Foundations for Designing and Evaluating User Interfaces Based on the Crossing Paradigm

GEORG APITZ and FRANÇOIS GUIMBRETIÈRE
HCIL, University of Maryland
and
SHUMIN ZHAI
IBM Almaden Research Center

Traditional graphical user interfaces have been designed with the desktop mouse in mind, a device well characterized by Fitts' law. Yet in recent years, hand-held devices and tablet personal computers using a pen (or fingers) as the primary mean of interaction have become more and more popular. These new interaction modalities have pushed the traditional focus on pointing to its limit. In this paper we explore whether a different paradigm—goal crossing-based on pen strokes—may substitute or complement pointing as another fundamental interaction method. First we describe a study in which we establish that goal crossing is dependent on an index of difficulty analogous to Fitts' law, and that in some settings, goal crossing completion time is shorter or comparable to pointing performance under the same index of difficulty. We then demonstrate the expressiveness of the crossing-based interaction paradigm by implementing CrossY, an application which only uses crossing for selecting commands. CrossY demonstrates that crossing-based interactions can be more expressive than the standard point and click approach. We also show how crossing-based interactions encourage the fluid composition of commands. Finally after observing that users' performance could be influenced by the general direction of travel, we report on the results of a study characterizing this effect. These latter results led us to propose a general guideline for dialog box interaction. Together, these results provide the foundation for the design of effective crossing-based interactions.

This work has been supported in part by Microsoft Research for the Microsoft Center for Interaction Design and Visualization at the University of Maryland and by NSF grant #0414699.
Early partial results of this work had appeared as conference publications Accot and Zhai [2002] and Apitz and Guimbretière [2004].
Authors' addresses: G. Apitz, Human-Computer Interaction Lab, University of Maryland, A.V. Williams Building, College Park, MD, 20742; email: geapi@cs.umd.edu; F. Guimbretière, Cornell University, Information Science, 301 College Avenue, Ithaca, NY, 14850; email: francois@cs.cornell.edu; S. Zhai, IBM Almaden Research Center, 650 Harry Road, San Jose, CA, 95120; email: zhai@almaden.ibm.com.

DOI 10.1145/1746259.1746263 http://doi.acm.org/10.1145/1746259.1746263

Categories and Subject Descriptors: H.5.2 [**Information Interfaces and Presentation**]: User interfaces—*Interaction styles*; H.1.2 [**Models and Principles**]: User/Machine Systems—*Human information processing*

General Terms: Human Factors, Performance, Theory

Additional Key Words and Phrases: Goal crossing, pointing, Fitts' law, graphical widgets, events, input, input performance, crossing-based interfaces, command composition, fluid interaction, pen-computing

ACM Reference Format:

Apitz, G., Guimbretière, F., and Zhai, S. 2010. Foundations for designing and evaluating user interfaces based on the crossing paradigm. ACM Trans. Comput.-Hum. Interact. 17, 2, Article 9 (May 2010), 42 pages.
DOI = 10.1145/1746259.1746263 http://doi.acm.org/10.1145/1746259.1746263

1. INTRODUCTION

Over the last few years, pen- and finger-based interactions have started to play an increasingly more important role in user interface design. In particular they have been the interactions of choice for a wide variety of devices from cell phone, to tablet computer, to table and wall interactive surfaces. Not surprisingly, these new modalities pushed the WIMP-interface (Windows, Icons, Menus, and Pointers) to its limit, since it was tuned to a typical desktop configuration. In that configuration, users control a cursor on the screen by moving a mouse on their desk. This indirect setting prevents any occlusion problems, while the accuracy of the mouse makes it easy to access very small targets such as the "close window" icon, double click, and the use of multiple buttons. Such interactions are often difficult to perform with a pen or a finger. Further, WIMP interfaces often rely extensively on keyboard shortcuts for expert interactions, making them slow to use on devices such as pen-based tablet computers. Noticing that pens are naturally suited to draw strokes, it seems natural to consider an interface paradigm in which the basic element of interaction is to draw strokes on a target to trigger an action. Here we are presenting an overview of our initial work on crossing-based interfaces.

The idea behind crossing-based interfaces is simple. Users cross a target on the screen to trigger the action associated with a given widget (Figure 1(b)) as opposed to clicking on a target. In this article, we first present a characterization of the overall performance of the crossing paradigm [Accot and Zhai 2002]. We show that crossing performance is well described by Fitts' law [Fitts 1954], and we present an analysis on how the relative performance of crossing-based interaction depends on factors such as the need to lift the input device between two targets or the orientation of the two targets with respect to the main direction of travel. Of particular interest are findings suggesting that crossing tasks can be as fast as, or faster than, clicking tasks of the same index of difficulty (ID).

We then demonstrate the expressiveness of the crossing paradigm, by describing CrossY [Apitz and Guimbretière 2004] a crossing-based drawing application. While examples of crossing-based interactions such as Lotus Notes [IBM 2004] and Baudisch's toggle map [Baudisch 1998] have been proposed

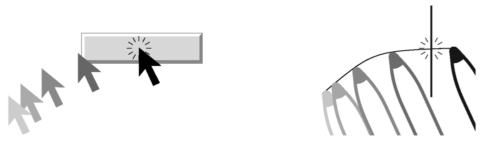


Fig. 1. Two different paradigms for triggering actions in a graphical user interface.

before, CrossY is the first application to rely exclusively on crossing to issue commands. Our work with CrossY demonstrates that crossing- based interfaces are more expressive than standard point and click interfaces. For example the interface can take into account the orientation and direction of the crossing stroke to distinguish between different types of commands on the same interface element. Crossing-based interaction also offers the option of composing several command invocations in a single stroke—something that is not possible in the point and click paradigm. Based on our experience building CrossY, we describe our initial recommendations on how to best leverage the unique aspects of crossing-based interactions.

During our work on CrossY we discovered that our original empirical characterization of crossing-based interactions needed refinement to better conceptualize the impact of layout on user performance. We conducted a follow-up study examining additional parameters of the crossing task, including target orientation in relation to movement direction. The empirical findings in this study provided additional guidelines and insights for crossing-based user interface design.

Together, this work lays the foundations needed by designers to design efficient crossing-based interfaces well suited to pen- and finger-based interactions.

2. PREVIOUS WORK

In the process of deriving the law of steering from Fitts' law, Accot and Zhai [1997] found that for a goal-crossing task the time to cross a goal can be determined by the goal distance and goal width. In fact, this relationship takes the same form as in Fitts' law. More precisely, both the time needed to reach and click on a target of width W that lies a distance D away (Figure 2(a)), and the time needed to cross a goal of width W that lies a distance D away (Figure 2(b)) are given by:

$$T = a + b \underbrace{\log_2\left(\frac{D}{W} + 1\right)}_{\text{Index of difficulty } (ID)}, \tag{1}$$

where a and b are experimentally determined performance constants. The logarithm factor in Equation (1) is called the index of difficulty (ID) of the pointing or crossing task.

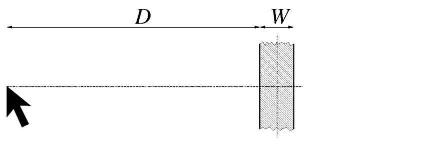
This model of goal crossing first presented in Accot and Zhai [1997] constitutes a necessary but not a sufficient foundation for studying and designing crossing-based interfaces. In order to establish crossing as a general paradigm of user interface design, more work is needed both in systematic theoretical analysis and in practical design exploration.

First, since the graphical targets to be crossed (goals) can be either orthogonal or collinear to the direction of pen's movement, it is important to understand the impact of target orientation to user performance. This issue has been at least indirectly addressed in traditional pointing tasks. Some Fitts' law research, such as Jagacinski and Monk [1985] or MacKenzie and Buxton [1992] and Hoffmann and Sheikh [1994], has tackled situations where the pointing target has constraints in two dimensions, implicitly incorporating directional error in pointing research. Most recently, in conjunction with the study presented here, one of us has conducted a study to refine Fitts' law to account for both directional and amplitude constraints more completely [Accot and Zhai 2003].

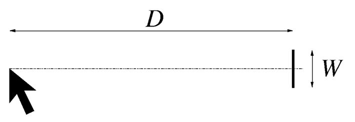
Previous work on crossing also goes beyond theoretical analysis. Although rare, crossing as an interaction method has been used in some commercial products. In Lotus Notes®, multiple emails can be selected by crossing their header in a designated area. Similar operations have been generalized to "area painting" as a way of turning multiple toggle switches in a research prototype by Baudisch [1998]. Researching interaction techniques based on a laser pointer, Winograd and Guimbretière [1999] proposed "gesture-and-sweep" instead of "point-and-click" as a selection technique in interactive rooms. Research has shown that point-and-click is particularly difficult with a laser pointer [Myers et al. 2002]. In the context of exploring pen-based interaction, Ren and Moriya [2000] studied entering and leaving a button as alternative strategies to clicking, and further pointed out the need for a theoretical model for studying these strategies.

Other examples that are well suited for pen use and show the application of crossing can be found in menu design. For example many pop-up menu systems are well adapted for pen-based interaction. Several systems, such as Pie Menu [Hopkins 1991] and Marking Menu [Kurtenbach 1993], use direction and pen-up transition to select commands. Other menu systems such as Control Menu [Pook et al. 2000] and FlowMenu [Guimbretière and Winograd 2000] use crossing as a way to select commands. More recently the Scriboli system [Hinckley et al. 2005, 2006] has studied the potential of self-crossing (to create a small pigtail) as a way to segment an ink stroke into a selection stroke and a command stroke.

In recent years, several systems also challenged the use of the point-and-click interface for whiteboard environments such as Tivoli [Pederson et al. 1993], FlatLand [Mynatt et al. 1999] and PostBrainstorm [Guimbretière et al. 2001], on the desktop [Ramos and Balakrishnan 2003], or for pen computing [Saund et al. 2003]. These systems are generally tuned to a certain class of applications (such as brainstorming, for example) and do not focus on crossing as the sole interaction paradigm. The work presented here, is an attempt to design a general pen centric-framework that will support such applications. Finally, several systems, such as SATIN [Hong and Landay 2000], have explored gesture-based interactions. Although, gestures are important to crossing-based



(a) Pointing: variability is allowed in the direction collinear to movement



(b) Crossing: variability is allowed in the direction orthogonal to movement

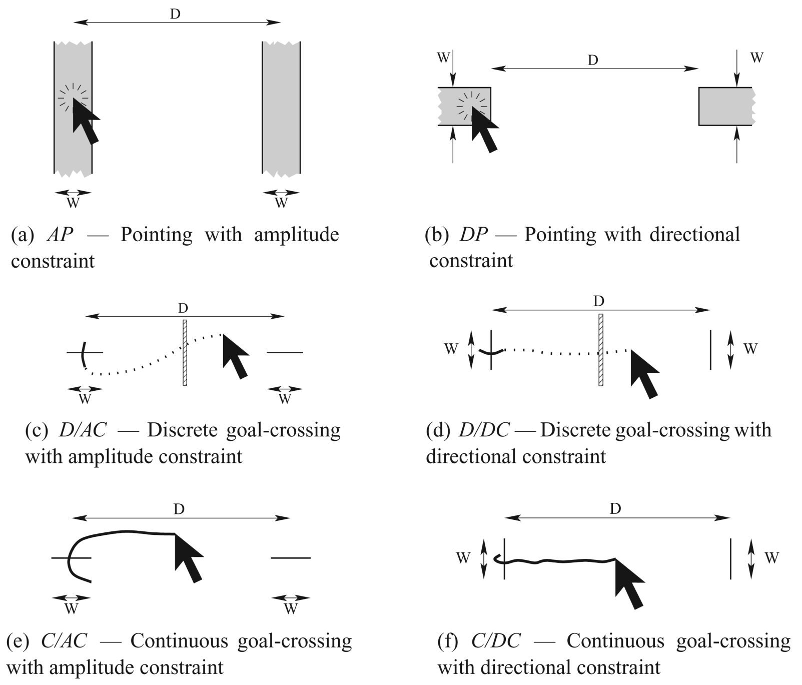
Fig. 2. Pointing vs. crossing: the experimental paradigms differ in the direction of the variability allowed in the termination of the movement.

interfaces, the gestures are relatively simple and, by adding a crossing requirement, ambiguity is reduced. In that respect, the requirements are similar to Gedrics [Geissler 1995], a system in which users can select the action performed by an icon by drawing a given gesture on top of it.

3. CROSSING AND POINTING WITH CONSTRAINTS

While Accot and Zhai (1997) have found that crossing-action follows a strong speed (time) and accuracy (size of the goal) tradeoff relationship (Equation (1), Figure 2), more modeling work is needed to systematically understand the difference between pointing and crossing. The two tasks in Figure 2 vary at least in two dimensions. One of the differences is pointing versus crossing. The other difference lies in the task precision constraint, one on the amplitude and the other on the direction of movement. When the orientation of precision constraint is *collinear* to movement, for either pointing (Figure 3(a)) or crossing (Figure 3(c)), the task performer has to primarily control the movement's amplitude. When the orientation of precision constraint is *orthogonal* to movement, for either pointing (Figure 3(b)) or crossing (Figure 3(d)) tasks, the performer has to primarily control the movement's direction. Because the two types of constraints could exist in both tasks, we cannot have a systematic understanding for crossing interfaces based on the study of Accot and Zhai [1997] alone (Figure 2). Instead, we need to understand the two types of action (pointing vs. crossing) under two types of constraint (amplitude and direction) in a factorial fashion.

Returning to the task of crossing, there is yet another task dimension that has to be considered if a study is to be truly systematic. Crossing can be done in two ways: either discretely or continuously. When there is nothing between the targeted goals, one can continuously stroke through these goals (continuous crossing, Figure 3(e),(f)). On the other hand, when there are nontarget objects (distractors) between the individual goals, one has to land the stylus (or finger) before an intended goal, cross it, and then lift up (discrete crossing, Figure 3(c),(d)). This fact is of highly practical interest. Imagine a widget that has several checkboxes, but the user only wants to select a subset of them. In this case discrete crossing or a combination of discrete and continuous crossing is necessary to give the user this flexibility. In addition to flexibility this aspect can be used to support an easy transition from novice to expert users. Novices can cross targets and issue commands in a discrete way while more advanced



(a) *AP* — Pointing with amplitude constraint

(b) *DP* — Pointing with directional constraint

(c) *D/AC* — Discrete goal-crossing with amplitude constraint

(d) *D/DC* — Discrete goal-crossing with directional constraint

(e) *C/AC* — Continuous goal-crossing with amplitude constraint

(f) *C/DC* — Continuous goal-crossing with directional constraint

Fig. 3. The six conditions tested. All tasks were reciprocal. Dotted lines indicate that the pen is not in contact with the tracking surface, for example to avoid a distracter represented by a stripped area in (c) and (d).

users can use single strokes to issue several commands and remember the shapes of the strokes that are necessary to achieve these commands. We do not see this difference in a point-and-click setting since it has to be discrete. Continuous crossing has another potential advantage which is the combination of commands. If the user is able to cross several targets in one stroke, it is also possible to issue several commands this way. We therefore investigated both discrete and continuous crossing in a systematic experiment as well as in our prototype application. There we give users the choice to issue commands either discretely or continuously. Finally, while we are only considering targets placed on the horizontal axis in this section, it is clear that this will not be always be the case in practice. We will consider the potential impact of the general direction of travel in Section 5.

