive modes for lasso and tap selections. A selection may be canceled by tapping once anywhere where there is no object in which case the tap will disappear, or by writing something which is not interpretable as the start of a gesture involving the selection in which case the writing will remain. In order for any of these selection starting gestures to be recognized at all, they must actually select something, otherwise writing things like "Jack O'Lantern" or "0.4" might not work.

$\text{scribble}_{\text{starts inside}}$	Deselect.
$\text{tap}_{\text{inside}}$	Toggle.
$\text{lasso}_{\text{overlapping}}$	Add.
$\text{crop}_{\text{inside}}\ \text{crop}_{\text{opposite}}$	Add.
$\text{crop}\ \text{crop}_{\text{opposite, in selection}}$	Add.
$\text{crop}_{\text{inside}}\ \text{crop}_{\text{same, outside selection}}$	Resizing add.
$\text{tap}_{\text{outside}}\ \text{tap}_{\text{in selection}}$	Add.
$\text{lasso}_{\text{outside}}\ \text{tap}_{\text{in selection}}$	Add.
$\text{crop}_{\text{outside}}\ \text{crop}_{\text{opposite, outside}}\ \text{tap}_{\text{in selection}}$	Add.

Table 5: Selection refinement.

The selection is drawn with a rectangle around the combined bounding box of all the selected objects, with a minimum size so that there's enough room to at least start a gesture conveniently inside. The idea is that the area inside the selection rectangle is a general gestural area, potentially extensible to other gestures than the ones we have described. In particular, we do not expect conflicts with our existing gesture set if letter gestures were to be added analogous to command-key shortcuts in a standard WIMP GUI.

All the subgestures that require context from Section 4.2.3 may additionally be used in conjunction with an existing selection. Except for delete and scale, all gestures used in such a way must start inside the selection rectangle. Further, none of the non-interactive variants of such gestures requires terminal punctuation except for scale, whose punctuating tap must fall within the selection rectangle. When the zoom gesture is used on an existing selection, the view zooms in on the selection's bounding box, not the original context used to make the selection. When moving an existing selection, the move gesture is not required to intersect anything (though the minimum length constraint still holds).

4.2.6 Selection Refinement Once a selection is made, we provide gestures for adding items outside the gesture set and also gestures that apply only inside the selection rectangle for modifying the selection by deselecting, toggling, and adding strokes. See Table 5. In general, all the selection starting gestures may be used to refine a selection as well. If the selection starting gesture is entered at least partly inside the existing selection, it does not need punctuation and will refine the selection. If a selection starting gesture is made entirely outside the existing selection, it requires final punctuation inside the existing selection to make it refine that selection; otherwise, it will deselect the old selection and make a new one, roughly following Saund and Lank [13]. We do not make use of any interactive variants of these gestures, though note that "tap pause" is used to invoke a FlowMenu as described above.

The "resizing add" command computes a rectangle made by

resizing the bounding box of the selection so that the corner corresponding to the orientation of the crop mark becomes at the apex of the second crop mark, and adds to the selection any objects enclosed by this rectangle but not by the old selection bounding box. Thus, all objects in the new rectangle are selected, except that objects that were deselected in the old bounding box remain deselected.

4.3 Gesture Primitive Implementation Details

This section describes some implementation details of our current gesture primitive recognition engine. We describe them for reproducibility and to indicate what the robustness of our gesture set allows us to get away with; this is not necessarily meant to indicate how one *should* recognize these gestures! The first step, before going through any other recognition process, is to remove all subpixel self-intersections from the copy of the stroke used for recognition.

Recognizing the strokehook primitive depends entirely on what cusps are reported by Microsoft's TabletPC Ink SDK. Given a list of those cusps, first, cusps are removed from that list until there are no two successive cusps that are closer than 6 pixels. A strokehook is then recognized if there are exactly 3 cusps (the SDK reports "cusps" at the start and end of a stroke) and the part of the stroke after the middle cusp goes back along the first part of the stroke (based on the dot product being negative).

