Can Friction Improve Mouse-Based Text Selection?

Dusty Phillips
Dept. of Computer Science & Engineering
York University, Toronto, Canada

Wolfgang Stuerzlinger
Dept. of Computer Science & Engineering
York University, Toronto, Canada

Abstract— **Text selection via caret positioning is a common task in modern word processing interfaces. It is a difficult task as target size – the distance between characters – is very small. We adapt Baudisch's snap-and-go technique for target acquisition to insert additional motor space at such targets when the mouse is decelerating. This increases the size of the target in motor space, thus potentially reducing the difficulty of text selection tasks. We expand this idea by introducing the concept of context-sensitive friction to improve target acquisition for common targets. We performed two pilot studies and a controlled user study to evaluate the new techniques. Our results indicate that selectively introducing friction into the interface can reduce total task time in common text selection tasks.**

Keywords: selection, snapping, control-display ratio, friction.

I. Introduction

Next to actually inserting and modifying text in a document, positioning the text-editing caret is arguably the most common task users undertake in text editing and word-processing. Another common and related task is text selection, which consists of two separate caret-positioning tasks: one each for the beginning and end of the target. In modern word-processing applications, two alternatives exist for caret positioning tasks: mouse- and keyboard-based techniques, and most applications support both. Mouse-based caret positioning entails moving the mouse cursor to the target area (between two characters) and clicking the button to position the caret. Text selection can occur in one of two ways: positioning the cursor at one end of the target, pressing the mouse button, dragging to the other end and releasing the button. Or alternatively positioning the cursor at one end, clicking the button, holding a modifier key (typically *Shift*), moving the cursor to the other end, and issuing a second click. Both techniques require positioning the cursor twice before text selection can occur and an error in either will require redoing both positioning tasks [9].

The emphasis of this work is on improving the mouse-based user interface for text selection. Most such tasks select blocks of words, sentences, or paragraphs. Users rarely start or end selection in the middle of a word. In contrast, other caret positioning tasks, e.g. for editing, have a more equal distribution of possible targets as editing may start anywhere. While it may be possible to adapt some of our work to caret positioning, this is beyond the scope of this paper.

A. Targets are Small and Numerous

In positioning the caret with a mouse, potential targets are all locations between characters in the document. These characters are tiled across the page with little space between them – the whitespace after the end of a line or paragraph being an exception. Since a page of English text can easily contain 5000 characters with a 10-point font, there is an equal number of possible targets, and each is relatively small. The target area for each caret position (between characters) is the line height by the sum of half the width of each character; see Fig. 1.

On a 1280x960 display at 100% zoom, text in 10 pt Times is about 15 pixels high, and characters are 4-10 pixels wide. The average Fitts' law index of difficulty for the caret positioning tasks is then in the range of 5-7 bits. This is high, especially for such a common pointing task, and finding ways to make it easier is worthwhile. Note that not all targets in a text are of equal interest to a user. Users generally select words and sentences more often than characters within words. Programmers generally select specific parameters, statements, or code blocks rather than characters in source code.

B. Related Work

Cursor positioning for text selection was first evaluated by Card [8], who compared four input devices including the mouse and cursor keys for hitting text targets. However, this was not a caret positioning task: targets were words, and a selection was counted when *any* portion of the target was hit. Mackenzie later revisited this data [11] and noted that 2D Fitts' law experiments should either use the approach angle or choose the smaller of width and height for target size [12]. Gillan analyzed point-drag movements and pointed out that the targets were the beginning and end of the target text [9]. Mackenzie also analyzed the differences between pointing and dragging, showing that Fitts law applies to both techniques [13].

Microsoft Word supports "select by words". When enabled, any selection that crosses a word boundary will automatically extend from the beginning of the first word to the end of the last. This is very useful for selecting whole words. However, for partial selection of words, it can be distracting. The feature can only be disabled via the "Options" dialog. Backtracking the selection will undo the word expansion, but this is not obvious and most users are unaware of it.