Another dimension of potential interest is the input device. In principle, crossing can be done with any input device, such as a mouse, but a pen (stylus) is the most obvious choice as a crossing input device. Thus, our crossing-based application was built for a pen-based setting, where users execute all interactions with a pen without having a keyboard or mouse at their disposal.

3.1 Experiment

We conducted an experiment involving all 6 tasks depicted in Figure 3. The two pointing tasks differed in the movement precision constraint—one directional and the other amplitude. The four goal-crossing tasks differed both in movement precision constraint (directional/amplitude) and in the nature of the action (discrete/continuous). The details of each condition are as follows.

AP: Pointing with amplitude constraint (Figure 3(a)). This is the traditional Fitts' tapping task [Fitts 1954]. Participants click alternately on two vertical rectangles with width W and "infinite" height. The two target centers are separated by distance D. We call this task amplitude pointing because the movement precision (or variability) constraint (W) is imposed on the movement amplitude.

DP: Pointing with directional constraint (Figure 3(b)). This is a variant of Fitts' original tapping task. Participants click alternately on two horizontal rectangles of height W and "infinite" width (to one side), separated by distance D measured by the gap between the two targets. The precision constraint in this task is imposed on the movement's direction.

D/AC: Discrete crossing with amplitude constraint (Figure 3(c)). Participants alternately cross, by a stroke, two horizontal goals of width W and distance D. For consistency, they are asked to perform the stroke downward for both goals. This crossing task is discrete since the stylus tip touches the tablet surface only when crossing the goal; the rest of the time the stylus is lifted from the tablet surface. An obstacle line, causing a beep when stroked through, is drawn between the two goals to remind the participants to use discrete strokes for crossing the goals.

D/DC: Discrete crossing with directional constraint (Figure 3(d)). Participants alternately stroke through two vertical goals with height W and distance D. They are asked to cross the goals from left to right for consistency. As in the previous condition, an obstacle, that is, a distractor, is drawn between the two goals to remind participants to lift up the stylus when traveling from one goal to the other.

C/AC: Continuous crossing with amplitude constraint (Figure 3(e)). Participants alternately move the cursor through two horizontal goals of width W over distance D. The crossing task is continuous as participants have to constantly slide the stylus tip on the tablet surface. If the stylus is lifted during a block of trials, the system will beep until stylus-tablet contact is resumed. Since the user has to keep the stylus on the screen one stroke is used to cross both targets which is not possible with point-and-click. Note that the setting does not make use of the stroke direction for an easier crossing since the targets are aligned against the crossing direction.

C/DC: Continuous crossing with directional constraint (Figure 3(f)). Participants move the cursor reciprocally through two vertical goals of height W over distance D. As in condition C/AC, the system will beep when the stylus is lifted during a block of trials until stylus-tablet contact is resumed. This task was introduced in Accot and Zhai [1997] and found to follow Fitts' law when performed nonreciprocally.

In all six tasks, participants were asked to perform as fast and as accurately as possible. When a target (or goal) was missed, a beep was played to remind the participant to improve accuracy. In case of a miss, participants continued the trial until they hit the target and their trial completion time increased as a result. When hit, the target changed color from green to orange. The time duration between the two alternate target clicks (or two goal crosses) was recorded as the trial completion time.

3.1.1 *Experimental Design and Participants.* A within-subject full factorial design with repeated measures was used. The independent variables were the task type (T = AP, DP, D/AC, D/DC, C/AC, C/DC), the distance D between targets or goals ($D = 256, 1024$ pixels) and the target/goal width ($W = 8, 16, 32, 64, 128$ pixels). For each task, participants performed three consecutive sets of 10 D-W combinations, the first set being a practice session and the later two, data collection sessions. The ten D-W combinations were presented in a random order within each session. With each D-W combination, participants performed a block of 9 trials. The order of testing of the six different tasks was balanced among six groups of participants according to a Latin square.

Twelve people, three female and nine male, all right-handed, participated in the experiment. They ranged in age from 21 to 51.

3.1.2 *Apparatus.* The experiment was conducted on an IBM PC running Linux, equipped with a Wacom® Intuos™ graphics tablet (model GD-0608-U, 20.3 cm × 15.2 cm active area, 2540lpi resolution) and a 19" IBM CRT monitor (model P76, 32 cm × 24 cm visual area, 127 dpi resolution). The tablet active area was mapped onto the display visual area, in absolute mode; the control gain was close to 1.6. The experiment was done in full-screen mode, with a black background color. The computer ran in single-user mode, with only a few active system processes. It was disconnected from the network.

3.2 Results and Analyses

3.2.1 *Learning, Time, and Error.* Figure 4 shows the average trial time over the three experimental sessions. The average trial completion time in the practice session was longer than the time in the two data-collection sessions, due to participants' inexperience and occasional experimentation with the tablet-stylus device and task strategy. The performance difference between the two data-collection sessions was relatively small, and hence both were used in the following data analyses.

Analysis of variance showed that mean trial completion times were significantly different across the six tasks ($F_{5,55} = 12.5$, $p < .001$). Task C/AC was the slowest (see Figure 4). Tasks DP and C/DC were the fastest, at least 10% faster than other tasks. The rest of the tasks, including the traditional Fitts' tapping task, fall in the middle range of performance.

Statistically, Fisher's PLSD test shows that each pair of tasks was significantly different from each other ($p < .05$), except DP vs. C/DC ($p = .52$), and AP vs. D/AC.

As illustrated in Figure 5(a), movement distance significantly changed mean trial completion time ($F_{1,11} = 898$, $p < .0001$). Across all tasks, the greater

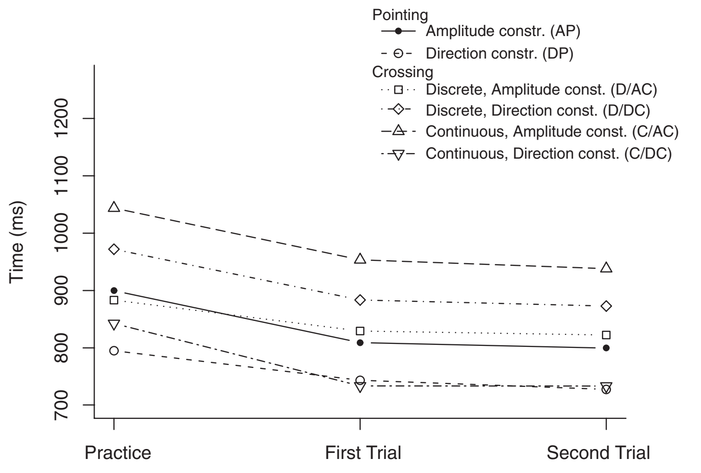


Fig. 4. Learning effect on average completion time.

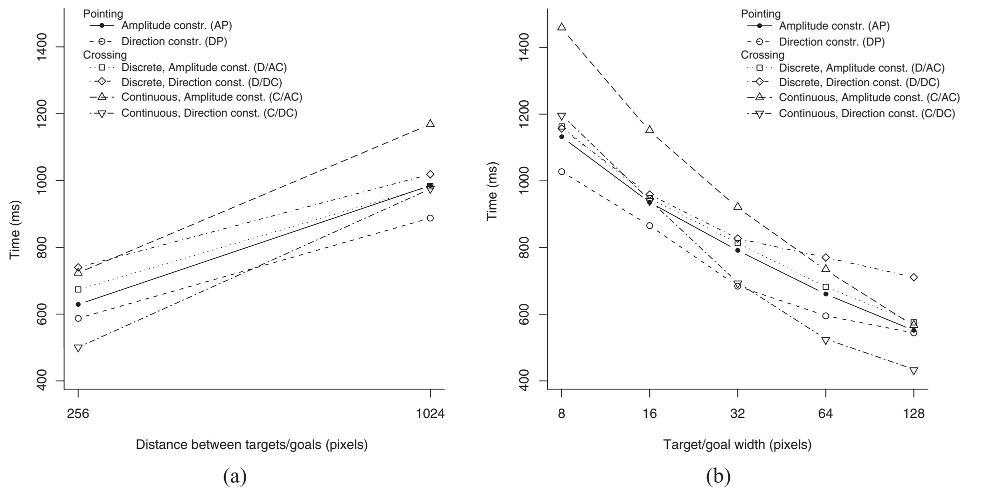


Fig. 5. Effect of distance and width on task completion time.

the distance between targets, the longer the time duration of the trial. As illustrated in Figure 5(b), target/goal width also significantly changed mean trial completion ($F_{4,44} = 485$, $p < .0001$). For all tasks, the greater the width of the target, the shorter the duration of the trial.

The error rate, measured by the percentage of trials that took more than one click or crossing to hit the target, varied significantly with task ($F_{5,55} = 7.76$, $p < .0001$), target distance ($F_{1,11} = 16.6$, $p < .01$) and target width ($F_{4,44} = 38.2$, $p < .0001$). As expected, smaller and more distant targets tended to cause more errors. As shown in Figure 6, except for Task C/AC (9.2%), all new tasks studied in this experiment had error rates close to and lower than that of Fitts' tapping task (AP, 7.6%). D/DC has the lowest error rate with 2.8%.

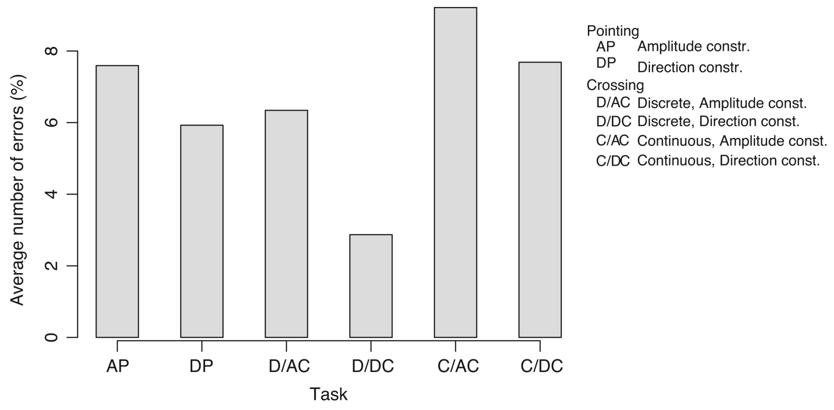


Fig. 6. Error rates for each task.

When participants made an error in a trial (missing the target), they were asked to continue the trial until they hit the target, with multiple clicks or multiple crossing attempts. The completion time of these trials does not reflect the same perceptual motor mechanism as successful trials without error; hence we did not include them in the time analysis in this section. As an extra caution for the robustness of the conclusions, we also repeated all time-related analyses with the error trial completion times included, but found no important or qualitative differences from the conclusions reported in this section.

3.2.2 *Lawful Regularities.* Most interestingly, the movement time in each and every of the six tasks could be largely accounted for by the target-distance to target-width ratio. More precisely, the difficulty in each task can be qualified by the following common index:

$$ID = \log_2\left(\frac{D}{W} + 1\right) \tag{2}$$

and the movement time can be determined by:

$$T = a + b\, ID, \tag{3}$$

where a and b are empirically determined in each task. Specifically, linear regression of the experimental data resulted in the following equations (in milliseconds):

$$AP : T = 103 + 172 \times IDr^2 = 0.998 \tag{4}$$
$$DP : T = 145 + 146 \times IDr^2 = 0.986 \tag{5}$$
$$D/AC : T = 155 + 165 \times IDr^2 = 0.994 \tag{6}$$
$$D/DC : T = 342 + 133 \times IDr^2 = 0.975 \tag{7}$$
$$C/AC : T = -41 + 242 \times IDr^2 = 0.995 \tag{8}$$
$$C/DC : T = -196 + 235 \times IDr^2 = 0.984. \tag{9}$$

In other words, there was a lawful regularity between movement time and D/W ratio in each of the six tasks. Furthermore, all six laws take the same

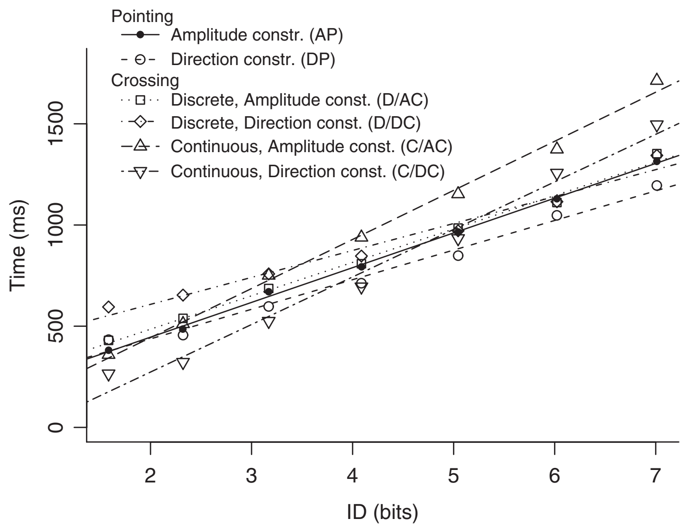


Fig. 7. Model fit for the six studied tasks.

form of logarithmic transformation of D/W ratio as in Fitts' law, with very high fitness values (r^2 ranging from .975 to .998). If we name laws by their mathematical appearance, we can say that crossing also follows Fitts' law—the law of pointing. If we name performance regularities by the tasks they describe, we can say there indeed exists a "law of crossing."

Figure 7 displays the regression lines of completion time as a function of *ID*. As we can see, despite the very different constraints and varying action patterns across the six tasks, the laws of these tasks all fall into a band that is even narrower than the range of data based on the same Fitts' tapping task reported by different authors in the literature [MacKenzie 1992; Zhai 2004].

3.2.3 *Task Comparison.* Although relatively small, there were important differences between the six tasks studied in this experiment. Instead of analyzing all pairs of comparison, we focus on a few comparisons most relevant to human-computer interaction tasks (illustrated by Figure 8).

