Object Pointing: A Complement to Bitmap Pointing in GUIs

Yves Guiard [a] Renaud Blanch [b] Michel Beaudouin-Lafon [b]

[a] Laboratoire Mouvement et Perception
CNRS & U. de la Méditerranée, Marseille, France
guiard@laps.univ-mrs.fr

[b] LRI & INRIA Futurs
Université Paris-Sud, Orsay, France
{blanch, mbl}@lri.f

Abstract

Pointing has been conceptualized and implemented so far as the act of selecting pixels in bitmap displays. We show that the current technique, which we call bitmap pointing (BMP), is often sub-optimal as it requires continuous information from the mouse while the system often just needs the discrete specification of objects. The paper introduces object pointing (OP), a novel interaction technique based on a special screen cursor that skips empty spaces, thus drastically reducing the waste of input information. We report data from 1D and 2D Fitts' law experiments showing that OP outperforms BMP and that the performance facilitation increases with the task's index of difficulty. We discuss the implementation of OP in current interfaces.

Key words: Input and Interaction Technologies, Pointing, Fitts' law.

1 Introduction

Since the advent, about two decades ago, of graphical user interfaces (GUIs), which led to the so-called WIMP paradigm (Windows, Icons, Menus, and Pointers) for human-computer interaction (HCI), sustained efforts have been made to optimize pointing, the key elemental act that permits selection among graphical objects such as icons, buttons, menu items, or hypertext links.

HCI researchers realized only recently that interface designers can pursue a more ambitious goal than just making pointing to a graphical object as easy as pointing to a real-life object: the emerging challenge is to make pointing in GUIs *easier* than normal [4, 5, 8, 14, 17].

It has been known, from the beginning of the WIMP era [6] that target acquisition time (or movement time, *MT*) in GUIs is almost entirely determined by the ratio of target distance *D* and target width *W*, as stated by Fitts' law [7, 15]:

$$MT = a + b \log_2 (D/W + 1), \quad (1)$$

with *a* and *b* standing for empirically adjustable coefficients ($b>0$) and the expression $\log_2(D/W +1)$ defining what Fitts termed the index of difficulty (*ID*).

Equation 1 suggests two non-exclusive ways of facilitating target acquisition in a GUI, both of which have been recently investigated in HCI research. One is to reduce *D*. If, as soon as the system detects cursor motion, it helps by shifting the set of objects that is likely to include the target toward the approaching cursor, the numerator of Equation 1 drops. This is the solution investigated in the Drag-and-Pop technique [4]. Second, the system may expand whatever object is being approached by the cursor, thus increasing *W* and hence reducing the *ID*. With this solution, implemented in the Mac OS X Dock,[1] performance can be facilitated even if the expansion is very late [14,17].

Semantic pointing [5], a more recent treatment of the pointing facilitation problem, simultaneously attacks the numerator and the denominator of the *D*/*W* ratio. The display-control (DC) ratio linking cursor motion to mouse motion is made to depend on cursor position across the landscape of graphical objects: the default DC ratio being set to a high level, each object is surrounded by a 'well' of reduced DC ratio. Thus, while mouse amplitude is saved for large cursor motion toward objects (i.e., *D* is reduced), more amplitude is needed in the vicinity of the target (i.e., *W* increases). One attractive feature of semantic pointing is that, unlike previous attempts [4, 14], it facilitates performance without the cost of perturbing the visual display.

The above solutions do have a potential for facilitating pointing, but they take it for granted that a screen cursor is a tool for pixel selection. Below we question this basic assumption.