The idea of dynamically modifying the mouse cursor position is not new. Snapping was introduced by Bier [5] and

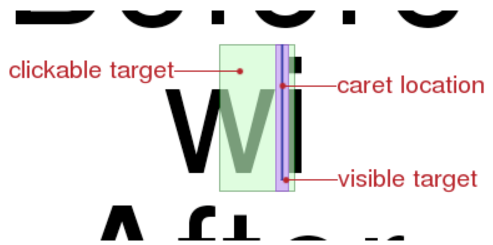


Figure 1. The target area when positioning the caret between two characters (green). The apparent (visible) target is much smaller: the distance between the two characters (purple).

has been used extensively in user interfaces. This snaps the mouse cursor to "important" locations during each mouse movement or drag. Semantic Pointing facilitates target acquisition by modifying the control-display ratio – the amount of distance covered by the mouse vs. the amount of distance the cursor moves onscreen [7]. Baudisch [4] described how control-display ratio adjustment can compensate for difficulties in snapping and how this idea can be applied to target acquisition. He introduced *friction* in the *Snap-and-Go* technique and coined the word *frixels*, i.e. friction pixels. There the trajectory of the mouse is adapted as it crosses important boundaries. The cursor pauses momentarily on the screen, while the mouse continues moving. Conceptually, the mouse cursor crosses invisible frixels at the current target, making it easier to acquire, without inhibiting the ability to target nearby positions. "Object pointing" is also based on the idea that certain pixels ("objects") on the screen may be more important than others [10]. Other mouse-adaptation techniques have been presented [1],[2],[3],[17],[19]. Noy's implicit disambiguation work [16] used a reversed Fitts' Law to determine the size of a target from movement time. This is useful when multiple targets occupy the same area and Noy discussed text selection as an obvious example: a character, a word, and a phrase all occupy the same location. With this, selection can occur with a single point-click movement, rather than a point-drag motion. However, no follow-up work has been presented. Miller described how a smart editor could intuit selections from previous selections for batch-style processing [14],[15]. Bier recently presented the Entity Quick Click system for rapid copy-and pasting of text [6].

C. *Contribution*

In this paper we present an algorithm to improve mouse-based text selection with friction. It offers two improvements over existing algorithms. First, the amount of friction applied at a specific target depends on the speed of the mouse as it crosses that target; more friction is applied at lower speeds. Second, the amount of friction applied at each target is context-sensitive, depending on the text content and mouse motion.

II. Improved Text Selection

To speed up text selection, we investigated techniques that extend standard mouse-based techniques. The first is a variation of snapping [5], where the cursor snaps only to character boundaries to minimize both cursor and text caret obfuscation. This is then similar to Object Pointing [10] in that the mouse cursor only moves between points where the caret can be placed. In practice, we found that snapping the cursor to the nearest target in the presence of many nearby targets is problematic, as the cursor frequently snaps back to the target that the user has just passed during a motion, which is undesirable. Accumulating the distance the mouse moved since the last snap action fixes this problem.

A. *Speed-dependent Friction*

We also adapted the *Snap-and-Go* technique [4] to text selection. This technique inserts *frixels* into motor space to increase the motor target size, and thus reduce acquisition difficulty. The main issue with adding friction to text editing or any interface with densely tiled targets is that simply adding

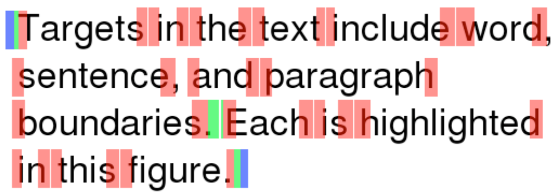

Targets in the text include word,
sentence, and paragraph
boundaries. Each is highlighted
in this figure.

Figure 2. The most salient targets in a paragraph of text are word, sentence, and paragraph boundaries. Some salient targets overlap. Target types are shown in different colours.

friction at *each* target causes all nearby targets to function as distractors. Hence, it is better to apply friction only during the final fine-tuning movements of caret positioning. In our new technique we adjust the amount of friction depending on speed.

B. *Context-Sensitive Friction*

To further improve text selection, we introduce the idea of context-sensitive friction. Here, the number of frixels added when the mouse passes over a target depends on the relative importance of the target. This enables the user to more easily acquire more salient targets while not limiting their ability to acquire arbitrary ones. E.g. the beginning or end of a word, or sentence, is more important than the space between two characters of an arbitrary word, see Fig. 2. Target importance can either be pre-computed or determined dynamically. There are many possibilities for determining the importance of a target. We implemented this with multiplicative factors for simplicity. Each target type has a base amount of friction. This is multiplied by a factor at certain locations depending on context. Currently, this includes the type of boundary, the type at which the drag operation began, if it is the complementary boundary relative to the start of the drag operation, and the direction of motion. E.g., if selection starts at the beginning of a word, the ends of words are emphasized with additional friction, and the end of the current word is emphasized by an even higher factor. Taking the direction of motion into account causes the beginnings of objects to be emphasized more when the mouse is moving from left to right, and vice versa.