First, Task DP, that is, pointing with directional constraint, not only followed Fitts' law closely ($r^2 = .986$), but also exhibited similar performance to Task AP, that is, pointing with amplitude constraint (the traditional Fitts' tapping task). On average, DP was 10% faster than AP (Figure 4). As displayed separately in Figure 8(a), starting from essentially zero, the difference between DP and AP increased with Index of Difficulty. The greater the *ID* (either smaller width or greater distance), the more pronounced this effect (see AP and DP lines in Figure 5(a)). Apparently it is "easier" to control directional error than amplitude error in pointing tasks. This is plausible because the former can be dealt with in the entire course of movement, while the latter can only be controlled at the very end of the movement. This is also consistent with previous findings in motor impulse variability research [Schmidt et al. 1979] which has demonstrated that directional variability is about half of amplitude variability in open-loop

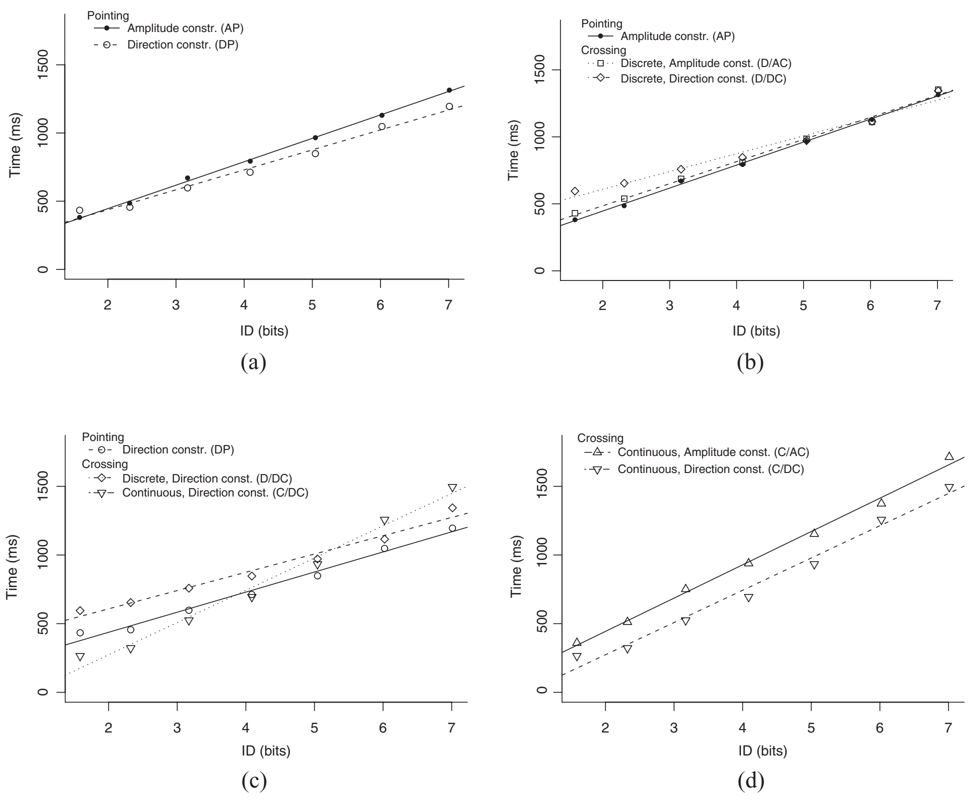
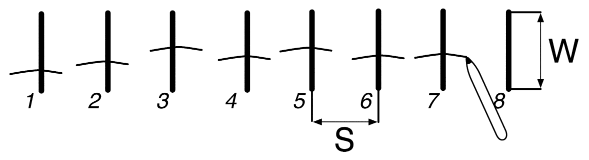


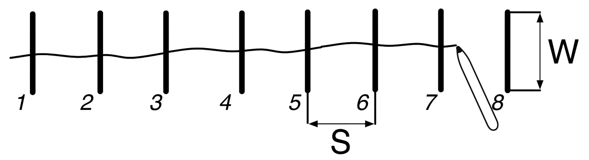
Fig. 8. Tasks comparisons.

reaching tasks. With closed-loop correction, the difference in this study is much smaller but the pattern is consistent.

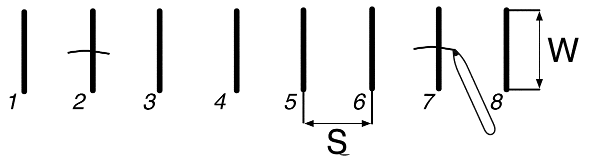
Second, the discrete crossing task with amplitude constraint and the discrete crossing task with directional constraint followed similar regression lines to the standard Fitts' tapping task, as separately shown in Figure 8(b). This suggests that it is possible to substitute pointing tasks with crossing tasks with essentially the same time but lower error rate (see Figure 6). This is a strong theoretical basis for a coexistence of pointing and crossing actions in the same user interface, since any selection task that can be accomplished by pointing can also be accomplished by crossing with the same, or shorter amount of time. Other advantages of crossing emerge as the user becomes more proficient with crossing actions. Between the two discrete crossing tasks, there was a trade-off swing by *ID*. D/DC tended to be faster than D/AC when *ID* was greater than 6 bits, and the reverse was true when *ID* was less than 5 bits. The fact that D/DC was slower than D/AC in low *ID* can be partially explained by the "obstacle" line positioned between the two goals (see Figures 3(c) and 3(d)). As the distance between the two goals reduces, the constraint of landing the stylus between the obstacle and the right goal increases. Implicitly there is a traditional Fitts' tapping task (AP) involved here with increasing difficulty. This is not true for the D/AC task because the stylus always lands above the



(a) Crossing all targets discretely.



(b) Crossing all targets in one stroke.



(c) Crossing only targets 2 and 7.

Fig. 9. Three scenarios of crossing multiple parallel targets.

horizontal goals. This distractor effect reflects realistic situations in computer interfaces where the target object can be surrounded by objects that are not targets at the moment.

Another large difference between the two discrete crossing tasks was that a much smaller number of errors was made with the D/DC task than with the D/AC task (Figure 6). This becomes an important aspect when designing low error rate layouts.

Third, there was a trade-off between C/DC and D/DC tasks (Figure 8(c)). C/DC was faster than D/DC when *ID* was lower than 5 bits, and C/DC was longer than D/DC when *ID* was greater than 5 bits. This again was partly due to the obstacle effect in the D/DC task. There was no obstacle in the C/DC task. Continuous strokes give better performance if the ID is kept low and obstacles are avoided. When the two goals were very close to each other(Figure 5(a)), it was possible to cross two goals in one stroke with C/DC. For D/DC, on the other hand, closer goals make it more difficult to place the stylus between the obstacle and the goal to be crossed. It is also conceivable that longer distance would give D/DC an advantage because the lifted stylus might travel faster than continuously sliding the stylus on the tablet surface in the D/DC condition. However, such a possibility is not supported by the data (Figure 5(a)).

Fourth, the C/AC task was uniformly slower than the C/DC task (Figure 8(d)). It also has the highest error rate (Figure 6). This suggests that, whenever possible, the goals to be crossed continuously should be positioned orthogonal to the movement direction.

We now consider a practical example, shown in Figure 9. In this case, a series of vertical goals of size W are horizontally laid-out in parallel with space S between them (the same analysis applies to a series of vertically laid-out goals). If the task is to select all of them at once, users might select them one by one using a series of small strokes (Figure 9(a)). This approach is similar to the discrete goal crossing with directional constraint (D/DC) condition of our experiment. Users might also decide to select them with one continuous stroke (Figure 9(b)). This is similar to the continuous goal crossing with directional constraint (C/DC) condition of our experiment. As shown before (see Figure 8(c)), the discontinuous approach will perform better if $log_2(\frac{S}{W}+1)$ is greater than about 5 or S is greater than 31W. It is important to note also that when $log_2(\frac{S}{W}+1)$ is smaller than 4 (see Figure 8(c)) or S smaller than 15W, continuous crossing, is not only faster than discontinuous crossing, but also faster than consecutively clicking on a series of targets of width W. The fact that continuous crossing is faster than (discrete) clicking in specific settings has been leveraged by Lotus Notes, which lets users select a series of continuous emails by continuously crossing check boxes.

Of course, users might also need to select non-continuous goals, for example, goal 2 and 7. In that case, the goals in-between are distractors or obstacles (Figure 9(c)). Then, the only crossing strategy one can employ is discrete crossing (D/DC). Of course, in order to cross goal 7, one first has to land between goal 6 and 7, a task similar to the pointing with amplitude constraint condition (AP) of our experiment. So the analysis of this task requires considering both a D/DC aspect with $ID_{D/DC} = log_2(\frac{5S}{W}+1)$ and an AP aspect with $ID_{AP} = log_2(\frac{4S}{S}+1)$. When S is greater than 0.8W, the impact of the obstacle created by the other crossing target(s) is less important. When S is smaller than 0.8W, the AP aspect becomes the dominant factor and the entire task is effectively an AP task with target width of S. In practice, if we consider a quite common situation in which each target is 5 characters long with one space at the end (or S $\approx$ 6W), then $ID_{D/DC} > 2ID_{AP}$. This means that in general, the constraints imposed by the D/DC aspect are predominant for design purposes.

4. APPLYING THE RESULTS TO BUILD AN APPLICATION

Drawing from the information and experience described thus far, we developed a prototype application, CrossY, that gives insights about how a real life crossing-based application can be designed. The main goal was to create an application for which all interface elements (including menus, buttons, scrollbars, and dialog boxes) rely solely on crossing (Figure 10).

CrossY not only demonstrates the feasibility of crossing as an interaction paradigm in a real life application, it also provides initial feedback on the unique challenges of developing such a crossing-based interface. We found that crossing is well adapted to both pen-based and mouse-based interactions, it is more expressive than the equivalent point-and-click interfaces, and it encourages a fluid composition of commands. We also found that, to leverage this latter advantage, special consideration of the interface layout is required.

4.1 Requirements for a Crossing-Based Application

Based on the study reported above (see Section 3.2.3), we found that the target for any crossing interaction should be orthogonal to the crossing direction.

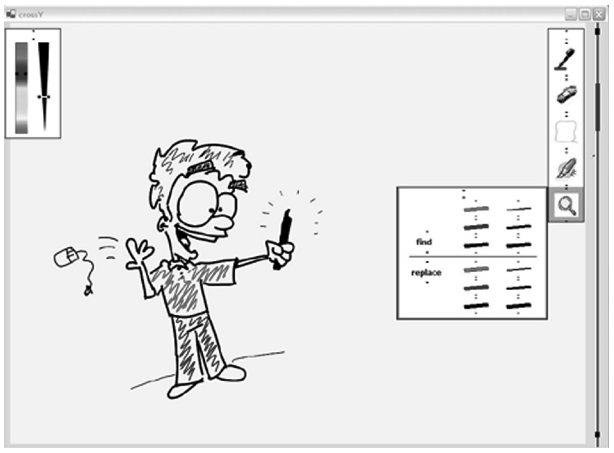


Fig. 10. A screenshot of the CrossY interface.

Since we assume a general horizontal flow of interaction, the alignment of the crossing target had to be vertical. Furthermore, as expected from Fitts' law and confirmed with the "law of crossing" in section 3.2.2, the distance and size of crossing targets was a crucial aspect as well. Based on these insights, we decided to use a low ID for the initial design to minimize the error rate and to provide the users with a satisfying experience.

Expressiveness. One of the most important questions to be addressed is: Can the new language express as rich a set of features as the language it means to replace? Therefore, we decided to examine how the key elements of a basic WIMP interaction can be implemented in a crossing interface. As a starting point, we decided to implement standard buttons, scrollbars, menu systems, dialog boxes (including selection of items from a list) and a simple set of window management tools. In each case, our initial goal was to mimic existing capabilities before developing new features.

Fluid composition of commands. As illustrated by Lotus Notes [IBM 2004] and the toggle maps system [Baudisch 1998] interfaces based on goal crossing promote the fluid, composition of commands. This allows users to issue several actions (e.g., selecting among a group of toggle switches) in one single stroke. As shown by systems such as Marking Menu [Kurtenbach 1993], and the SHARK system [Zhai and Kristensson 2003] there are advantages to encouraging transition from a a visual interface to a gesture-based interface. Thus our goal was to determine if crossing based interactions could extend this benefit to a wider set of interactions such as a search and replace task.

We also examined if the advantages of transitioning from a visual interface to a gesture-based interface (as demonstrated in the Marking Menu [Kurtenbach 1993]) could be extended to the selection of several commands inside a dialog box using continuous crossing.

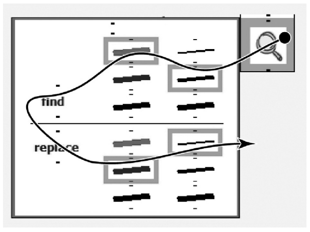


Fig. 11. Selecting the values for find and replace and applying the command in a continuous stroke.

Efficiency. Expressiveness and fluidity are of little use if they come at the price of an inefficient interface. Therefore, efficiency was an important consideration during the design process.

Visual footprint. Screen real estate is a valuable resource and the new interaction language needs to use it efficiently. Crossing-based interfaces are unique, since the visual layout affects efficiency when a user selects several commands in one stroke. For the discrete case the design is as important for crossing as it is for pointing.

4.2 CrossY in Detail

CrossY[1] is a simple sketching program offering several tools (e.g., a pen, a highlighter, an eraser). It was designed to run on the Tablet-PC platform without a keyboard. CrossY allows users to modify tool attributes. It also offers a simple search-and-replace feature which lets users find strokes based on their attributes (color and thickness) and replace them. Although this drawing system is primitive by today's standards, CrossY demonstrates how most of the standard widgets of point-and-click interfaces can be implemented in a goal crossing framework.

4.3 Crossing Direction

In contrast to point-and-click designs, the direction of the crossing matters and can hold information. In the case of CrossY, we decided to use right to left as the main crossing direction to minimize hand occlusion for right handed people. There is however one exception: we reverse the stroke direction when continuous strokes are used for command combinations. An example of this is our find-and-replace dialog where the crossing direction is reversed for some actions to allow for the composition of commands. In other words, in the upper part of this dialogue box the selections are made by crossing from right to left and in the lower part the selections are made by crossing from left to right. This enables users to select the stroke width and color to find, issue the find and replace commands and select the replacement values in one stroke that is shaped roughly like a "c," see Figure 11.

[1]Although the descriptions in the following sections on CrossY are self-contained, a video demonstration of CrossY is available at http://www.cs.umd.edu/hcil/crossy/.

