More than dotting the i's — Foundations for crossing-based interfaces

Johnny Accot **Shumin Zhai**
IBM Almaden Research Center
650 Harry Road, San Jose CA 95120, USA
{accot, zhai}@almaden.ibm.com

ABSTRACT
Today's graphical interactive systems largely depend upon pointing actions, i.e. entering an object and selecting it. In this paper we explore whether an alternate paradigm — crossing boundaries — may substitute or complement pointing as another fundamental interaction method. We describe an experiment in which we systematically evaluate two target-pointing tasks and four goal-crossing tasks, which differ by the direction of the movement variability constraint (collinear vs. orthogonal) and by the nature of the action (pointing vs. crossing, discrete vs. continuous). We found that participants' temporal performance in each of the six tasks was dependent on the index of difficulty formulated in the same way as in Fitts' law, but that the parameters differ by task. We also found that goal crossing completion time was shorter or no longer than pointing performance under the same index of difficulty. These regularities, as well as qualitative characterizations of crossing actions and their application in HCI, lay the foundation for designing crossing-based user interfaces.

Keywords
Graphical user interfaces, interaction techniques, goal crossing, goal passing, pointing, Fitts' law, widgets, events, input, input performance

INTRODUCTION

Modern human computer interfaces are based almost exclusively on one type of motor actions — pointing. Loosely speaking, pointing in human computer interaction consists of moving a cursor into a graphical object with an input device and clicking a button (Figure 1.a). This simple sequence of action constitutes the basic interaction scheme for designing most traditional interactive widgets, such as buttons, menus, scrollbars, and folders. Attempts have been made to create alternative interaction techniques (e.g. ray/cone casting [5], bounding box or target inclusion [8, 15], or gestures), but pointing undoubtedly remains the most universal interaction paradigm across diverse application domains and contexts. However, pointing has a number of inherent drawbacks: for instance, it may be time-consuming if the object to be pointed is small; pointing-driven widgets can use a significant screen real-estate; and some users find double click difficult to handle because it requires rapid successive clicking actions without accidentally moving the cursor position. Finding an interaction paradigm that improves performance or circumvents the drawbacks of pointing thus remains a worthy challenge. In this paper, we explore the possibility of using "crossing" as such a paradigm.

Crossing events, which signal that the cursor intersected the boundary of a graphical object (Figure 1.b), are already used in modern interfaces, e.g. for detecting *Enter/Leave* events on graphical objects. For another example, Ren and Moriya [12] studied entering and leaving a button as alternative strategies to clicking, and further pointed out the need of a theoretical model for studying these strategies. We call the process of moving a cursor beyond the boundary of a targeted graphical object a *goal-crossing task* [1, 2].

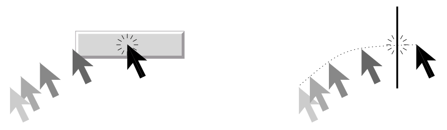


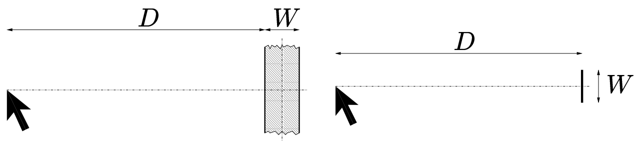
Figure 1: Two different paradigms for triggering actions in a graphical user interface

Fitts' law [3], a quantitative human performance model, has provided a scientific foundation for studying and designing pointing-based user interfaces. A similar scientific foundation is needed if crossing is to be considered as another major paradigm for interaction, particularly if we want to compare the two paradigms systematically.

As a stepping stone towards devising the steering law, Accot and Zhai [2] found that goal-crossing indeed follows a quantitative relationship among movement time, movement distance, and the constraint of the goal width. In fact, such a relationship takes the same form as in Fitts' law. More precisely, both the time needed to go 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 which lies a distance D away (Figure 2.b) are given by:

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

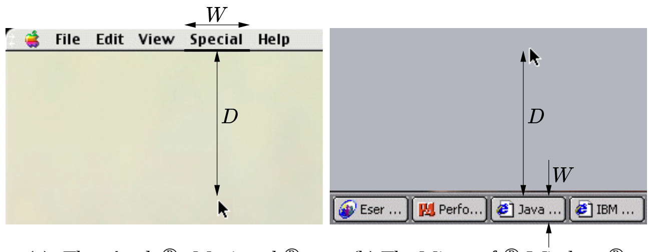
where a and b are experimentally-determined psychological constants. The logarithm factor in Equation 1 is called the index of difficulty (ID) of the pointing or crossing task.