C. *IMPLEMENTATION*

We implemented the above snapping and friction techniques in Python and PyGTK. The implementation of snapping was straightforward. First, the nearest target is identified. If the mouse is moving away from that target, the distance moved is accumulated and added to the cursor location when the next event is processed. We also use a threshold to preclude snapping if the speed is high. This allows the user to cover large areas of whitespace (specifically the distance from the end of a line to the edge of the window) without snapping.

For friction, we track horizontal and vertical motion separately and also insert frixels at character locations in both dimensions independently. We first calculate the actual screen distance the mouse has moved since the previous event. If the mouse was previously in a friction location, we subtract the number of frixels not yet accounted for. Otherwise, we search all screen positions traversed by the mouse and subtract all corresponding frixel values. This may leave the cursor in a friction location, in which case, we store the number of frixels not yet accounted. Pseudo code is shown in Figure 3.

This basic friction algorithm is common, but an external function is used to determine the amount of friction for a given character location. The function implements either speed-dependency or context-sensitivity. The first method inserts fewer frixels when the mouse is moving quickly, and more when it is slowing down. For this, we take the physical screen distance moved since the previous motion event and compute the percentage of this maximum value that should be inserted into motor space. If the motion is less than a certain threshold, full friction is inserted, if it is more than a second threshold, no friction is inserted. Between these values, the amount of friction is interpolated linearly. For context-sensitive friction, we pre-parse the text at each character boundary into types (i.e. start/end for word, sentence, block, etc) and use this as a lookup table when the mouse crosses a specific character location. Alternatively, one could integrate this into an editor's syntax highlighting and tokenizing routines.

III. PILOT STUDIES

To investigate the new snapping and friction methods, we performed two pilot studies. In each study, the basic task was to select a text target by a click and drag sequence. The user interface for all studies was a 600x600 pixel text view window, with a 32x32 pixel task initialization button vertically centered on the right, see Fig. 4. The purpose of the first pilot was to compare both snapping- and speed-dependent friction-based interfaces to traditional cursor positioning. The second compared context sensitive to speed-dependent friction and the traditional interface. The intent of both pilots was to explore the domain and determine the potential of the techniques.

A. *Task*

The participants' task was to perform a series of selections in each of four conditions. Each trial began with a click on the initialization button. The target was highlighted, see Fig. 4. Users then moved the mouse to either end of the target and initiated a click-drag sequence to select it. Time, distance, target size and errors were recorded separately for both the click and the drag movement. We used four types of text: parts of a Wikipedia article, lines from a classic play, HTML, and Java source code. There were four types of selection target: words, sentences, paragraphs, and targets starting and ending at arbitrary positions. Participants were asked to perform each trial as quickly and accurately as possible.

B. *Conditions & Apparatus*

There were four main conditions for each of the pilots. The first compared traditional mouse motions, snapping, symmetric and asymmetric friction-based positioning. The second extended this and compared traditional positioning with a revised symmetric friction-based positioning method along with two variations of the context sensitive methods.

In the traditional method, mouse motion was unmodified. In snapping, the cursor moves only to character boundaries. The two basic friction conditions inserted frixels in each dimension at all character boundaries, using speed-dependent friction. The symmetric friction condition (Friction-1) inserted the same number of frixels in each dimension, while the asymmetric one (Friction-2) inserted more frixels for horizontal than vertical. Caret targets are generally smaller in width than

```
function friction_motion(location):
  set pixelsMoved to distance to previous
  if pixelsMoved <= frictionLeft from last motion
    remove pixelsMoved from frictionLeft
    reposition cursor to previous location
  else: //need to account for extra friction
    remove frictionLeft from pixelsMoved
    reset frictionLeft to 0
    initialize actualToMove to 0
     //Check each pixel between previous and end
    while pixelsMoved > 0:
      //set pix to location we are checking next
      pix = previous + actualToMove + 1
      set frictionLoc to character nearest to pix
      increment actualToMove by one
      if pix is same as frictionLoc:
        extra = getNumFrixels(frictionLoc)
        if pixelsMoved more than extra to insert:
          remove extra from pixelsMoved
        else: //moved into a friction location
          reset frictionLeft for next motion
          reset pixelsMoved to 0
      else: //Crossed pixel with no friction
        decrement pixelsMoved by 1
    reposition cursor to previous + actualToMove
    reset previous to new location
    reposition cursor to previous + actualToMove
    reset previous to new location
```