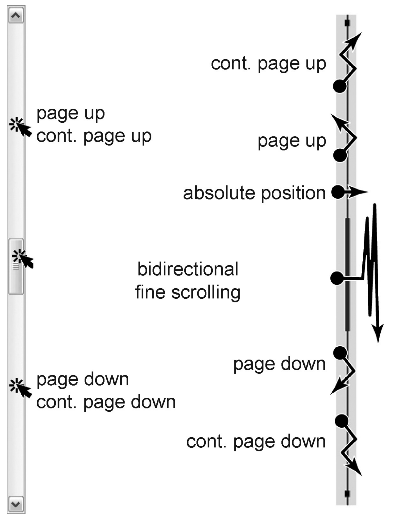


Fig. 12. A traditional scrollbar and the crossbar compared.

4.3.1 *Command Selection.* Like many drawing applications, the CrossY interface implements two kinds of menu systems. Common tools are accessed through a tool palette placed on the right side of the display (Figure 10) and all the crossing targets except for the checkboxes are vertical to allow a horizontal crossing direction. This layout was adopted to limit potential hand occlusion. CrossY offers five basic tools to choose from: a pen, an eraser, a lasso, a highlighter, and a search tool. Each of these tools can be selected by simply crossing its icon from right to left. Users can also move the palette to a more convenient place. To do so, users cross the center of the title bar between the two black marks from left to right. This action starts the dragging interaction which will stop as soon as the pen is lifted from the screen. Crossing the same area from right to left brings the palette back in its original position. This behavior is present for all palettes. In addition, CrossY uses FlowMenu as the primary command selection mechanism to control the application. This includes commands for lasso, open a file, save the current file, and quit the application.

4.3.2 *Navigating within the Document.* Users navigate the document with a crossbar, the equivalent of the standard scrollbar shown in Figure 12. The crossbar looks like a simple bar spanning the length of the document viewport and shows the current location inside the document. To interact with it, users perform gestures crossing the bar. Most of these gestures consist of single strokes to support the notion of continuous stroking. We provide several standard features such as page up and page down. These commands are triggered by open triangles drawn on top of the crossbar in the direction of the desired movement (see Figure 12). To start a continuous page down or page up, the user simply crosses the bar a third time after issuing the initial command. The document now scrolls continuously until the pen is lifted. To jump to a specific position inside the document, the user crosses the bar in the vicinity of the

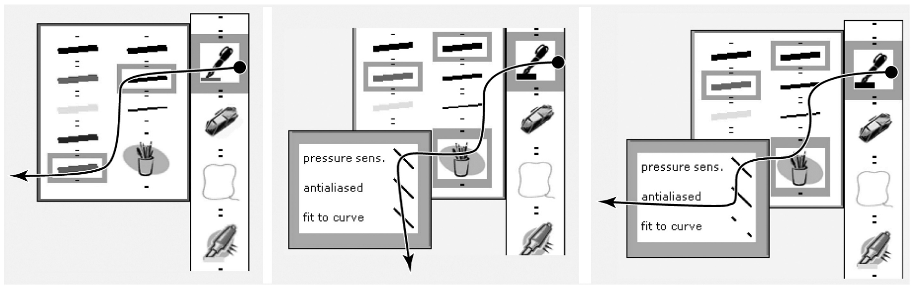


Fig. 13. Left: The CrossY palette with the pen-panel opened. A single stroke opens the pen-panel, selects width and color of the strokes, and validates the selection. By convention, the left and bottom edges of each dialog box are validation edges (colored green), and the top and right edges are cancellation edges (colored red). Middle: The dialog box with check boxes to set the stroke-rendering attributes. A single stroke selects all items. Right: The dialog box with the check boxes to set the stroke-rendering attributes. A single stroke selects only two items.

target location and then finely adjusts the position by simple dragging motions on the right side of the bar. Because absolute access and adjustment are now two different parts of the same interaction, it is possible to provide a different gain for both phases, enabling either precision or speed amplification similar in effects to those techniques explored by Albinsson and Zhai [2003]. While the initial gain is defined by the ratio of the document length to the scroll-bar length, the gain can be reduced during the adjustment phase to allow for finer adjustments, by moving the cursor further away from the crossbar. While some experimental scrollbars such as the FineSlider [Masui et al. 1995] provide similar options, the fluid integration of the two phases is typically difficult to achieve in a point-and-click interface. Another advantage of the crossbar is that users are not required to reach a given area of the bar before interacting. For example, they can initiate scrolling commands anywhere on the scrolling area. They also do not need to acquire the crossbar's slider before moving to an absolute position in the document; they just need to cross the crossbar at the target position. This makes the scrolling process faster and reduces the reliance on visual feedback.

4.3.3 *Selecting Pen Attributes.* In CrossY, users can select pen attributes by using either the pen attribute dialog box or the brush palette.

The Pen attribute dialog box is opened by crossing the pen tool button and extending the stroke towards the left over the edge of the button. Unlike current implementations, which present "dual-use" in a tool palette (such as in Adobe Illustrator [Adobe 2005]), our implementation does not force the user to dwell over the button to access the extended features. This increases the fluidity of the interaction and promotes chunking. The pen attribute dialog box is presented in Figure 13, left. It contains a set of crossing-based radio buttons used to select the size and color of the stroke. The layout of the buttons and radio buttons was a result of the findings from the experiment that showed an advantage of crossing targets aligned orthogonally to the pen movement for the discrete and continuous tasks (C/DC, D/DC see Figure 8). Thus, radio buttons

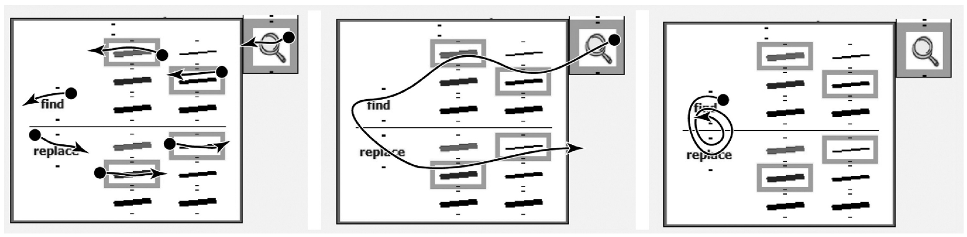


Fig. 14. Left: Separate strokes are shown in the find-and-replace dialog box. Users select the values for the target stroke in the upper panel and the replacement values in the lower panel. Middle: The separate strokes are combined into one single stroke. Right: Repeated find-and-replace operations are carried out with one continuous stroke.

are designed such that crossing along the horizontal axis of the label (in either direction) will toggle the button. This feature is not only a performance choice but also reinforces the notion that radio buttons represent exclusive choices (Figure 13, left). For check boxes, on the other hand, the vertically aligned layout would work against the fact that they are not mutually exclusive and make it more difficult to select several of them at once in a continuous stroke. Aligning them horizontally would make single choices difficult. Therefore, we decided to use tilted crossing targets for check boxes, this way the user can cross either horizontally to make just a single selection or vertically to make several selections (Figure 13, middle). Making a stroke that starts vertically and ends horizontally or similar combinations allow the selection of a subset of check boxes (Figure 13, right). An unusual aspect of the dialog boxes presented in Figure 13 is that they do not seem to include an OK/Cancel mechanism. The corresponding buttons are in fact very close to the edge of the window. Both the bottom and left border are validating borders (shown in green in our implementation), while the top and right border are cancellation borders (shown in red in our implementation). This layout lets users select all relevant options and validate the selection in one stroke. Even though “the single stroke” feature is provided, the user can still use discrete strokes to make selections and confirm or cancel.

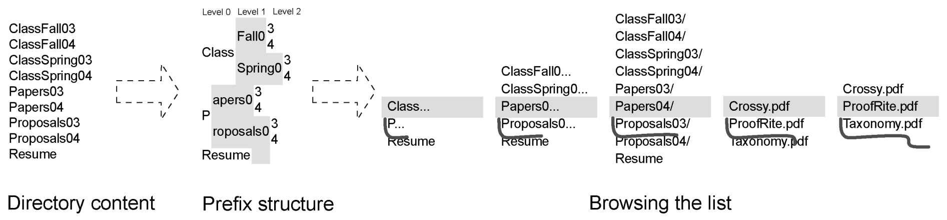
4.3.4 *Finding and Replacing Stroke Attributes.* Our application also provides a simple “find-and-replace” function which lets users change the attributes of some strokes on the screen. The function is accessible through a dialog box which is structured around two panels (Figure 14). On the top panel, the user can select the width and color of the target strokes using a set of radio buttons. On the bottom panel, the user can select the new width and color for the selected strokes. After setting the target attributes, the user can find the next stroke forward by crossing the “find” button from right to left. Similarly, replacement is triggered by crossing the “replace” button from left to right (Figure 14, left). While this layout seems somewhat unusual, it has been selected to encourage command composition. For example a user can in one single gesture select “medium” and “red”, cross the “find” button to find the first occurrence of this type of line, cross the “replace” button to indicate the need for replacement, and select “blue” and “thin” as the replacement values

(Figure 14, middle). The command is executed as the pen is lifted from the panel. Once the parameters have been correctly selected, there is no need to reselect them, and a simple circular motion between the “find” and “replace” button will trigger the replacement (Figure 14, right). It is also easy to skip some replacements by only circling around the “find” button without crossing the “replace” button. Backwards search is provided by crossing the “find” button from left to right. An undo for replacements is achieved by crossing the “replace” button from right to left. Since there are no distracters between the crossing targets the performance of the continuous stroke is likely to be better than of single strokes since the overhead of lifting and putting the pen back on the screen is avoided. This is supported by the experimental results from Section 3.2.2.

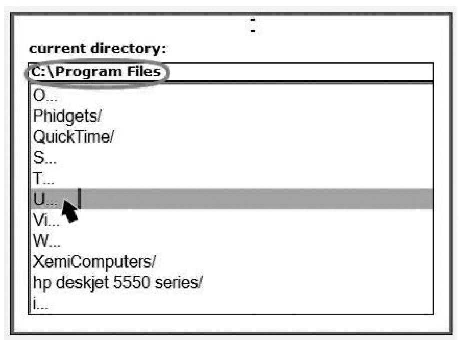
4.3.5 *Loading an Existing Drawing.* The file dialog box (Figure 15(b–d)) is called up through a FlowMenu. It lets users navigate the file system and load an existing drawing. At first glance, using crossing to navigate the file system hierarchy seems like a challenge since current interfaces rely heavily on the use of sequential point-and-click operations for this function. In traditional navigation systems, users first have to search through the list of files that is present at the current level. This is typically achieved by using the scrollbar tab for coarse adjustment and the arrow at the end of the scrollbar for line by line movement. Next, users have to select the next directory (or the target file) by double-clicking on its name. For directories containing a large number of items, this method can be quite cumbersome and is far less efficient than a text based interface with auto-completion enabled. We believe that the crossing paradigm provides ways to combine the convenience of the graphical interface with the speed of auto-completion.

In our directory navigation tool, the local directory is scanned and its contents are parsed into a hierarchy of display levels. Exploration of the file hierarchy works on the same basis as auto-completion in text-based systems. At the first level, we include all the names which are unambiguous (i.e., which do not share a common prefix with any other name) as well as the maximum common prefixes for all other names in the directory. For example in Figure 15(a) top,“ClassFall03,” “ClassFall04,” “ClassSpring04,” and “ClassSpring04” will be represented by their common prefix “Class...,” “Papers03,” “Papers04,” “Proposal03,” and “Proposal04” are represented by their common prefix “P...,” and “Resume” is presented as is since it is not ambiguous.

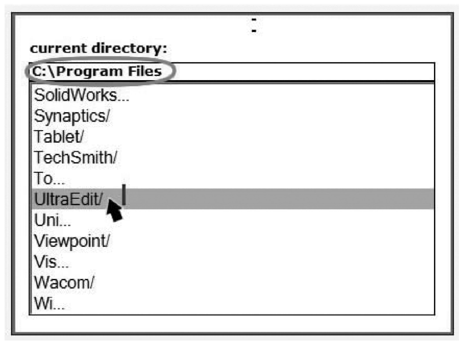
Exploration is performed by expansion of successive prefixes as users move through successive levels. For each prefix, we add the list of unambiguous names and maximum common prefixes derived from that prefix by adding in turn all possible letters following this prefix (see Figure 15(a), left/middle). It is important to note that there are only a limited set of possible characters (256 in theory but far less in practice) that may follow a given prefix. As a consequence, moving from one level to the next only adds a small number of new options for each prefix (often less than ten). Yet, assuming an average of 10 new words per prefix, after crossing only 3 levels 1000 elements can be accessed. Once created, this hierarchy can be navigated as follows (Figure 15(a), right): At all times,



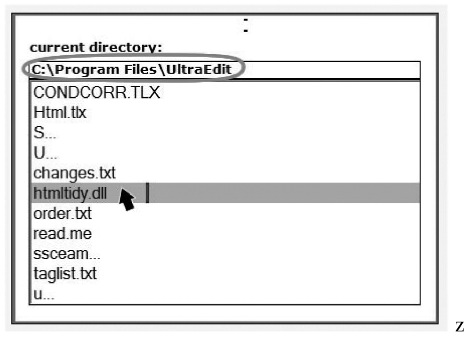
(a) Exploring a directory. Left: The directory content. Middle: The corresponding prefix structure. Right: Navigating through a directory to open the file Papers03/ProofRite.pdf. (The black line shows the pen movement for the described selection.) First, the prefix P is set in focus by moving the pen downwards and expanded by moving the pen to the right. Second, the prefix Papers0... is expanded by moving the pen right. Next, by moving the pen to the right the Papers04 directory is opened. Finally, with an downwards movement ProofRite.pdf is selected and opened with a rightwards movement.


current directory:
C:\Program Files
O...
Phidgets/
QuickTime/
S...
T...
U...
Vi...
W...
XemiComputers/
hp deskjet 5550 series/
i...

(b) The user is in the directory "Program Files" (seen at the top, circled) and "U" is selected (the gray line highlights the selection), the arrow shows the cursor position, the small vertical line indicates the crossing target upon crossing of which the next level in the file hierarchy is entered.


current directory:
C:\Program Files
SolidWorks...
Synaptics/
Tablet/
TechSmith/
To...
UltraEdit/
Uni...
Viewpoint/
Vis...
Wacom/
Wi...

(c) The user crossed the small line and expanded the prefix "U, " is still in the same directory and has "UltraEdit" selected.


current directory:
C:\Program Files\UltraEdit
CONDCORR.TLX
Html.tlx
S...
U...
changes.txt
htmltidy.dll
order.txt
read.me
ssceam...
taglist.txt
u...
z

(d) The small target was crossed again and the user switched the directory to "UltraEdit" (circled at the top), a next crossing of the small line would open "htmltidy.dll."