2 Input Information Waste in Current GUIs

The present study was triggered by the observation that in current GUIs the amount of information received by the system is generally far less than that emitted by the mouse. Consider a graphical desktop with 40 icons, each 20x30 pixel (px) large, on a 1600x1200px screen. Selecting one of these icons means selecting 600px

[1] However, the Dock implementation of this technique fails to increase actual pointing tolerance [17].

within a set of 1,920,000px, and this amounts to sending $\log_2(1{,}920{,}000/600)=11.6$ bits to the system. The system, however, just needs the specification of one icon among the 40 icons, that, $\log_2 40=5.3$ bits. So in this example, representative of many real HCI situations, the system will use hardly half of the information produced in pointing: a substantial proportion of the input information is wasted. This waste amounts quantitatively to $\log_2(S_s/S_o)$ - $\log_2 N$, with S_s denoting the surface area of the screen and S_o the surface area of the objects (assumed to be all the same size), and with N denoting the number of displayed objects.

Obviously, what is ignored by the system is unnecessary information. Consider the case in which the graphical desktop is entirely tiled with objects, leaving no empty spaces. In that limiting case, we have $\log_2(S_s/S_o)=\log_2 N$. If, however, there are empty spaces, as is the case in many real displays, then the information emitted with the mouse will necessarily exceed that received by the system, and the larger the proportion of empty space in the display, the more bits wasted. So it is the information delivered by cursor motion through the voids of our graphical displays that the system (justly) ignores.

Put differently, the continuous input from the hand contains a proportion of gratuitous information. As the cursor crosses an empty region of the display, the system keeps on updating the cursor position to reflect mouse motion. However, this is information just for the user—not to the system, which has to keep waiting for some discrete selection. The gratuitous component of pointing, we believe, lies in the fact that current GUIs force the user to select pixels whereas the system, more often than not, just needs the selection of discrete objects.

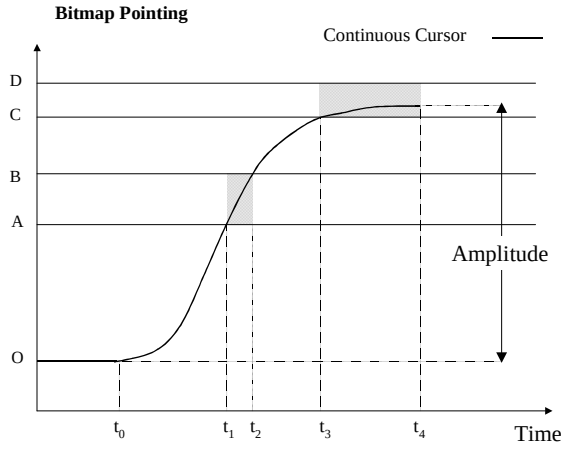

Figure 1: Cursor motion in an elemental 1D pointing movement, using the current BMP mode.

Our GUIs being based on bitmap graphics, let us call the current pointing mode *bitmap pointing* (BMP). Figure 1 shows what happens when the cursor moves to some object to select it. Time elapses along the figure's horizontal axis. For the sake of simplicity, assume pointing to take place on a 1D desktop, and so space can be represented on the vertical axis. Suppose that the desktop displays two targets, which occupy intervals [AB] and [CD], and the cursor's target is object [CD]. The starting point is O.

At first, suppose the cursor is stationary. At time t_0, when the cursor starts to move upwards, anyone—and hence the software that controls onscreen cursor motion—could safely guess that the user is starting a movement that will end up in either object [AB] or object [CD]: the only uncertainty left at t_0 is target identity (1 bit, ignoring pointing abortion).

The system gains no information as the cursor crosses the empty space interval [OA]. At t_1, when the cursor reaches the proximal boundary of the first object, this object gets highlighted (gray shading in the figure), signaling that if a click or an ENTER key press were to happen now, the system would activate that particular object. However, this is information *from* the system—not to the system. What is occurring at t_1 was trivially predictable from t_0 because [AB] had to be reached anyway, regardless of whether or not it was the movement's target. The message emitted by the user over the $[t_0, t_1]$ interval is essentially redundant.

In fact, the single bit of information the system has been waiting for since t_0 is delivered at time t_2, when the cursor crosses the upper boundary of target [AB]: at that moment, the cursor is moving rather fast (i.e., the slope of the trajectory is steep), so there is no doubt that the cursor is leaving the object—the probability that the cursor aims at target [CD] switches to 1. Had its velocity been low and its acceleration negative (braking) at t_2, one would have been able to guess that the target was interval [AB] with an overshoot. At any rate, the remainder of cursor motion from t_2 to t_4 will deliver no information whatsoever. Yet the standard BMP technology demands that users carefully finish up their movement, even though the final deceleration of a targeted movement is known to be the most costly phase, in terms of time and control effort [16].