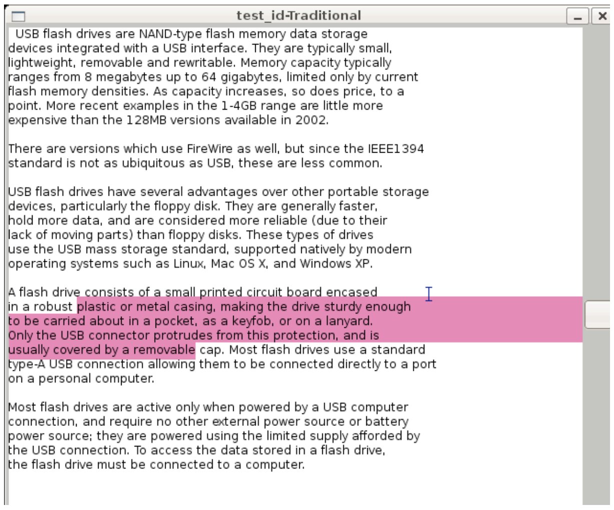
Figure 3. Simplified pseudocode for friction

in height, and the smaller of width and height has been shown to be a good approximation for two-dimensional Fitts tasks [11]. The two context-sensitive friction techniques for the second pilot added friction only at word, sentence, and paragraph boundaries. No friction was added for other targets. Higher-level targets were considered more salient and therefore received more friction. But as they are less frequent, this does not add much distraction. One technique considered the direction of mouse movement, the other did not.

Experiments were run on a Linux PC with two 19" 96dpi monitors at 1280x1024@75Hz. The main window was presented centered on the left-most monitor. A Microsoft IntelliMouse was used as input device. The keyboard was moved out of the way for the duration of the study.

C. *Experimental Design*

Both studies utilized a within-subjects 4x4x4 design (Positioning Technique by Text Type by Selection Type),



USB flash drives are NAND-type flash memory data storage
devices integrated with a USB interface. They are typically small,
lightweight, removable and rewritable. Memory capacity typically
ranges from 8 megabytes up to 64 gigabytes, limited only by current
flash memory densities. As capacity increases, so does price, to a
point. More recent examples in the 1-4GB range are little more
expensive than the 128MB versions available in 2002.

There are versions which use FireWire as well, but since the IEEE1394
standard is not as ubiquitous as USB, these are less common.

USB flash drives have several advantages over other portable storage
devices, particularly the floppy disk. They are generally faster,
hold more data, and are considered more reliable (due to their
lack of moving parts) than floppy disks. These types of drives
use the USB mass storage standard, supported natively by modern
operating systems such as Linux, Mac OS X, and Windows XP.

A flash drive consists of a small printed circuit board encased
in a robust plastic or metal casing, making the drive sturdy enough
to be carried about in a pocket, as a keyfob, or on a lanyard.
Only the USB connector protrudes from this protection, and is
usually covered by a removable cap. Most flash drives use a standard
type-A USB connection allowing them to be connected directly to a port
on a personal computer.

Most flash drives are active only when powered by a USB computer
connection, and require no other external power source or battery
power source; they are powered using the limited supply afforded by
the USB connection. To access the data stored in a flash drive,
the flash drive must be connected to a computer.

Figure 4. The user interface used in the study with selected text. The task initialization button appears to the right of the text area.

where positioning technique was counterbalanced to account for potential learning effects. Six different selections were performed for each selection type, with no repetitions. Hence, there were 96 trials for each technique for a total of 384 trials per user. Task times and distances were recorded separately for the click and drag portions of each trial. Task time for clicking starts with the release of the initialization button and ends when the mouse button is pressed over the text area. Drag time starts with a mouse button press and ends on release. Distance is calculated from the location of the mouse event to the center of the target. Errors are logged independently.