A tap is recognized if a stroke is made either in 100 ms or less with a bounding box of less than 15 pixels^2, or in 200 ms or less with a bounding box of less than 10 pixels^2.

A pause is detected if, for the past 200 ms, the stylus has been down and no input point during that period was more than 4 pixels away from the current point. If it's an initial pause (meaning the time period includes the start of the stroke), then the threshold increases to 400 ms. All three of these thresholds probably should be user preferences, but it seems more important to investigate using a dynamic speed threshold (based on the speed history of the stroke) in case someone happens to be drawing something particularly carefully.

The scribble has a particularly elaborate recognition process that is based on the stroke having 5 or more cusps including the start and end points. (Unlike for the strokehook, no filtering is done.) We then form a triangle strip consisting of the set of triangles obtained by taking successive triplets of cusps (cusps 0, 1, and 2, then 1, 2, and 3, etc). The gesture is recognized as a scribble if at least 75% of the triangles each contain some part of some other object, or if both the first and last 25% of the triangles have at least one member containing something.

A "wide delete" will delete everything contained at least in part by any triangle from the above triangle list. The "narrow delete" takes the set of objects the wide version would have used and only deletes those members of the set which are completely contained in the convex hull of the scribble. However, if there are no such objects a normal wide delete is done.

A lasso is recognized when a stroke gets "close" to the starting point after first getting "far" from the starting point, and when the stroke contains at least half of some object. A point on the stroke is "close" to the starting point if its distance from the start is less than 25% of the maximum dimension of the bounding box of the stroke as a whole. A point on the stroke is "far" from the start when its distance is more than 45% of the maximum dimension of the bounding box. We use the Microsoft Ink SDK to determine containment for strokes; due to issues with the SDK we check for containment by both the stroke the user drew and the stroke with the same points in reverse order.

There are any number of ways to implement the other gesture primitives; we use the Microsoft Ink SDK character recognizer in a hacky way. A "$\supset$" is recognized as a rotated 'C'. A crop is recognized by looking for an 'l' or 'L' or '7' or 'c' with the appropriate rotations and with dimensional restrictions (and if it doesn't match the "$\supset$"). A "Z" is recognized as a '2' or 'Z'. This is used as a first pass, with additional filtration to avoid things like a script 'l' or an 'l' with no distinguished cusp.

5 USER STUDY

We conducted a pilot user study to judge relative user-preference between Fluid Inking gestures and a more conventional widget-based approach, with particular interest in user reaction to pause- and tap-based gestures. We did not compare to a stylus-mounted button interface because our functionality would nevertheless have required widgets or punctuation similar to those tested.

5.1 The Techniques

We compared four different ink stroke selection and manipulation techniques and two different techniques for selecting items from a marking menu. Common to all of these techniques was the requirement that a lasso be drawn around the ink strokes of interest. After drawing a lasso, two 20×20 pixel buttons, labeled "M" and "C", appear on the top of the lasso's bounding box. For manipulation, users could click on the "M" button and either drag to interactively move the ink strokes to the desired location, or simply make a second click anywhere on the tablet screen to place the ink. Alternatively, two gestural techniques were available. With the first gestural technique, a line could be drawn from inside the lasso to a destination location, followed by a tap to complete the gesture and perform the manipulation. With the second gestural technique, pausing at any time while drawing the line with the pen still on the screen resulted in the lassoed ink strokes snapping to the pen tip; further pen motion would interactively drag the strokes until the pen was lifted from the screen.

To select items from a marking menu, the user could either click on the "C" button and then drag and release in a given direction to invoke a radial menu item, or use a more gestural technique by drawing a strokehook in a given direction as long as it started within the lasso followed by a tap to invoke a menu item. For the purposes of this study, the menus contained two items (for changing ink stroke color to red or blue) corresponding to the upper and lower halves of the radial menu.

5.2 Experimental Methodology

Ten subjects (6 male and 4 female) were used in our study, five of whom had never used a pen-based computer before. We used a HP Compaq Tablet PC tc1100 for all subjects. The testing environment used in the study included five numbers in a column on the left side of the screen and a box drawn in digital ink on the right. Users were asked to move the numbers into the box and change their colors.