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

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

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



(a) The Apple® Macintosh® menu bar

(b) The Microsoft® Windows® task bar

Figure 3: Accessing menu bar and task bar items on the Macintosh and Windows

The above model discovered in [2] on goal-crossing constitutes a necessary, but not a sufficient foundation for studying and designing crossing-based interfaces. First, it is not known if the coefficients in the above equation are identical for pointing and crossing tasks. If not, using (a_p, b_p) and (a_c, b_c) to denote the pointing and crossing performance respectively, how do these coefficients compare? Is it easier to point a target than cross an equivalent goal? Does the comparison depend on the index of difficulty? If crossing provides a superior performance, how would it be used in practice for the design of graphical interactive systems?

Upon examining the two tasks in Figure 2, we realized that they vary in two dimensions. One of the differences is pointing vs. crossing. The other difference lies in the direction of task constraint, either orthogonal or collinear to movement. Because the two directions of constraint could exist in both two tasks, we need a 2×2 factorial comparison.

One of these 2×2 combinations is pointing with orthogonal target constraint. Although pointing is typically studied with collinear constraint, pointing with orthogonal constraint does exist. For example (Figure 3), as noted by Tognazzini [13], "*Apple's menu bar has, in essence, infinite height, since the mouse pointer pins once it arrives.*" The width of these menu bars, however, is finite and hence become the main constraint, albeit in the orthogonal direction to cursor movement when the cursor is moved upwards to the menu bars. Tognazzini further remarked that "*the worst 'feature' of the Windows UI is their having the menu bar on line 2, instead of at the top of the display. This makes menu bar acquisition perhaps five to ten times slower than on the Macintosh.*" A model for pointing with orthogonal constraint, particularly if made comparable to the traditional Fitts pointing task with collinear constraint, would put the comparison between the menu bars in Macintosh and Windows on a more scientific ground.

Pragmatically, 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). On the other hand, when there are non-target 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). We therefore investigated both discrete and continuous crossing in a systematic study presented in the next section.

EXPERIMENT

Participants

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

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, 2540 lpi 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 was running in single-user mode, with only a few active system processes. It was disconnected from the network.

Procedure and design

Participants performed six different tasks, including two pointing tasks and four goal-crossing tasks. The two pointing tasks differed in the direction of movement variability constraint — one collinear and the other orthogonal to the main movement direction. The four goal-crossing tasks differed by both the constraint direction (collinear/orthogonal) and the nature of the action (discrete/continuous). The details of each condition are as follows:

CP *Pointing with collinear constraint* (Figure 4.a): This is the traditional Fitts' tapping task [3]. Participants click alternately on two vertical rectangles with width W and "infinite" height. The two target centers is separated by distance D. We call this task collinear pointing because the variability constraint (W) imposed on the termination of the movement is in the direction collinear to the hand movement.

OP *Pointing with orthogonal constraint* (Figure 4.b): This is a variant of Fitts' original tapping task. Participants click alternately on two horizontal rectangles of height W and "infinite" width (by one side), separated by distance D measured by the gap between the two targets. The variability constraint imposed on this task is in the direction orthogonal to the movement. To our knowledge, no performance model has been previously established for this task (or for the next three tasks).

D/CC *Discrete collinear crossing* (Figure 4.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, was drawn between the two goals to remind the participants to use discrete strokes for crossing the goals.

D/OC *Discrete orthogonal crossing* (Figure 4.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, i.e. a distractor, was drawn between the two goals to remind participants to lift up the stylus when traveling from one goal to the other.

C/CC *Continuous collinear crossing* (Figure 4.e): Participants alternately move the cursor through two horizontal goals of width W over distance D. The crossing task is called 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.

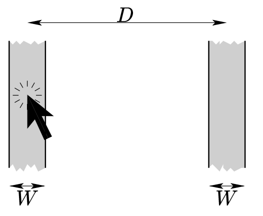
C/OC *Continuous orthogonal crossing* (Figure 4.f): Participants move the cursor reciprocally through two vertical goals of height W over distance D. Same as in condition *C/CC*, the system will beep when the stylus is lifted during a block of trials until stylus-tablet contact is resumed. This task was introduced in [2] and found to follow Fitts' law when performed non-reciprocally.

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. When hit, the target changed color from green to orange. The time at which participants successfully clicked on the target or crossed the goal was recorded.

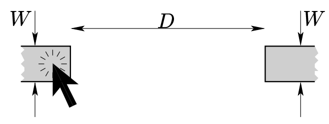
A within-subject full factorial design with repeated measures was used. The independent variables were the task type ($T =$ *CP, OP, D/CC, D/OC, C/CC, C/OC*), the distance D between targets or goals ($D = 256, 1024$ pixels) and the target/goal width ($W = 8, 16, 32, 64, 128$ pixels). For each task, participants 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 between six groups of participants according to a Latin square.