3 Object pointing

We now introduce what we call *object pointing* (OP). The OP mode involves an extra cursor, which we call the 'timorous' cursor (Tim) because it behaves on the screen as if it were 'void'- phobic. Tim's original feature is that it *never* visits empty regions of graphical space, thanks to an algorithm that uninterruptedly analyzes its kinematics. As soon as the algorithm detects that Tim has left an object, it identifies the current direction of Tim's motion and makes it jump to the first object located in that direction. However, as long as Tim is within an object, it moves normally in parallel to the standard, continuous cursor. Note that in OP mode, the

usual system cursor is ineffective for object selection: only Tim's visit to objects can cause their highlight.

3.1 Object Pointing in 1D space

Figure 2 illustrates OP in the case of the simple 1D pointing task of Figure 1. We suppose the user is initially in BMP mode. At time t_0, with the standard, continuous cursor still at rest, the user switches to the OP mode. Tim could have popped up at the same location as the standard cursor, but because it is not within an object, it immediately jumps to the nearest boundary of the nearest object, causing this object to highlight. At t_1, the mouse starts to move, causing the two cursors, now in different locations, to move upward in parallel, with Tim 'safely' within the object [AB]. At t_2, Tim reaches point B, the upper boundary of the object, and hence jumps instantaneously to point C, the proximal boundary of object [CD], in which it can resume its continuous course parallel to the standard cursor. Note that when Tim reaches point D, the mouse is still pushing it upward (as revealed by the trajectory of the other cursor), but there is nothing beyond object [CD] and so Tim stays at D.

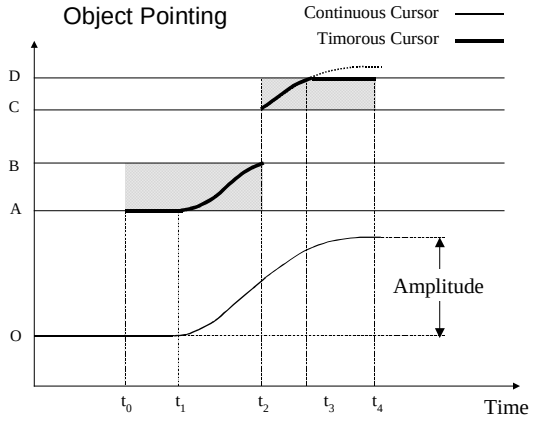

Figure 2. Tim's jumping course in the OP mode, along with the continuous trajectory followed by the standard cursor.

This rather simple example in 1D space captures the main properties and potentialities of OP, which still hold in 2D or 3D space. One noteworthy property is that neither of the two screen cursors need to be shown to the user. Displaying the standard cursor in OP mode is superfluous because this cursor is ineffective. As for Tim, it is always in some object and that object is highlighted, so at every single instant the user receives the appropriate visual feedback, in the form of a highlight that jumps from object to object as the mouse moves.

The main advantage of the OP mode is that it makes pointing work as though all the objects of the display had been packed together as a compact keyboard. Therefore distance as measured on the screen no longer constrains the *ID*. The only distance Tim actually needs to cover to reach its target is the sum of the widths of the obstacles it may have to cross. Obviously this distance depends on object density in the display, but it can only be shorter than *D*. Compare in Figures 1 and 2 the amplitudes covered by the continuous cursor, which in either mode faithfully reflects mouse movements.

Second, if the target is the last object in the direction of the motion, OP makes pointing tolerance infinite in this direction. Thus, in Figure 2 the user slightly overshoots object [CD], but this error has no impact on target acquisition—after t_3, Tim stays at the boundary of the object, which therefore remains highlighted.