Subjects were first shown the four manipulation techniques and two marking menu selection techniques by the experimenter. After showing the techniques to a subject, the experimenter handed the Tablet PC to the subject to begin trials. Subjects first underwent a training and acclamation phase. They were asked to move all five numbers into the box using one of the manipulation techniques at random so as to ensure that subject preferences were not biased based on technique ordering. The user would reset the numbers and repeat the process for each remaining technique. After all of the manipulation techniques were completed, subjects were then asked to change the color of all numbers using one of the marking menu selection techniques and then the other, once again in random order.

To insure that subjects had a reasonable understanding of all manipulation and marking menu techniques, we asked each subject to recall and perform the various techniques. If the subject did not remember how to perform a particular technique (this occurred twice out of our ten subjects), we had the subject practice the technique again.

The last part of the experiment asked the subject to choose the manipulation and marking menu selection techniques that they liked the best and then use them to move all five numbers into the box and change their colors to red or blue. Subjects' preferred techniques and why they chose them were recorded.

5.3 Results

Subject preferences are summarized in Figure 3. The results show that the majority of subjects (80%) preferred the gestural approach to ink stroke manipulation over the widget-based approach. Additionally, 70% of our subjects preferred using the "lasso strokehook tap" technique to make a marking menu selection. For those subjects who preferred the gestural approach to ink stroke manipulation, 75% of them found the "lasso stroke tap" technique to be the easiest and most intuitive to use.

When subjects were asked why they preferred one technique over another, we received several interesting responses. The main comment from subjects who preferred gestural techniques was that they did not like having to first click on a button to move an ink stroke. They felt that it was an extra step and harder to perform quickly. In some cases, they found it difficult to click on the button. In fact, one of the more interesting observations regarding subject preferences is that, even though subjects had some difficulty using the "lasso strokehook tap" technique for marking menu selection, the majority of them preferred the technique, indicating that, in the context of pen-based computing, subjects disliked having to press buttons to select menu items. Much of

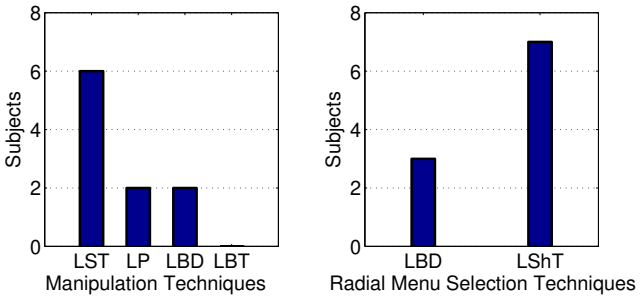

Figure 3: Technique preferences. "L" stands for lasso, "S" for stroke, "T" for tap, "P" for pause, "Sh" for strokehook, "B" for using the button widget, and "D" for dragging.

the difficulty appeared to be due to the use of an unreliable recognition algorithm for strokehook, which has since been improved.

Subjects who preferred the gestural manipulation approach also said they did not like having to pause to move the ink. However, several subjects commented that they did like the fine grained control which pausing affords and would consider using the technique if precise movements are required. These comments indicate that further study is needed to determine the precise conditions in which the pausing technique is most preferred. For the subjects who preferred the button-based techniques, they found these approaches intuitive and easy to learn because clicking on buttons was what they were used to performing.

6 DISCUSSION

We used our principles to design and formatively evaluate a gesture set for note-taking. We now discuss our principles in terms of our experience with the resulting gesture set.

We were able to find paper and pencil idioms that closely matched the notions of indicating movement, context selection and deletion. Our corresponding gestures have been generally well received as people commented on how natural they seemed and easy they were to perform. The paste and insert space actions were more loosely related to existing idioms—paste by its relation to using crop marks to indicate the boundaries of pasted pictures, and insert space by its relation to overdrawing division lines in a notebook to separate content. The zoom gesture doesn't directly correspond to a notational convention, but instead leverages a mnemonic relationship with the name of the action, and in fact, people found memorable and fluid. We used a similar approach to design the menu gesture—the strokehook corresponds to a physical memory of the motion required to invoke a FlowMenu.