Results and Analyses

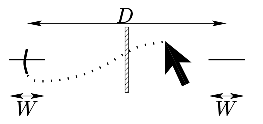
Learning, Time and Error Figure 5 shows the average trial time over the three experimental sessions. The average trial completion time in the practice session was longer than the time in the two data-collection sessions, due to participants' inexperience and occasional experimentation with the tablet-stylus 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.



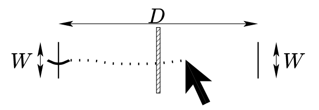
(a) ***CP*** — Pointing with collinear variability constraint



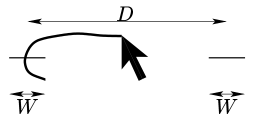
(b) ***OP*** — Pointing with orthogonal variability constraint



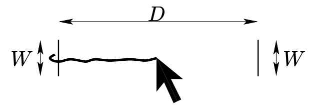
(c) ***D/CC*** — Discrete collinear goal-crossing



(d) ***D/OC*** — Discrete orthogonal goal-crossing task



(e) ***C/CC*** — Continuous collinear goal-crossing



(f) ***C/OC*** — Continuous orthogonal goal-crossing

Figure 4: The six tested conditions. All tasks were reciprocal.

Variance analysis showed that mean trial completion times were significantly different across the six tasks ($F_{5,55} = 12.5, p < .001$). Task *C/CC* was the slowest (see Figure 5). Tasks *OP* and *C/OC* 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.

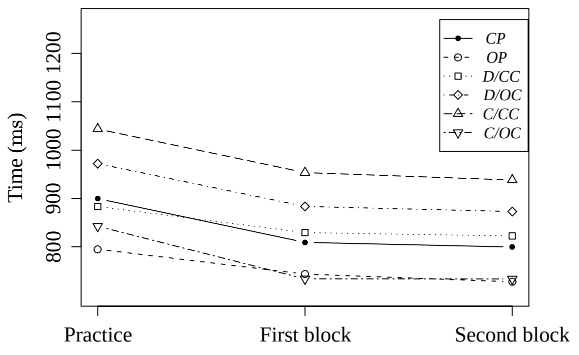
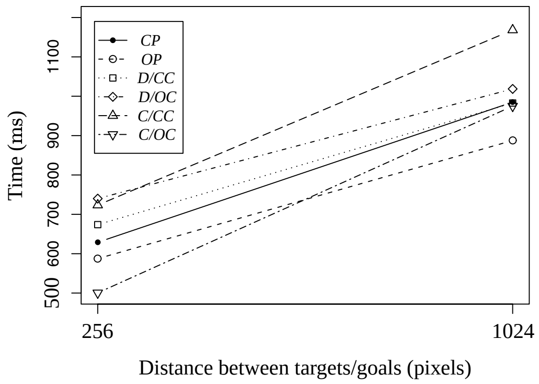


Figure 5: Learning effect on average completion time

As illustrated in Figure 6.a, movement distance significantly changed mean trial completion ($F_{1,11} = 898, p < .0001$). Across all tasks, the greater the distance between targets, the longer time duration of the trial. As illustrated in Figure 6.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.

(a)


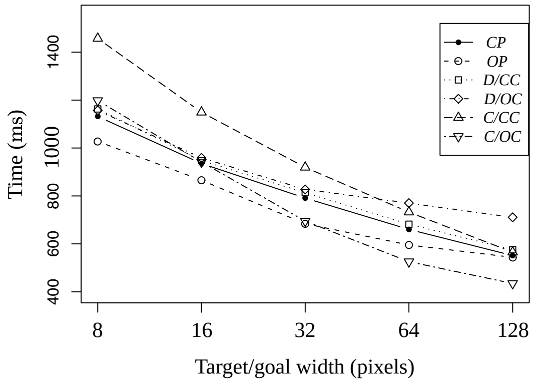
(b)


Figure 6: Effect of distance and width on task completion time

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 7, except for Task *C/CC* (9.2%), all new tasks studied in this experiment had error rates close to and lower than that of Fitts' tapping task (*CP*, 7.6%). *D/OC* has the lowest error rate with 2.8%.

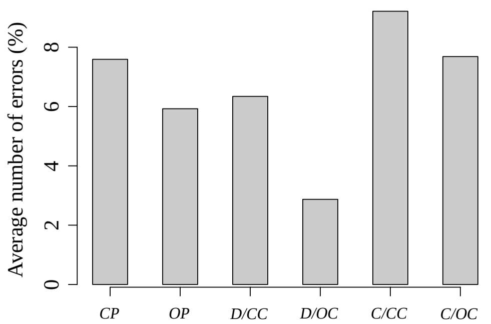


Figure 7: Error rates for each task

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.