D. *Participants*

Twelve paid volunteers were used for each pilot, recruited from the local campus. In the first, 7 males and 5 females between the ages of 21 and 34 (mean: 25) participated, in the second, 4 males and 8 females between the ages of 18 and 31 (mean: 24). All participants had previous experience with graphical user interfaces and mice and all were right handed.

E. *Hypotheses*

For the first pilot, we had three hypotheses: (**1**) *Task times for all four conditions would be about the same.* The traditional and snapping interfaces have roughly the same index of difficulty. Friction-based targets have larger target size in motor space but the mouse may also have to cover larger distances; thus the index of difficulty should be roughly the same. While the evaluation of Snap-and-go showed substantial effects [4], we cannot expect to replicate this as our targets are much denser. (**2**) *Participants would have a lower error rate with snapping.* With less visual interference between mouse cursor and characters, participants should be able to locate targets more easily. (**3**) *The error rate would be lower for friction than for the other conditions.* The larger target size in motor space should make targets easier to acquire.

For the second pilot we had the following two hypotheses: (**4**) *The context sensitive conditions would show lower time for common targets (words, sentences, and paragraphs), but not for random targets.* Friction emphasizes targets most likely to be selected by users, thus making them easier to acquire. For random targets, the context-sensitive method should not have a noticeable effect because as its targets are separate from those enhanced by friction. (**5**) *Directional context sensitive friction would be faster.* Context sensitive friction may suffer from nearby distracters, e.g. the end of the previous word when selecting the beginning of the next. Making it dependent on friction should reduce this effect.

F. *Results*

1) *Selection Time*

For the first pilot, a within-subjects ANOVA on selection time shows a significant difference, $F_{3,33}=41.83,\ p<<0.01$. A Tukey-Kramer test revealed that snapping was significantly slower than all other techniques. Mean times were Traditional: 3.39, Friction-1: 3.62, Friction-2: 4.00, Snap: 4.92 seconds. Hence, hypothesis (**1**) did not hold. There was also a significant effect in the second pilot, $F_{3,33}=10.09,\ p<<0.01$, where the traditional interface was significantly faster, but there was no difference between the friction conditions. The mean times were Traditional: 3.31, Context-Sensitive: 3.49, Directional: 3.59, Friction: 3.61 seconds. Analyzing the tasks by target type for hypothesis (**4**), we found a significant effect only for random selections, $F_{3,33}=10.40,\ p<<0.01$. Mean times were: Traditional: 3.39, Friction: 3.65, Context-Sensitive: 3.71, Directional: 3.95 seconds. There was a main effect for selection time on non-random targets (words, paragraphs, sentences), $F_{3,33}=7.24,\ p<<0.01$, and only the simple friction and traditional interfaces were different. Average movement times were Traditional: 3.29, Context-Sensitive: 3.42, Directional: 3.47, Friction: 3.60 seconds. Hence, Hypothesis (**4**) did not hold. There was also no significant effect between the two friction tasks and the directional interface tended to be slower, disconfirming Hypothesis (**5**).

2) *Error Rate*

There was a slight main effect on error rate in the first pilot, $F_{3,33}=3.00\ p<0.05$. The Tukey-Kramer test indicated that snapping was significantly worse than the traditional interface. The error rates are Traditional: 7.4%, Friction-1: 8.8%, Friction-2: 8.9%, Snap: 12.9%. In the second pilot, there was no main effect on error rate, with error rates Context-Sensitive: 9.9%, Directional: 9.9%, Traditional: 10.5%, Friction: 10.7%. There was no main effect on error rate for the friction conditions, so Hypothesis (**2**) and (**3**) did not hold. While error rates appear to be high compared to the 4% expected for Fitts' tasks, we note that each task is composed of two positioning tasks, which increases the potential for errors substantially.

3) *Other Effects*

The plain text article was significantly slower than the others in the first pilot, $F_{3,33}=7.65,\ p<<0.01$. Means were: Play: 3.8, Java: 3.92, HTML: 3.94, Plain Text: 4.26 seconds. The text article was also significantly slower than HTML in the second pilot, but a Tukey-Kramer test indicated no further differences. We also noticed additional interesting effects in our second pilot. There was a main effect for movement time on target type, $F_{3,33}=33.34,\ p<<0.01$, and on text type, $F_{3,33}=20.97,\ p<<0.01$. The former is to be expected, since target size varies between the four target types and size has a direct impact on movement time. The effect on text-type is somewhat surprising, but can be explained by the fact that the play and source code have more line breaks and hence more easy-to-hit targets.