Fig. 15. The file dialog box in CrossY.

the currently selected item is presented highlighted at the center of the widget. Users can change the currently selected item by moving the pen up and down anywhere on the widget. To move one level downward in the display hierarchy, users simply make a left-to-right horizontal movement in the current gesture. This causes the current highlighted prefix (represented with an ellipsis, e.g., "P...") to extend one level. A movement to the right while an unambiguous name is selected, loads the corresponding directory or file. To move one level upward in the display hierarchy, users need to make a small right-to-left horizontal movement in their gesture. Going upward at the root display level loads the parent directory. During navigation, feedback is provided in several ways: when the user starts a horizontal segment, a crossing goal is displayed in form of a little bar indicating the point at which the transition to the next level will be triggered. This feedback is mostly useful for the novice. For more expert users, we also provide a "click" sound each time a transition between levels occurs and a "select" sound each time a directory (or a file) is selected. To distinguish between files and directories, we display a slash at the end of directory names.

This system is very efficient to navigate through large directory structures given that the number of levels in the prefix structure of each directory is typically small. This allows the user to navigate through several directory levels in the space of a small window.

4.4 Dispatching Events

From our early initial experience with CrossY, we learned that the standard event dispatch mechanism can result in the loss of crossing events. The problem is a combination of sampling rate of the pen or pointing device in general and how the sampled events are distributed to the different recipients. Moving the device quickly over the screen results in just a few sparsely distributed points. Distributing only these points can result in the "loss" of crossing events. This can happen if the pen is moved in such a manner that the points captured are outside the widget that contains the line to cross. The solution we applied to this problem is a completely customized dispatch mechanism. Starting off with a distribution of line segments[2] instead of points simplifies the handling of the crossing events since all the actions that need to be taken are encapsulated in each widget and are based on the information passed into the widget (see Figure 16). The central dispatch checks if the current line segment crossed any widgets and then passes the line segment information on to the widget (see Figure 17). The mechanisms encapsulated into the widget then trigger the necessary action. This functionality is provided by a recursive descent of the windowing hierarchy and thus reaches all involved widgets. One advantage of this approach is the fact that even in case where the "actual" points are not on the widget, like in the case of several option boxes in a panel, each widget gets the information about the line segment if it somehow intersects with it. The correct receiving of all crossing activity is especially crucial when it comes to command combination, where the user issues several with one stroke. An

[2]These are segments rather than lines since we do not dispatch the whole current stroke but only the last *segment* of it, thus the naming.

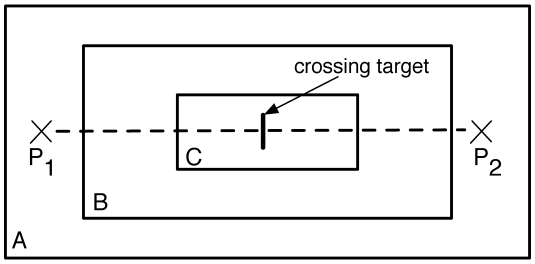


Fig. 16. An example that visualizes how the dispatch mechanism works. Boxes A–C represent widgets on the screen. Widget A receives two points P_1 and P_2, widgets B and C do not receive the events directly, but the dispatch in widget A computes that the line between P_1 and P_2 touches B and thus dispatches the line (P_1,P_2) to B. The dispatch in B computes that (P_1, P_2) crosses C and dispatches it into C, in C the actual crossing is detected without ever receiving a direct event of points within its boundaries.

```
public void OnNewLine(LineSegment currentLineSegment){
      counter = 0;
      while(GoThroughWidgets(counter)!=null)
       {
             Dispatchable aWidget = GoThroughWidgets(counter);
             if(lineIntersects(aWidget))
             {
                   aWidget.OnNewLine(currentLineSegment);
             }
             counter++;
       }
}
```

Fig. 17. A pseudocode listing of the main loop in the dispatcher which notifies widgets of line events if they are intersecting in any way with the widget. The LineSegment class ensures that the translation between window and screen coordinate system is handled correctly when switching from widgets to their children.

example is the toolbox where the stroke width and color are selected in one stroke. If some of the points were not on the widget, either the width or the color would not be selected properly and the user would be forced to reissue.

4.5 Implementation

CrossY was implemented in C# on a Tablet-PC. To achieve the intended behavior we created our own stroke/stroke collector class which allows us to have a standard functionality of adding and removing strokes but also enables us to add crossing specific features. These are for example the possibility to treat strokes as a sequence of lines and not as a sequence of points. The whole system was developed using the .NET framework [Jarett and Su 2002] as the basis for our design.

4.6 Lessons Learned in Building CrossY

From our experiences gained while implementing CrossY, it is clear that the crossing paradigm is at least as expressive as the standard point-and-click interface and provides the same level of functionality as the latter. It is possible

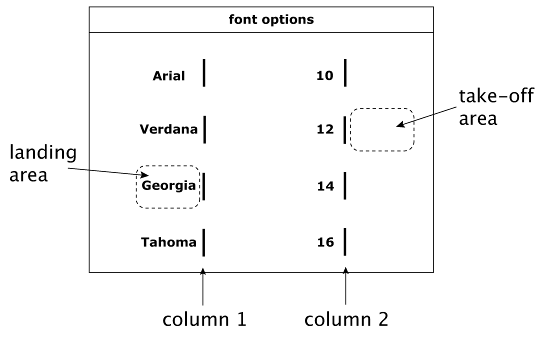


Fig. 18. The design of a simple dialog box.

to offer a wide range of features with a minimal visual footprint on the screen because the system accounts for the crossing location (in the crossbar and the palette for selecting pen attributes), the performed gesture (in the crossbar), and the direction of the stroke. Similar advantages were achieved in Gedrics [Geissler 1995] which uses gestures on top of icons.

As mentioned earlier, an interesting aspect of the crossing paradigm is the possible composition of commands in one single stroke.

The feature of command composition is a unique and fundamental aspect of this approach since it allows users to smoothly move from novice to expert. Novice users will perform one command at a time, while relying heavily on visual feedback. As they become more and more proficient, they start to remember the shape of the strokes corresponding to a particular dialog box and rely less and less on visual feedback. As described earlier, each command combination can also be executed in separate steps. This reduces the cost of making a mistake while performing a long sequence of actions because the user only needs to restart the gesture at the point where the error occurred. This is possible because the actual confirm or reject of a series of commands is done by crossing the border of the widget as described in Section 4.3.3. While menu systems such as the Marking Menu were designed to encourage such transitions in the case of single command selections, we believe that our work is the first to explore how the same effect can be obtained for a succession of commands.

5. STUDY OF PARAMETERS THAT INFLUENCE CROSSING

In the process of building CrossY, we realized that the data gathered during our first experiment was not sufficient to understand how the interface layout might influence users' performance. In particular, we were interested in the parameters influencing performance when selecting options in a dialog box such as CrossY's pen attribute dialog box. To this end, we designed an empirical study of parameters influencing the selection performance for a simple, two-parameter crossing-based dialog box (Figure 18).

When designing CrossY, empirical evidence seemed to imply that the angle α between the target centerline (see Figure 19) and the horizontal might influence the values of a and/or b in Fitts' law (see Equations (2) and (3) in Section 3.2.2). Based on this observation, we structured our experiment around the angle

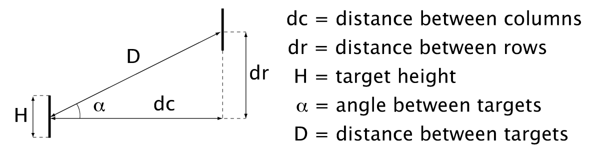

D
dr
H
α
dc
dc = distance between columns
dr = distance between rows
H = target height
α = angle between targets
D = distance between targets

Fig. 19. Key variables for designing crossing-based widgets.

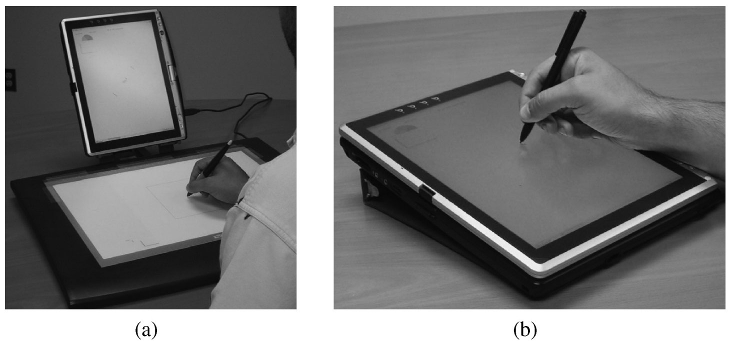


Fig. 20. Screen shot of the two settings in our test application.

(α) between the centerline of the targets and the horizontal while taking into account the index of difficulty based on height and distance of the crossing targets. This choice allowed us to compare our results with previous studies [Jagacinski and Monk 1985; MacKenzie and Buxton 1992] and to explore the influence of α in the direct setting.

Another important parameter is target orientation. As discussed in Section 4.3, due to layout constraints in crossing interfaces, it is advantageous to use vertical targets. Yet as α increase it might become more advantageous to have targets orthogonal to the direction of travel, since our initial experiment showed that the C/DC setting was faster than the C/AC setting (Section 3.2.2).

Finally, we felt it was also important to explore the possible influence of the interaction setting (either direct or indirect). This is because while the initial empirical study was performed in the indirect setting, CrossY was designed with a direct setting in mind (Table PC).

In summary, we identified 4 variables that may influence user performance in crossing interfaces:

(1) α, the angle between the target centerline and the horizontal line, see Figure 19.
(2) ID, the index of difficulty based on height of targets and the distance, center to center, between two crossing targets;
(3) orientation, which could be either vertical, or orthogonal to the target centerline,
(4) setting which can be either indirect setting (interaction on a tablet in front of the display (Figure 20(a)) or direct setting (interaction directly on the display, (Figure 20(b)).

Table I. Height Distance Combination for the Experiment

ID	Distance	Height
2	120	40
	180	60
3	280	40
	420	60
4	300	20
	600	40
5	310	10
	620	20
6	630	10
	788	13

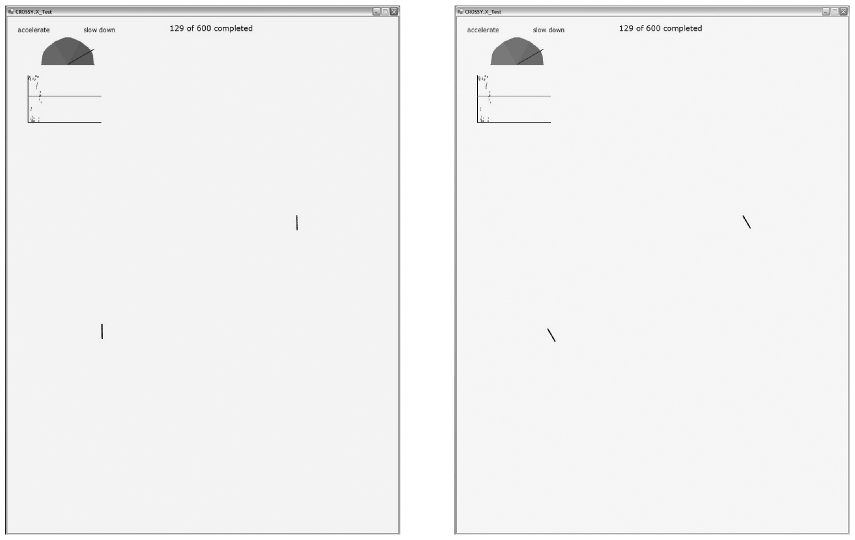
5.1 Experiment

Fully crossing all these variables would, of course, be unpractical. Instead we focused on the interaction of ID (Distance, Height) x Angle within three typical configurations of orientation and setting. First, we were interested in a situation closely simulating a typical crossing-based interface on a tablet computer. In the direct vertical condition (DirVert) participants interacted directly on the screen and the targets were always vertical (Figure 21(a)). To study the possible effects of target orientation on performance, we also considered the direct orthogonal condition (DirOrtho), where the targets were always orthogonal to the centerline between the two targets (Figure 21(b)). In both conditions, interactions took place directly on the display of the tablet computer.

Because our first study has focused on indirect settings with crossing targets orthogonal to the main direction of movement, we included this setting as a reference point. This condition, IndOrtho, is identical to the DirOrtho setting except that users were interacting on a WACOM tablet placed in front of the computer. Figure 20 shows pictures of the direct and indirect setting. We focused on continuous crossing since we were interested in complex interactions such as the search and replace dialog box in which the stroke "accumulates" context.

For ID, we selected ten possible Distance-Height combinations corresponding to 5 indices of difficulty (2–6), see Table I. Our choice was constrained both by practical concerns (e.g. in dialogue boxes, targets need to be far enough apart to allow for a label), and the size of the screen of our tablets. For the Angle, we selected 0, 15, 30, 45, 60, 90 degrees. While CrossY was built around the idea of a right-to-left traversal (angles from 180 to 270 degrees), we felt that this might be too unusual for untrained users. Instead we focused on the upper right quadrant (0 to 90 degrees) as a compromise between limiting the potential for occlusion and familiarity.

5.1.1 *Task.* Users were asked to cross two targets with a single stroke and without lifting the pen. After crossing the first target, a feedback sound was played and the color of the target changed. Upon crossing the second target,



(a) In the vertical setting the crossing targets stayed vertical within the context of the screen.

(b) In the orthogonal setting the targets stayed orthogonal to the main direction of travel.

Fig. 21. The setup for our experiment. a) indirect setting; b) direct setting.

the task was completed and the next trial appeared. An error was counted when the user lifted the pen before crossing the second target, crossed the target in the wrong direction, or did strokes that involved no crossing at all. To avoid counting simple landing (i.e., touching the pen to the screen) as an error, we introduced a minimum stroke length of 10 pixels. Thus, a simple touch of the screen, as it may occur between blocks, did not count as an error. To provide the user with feedback about their performance we provided two scales in the upper left corner of the application window (Figure 21). One represented the cumulative error rate in the current condition and the other the completion time for each connection. The first scale was a half-circle consisting of three evenly sized parts colored in blue, green and red with a representation of an error rate from 0% on the left side to 8% on the right side of the half circle. The second feedback widget showed a diagram where the y-axis reflected completion time and the x-axis represented the trial number. Parallel to the x-axis, a line indicated the average completion time for the task (based on pilot studies). After each trial, a dot appeared on the diagram and gave feedback about the current speed of the user. This was done to ensure that participants maintained aproximately a 4% error rate to avoid possible problems with speed accuracy trade-off.