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 (all time values in this paper are in milliseconds):

$$\textit{CP}:\quad T = 103 + 172 \times ID \quad r^2 = 0.998 \tag{4}$$
$$\textit{OP}:\quad T = 145 + 146 \times ID \quad r^2 = 0.986 \tag{5}$$
$$\textit{D/CC}:\quad T = 155 + 165 \times ID \quad r^2 = 0.994 \tag{6}$$
$$\textit{D/OC}:\quad T = 342 + 133 \times ID \quad r^2 = 0.975 \tag{7}$$
$$\textit{C/CC}:\quad T = -41 + 242 \times ID \quad r^2 = 0.995 \tag{8}$$
$$\textit{C/OC}:\quad T = -196 + 235 \times ID \quad r^2 = 0.984 \tag{9}$$

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 form of logarithmic transformation of D/W ratio as in Fitts' law, with very high fitness values (r^2 ranging from 0.975 to .998).

Figure 8 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 (see [9]).

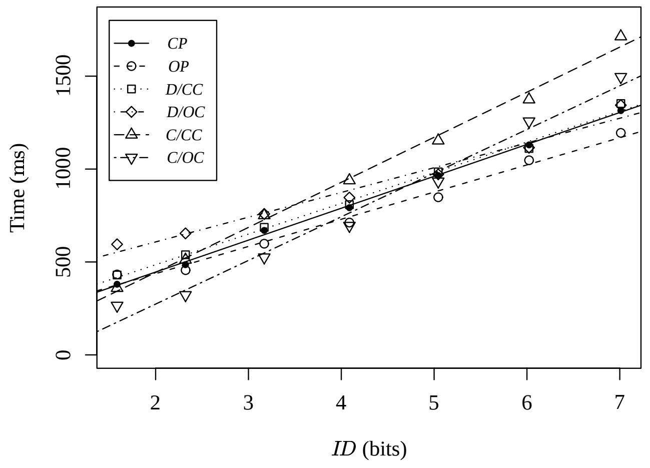


Figure 8: The fit to the model for the six studied tasks

Task Comparison Although relatively small, there were important differences between the six tasks studied in this experiment. Due to space limitations, we report only the analyses of a few comparisons most relevant to human-computer interaction tasks (illustrated by Figure 9).

First, Task *OP*, i.e. pointing with orthogonal constraint, had similar performance to Task *CP*, i.e. pointing with constraint collinear to the movement direction (the traditional Fitts' tapping task). On average, *OP* was 10% faster than *CP* (Figure 5). As displayed separately in Figure 9.a, starting from essentially zero, the difference between *OP* and *CP* increased with Index of Difficulty. It appeared that variability constraints orthogonal to the pointing movement direction is "easier" to deal with than constraints in the same direction as the movement. 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. The greater the *ID* (either smaller width or greater distance), the more pronounced this effect (see *CP* and *OP* lines in Figure 6.a).

Second, the two discrete crossing tasks, with collinear and orthogonal constraint respectively, followed similar regression lines to the standard Fitts' tapping task, as separately shown in Figure 9.b, suggesting that it is possible to substitute pointing tasks with crossing tasks with essentially the same time but lower error rate (see Figure 7). Between the two discrete crossing tasks, there was a trade-off swung by *ID*. *D/OC* tended to be faster than *D/CC* when *ID* was greater than 6 bits, and the reverse was true when *ID* was less than 5 bits. The fact that *D/OC* was slower than *D/CC* in low *ID* can be partially explained by the "obstacle" line positioned between the two goals (see Figures 4.c and 4.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 (*CP*) involved here with increasing difficulty. This is not true to the *D/CC* task because the stylus always lands above the horizontal goals. This effect is partly an experimental manipulation artifact, and partly a reflection of 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 much smaller number of errors were made with the *D/OC* task than with the *D/CC* task.

Third, there was a trade-off between *C/OC* and *D/OC* tasks (Figure 9.c). *C/OC* was much faster than *D/OC* when *ID* was much lower than 5 bits, and *C/OC* was longer than *D/OC* when *ID* was greater than 5 bits. This again was partly due to the obstacle effect in the *D/OC* task. There was no obstacle in the *C/OC* task. When the two goals were very close to each other, it was possible to cross the two in one stroke. This is an advantage that should be taken in interface design whenever possible.