Kinematic Conditions for the Jump. The algorithm we use for the control of Tim's jumps in OP mode considers not only Tim's current position, but also its instantaneous velocity and acceleration at the time it is found to leave an object. As soon as Tim is found to occupy a pixel that does not belong to some object, the algorithm evaluates whether two kinematic conditions are met. First, velocity must have reached a threshold: if not, Tim returns to the object's boundary it has just left—so no jump will take place when Tim is just wandering in the vicinity of some object. Second, Tim's current velocity must not be dropping too steeply: if Tim is decelerating while leaving an object, it returns to the boundary of the previous object—so no jump can take place in the case of a mere overshoot during object landing. If the above two kinematic conditions are met, then the algorithm starts to evaluate if a jump is in order.

3.2 Object Pointing in 2D Space

The basic features of OP are essentially the same in 1D, 2D and 3D spaces. Below, we focus on the 2D case, keeping in mind the importance of 2D displays in current GUIs.

Identifying the Target Object. In 2D space the direction of Tim's motion must be analyzed in angular terms. Provided the position, velocity and acceleration requirements are met, our algorithm performs a simple linear regression analysis using a small sample of *xy* coordinates from the recent history of Tim's positions (e.g., the last five samples). Once the instantaneous direction of Tim's motion has been identi-fied, the algorithm determines an angular sector centered around Tim's current direction to search for a new object. If the search sector detects at least one object, Tim jumps to the proximal boundary of the nearest object. If the search is unsuccessful, the angular sector is incremented and the search starts again, up to a certain maximum angular sector. If the search fails to find any object in the largest angular sector, then Tim returns to its departure point.

Eye movements. One argument for the workability of OP is the well-known behavioral fact that in pointing tasks

the gaze precedes the hand. Typically, the target becomes the gaze fixation point before or at about the time the cursor starts to move [1,12]. This means that in OP mode, there should be no difficulty for the user to follow the jumping motion of the highlight across the layout of objects (regardless of whether or not Tim is visible): the user's task will be to move the highlight from the periphery of the visual field to her/his current point of fixation.

4 Experiment 1: Reciprocal Pointing in 1D Space

The goal of Exp.1 was to quantitatively assess OP vs. BMP performance in a highly simplified laboratory task so designed as to allow a rigorous experimental control over a minimal number of relevant variables. We investigated three pointing conditions: the standard BMP mode (baseline) and two variants of the OP technique, one with the timorous cursor permanently visible (OP_c) and the other with no cursor at all (the standard, continuous cursor was never displayed).

4.1 Dependent Variables

Movement time (*MT*), defined as the time elapsed between two correctly located clicks, was the main dependent variable. The other dependent variables were the amplitude of on-screen cursor motion, the amplitude of the hand movement, and the subjective rating of the difficulty of pointing in the various conditions.

4.2 Independent Variables

Exp.1 was run in two parts (Table 1). Exp.1A served to manipulate *Task difficulty* by just varying *W* at a constant level of movement amplitude, with *D* set to a constant 800px, to avoid any experimental confound between the effects of difficulty and scale [10]. We used three levels of *ID*: 3.0, 5.5, and 8.1 bits (i.e., *W*=114, 18, and 3px, respectively).

Table 1. Task parameters for Exp.1. Underlined are the values that were expected to be problematic for participants.

	Screen Space (pixels)		Tablet Space (mm)		Task Difficulty	
	D	*W*	*D*	*W*	Ratio *D*/*W*	*ID* (bit)
Exp. 1A	800	114	238.7	34.0	7	3.00
	800	18	238.7	5.4	44	5.51
	800	3	238.7	0.9	267	<u>8.06</u>

	Screen Space (pixels)		Tablet Space (mm)		Task Difficulty	
	D	*W*	*D*	*W*	Ratio *D*/*W*	*ID* (bit)
Exp. 1B	10	1	3.0	<u>0.3</u>	10	3.46
	250	25	74.6	7.5	10	3.46
	1230	123	<u>367.0</u>	36.7	10	3.46