5.1.2 *Method.* For this experiment we adopted a within-subject design. This was based on the assumption that skill variation between subjects might be important and little or no asymmetrical skill transfer was expected. The presentation of different IDs was randomized. To reduce measurement noise, each combination was repeated 10 times in a row. As a result, participants performed 600 connections in each device setting. To limit the influence of skill transfer, we used a fully counterbalanced design for the order of conditions (IndOrtho, DirOrtho and DirVert). In summary, the experiment collected: 5 (distance) $\times$ 6 (height) $\times$ 8 (angles) $\times$ 12 (participants) $\times$ 10 (repetitions) = 28'800 trials. Overall, each participant was asked to complete 2400 connections, including practice crossings.

5.1.3 *Apparatus.* The main apparatus of the experiment was our test application running on two Toshiba Protégé Tablet PCs, both with 1.25 GB RAM and one with 1.7 GHz CPU frequency and the other with 1.5 GHz. To ensure that the different CPU frequencies had no influence on our results we fully balanced the tasks that were done on the different computers. The diagonal of the screen was 307mm and the resolution was set to 1400 $\times$ 1050 pixels. The tablet PCs were used in the folded configuration when they appear as a slate to users (see Figure 21). For the indirect condition, we used a Wacom IntuosII tablet where the gain between pen movement and cursor movement was set to 1.8. This value corresponds to an average of the values reported in the literature [Accot and Zhai 2002; MacKenzie and Buxton 1992]. To prevent confusion and errors, the buttons on the pen for the Tablet PCs and the pen for the Wacom tablet were disabled. The test application was written in C#. Apart from presenting the different tasks, it also logged all actions performed by the users.

5.1.4 *Protocol.* The participants were 12 students at the University of Maryland (5 female; age range 18–30 years). One of the participants was left-handed. The three configurations (IndOrtho, DirOrtho, DirVert) were presented in blocks. Before completing the actual task, users completed a training session of 3 connections per ID/Angle combination. participants were asked to complete the crossing as fast and precise as possible. They were asked to keep the error rate within the middle area of the error scale described above. This represents an error rate of 4%. Since we were measuring only the times from first crossing to second crossing, users had a chance to rest as soon as they completed a crossing. Auditory feedback was provided through headphones connected to their tablet as some users were run in pairs in the same room. Users received $20 for their participation.

5.2 Results and Analysis

For our analysis, we focused on the average completion times for the connections performed under each combination of Angle, ID, and condition. Bonferroni adjustments for multiple comparisons were used.

To evaluate the possibility of asymmetrical skill transfer [Poulton and Freeman 1966], we performed a condition (IndOrtho, DirOrtho, DirVert) x order analysis of variance (ANOVA). There was no main effect of presentation order ($F_{2,15} = 1.61$, $p = .233$) and no interaction between condition and presentation order ($F_{4,15} = 1.12$, $p = .385$).

To compare the error rate across the different conditions we performed a within-subject ANOVA on error rate for each condition. We found that there was a significant effect ($F_{2,22} = 5.46$, $p = .012$), driven by a high value (8.54%) for the indirect setting (IndOrtho) compared to the direct settings (5.0% and 5.6% for DirOrtho and DirVert respectively). Yet no *significant* pairwise differences were found. This indicates that we were only moderately successful in holding error rates constant across conditions.

To examine the influence of different CPU frequencies we performed an ANOVA with condition as a within-subject factor and CPU frequency (1.7 vs. 1.5 GHz) as a between-subject factor with average crossing time as the dependent variable. We found no significant main effect of CPU frequency on total crossing time ($F_{1,20} = 1.44$, $p = .244$) No interaction between CPU frequency and condition was found ($F_{1,20} = 1.437$, $p = .245$).

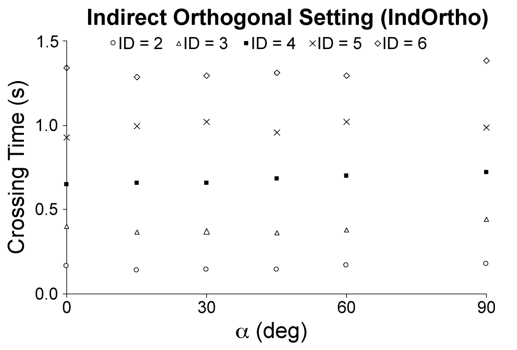
Next we performed a (Condition $\times$ ID $\times$ Angle) within-subject ANOVA on average crossing time. There was a significant main effect for condition ($F_{2,22} = 15.7, p < .001$). Post-hoc tests indicated that the indirect condition was significantly slower than either of the direct conditions ($p < .005$). There were no significant differences between the direct conditions ($p = .34$). These findings are not surprising, because hand-eye coordination provides a significant advantage in the case of the direct conditions. This is consistent with the comments of users who reported that they felt more control over the interaction in these settings. We also found a significant main effect of ID, ($F_{4,44} = 656, p < .001$), reflecting Fitts' law. In addition, there was a main effect of Angle ($F_{5,55} = 34.4, p < .001$). These main effects were qualified by a complex interaction pattern. We observed a significant (Condition $\times$ Angle) interaction ($F_{10,110} = 11.3, p < .001$), a (Condition $\times$ ID) interaction ($F_{8,88} = 8.58$, $p < .001$) as well as a significant 3-way (Condition $\times$ ID $\times$ Angle) interaction ($F_{40,440} = 1.73$, $p < .005$). To explocate this interactions, we are now proceeding with an analysis on a method per method basis.

5.2.1 *The Indirect Orthogonal Condition.* We performed an (ID $\times$ Angle) within-subject ANOVA on average crossing time in the indirect orthogonal condition (Figure 22(a)). It revealed a strong main effect of ID ($F_{4,44} = 639$, $p < .001$) suggesting a direct linear dependency between total crossing time and ID which reflects Fitts' law:

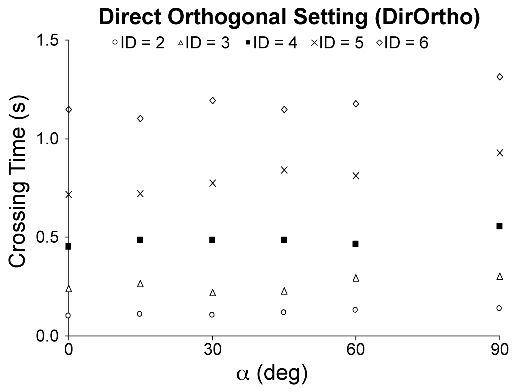
$$T = 292ID - 463, (r^2 = 0.996).$$

The main effect of Angle ($F_{5,55} = 1.62$, $p = .21$) and the (ID $\times$ Angle) interaction ($F_{20,220} = 1.34, p = .32$) were not significant. In general, these findings on crossing with different approaching angles are consistent with previous studies in Fitts' law pointing tasks [Jagacinski and Monk 1985; MacKenzie and Buxton 1992], although in our case, the 45 degree angle does not seem to stand out.

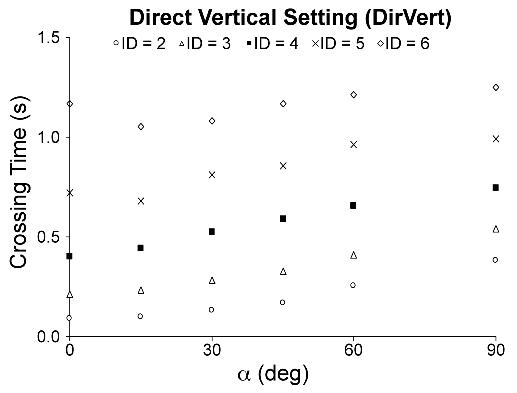
5.2.2 *The Direct Orthogonal Condition.* We performed an (ID $\times$ Angle) within-subject ANOVA on average crossing time in the direct orthogonal condition (Figure 22b). There was a main effect of Angle ($F_{5,55} = 7.89$, $p < .001$) and a (ID $\times$ Angle) interaction ($F_{20,220} = 1.93$, $p < .012$). Post-hoc comparisons suggested that the only significant differences were between angles lower than 15 degrees and the 90 degree condition ($p < .03$; mean difference around 115 ms.). Further, there was a main effect of ID ($F_{4,44} = 332, p < .001$)



(a) The effects of ID and Angle on Crossing Time in the indirect orthogonal setting.



(b) The effects of ID and Angle on Crossing Time in the direct orthogonal setting.



(c) The effects of ID and Angle on Crossing Time in the direct vertical setting.

Fig. 22. The effects of ID and angle on crossing time.

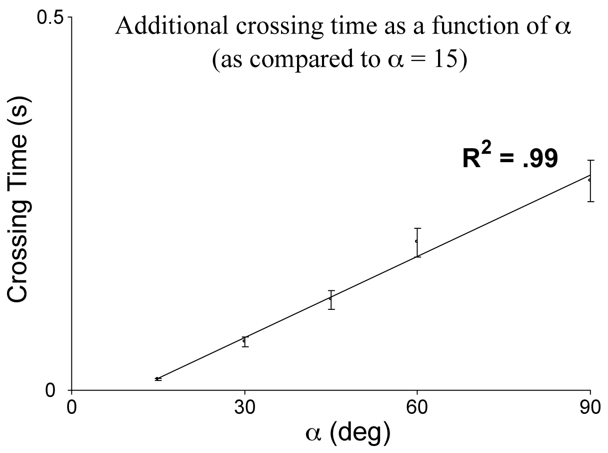


Fig. 23. The influence of angle on crossing time averaged over all IDs using the crossing time at Angle = 0 degree as a reference point. ($\triangle t_{ref=15deg}(Angle) = 3.64(Angle - 15) + 15.9,\ Angle \geq 15degree, r^2 = 0.99$).

reflecting that, as predicted by Fitts' law, there was a strong linear dependency between the total crossing time and ID for each angle:

$$
\begin{aligned}
T &= 257ID - 498, (r^2 = 0.956), Angle = 0degree\\
T &= 245ID - 444, (r^2 = 0.973), Angle = 15degree\\
T &= 274ID - 539, (r^2 = 0.962), Angle = 30degree\\
T &= 262ID - 471, (r^2 = 0.964), Angle = 60degree\\
T &= 298ID - 544, (r^2 = 0.975), Angle = 90degree.
\end{aligned}
$$

In a direct setting (and without acceleration), the two extreme directions (0 and 90 degree) different directions involve different muscle groups, particularly when ID (and hence the distance to travel) is high. Specifically, we observed that as the angle nears 90 degrees, the upper arm and the shoulder participate in the movement. These proximal joins have slower bit rates than more distal joins like the wrist [Langolf and Chaffin 1976] and this may account for the slower crossing times.

5.2.3 *The Direct Vertical Condition.* We performed an (ID × Angle) within-subject ANOVA on average crossing time in the direct vertical condition (Figure 22c). Again, there was a main effect for Angle ($F_{5,55} = 54.2$, $p < .001$) and an a (ID × Angle) interaction ($F_{20,220} = 1.80$, $p < .023$). The effect of Angle was considerably stronger than in the direct orthogonal condition. For example the difference between angle = 0 and angle = 90 degree was 262 ms ($p < .001$). Setting aside the special case of angle = 0 (for which we observed an unexplained "bump" at ID 6), we observed that the difference between measured task completion time and the observed task completion time at angle = 15 degree increase linearly as a function of Angle. (Figure 23). Again there was a strong main effect on ID ($F_{4,44} = 380$, $p < .001$) reflecting Fitts' law. For each angle we observed a strong linear dependency between the total crossing time

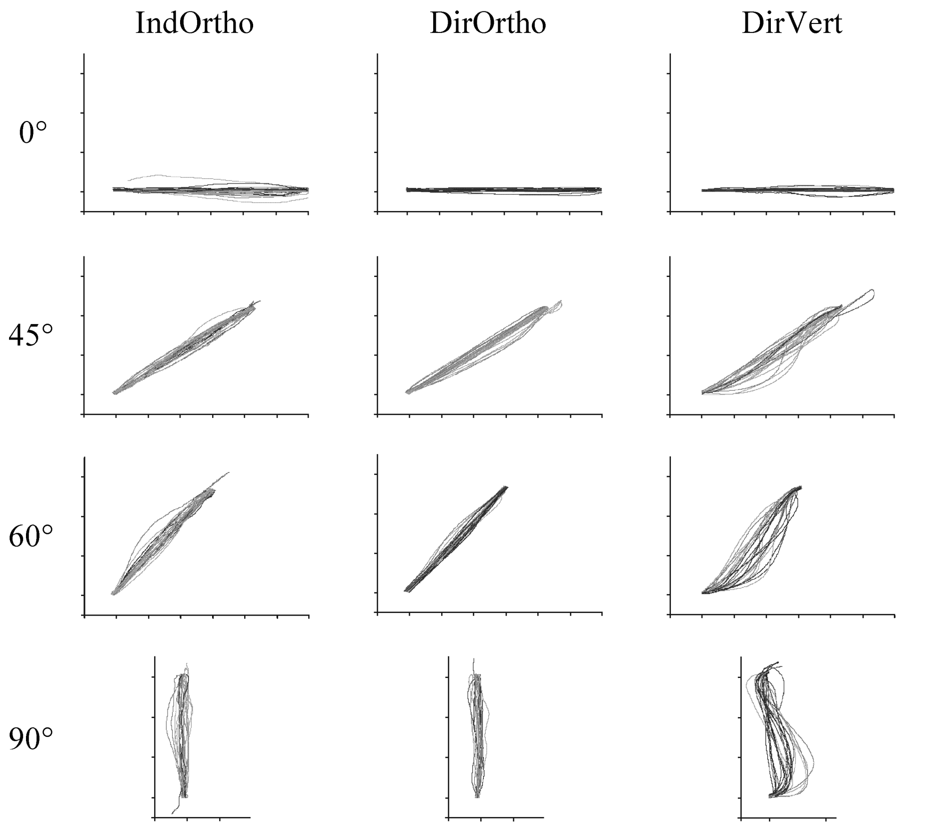


Fig. 24. Users take different paths depending on the setting and path angle. In the first and second column (Orthogonal setting), paths are relatively straight for all angles. In the third column (Vertical setting) we see an increasing in curvature with increasing angles. The paths shown are the 9th and 10th strokes of each user for each setting for ID = 5 (Height = 20 pixels).

and ID:

$$T = 266ID - 546, (r^2 = 0.942), Angle = 0degree$$
$$T = 235ID - 440, (r^2 = 0.968), Angle = 15degree$$
$$T = 243ID - 404, (r^2 = 0.989), Angle = 30degree$$
$$T = 254ID - 392, (r^2 = 0.989), Angle = 45degree$$
$$T = 247ID - 289, (r^2 = 0.989), Angle = 60degree$$
$$T = 218ID - 91, (r^2 = 0.991), Angle = 90degree.$$

As for the DirOrtho condition, these results can be explained by the progressive involvement of more proximal joints in the movement. Yet, the effect is much stronger than in the DirOrtho condition and additional factors may play a role. One possible explanation might lie in the dynamic of the movement itself [Viviani and Terzuolo 1982] (see Figure 24). As expected, the IndOrtho condition appears noisier than the DirOrtho—probably because of the high gain for this condition. However, the most interesting difference is between the

DirOrtho and DirVert condition. Participants no longer take a straight path between the two crossing targets but start to use more S-like strokes to achieve shallower entry and exit angles into the crossing targets.