Fourth, the *C/CC* task was uniformly slower than the *C/OC* task (Figure 9.d). It also has the highest error rate (Figure 7). This suggests that, whenever possible, the goals to be crossed should be positioned orthogonal to the movement direction.

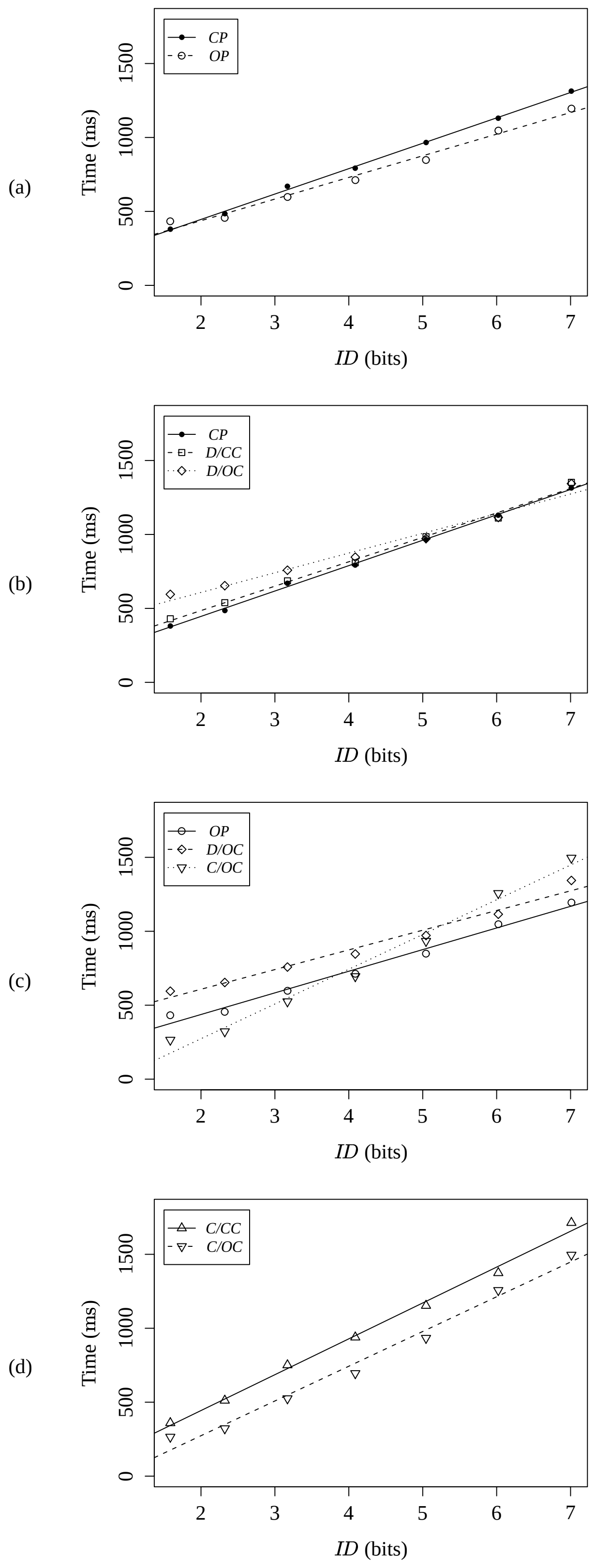


Figure 9: Tasks comparisons

Note on the *OP* task

The *OP* task can be parameterized in two ways. The one we used, which defines the movement amplitude as the distance between the two edges of the horizontal targets (Figure 4.b), has the advantage of being closer to the orthogonal goal-crossing tasks. But it also has a drawback: when performing the task, the actual movement amplitude is a little higher than the one controlled in the experiment, possibly making the pointing task more difficult than predicted. A way to cope with this issue is to introduce the concept of effective amplitude, similar to the idea of effective width (see [9]), and use the average actual movement amplitude to compute the difficulty of the task. The drawback of this method is that the difficulty is computed *a posteriori*, which goes against the idea of *a priori* performance modeling and prediction. A second way is to change the definition of distance between targets. A definition that appears particularly suited is to add half a target width on both sides of the movement amplitude in Figure 4.b, as illustrated by Figure 10. This new definition has an increased compatibility with the *CP* task (Figure 4.a): when D is equal to W, the two targets are adjacent and the task consists in merely jumping over a single line from one target to the other; when D is equal to 0, the two targets are partially overlapping, and the task can be performed by clicking repeatedly without even moving. This definition of the *OP* task may then be more adequate for comparing performance within the pointing task class. With this definition of movement amplitude, the linear regression between time and index of difficulty is:

$$OP^\star: \qquad T = 75 + 158 \times I\!D \qquad r^2 = 0.992 \tag{10}$$

The fitness is even greater than the one of Equation 5, suggesting that the variant definition may be a more appropriate model. If we use this model, conclusions regarding *OP* task in relation to others in this paper are qualitatively the same but quantitatively stronger.