In contrast, we had a difficult time finding any basis for an undo gesture. Our initial attempt involved drawing a right-to-left line thinking that this was easy to perform and analogous to "backing-up" in time. However, our supposition that it would not conflict with regular note-taking was disproven during formative testing when a left-handed user drew the lines in the "=" sign in either direction, which caused con-

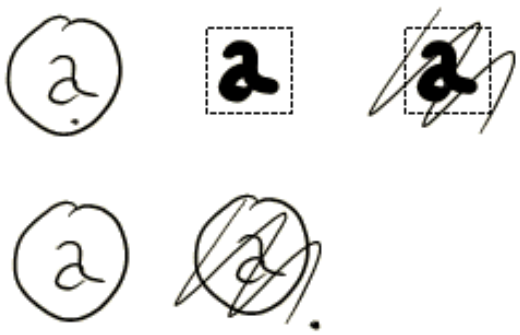

Figure 4: Top row: create an explicit selection by lassoing and tapping, then delete with a scribble. Bottom row: use same gestural actions in a different order (lasso, scribble, tap) to chunk selection and deletion into one cognitive activity.

flicts when writing expressions like "$x = .4$". In addition, our technique of repeating undo actions required targeting the original line stroke, which many people found difficult and stressful. Our next idea was a gesture that vaguely resembles a "U-turn" and provides a much larger area to target. However, during formative testing we noted that the interactive variant of the gesture was often triggered accidentally, for when drawing a capital "P" or a loop attached to another shape. Our current gesture is the same shape, but must be drawn bottom-to-top; although this forces users to do something that is unnatural to avoid conflicts, people seem to learn and have little trouble performing this gesture.

We were successfully able to avoid precise targeting except in the case of the second tap of a double tap, but this seems like a special case of motor control because no hand movement is required between taps. In fact, some user study participants specifically noted that the button techniques that required targeting were slower and harder to perform even though we made special efforts to make the buttons larger than they would typically be in a real application. We struggled with the idea of using precise tap location to indicate different operations—on the one hand, they make more functionality directly accessible, but whenever we tested them, we found the difficulties in targeting overwhelmed the benefit.

We have not fully evaluated the impact of chunking gestural actions or explored their implications. However, we informally observed that chunking does seem to make operations seem more fluid. Consider Figure 4. In addition, although we didn't contrast with our explicit selection technique, the chunked movement gesture was the most preferred variant in our user study.

Although providing feedback is a staple of user interface design, we note that it can be detrimental if users have to wait for it. Previous results [6] have found that there is an advantage if users are able to invoke radial marking menus without the disruption of having the menu appear. We noticed this both with our FlowMenu gesture and when chunking actions together, for instance to move an object. Nonetheless, we experimented with providing feedback for the undo gesture before it is punctuated to notify users that they had performed the unusual gesture correctly. It seems that such feedback may be useful during training, but it does not seem to be useful for trained users and in fact is sometimes distracting. A current problem with our implementation is that some operations may take a half-second or more to compute, such as deleting a large group of strokes at one time. In these situations, a user may perform the gesture and feel that it was not recognized because we provide no feedback during the computation. We also feel that it might be nice to have some form of explicit feedback for operations like copy, which currently are only indicated by the disappearance of the gesture strokes.

We had been concerned that the fundamental principle for our gesture set, terminal punctuation, might not be generally accepted by users. Our user study showed that a majority of users preferred our punctuated gesture techniques over more traditional alternatives. We believe that the group that chose the other techniques would accept punctuated gestures and might actually prefer them when used in the actual context of note-taking, since our user study did not even involve switching between gestures and ink. Although we have characterized most of the infrequent mis-classification of gestures vs. notations as errors in our recognition algorithms, we believe that there may still be cases where errors will occur. Thus, we may consider having an explicit mode which turns off interpretation if the user knows they are about to draw something that will conflict. The converse mode, which interprets everything as gestures, might also be useful when the user is cleaning up or reorganizing their notes. We also note that it is difficult to unlearn gestures in general, and even casual users of our system have tried to use our gestures in Journal and wondered why nothing happened. Thus, since some users reflexively tap when performing an action on an existing selection because the gesture is so similar to gestures requiring terminal punctuation, we plan to allow optional terminating taps.