5.2.4 *User Feedback.* The results of the user testing show that the crossing paradigm is easy to understand and to transfer into action even by novice users. Overall, the direct setting was in all cases preferred to the indirect setting. This suggests that direct pen-based interactions are easier to perform than indirect pen-based interactions. All participants agreed that the indirect setting was more difficult to handle and more error prone. This is reflected in the increased crossing time (see Figure 22(a)) and a higher error rate.

6. DISCUSSION AND IMPLICATIONS FOR CROSSING-BASED INTERFACES

In this section we are reporting the insights we gathered while designing CrossY and how these insights, combined with the findings from our second study, can be applied to crossing-based interfaces in general. For illustration purposes, we use the simple case of a two-column dialog box as shown in Figure 18 as a running example, and restrict ourselves to the case of interactions in the first quadrant (e.g., selecting the font "Georgia" and font size "12" in sequence).

6.1 Space and Time Efficiency

When designing the dialog box in this example, one first needs to consider how much space is required to comfortably perform selections. This includes the space needed to land the pen before the connection, the minimum space required between the two columns, and the space needed to take off the pen before crossing the dialog box boundary. In our experience, if one considers novice users, the space requirements of a crossing-based interface will be similar to the equivalent point-and-click interface. This is derived from the fact that the crossing efficiency is similar to that of pointing which was described earlier (DP task, see Figure 3(b)). Thus one can simply substitute every standard button with a crossing button of the same size. Yet, when one wishes to leverage command composition, a space vs. speed trade-off will appear because some space will be needed due to the sloppiness of rapid gestures. This means that we need to provide more space for each widget in order to ensure reliability. Based on our experience, we believe that a slightly larger footprint may be acceptable as the expected speed benefits from command composition are substantial. Furthermore, natural constraints of efficient visual layout (such as the use of negative space Mullet and Sano [1994]), may be all that is needed.

6.2 Landing and Takeoff Space

As seen in Figure 18, a certain area needs be reserved for landing and taking off before and after the actual crossing interaction. To better understand the user requirements for landing and takeoff, we plotted the distribution of landing and takeoff points from our follow-up study (Figure 25). As the target grows bigger, the users are able to comfortably land their pen further away. This can be

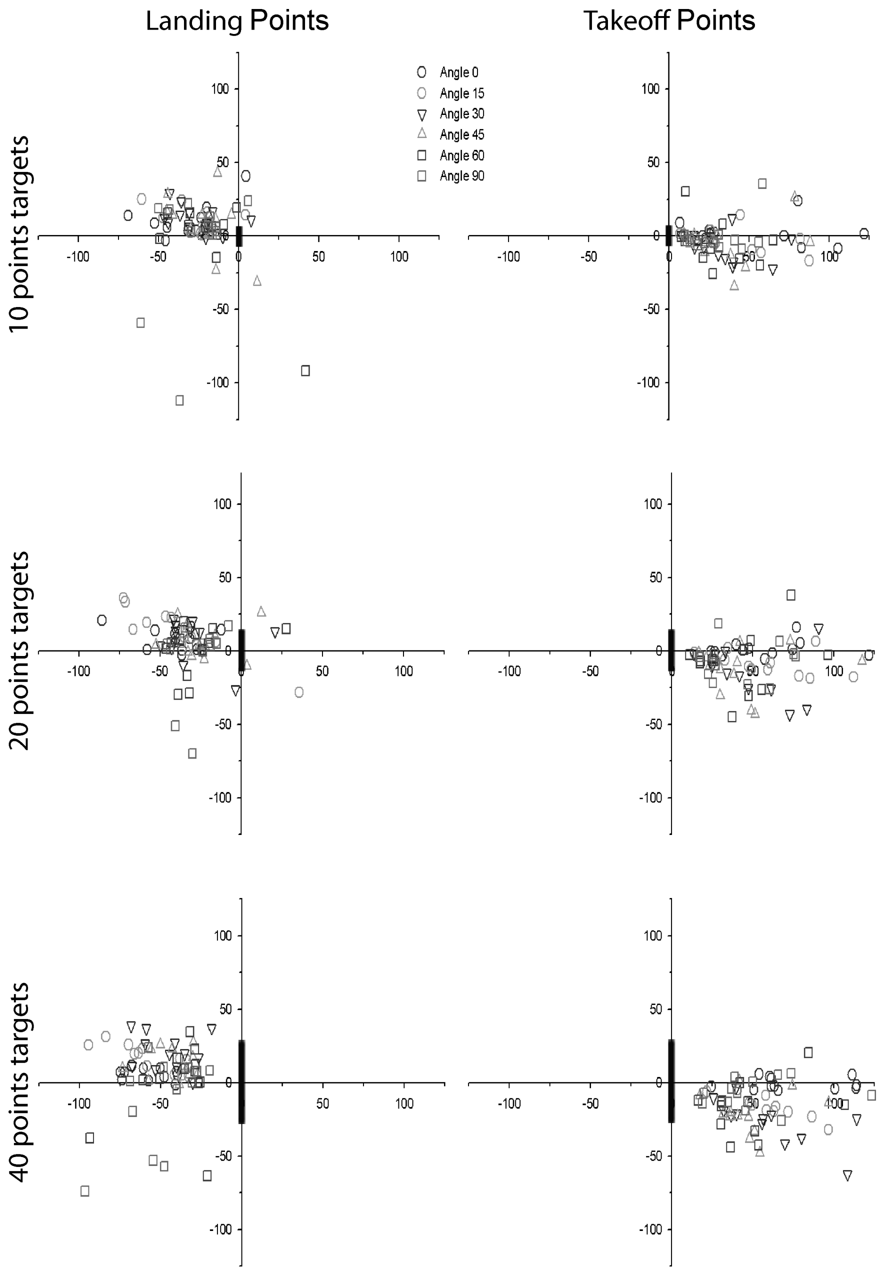


Fig. 25. Distribution of the landing points (left column) and take-off points (right column) relative to the crossing target height and the connection angle. Black vertical lines represent the crossing target position. Points further than 125 pixels from target center are not shown (less than 10% of all points).

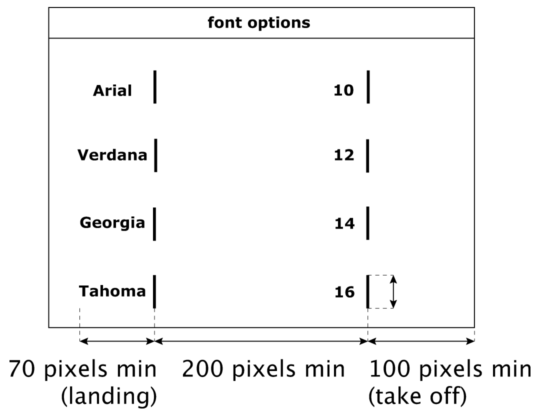
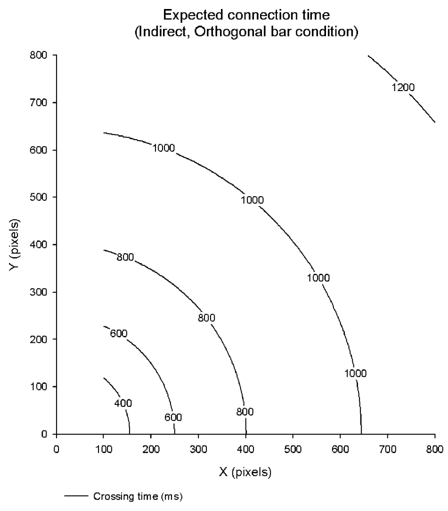


Fig. 26. Sample dialog box with measurements.

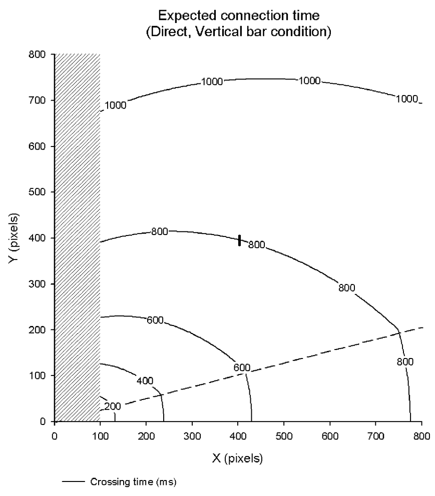
explained by the fact that they are trying to maintain a comfortable angle through which they are "seeing" the target upon landing [Accot 2001]. A similar behavior is observed for takeoff. From a design perspective, this means that one has to reserve a 70 pixel (12.3mm) margin to the left of the first column and a 100 pixel (17.5 mm) margin to the right of the second column if one assumes a typical crossing target height of 30 pixels, as in CrossY. We also measured the maximum bounding box for the S shape connection curve. For the 90 degree angle, this box is 100 pixels (17.5mm) by 70 pixels (12.3mm), which implies that the distance between columns should be at least 200 pixels (35mm), see Figure 26. These values are based on our findings but need to be validated experimentally.

6.3 How Do Users Cross and React to Errors?

Based on the results from our first study, we designed CrossY so that the majority of crossing targets was directly orthogonal to the crossing direction. In cases where several commands had to be selected consecutively we used targets with a low index of difficulty. From informal interviews with our users, we learned that the initial layout was successful and was well perceived. We observed that the strategies for crossing, reacting to errors and interaction with the tilted lines in the checkboxes (see Figure 13) varied between users. Some examples of different paths users drew are shown in Figure 24. It is important to note here that the findings from the C/AC and C/DC settings in our first experiment were confirmed: performance, in term of time and error rate decreased the further the crossing targets were away from being orthogonal to the crossing direction. The most extreme cases are the angles close to 90° where users perform a S-like stroke to cross both targets. This slows them down and increases the chance of errors (see Section 3.2.3). Also, in terms of error correction, some users simply continued with a loop back after missing the second target, instead of starting over again (See Figure 24 (DirVert, 45°), where one stroke extends longer than the others. This stroke was not counted as an error since, after missing the second target, the user traveled back and crossed it successfully. This suggests that a backtracking path (e.g., for checkboxes) has to be considered.



(a) The predicted times for the IndOrtho condition.



(b) The predicted times for the DirVert condition. The gray area shows where in the case of dialog boxes, the movement might be limited by Drury's law (see text).

Fig. 27. Each contour plot shows the predicted crossing time (in ms) it will take a participant to connect two targets (with the first target at the origin and the X,Y coordinates describing the center of the second target).

6.4 Influence of Layout on Performance

As discussed for the DirVert condition (Section 5.2.3), the crossing time between two targets whose centerline is α degree from the horizontal is given by the following formula (for $\alpha \geq 15$ degree[3]):

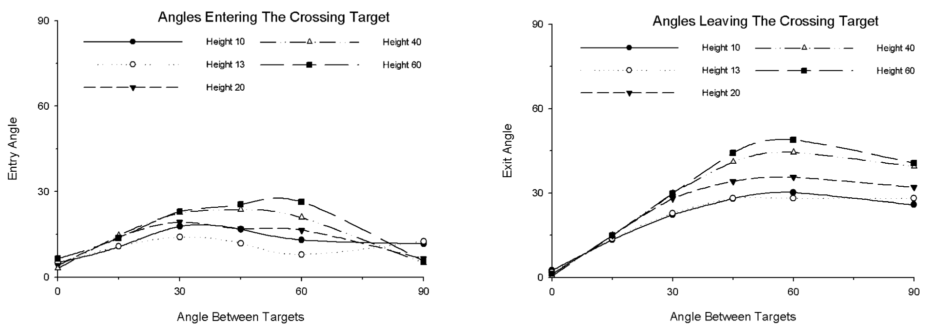
$$T = 235ID - 440 + 3.64(\alpha - 15).$$

We used this formula to draw the contour plot presented in Figure 27(b). Given two targets (of height 20), one with its center at the origin of the graph and one with its center at the (x, y) coordinate, the value read at the (x, y) will provide an estimation of the crossing time. For example if one places a target at (400, 386) one might expect a 800ms crossing time. As a reference, we also show in Figure 27(a) the same contour plot for the indirect condition. This graph confirms our informal observations during the development of CrossY that as the dialog box becomes taller it also become more cumbersome to use.

6.5 Relationship to Pointing and Steering

While we have focused on performance regularities as a function of crossing target characteristics, it is obvious that in real system design, other constraints may play a role, and sometimes dominate the crossing constraints. For example, if one has to cross a goal with distracter close by, Fitts' law for pointing (placing

[3]For $\alpha < 15$ degrees our data were inconclusive suggesting that users performed similar as in the horizontal condition.



(a) The influence of the target height and α on entry angle. (b) The influence of target height and α on exit angle.

Fig. 28. The influence between angle and target height.

the pen between target and the distractor goals) may dominate, as we have previously shown in Section 3.2.3. Similarly, in the case of a two column design for a cro in ssing dialog (Figure 18), if the two sets of goals are too close to one another, they will form a "tunnel" users will have to negotiate (i.e. selecting Tahoma 10 in one stroke in Figure 18). In such cases the steering law [Rashevsky 1959; Drury 1971; Accot and Zhai 1997] effect may dominate the total task completion time. Note that the steering law effect levels off when the steering ID (tunnel distance to tunnel width ratio in the simple cases) is below 4 ([Accot and Zhai 1997], page 301). In the present set-up of CrossY, the steering law effect therefore should not dominate the crossing effect.

6.6 Beyond Two Columns

Of course, applications may use more than 2 columns in their dialog boxes.[4] We did not investigate situations in which more than two marks are crossed in succession. Yet, our data provide us with two important pieces of information: the target entry angles and the target exit angles. These values are graphed in Figure 28. The figures show that for low angles, the entry and exit angles roughly correspond to the angle of the target centerline. However, after the angle reaches 20 degrees, users limit the entry angles around 30 degrees. A simple additive formula for successive connections will at least require input and output angle to be similar (so that there will be very little influence from the first path to the second). Our data suggest that even within the restrictions of the first quadrant, interactions between successive connections are to be expected. Of course, the problem will be even more acute in the case of connections which require users to first go up and then go down. Additional experiments are needed to explore these cases.

[4]The empirical evaluation of the 3 columns case [Dixon et al. 2008] was published after this work was done.

6.7 A Quick Reference for Designers

Based on the considerations outlined so far we now present a summary of the most important points to keep in mind when designing crossing-based interfaces for quick reference.