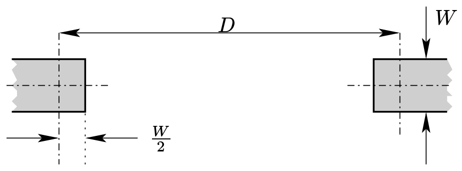


Figure 10: A variant of the *OP* task, with a different measure of movement amplitude

APPLICATIONS AND DISCUSSION

As with other laws of natural phenomena, the regularities of human movement revealed in this study can be applied in ways and scope beyond expectations at the time of discovery. In this section we explore only a few obvious implications of these movement laws to the design of human computer interfaces.

The *OP* task and the primacy of border locations

As we found in the experiment, an *OP* task takes the same or less time than a *CP* task for the same amount of constraint W (Figure 4.a, 4.b, Figure 9). This can be very informative to interface designers. When choosing the orientations and locations of interaction objects (widgets), they can use this empirical relationship to estimate the impact of their choices.

Consider for instance the two pointing tasks shown in Figure 3 and discussed in the Introduction section. For the Macintosh menu bar (Figure 3.a), the menus are virtually infinitely extended beyond the edge of the screen where the mouse cursor is stopped. The primary pointing variability constraint being orthogonal to the movement direction, pointing at these menus is an *OP* task, modeled by Fitts' law with $a_{OP} = 75$ and $b_{OP} = 158$ (Equations 1 and 10). On the other hand, since the buttons of the Windows task bar are not pushed onto the very edge of the screen, the pointing task has both orthogonal and collinear constraints (see Figure 3). The collinear variability allowed here is much smaller than the orthogonal one, such that pointing a button is mainly a *CP* task (see also [6, 10, 11, 14]) and modeled by Fitts' law with $a_{CP} = 103$ and $b_{CP} = 172$ (Equations 1 and 4). Now let us assume we want to select an item of width 160 pixels and height 40 pixels[1]. Then, for a movement distance of D, the time to select a Macintosh menu item is:

$$T = \ 75 + 158 \log_2 \left(\frac{D}{160} + 1 \right) \tag{11}$$

while the time to select a Windows task button is:

$$T = 103 + 172 \log_2 \left(\frac{D}{40} + 1 \right) \tag{12}$$

These two time models, plotted in Figure 11, clearly show the advantage of the Macintosh menu bar over the Windows task bar, the former being about twice as fast as the latter for most amplitudes.

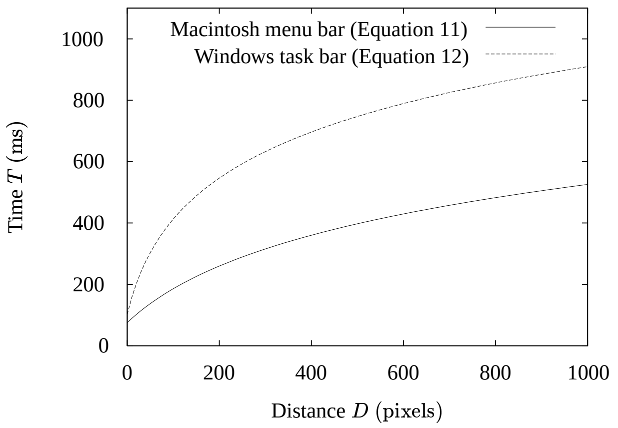


Figure 11: Comparison of the access time for the Macintosh menu bar and the Windows task bar

This simple exercise shows the primacy of the border locations of the computer screen real estate. Widgets on the border locations should take the maximum advantage of extending beyond where the cursor stops. To take another example, scrollbars on the very edge of screen should also extend beyond the screen pixels.

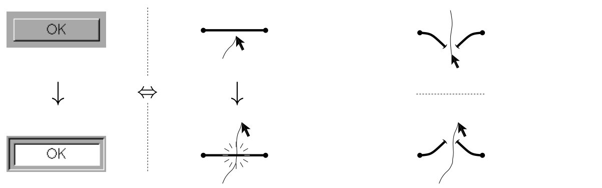
Note that fundamentally all 2D widgets are constrained in two dimension and pointing at these targets is ultimately "bivariate". The above analysis focuses on only the primary

[1] As a reference, a standard task button in Microsoft® Windows NT® has a width of 320 pixels and a height of 44 pixels.

constraint, which is valid only when the objects are elongated. For objects that are close square, more empirical work is needed (see [10] for a previous investigation).