4) *User Preferences*

In the first pilot eleven of the twelve users preferred the traditional interface, and eleven ranked the snap interface in last place. Seven ranked symmetric friction ahead of asymmetric. User preference in the second pilot varied more. Half of the 12 users rated the traditional interface as their first choice, and four preferred the context sensitive condition first. Overall, the traditional and context sensitive techniques were preferred over the friction and directional friction techniques.

G. *Discussion*

These pilots had several uncontrolled variables and the results are somewhat inconclusive. The only fact we can derive from the first pilot is that snapping is clearly inferior. Users had a great deal of difficulty with this interface as the cursor tended to jump between lines. We suspect that an interface that only snaps in the *X* dimension might be better received. The symmetric friction algorithm outperformed the asymmetric friction algorithm. While significant, these results are not conclusive; it is probable that the added friction in the horizontal dimension caused more interference, and not because asymmetry in friction is a bad idea per se. Without taking context into account, speed-dependent friction performs less effectively than the traditional method. However, observations during the pilots indicate that this may be due to inappropriate parameter settings (too much friction, wrong speed-dependent thresholds, or multiplicative constants too big). We also noticed that many users did not move the mouse in a way that allows prediction of target type from direction. Hence, we are uncertain if direction is a good design factor.

IV. USER STUDY: CONTEXT SENSITIVE FRICTION

Users are experienced with the traditional interface. As the parameter settings emerged as a critical issue in our pilots, we incorporated a tuning task in our main user study, where we adjusted parameters for each user. This helps also to identify which parameters are user-specific, and which can be set globally. The results from our pilots could not be easily analyzed as a Fitts' task due to several issues, such as the freedom of users to select text in either direction or the lack of repeated indexes of difficulty. Hence, we designed the main study to enable such analysis as well.

A. *Task*

The task was a restricted version of that used in the pilots. In particular, users were asked to select only from left to right in this study, to fix the indexes of difficulty. We only evaluated two techniques in this study, traditional and a revised form of context-sensitive text selection. Each trial was repeated four times to enable users to learn the new technique. We used only regular text, as this was the slowest condition in our pilots and is also the most common task. This also allowed us to fix the types of boundaries we targeted and reduced the potential for interaction effects. We chose four different Wikipedia article excerpts to alleviate participant boredom. We limited ourselves to targets aligned to word boundaries. These are the most common targets, and are easy to define and mark up in context. Target lengths ranged from one three-letter word to half a paragraph. Targets were selected to ensure they did not begin or end on artificially large targets like line and paragraph boundaries, as such targets are much easer to acquire and the varying width complicates a Fitts' analysis.

B. *Experimental Design*

This study was performed in three parts. We first performed a 2x4x8x4 (condition by text by target by repetitions) within-subjects experiment with counterbalanced conditions using default parameters for the context sensitive algorithm. For five minutes we then performed interactive parameter tuning. Participants would make a few selections; we would adjust parameters; have the user make more selections, etc. We based tuning on user feedback as to how the adjustments felt, speed and error measures, and observation of problems during selection. Four parameters were modified. *Speed thresholds*: The minimum and maximum speed at which friction is fully applied or not applied at all, with linear interpolation between. Defaults were 2 and 32 pixels per motion event. *Line friction:* The amount of friction inserted at each line in the *Y* dimension. Increasing this helps users stay on the same line. Initially set to 1 frixel. *Word friction:* the number of frixels to insert at the beginnings and ends of words. Default was 5 frixels. *Drag-factor:* A multiplicative friction factor to emphasize end-of-word targets, once the beginning of a word had been selected. The default was 2. *Direction-factor:* A multiplicative friction factor to emphasize motion towards the beginning or end of a word. The default was 1.5. In the third part of the study, we again evaluated the traditional interface against the context sensitive interface using the design of the first part, this time using the custom parameters identified during tuning.