(1) The predominant constraints for the interface layout will be imposed by interactions requiring discrete directional crossing. As a result, support for continuous crossing can lead to more compact interface design.
(2) Crossing targets that are to be crossed continuously should be positioned orthogonal to the movement direction whenever possible.
(3) The further a crossing path deviates from the horizontal, the more difficult it will be for users to perform. As a results, application designers should be sure to place the most frequently used sequence on a more or less horizontal path.
(4) The design should provide enough space to allow for “recovery” strategies for missed crossings during multitargets interactions.
(5) The normal dispatch of mouse events might not always provide for accurate crossing detection during fast gestures.
(6) Crossing and pointing can be used as parallel interaction methods.
(7) Crossing as an interaction method works in a direct and indirect setting, even though our users preferred the direct setting.

7. FUTURE WORK

Although it is too early to judge the success of the command composition approach, this work illustrate the potential of this technique. For example, our implementation demonstrates how crossing may alleviate the need for dwell time for several types of interactions. Somewhat like the text input systems proposed in Zhai and Kristensson [2003] and Perlin [1998], we envision a system in which, as novice users discover the interface, they also train themselves towards generating accurate gestures for the most commonly used commands. At some point, users will be able to remember the shape of the gesture well enough to be able to generate it on top of the interface elements without the need for visual feedback. We believe that such a system could be implemented by having two concurrent tracking mechanisms for user input. The first mechanism will be based on the techniques we used while implementing CrossY and will track the crossing of each interface element. This mechanism will typically require visual feedback. The second tracking mechanism will use a gesture recognition engine to classify user input into possible strings of commands. Depending on specific aspects such as the start of the stroke, the scale of the stroke, or its overall speed, the input of both systems can be integrated to infer users’ commands. Relying only on visual feedback limits the performance of crossing to the speed of hand-eye coordination. Therefore crossing interfaces could potentially benefit from simple haptic or tactile feedback on the stylus when a goal is crossed. There are various ways to add tactile feedback to a pen [Lee et al. 2004] or touch-screen [Poupyrev and Maruyama 2003]. With these tactile feedback enabled digital pens, it is possible to emit a click to the pen as it crosses a graphical line, as if it crosses a physical edge. This can be particularly powerful when visual appearance of the goals, such as a "raised" edge, and the form of the tactile feedback, such as a click, are consistent [Campbell et al. 1999]. With such feedback, the "tail" end of crossing could be shorter due to the feedback effect.

7.1 Special Application Domains of Crossing Interfaces

While CrossY is a "pure" crossing-based application it is important to note that pointing and crossing do not have to be mutually exclusive. One could have a "double representation" interface where both actions are enabled.[5] First, the increased flexibility of both pointing and clicking is beneficial from the perspective of reducing repetitions of the same action. The same user may choose either crossing or pointing depending on the idiosyncrasies of the task at hand. Second, users may tend to use one mode or the other depending on individual preference. Some users have gotten used to pointing and will continue their habitual primary mode of action. Others, including some elderly users and users with certain motor disabilities, may have difficulty clicking without moving the cursor position, or worse, double-clicking [Walker et al. 1996, 1997]. For these users, crossing may become the main interaction mode. Second, hyperlinks in Web pages are especially difficult for pointing due to their narrow height (one line of text) but easy for crossing due to the much greater width (one word long and often much longer). It is also possible to cross a list of links in a cascade. This, in fact, has been demonstrated in the "Elastic window" prototype [Kandogan and Shneiderman 1997].

Finally, traditional GUI widgets are very difficult to integrate in virtual reality types of 3D interfaces—partly because point and click is necessarily dependent on a solid 2D surface. In contrast, 1D goals (bars) can be easily crossed ("chopped") without having to be on a 2D surface. Similarly, actions may be triggered when crossing a 2D surface, like a door or portal (e.g., Haisenko and Musgrave [2001]).

8. CONCLUSION

In experiment 1, we compared human performance in both pointing and crossing tasks, with an emphasis on systematic variations in terms of task difficulty, the direction of movement constraint, and the nature of the task (discrete vs. continuous). We found a robust regularity among trial completion time, distance between targets, and target width in all six tasks under examination. Our results show that all tasks follow Fitts' law with parameters which are task dependent. Observing that in some settings, crossing can be faster than pointing and clicking, we explored crossing as the primary building block of a graphical user interface in CrossY. We found that crossing as interaction technique is as expressive as the more traditional point-and-click interface and provides designers with more flexibility than the latter because it takes into

[5]This idea has been explored further after the work presented here was conducted: Dixon et al. [2008] demonstrated that it can also improve the speed of interaction.

account the shape and direction of the strokes. We also found that a crossing-based interface can encourage the fluid composition of commands in one stroke, as illustrated in our find-and-replace dialog box. Ultimately, we believe that the fluid composition of commands will lead to more efficient and natural interfaces for pen computing.

Our experience while designing CrossY showed us that more empirical data was needed to better understand the design parameters influencing the performance of crossing based design. We explored them in experiment 2, which showed that that for the direct configuration with vertical targets, connection time depended on the angle between the centerline of the two targets and the horizontal. We also showed how the empirical data gathered in experiment 2 can be applied to better understand the implications of a given layout on overall crossing performance. This work lays the foundation for designing crossing-based user interfaces in which, crossing is used instead of or in addition to pointing, as a fundamental interaction technique. The use of crossing action may not be limited to a certain device but applicable to many pen-based or touch-screen devices from small display smart phones to large-sized wall mounted displays.

ACKNOWLEDGMENTS

We would like to thank Anja Apitz, Corinna Löckenhoff and all other reviewers of early drafts of this article for their comments. The present work is based on a synthesis of Accot and Zhai [2002] and Apitz and Guimbretière [2004], with significant expansion and a new experiment. Dave Levin drew the screen content of Figure 10.

REFERENCES

Accot, J. 2001. Les tâches trajectorielles en interaction homme-machine—cas des tâches de navigation. Ph.D. thesis, Université de Toulouse 1, France.

Accot, J. and Zhai, S. 1997. Beyond Fitts' law: models for trajectory-based HCI tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'97)*. 295–302.

Accot, J. and Zhai, S. 2002. More than dotting the i's—foundations for crossing-based interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'02)*. ACM Press New York, NY, 73–80.

Accot, J. and Zhai, S. 2003. Refining Fitts' law models for bivariate pointing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'03)*. ACM Press, New York, NY, 193–200.

Adobe, S. I. 2005. Illustrator 10.

Albinsson, P.-A. and Zhai, S. 2003. High precision touch screen interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'03)*. ACM Press, New York, NY, 105–112.

Apitz, G. and Guimbretière, F. 2004. CrossY: a crossing-based drawing application. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'04)*. ACM Press, New York, NY, 3–12.

Baudisch, P. 1998. Don't click, paint! using toggle maps to manipulate sets of toggle switches. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'98)*. ACM Press New York, NY, 65–66.

Campbell, C. S., Zhai, S., May, K. W., and Maglio, P. P. 1999. What you feel must be what you see: adding tactile feedback to the trackpoint. In *Proceedings of the 7th IFIP Conference on Human Computer Interaction (INTERACT'99)*. 383–390.

DIXON, M., GUIMBRETIÈRE, F., AND CHEN, N. 2008. Optimal parameters for efficient crossing-based dialog boxes. In *Proceeding of the 26th Annual SIGCHI Conference on Human Factors in Computing Systems (CHI'08)*. ACM, New York, NY, 1623–1632.

DRURY, C. G. 1971. Movements with lateral constraint. *Ergonomics 14,* 2, 293–305.

FITTS, P. M. 1954. The information capacity of the human motor system in controlling the amplitude of movement. *J. Exper. Psych. 47*, 381–391.

GEISSLER, J. 1995. Gedrics: the next generation of icons. In *Proceedings of the IFIP conference on Human Computer Interaction (INTERACT'95)*. 73–78.

GUIMBRETIÈRE, F., STONE, M., AND WINOGRAD, T. 2001. Fluid interaction with high-resolution wall-size displays. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'01)*. ACM Press New York, NY, 21–30.

GUIMBRETIÈRE, F. AND WINOGRAD, T. 2000. Flowmenu: combining command, text, and data entry. In *Proceedings of the ACM symposium on User Interface Software and Technology (UIST'00)*. ACM Press New York, NY, 213–216.

HAISENKO, M. AND MUSGRAVE, P. 2001. The 3Dsia project. http://threedsia.sourceforge.net.

HINCKLEY, K., BAUDISCH, P., RAMOS, G., AND GUIMBRETIERE, F. 2005. Design and analysis of delimiters for selection-action pen gesture phrases in scriboli. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'05)*. ACM Press, New York, NY, 451–460.

HINCKLEY, K., GUIMBRETIERE, F., AGRAWALA, M., APITZ, G., AND CHEN, N. 2006. Phrasing techniques for multi-stroke selection gestures. In *Proceedings of the Conference on Graphics Interface (GI'06)*. Canadian Information Processing Society, Toronto, Ont., 147–154.

HOFFMANN, E. R. AND SHEIKH, I. H. 1994. Effect of varying target height in a Fitts' movement task. *Ergonomics 37,* 6, 1071–1088.

HONG, J. I. AND LANDAY, J. A. 2000. Satin: A toolkit for informal ink-based applications. In *Proceedings of the ACM symposium on User Interface Software and Technology (UIST'00)*. 63–72.

HOPKINS, D. 1991. The design and implementation of pie-menus. *Dr. Dobb's J. 16,* 12, 16 – 26.

IBM. 2004. Lotus notes (http://www.lotus.com).

JAGACINSKI, R. J. AND MONK, D. L. 1985. Fitts' law in two dimensions with hand and head movements. *J. Motor. Behav. 17*, 77–95.

JARETT, R. AND SU, P. 2002. *Building Tablet PC Applications*. Microsoft Press.

KANDOGAN, E. AND SHNEIDERMAN, B. 1997. Elastic windows: A hierarchical multi-window world-wide web browser. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'97)*. 169–177.

KURTENBACH, G. 1993. The design and evaluation of marking menus. Ph.D. thesis, University of Toronto.

LANGOLF, G. D. AND CHAFFIN, D. B. 1976. An investigation of Fitts' law using a wide range of movement amplitudes. *J. Motor. Behav. 8*, 113–128.

LEE, J. C., DIETZ, P. H., LEIGH, D., YERAZUNIS, W. S., AND HUDSON, S. E. 2004. Haptic pen: a tactile feedback stylus for touch screens. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology (UIST'04)*. ACM Press, New York, NY, 291–294.

MACKENZIE, I. S. 1992. Fitts' law as a research and design tool in human-computer interaction. *Hum.-Comput. Interac. 7*, 91–139.

MACKENZIE, I. S. AND BUXTON, W. 1992. Extending Fitts' law to two-dimensional tasks. In *Proceedings of the Conference on Human Factors in Computing Systems*. 219–226.

MASUI, T., KASHIWAGI, K., AND GEORGE, R. BORDEN, I. 1995. Elastic graphical interfaces to precise data manipulation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'95)*. ACM Press New York, NY, 143–144.

MULLET, K. AND SANO, D. 1994. *Designing Visual Interfaces: Communication Oriented Techniques*. Prentice Hall.

MYERS, B. A., BHATNAGAR, R., NICHOLS, J., PECK, C. H., KONG, D., MILLER, R., AND LONG, A. C. 2002. Interacting at a distance: Measuring the performance of laser pointers and other devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'02)*.

MYNATT, E. D., IGARASHI, T., EDWARDS, W. K., AND LAMARCA, A. 1999. Flatland: new dimensions in office whiteboards. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'99)*. 346–353.

PEDERSON, E. R., MCCALL, K., MORAN, T. P., AND HALAS, F. G. 1993. Tivoli: an electronic whiteboard for informal workgroup meetings. In *Proceedings of the Conference on Human Factors in Computing Systems (INTERCHI'93)*. IOS Press, Amsterdam, Netherlands, 391–8.

PERLIN, K. 1998. Quikwriting: continuous stylus-based text entry. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'98)*. ACM Press, New York, NY, 215–216.

POOK, S., LECOLINET, E., VAYSSEIX, G., AND BARILLOT, E. 2000. Control menus: excecution and control in a single interactor. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (Extended Abstracts) (CHI'00)*. ACM Press, New York, NY, 263–264.

POULTON, E. C. AND FREEMAN, P. 1966. Unwanted asymmetrical transfer effects with balanced experimental designs. *Psych. Bull. 66,* 1, 1–8.

POUPYREV, I. AND MARUYAMA, S. 2003. Tactile interfaces for small touch screens. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'03)*. ACM Press, New York, NY, 217–220.

RAMOS, G. AND BALAKRISHNAN, R. 2003. Fluid interaction techniques for the control and annotation of digital video. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'03)*. ACM Press, New York, NY, 105–114.

RASHEVSKY, N. 1959. Mathematical biophysics of automobile driving. *Bull. Mathe. Biophys. 21*, 375–385.

REN, X. AND MORIYA, S. 2000. Improving selection performance on pen-based systems: a study of pen-based interaction for selection tasks. *ACM Trans. Comput.-Hum. Interac. 7,* 3, 384–416.

SAUND, E., FLEET, D., LARNER, D., AND MAHONEY, J. 2003. Perceptually-supported image editing of text and graphics. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'03)*. 183–192.

SCHMIDT, R. A., ZELAZNIK, H. N., HAWKINS, B., AND FRANK, J. S. 1979. Motor output variability: A theory for the accuracy of rapid motor acts. *Psych. Rev. 86*, 415–51.

VIVIANI, P. AND TERZUOLO, C. A. 1982. Trajectory determines movement dynamics. *Neuroscience 7*, 431–437.

WALKER, N., MILLIANS, J., AND WORDEN, A. 1996. Mouse gain functions and cursor positioning: performance of older computer user. In *Proceedings of the Annual Meeting of the Human Factors and Ergonomics Society*.

WALKER, N., PHILBIN, D., AND FISK, A. 1997. Age-related differences in movement control: adjusting submovement structure to optimize performance. *J. Gerontol. Series B: Psych. Sci. Soc. Sci. 52,* 1, 40–52.

WINOGRAD, T. AND GUIMBRETIÈRE, F. 1999. Visual instruments for an interactive mural. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (Extended Abstracts) (CHI'99)*. ACM Press, New York, NY, 234–235.

ZHAI, S. 2004. Characterizing computer input with fitts' law parameters? the information and non-information aspects of pointing. *Int. J. Hum.-Comput. Stud. 61*, 6, 791–809.

ZHAI, S. AND KRISTENSSON, P.-O. 2003. Shorthand writing on stylus keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'03)*. ACM Press, New York, NY, 97–104.

Received March 2006; revised January 2009; accepted May 2009 by Wendy Mackay