Crossing Interfaces

Perhaps the most important HCI implications of the current study is a paradigm of interaction based on goal crossing interfaces where the basic interactive widget is a goal (1D bar) instead of target (2D area). Just as Fitts' law in the Fitts' tapping task has served as a foundation for analyzing pointing based interface, the laws of goal-crossing tasks can be the basis for designing and analyzing interfaces based on goal crossing (Figure 12.a).



(a) To trigger an action: on the left we push the button; on the right we cross the goal.

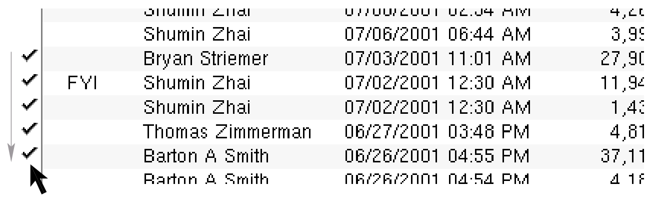
(b) Unlike a traditional check box, a goal can "store" two visual states depending on the crossing direction.

Figure 12: Using goal-crossing in graphical widgets

Quantitatively, as the results of the experiment show, there are situations where user performance in goal-crossing tasks is superior to that of target-pointing tasks. Depending on the idiosyncrasies of the particular cases, user interface designs can be grounded on the relative pros and cons of different types of pointing or crossing tasks. In some cases pointing is preferred; in other cases, a particular type of crossing (discrete or continuous) is preferred.

Qualitatively, crossing — a continuous action occurring more often in the natural world than discrete pointing and clicking — affords some unique characteristics. First, crossing can be bi-directional (Figure 12.b). This means that richer semantics can be assigned to crossing than pointing. Further, it is also possible to cross a goal back and forth once (double crossing) or even more than once to represent a variety of commands. Indeed, in some sense crossing is a step from pointing towards free gesturing, which is even richer but harder to quantify and disambiguate than crossing.

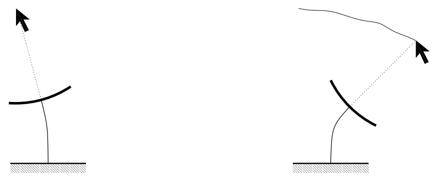
Another characteristic of crossing is that multiple goals can be crossed in a cascaded fashion. Such an application can in fact be found in existing software products. For example, in Lotus Notes®, multiple messages can be selected by one continuous cursor drag, crossing a series of implicit goals (Figure 13). To select many messages that are not consecutive, one can first select a whole set of continuous messages, and then deselect the ones that are not wanted. Such mixed pointing and crossing performance can be quantified by a combination of the laws discussed in the previous section. In comparison, using pointing alone to select multiple messages would require very precise and repetitive pointing actions. It is possible to use a bounding box technique to select multiple targets and then use pointing to activate all of them, but this is less fluid than one continuous crossing stroke.



		Shumin Zhai	07/06/2001 02:34 AM	4,2
		Shumin Zhai	07/06/2001 06:44 AM	3,9
✓		Bryan Striemer	07/03/2001 11:01 AM	27,9
✓	FYI	Shumin Zhai	07/02/2001 12:30 AM	11,9
✓		Shumin Zhai	07/02/2001 12:30 AM	1,4
✓		Thomas Zimmerman	06/27/2001 03:48 PM	4,81
✓		Barton A Smith	06/26/2001 04:55 PM	37,11
		Barton A Smith	06/26/2001 04:54 PM	4,1

Figure 13: Selection of multiple messages by a continuous goal-crossing action in Lotus Notes

While it is possible to have both orthogonal and collinear crossing, our experiment clearly showed the advantage of orthogonal crossing. Figure 14 illustrates the possibility of dynamically orienting the goals towards the cursor to maximize this advantage (a "sunflower" interface, so to speak).

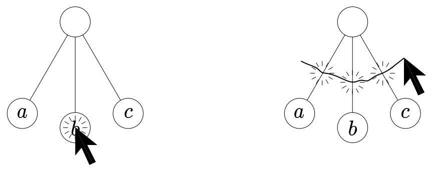


(a) As the cursor moves, ...

(b) ... the goal rotates to offer maximum width

Figure 14: Dynamically orienting goals may minimize crossing time for any cursor position

It should be noted that pointing and crossing do not have to be mutually exclusive. It is in fact possible to have a "double representation" interface where both actions are enabled. To select individual objects a, b, or c in Figure 15, the user can either point and click on the circles or cross the links, depending on subjective preference. To select a group of objects, the user may cross multiple links in one stroke, taking advantage of the cascading property of crossing.