C. *Participants*

Ten paid volunteers (3 male, 7 female) between the ages of 20 and 28 (mean: 23) were recruited from our campus. All participants had previous experience with graphical interfaces and mice. All used the mouse with their right hand.

D. *Hypotheses*

We had 3 hypotheses. (**6**) *Movement time for both initial positioning and drag positioning tasks would be lower for the context sensitive interface than for the traditional interface.* A larger target size in motor space gives friction an advantage, and all targets are at friction boundaries. (**7**) *Error rates would be lower for the context sensitive.* A larger target size in motor space again makes targets easier to acquire. Blanch's study on Semantic Pointing showed improvements in both movement time and error rate [7]. (**8**) *Users would perform better on the context sensitive task after tuning than before.* Tuning was designed to determine better parameter settings. Tuning also gave users a chance to learn the interface better.

E. *Results*

1) *Movement Time*

An ANOVA on total selection time found a main effect, $F_{1,10}=7.69, p<0.05$. Mean movement times were Friction: 2.85, Traditional: 3.00 seconds. We removed trials as outliers, if the user gave up on the task due to an error in the initial press. Interestingly, after optimizing the parameters selection time had only a marginally significant effect, $F_{1,10}=4.24, p<0.1$. The means were Friction: 2.72, Traditional: 2.82 seconds. Breaking times down, we found that the positioning time for the initial click was not different, only the drag times showed an effect both before and after optimization, $F_{1,10}=6.77, p<0.05$ and $F_{1,10}=7.55, p<0.05$ respectively. Means before optimization were Friction: 1.32, Traditional: 1.42 and after Friction: 1.25, Traditional: 1.34 seconds. Hence, hypothesis (**6**) is supported.

2) *Error Rate*

The error rates for the first part of the study were Friction: 10%, Traditional: 11.6%, with no main effect. There was a significant effect between error rates in the post-optimization part of the study, $F_{1,10}=6.66, p<0.05$. The error rates were Friction: 8.7%, Traditional: 12%. Hypothesis (**7**) is therefore supported. Both time and error rates were significantly lower for the context-sensitive friction condition after optimization and thus Hypothesis (**8**) is supported.

3) *Fitts' Analysis*

We calculated the effective width and index of difficulty for the press and drag events. We then calculated throughputs for each trial and performed an ANOVA. There was no significant difference in the throughputs for the press portion of the task before optimization, but there was a main effect after the tuning process, $F_{1,10}=5.43, p<0.05$. There was a significant difference for the drag portions both before and after optimization, $F_{1,10}=6.27, p<0.06$ and $F_{1,10}=18.17, p<<0.01$, respectively. The throughputs are summarized in Table 1.

4) *Parameter Optimization*

The factors for dragging and direction both converged towards 1.5 during the tuning process. The amount of friction at word boundaries did not vary much between users and ranged from 3 to 7 frixels, most seemed to prefer 5. The speed threshold ranged from 2 to 5 pixels for the minimum and from 32 to 64 for the maximum. We noticed that the minimum tended to affect users more, and also that the setting seemed to be deeply personal. Some users appreciated having the cursor "stick" to the line via friction in the vertical dimension, but most did not.

5) *User Preferences*

Data from the questionnaire on user preference does not tell us much. Before optimization, two participants had no preference and the remaining eight were divided equally. Most indicated that the difference was not terribly noticeable. After optimization, four participants indicated no preference, and the remaining six were again divided equally. One user switched from friction to traditional after optimization, but the remaining nine either did not change their preference or migrated towards friction after optimization.

F. *Discussion*

In this experiment, the context sensitive interface outperformed the traditional interface, especially after optimizing the parameters. Most of the parameters can be set to default values, except the speed thresholds, which seems to be individual. Also, we found evidence that the values we chose in our pilot were indeed too "strong". From our pilots we believe that context sensitive friction does not interfere significantly with other selection tasks. Therefore, context sensitive friction can be added to text editing interfaces without noticeable negative effects. The most common targets can be emphasized without impeding the ability to select other arbitrary targets.

Possibly the most surprising result is the fact that the real difference is in the dragging portion of the task. This suggests that friction is more useful for text selection than it is for caret positioning. Also, the throughput results support previous findings that users tend to drag more slowly than they position [9], [13]. We suspect that either the chosen friction parameters or the friction idea in general does not work as well during the (faster) positioning task. It is possible that different friction speed thresholds are needed for the caret positioning and text selection portions of the task.