7 CONCLUSION

We have presented *Fluid Inking*, a set of design principles for modelessly combining gestures with free-form notetaking. These principles include exploiting common pencil and paper idioms, avoiding precise targeting, supporting chunked actions, suppressing intermediate feedback, and using terminal punctuation. We used these principles to develop a novel note-taking application user interface featuring a full gesture set defined as short sequences of drawn ink marks and terminal punctuation. Although individual gesture primitives may be identical to common note-taking marks, collective gesture sequences were designed to be easily disambiguated from notes. Based on our informal feedback and user study, we believe our gesture design principles are a powerful approach for creating a novel style of gesture sets that are effective for free-form note-taking because of the fluid transitions they provide between gesturing and marking.

8 ACKNOWLEDGEMENTS

This work was supported in part by the National Science Foundation, Microsoft, and the Joint Advanced Distributed Learning Co-Laboratory. Thanks to David Laidlaw for discussions and advice about the user study; that work was also supported in part by NSF (CCR-0086065).

REFERENCES

1. C. Alvarado and R. Davis. Resolving ambiguities to create a natural computer-based sketching environment. In *Proceedings of IJCAI*, pages 1365–1371, 2001.

2. W. Buxton. Chunking and phrasing and the design of human-computer dialogues. In *Proceedings of the IFIP World Computer Congress*, pages 475–480, Dublin, Ireland, 1986.

3. F. Guimbretière and T. Winograd. FlowMenu: combining command, text, and data entry. In *Symposium on User Interface Software and Technology*, pages 213–216, San Diego, California, US, 2000. ACM.

4. T. Igarashi and J. F. Hughes. A suggestive interface for 3D drawing. In *Symposium on User Interface Software and Technology*, pages 137–181, Orlando, Florida, US, 2001. ACM.

5. G. Kurtenbach and W. Buxton. Issues in combining marking and direct manipulation techniques. In *Symposium on User Interface Software and Technology*, pages 137–144. ACM, 1991.

6. G. Kurtenbach and W. Buxton. User learning and performance with marking menus. In *Proceedings of CHI '94*, pages 258–264. ACM SIGCHI, 1994.

7. J. A. Landay. SILK: sketching interfaces like krazy. In *CHI '96*, pages 398–399, Vancouver, Canada, 1996. ACM.

8. J. J. LaViola, Jr. and R. C. Zeleznik. MathPad^2: a system for the creation and exploration of mathematical sketches. In *Computer Graphics (SIGGRAPH 2004 Conference Proceedings)*. ACM SIGGRAPH, 2004.

9. J. Mankoff, S. E. Hudson, and G. D. Abowd. Interaction techniques for ambiguity resolution in recognition-based interfaces. In *Symposium on User Interface Software and Technology*, pages 11–20. ACM, 2000.

10. Microsoft Corporation. OneNote. Part of Microsoft Office, 2003.

11. T. P. Moran, P. Chiu, and W. van Melle. Pen-based interaction techniques for organizing material on an electronic whiteboard. In *Symposium on User Interface Software and Technology*, pages 45–54, Banff, Alberta, Canada, 1997. ACM.

12. E. D. Mynatt, T. Igarashi, W. K. Edwards, and A. LaMarca. Flatland: New dimensions in office computing. In *CHI '99*, pages 346–353, Pittsburgh, PA, 1999. ACM.

13. E. Saund and E. Lank. Stylus input and editing without prior selection of mode. In *Symposium on User Interface Software and Technology*, pages 213–216, Vancouver, BC, Canada, 2003. ACM.