(a) Pointing at targets

(b) Crossing the arcs

Figure 15: Input polymorphism: choosing the input method that is best suited to the context

Special application domains of crossing interfaces

While goal-crossing based interaction techniques can be used in any graphical user interface, there are several applications to which they are particularly suited.

One follows the idea of double representation for universal accessibility of interfaces. Some users have gotten used to pointing and will continue their habitual mode of action. Others, including some elderly users and users with certain motor disability may have difficulty with clicking without moving the cursor position, or worse, double clicking. For these users crossing may become the main interaction mode.

The second domain of special application is interfaces for mobile or handheld devices. These devices usually come with a stylus (pen), which is more suited for crossing actions than a mouse. Furthermore, double-click with a stylus is often very difficult for any user. The more compact representation of bars or links in a goal crossing interface also requires less of the small screens in these devices.

Third, the text of hyperlinks in web pages are especially difficult for pointing due to the their narrow height (one line of text) but easy for crossing due to the much greater width of (one word long and often much longer). It is also possible to cross a list of links in a cascade (Figure 16). This in fact has been demonstrated in the "Elastic window" prototype [7].

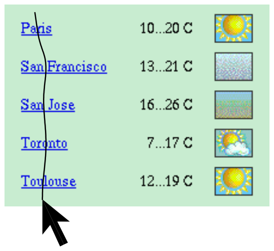


Figure 16: Crossing multiple city names to get their detailed weather forecast

Finally, traditional GUI widgets are very difficult to be integrated in virtual reality type 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. [4]).

CONCLUSION

In this paper, we studied human performance in both pointing and crossing tasks, with 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 the target width in all six tasks studied. If we refer to natural laws by their mathematical appearance, we can say all of the six tasks follow Fitts' law. If we refer to natural laws by the task (or phenomenon) they describe, we can say there is a law to each of the six tasks. Many subjective impressions, such as the superior usability of the menus on the screen border, can be in fact quantified by some of these laws. Most significantly, together with the qualitative characterizations of the crossing action, these laws lay the foundation for designing crossing-based user interfaces in which crossing, instead of or in addition to pointing, is the fundamental interaction technique.

ACKNOWLEDGMENTS

We would like to thank Barton A. Smith, Paul Mauglio and Teenie Matlock for their helpful comments on this work.

REFERENCES

1. J. Accot. *Les Tâches Trajectorielles en Interaction Homme-Machine — Cas des tâches de navigation*. PhD thesis, Université de Toulouse 1, France, January 2001.

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

3. P. M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47:381–391, 1954.

4. M. Haisenko and P. Musgrave. The 3Dsia project. `http://threedsia.sourceforge.net`, 2001.

5. K. Hinckley, R. Pausch, J. C. Goble, and N. F. Kassell. A survey of design issues in spatial input. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 213–222, 1994.

6. E. R. Hoffmann and I. H. Sheikh. Effect of varying target height in a Fitts' movement task. *Ergonomics*, 37(6):1071–1088, 1994.

7. E. Kandogan and B. Shneiderman. Elastic windows: A hierarchical multi-window world-wide web browser. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 169–177, 1997.

8. A. Leganchuk, S. Zhai, and W. Buxton. Manual and cognitive benefits of two-handed input: An experimental study. *ACM Transactions on Computer-Human Interaction*, 5(4):326–359, Dec. 1998.

9. I. S. MacKenzie. Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, 7:91–139, 1992.

10. I. S. MacKenzie and W. Buxton. Extending Fitts' law to two-dimensional tasks. In *Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems*, pages 219–226, 1992.

11. A. Murata. Extending effective target width in Fitts' law to a two-dimensional pointing task. *International journal of human-computer interaction*, 11(2):137–152, 1999.

12. X. Ren and S. Moriya. Improving selection performance on pen-based systems: a study of pen-based interaction for selection tasks. *ACM Transactions on Computer-Human Interaction*, 7(3):384–416, 2000.

13. B. Tognazzini. Club Wired — Bruce Tognazzini Transcript. `http://hotwired.lycos.com/talk/club/special/transcripts/95-06-14.tog.html`, June 1995.

14. N. Walker and J. B. Smelcer. A comparison of selection times from walking and pull-down menus. In *Proceedings of ACM CHI'90 Conference on Human Factors in Computing Systems*, pages 221–225, 1990.

15. S. Zhai, W. Buxton, and P. Milgram. The "silk cursor": Investigating transparency for 3D target acquisition. In *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems*, pages 459–464, 1994.