V. Conclusion

This work targets the common task of mouse-based text selection. We compared the traditional technique to snapping and various new, iteratively designed, friction-based interfaces. Our pilot studies indicated that snapping is inferior and that simple friction methods do not work well. Our main study showed that a well-tuned implementation of context sensitive friction could significantly outperform traditional text selection techniques for targets aligned at word boundaries. For future work, we plan to investigate the concept of decreasing friction over less important areas, i.e. introduce "anti-friction", which would speed the cursor up on such areas. Also, we will focus on caret positioning separately from the text selection. Finally, we want to investigate how syntax highlighting can coexist with the identification of context-sensitive targets.

	Before Optimization		After Optimization	
	Press	Drag	Press	Drag
Traditional	4.55	3.42	4.85	3.55
Friction	4.74	3.81	5.16	4.00

Table 1: Grand throughputs for the two conditions before and after optimization.

References

[1] Aliakseyeu, D. Nacenta, M. A. Subramanian, S. and Gutwin, C., Bubble Radar: Efficient Pen-Based Interaction, AVI 2006, 19–26.

[2] Asano, T. Sharlin, E. Kitamura, Y. Takashima, K. Kishino, F., Predictive Interaction Using the Delphian Desktop, UIST '05, 133-141.

[3] Balakrishnan, R., "Beating" Fitts' law: virtual enhancements for pointing facilitation, Human-Computer Studies, 61(6): 857-874, 2004.

[4] Baudish, P. Cutrell, E. Hinckly, K. and Eversole, A., Snap-and-go: Helping users align objects without the modality of traditional snapping, CHI 2005, 301-310.

[5] Bier, E., Stone, M., Snap-dragging, SIGGRAPH 1986, 233-240.

[6] Bier, E. Ishak, E. Chi, E., Entity Quick Click: Rapid Text Copying Based on Automatic Entity Extraction., CHI Abstracts 2006, 562-567.

[7] Blanch, R. Guiard, Y. and Beaudouin-Lafon, M., Semantic Pointing: Improving Target Acquisition with Control-Display Ratio Adaptation, CHI 2004, 519–526.

[8] Card, S. English, W. and Burr, B., Evaluation of Mouse, Rate-Controlled Isometric Joystick, Step Keys, and Text Keys for Text Selection on a CRT, Ergonomics 21(8): 601-613, 1978

[9] Gillan, D. Holden, K. Adam, S. Rudisill, M. and Magee, L., How Does Fitts Law fit Pointing and Dragging?, CHI 1990, 227–234.

[10] Guiard, Y. Blanch, R. and Beaudouin-Lafon, M., Object Pointing: A complement to Bitmap Pointing in GUIs, Graphics Interface 2004, 9-16.

[11] MacKenzie, I. S., Fitts' Law as a Research and Design Tool in Human-Computer Interaction, Human-Computer Interaction, 7:91-139, 1992

[12] MacKenzie, I. S. and Buxton, W., Extending Fitts' Law to Two-Dimensional Tasks, CHI 1992, 219–226.

[13] MacKenzie, I. S. Sellen, A. and Buxton, W., A Comparison of Input Devices in Elemental Pointing And Dragging Tasks, CHI '91, 161–166.

[14] Miller, R. Myers, B., Multiple Selections in Smart Text Editing, IUI 2002, 103-110.

[15] Miller, R. and Myers, B., LAPIS: Smart Editing with Text Structures, CHI 2002 Extended Abstract, 496–497.

[16] Noy, D., Predicting User Intentions in Graphical User Interfaces Using Implicit Disambiguation, CHI 2001 Extended Abstract, 455–456.

[17] Rodgers, M. Mandryk, R. L. and Inkpen, K., Smart Sticky Widgets: Pseudo-haptic Enhancements for Multi-Monitor Displays, Smart Graphics 2006.

[18] Soukoreff, R. W. and MacKenzie, I. S. Towards a standard for pointing device evaluaton, perspectives on 27 years of Fitts' law research in HCI. Human-Computer Studies, 61:751-789, 2004.

[19] Worden, A. Walker, N. Bharat, K. and Hudson, S., Making Computers Easier for Older Adults to Use: Area Cursors and Sticky Icons, CHI 1997, 266–271.