Exp.1B served to manipulate *Task scale* by varying *D* and *W* proportionally, so that the *ID* was a constant 3.46 bits (see Table 1). We used three levels of scale. For the lowest, *D* and *W* were set to 10 and 1px, respectively, creating a miniaturized task—the problematic feature was obviously *W*=1px, corresponding to 0.3mm on the tablet. For the next scale level, *D* and *W* were set to a comfortable 250 and 25px, corresponding on the tablet to 75 and 7.5mm, respectively. In the third condition, *D* and *W* were set to 1230 and 123px to force the puck to cover a rather large 367mm amplitude on the tablet. This extended scale range was expected to give rise, in the baseline BMP condition, to an optimum, V-shaped, relationship between *MT* and scale. Below, *D* identifies task scale (but recall that in Exp.1B *W* always equaled *D*/10).

Since display-control gain was a constant 3.35px/mm in Exp.1, the scale manipulation of Exp.1B simultaneously affected display size and movement amplitude on the tablet (at least for BMP). We wished to evaluate OP in situations that capture the properties of real-world interfaces such as PDAs (resp. wall interfaces), which miniaturize (resp. magnify) both the display and the movement.

4.3 Hypotheses

Three working hypotheses were derived for Exp.1 from our theoretical analysis of OP.

- Overall, pointing performance should be faster in OP mode than in standard BMP mode, thanks to the reduction of movement amplitude and—both targets of the reciprocal pointing paradigm being peripheral—the increase of target tolerance.
- In OP mode, the slope of Fitts' law should be zero, because the variations of *D* and *W* are handled with immaterial time costs by the system. Since, in contrast, *MT* is known to follow Fitts' law in the standard BMP mode, one predicts that the higher the *ID*, the larger the superiority of OP over BMP (Exp.1A).
- In OP mode, *MT* should become scale-insensitive for the same reason as above. Since both interface miniaturization and magnification are known to impair BMP performance [3], the larger the deviation from optimal size in either direction, the larger the expected superiority of OP over BMP (Exp.1B).

4.4 Methods

Equipment. Exp. 1 was run on a PC running UNIX with a screen set to a 1600x1200 pixel resolution. The input device was a puck to be moved on an A3 Intuos 12x18' Wacom tablet programmed in relative mode.

Display and Task. We used Fitts' (1954) classic 1D reciprocal-pointing paradigm [7]. The participants had to click back and forth on two dark-blue-colored vertical strips that extended from the top to the bottom of the screen, appearing in full-screen mode on a black background. The pointing movement had to be controlled exclusively along the left-right dimension. The target turned light-red when it was reached by the cursor, as if high-lighted—a necessary feature for the OP mode to work in the absence of any visible cursor. The cursor that controlled the highlight was a 1-pixel thick

vertical line drawn in white from top to bottom. In the BMP condition the line cursor could be moved continuously across screen space. In contrast, in the two OP conditions the line cursor—which could be moved continuously within a target— could only jump from one target to the other. The cursor was displayed in the OP_c condition but not in the OP condition. Note that a miss, being ineffective, had to be immediately corrected, simply resulting in an increased *MT*: error rate, a constant 0%, can be ignored.

Design and Procedure. Exp.1A involved three pointing modes (BMP, OP_c with Tim visible, and OP with Tim invisible) and three *ID*s, yielding a nine-cell within-participant experimental design, which we recycled in Exp.1B with scale replacing the *ID* (Table 1).

Twelve unpaid volunteers participated in the experiment. However, due to errors in handling log files, we were left with a sample of 9 participants (Exp.1A) and 8 participants (Exp.1B) to assess the performances. Each participant ran two sets of 27 blocks of trials, the first set for Exp.1A and the second for Exp.1B. Each block comprised 10 movements, and so Exp.1 totaled 540 target acquisitions per participant. Within each set of 27 blocks, order effects were counterbalanced over conditions with Latin squares. The two sets, separated by a comfortable rest, made up a session that lasted approximately one and a half hour.

4.5 Results and Discussion

Performance Speed. Figure 3 shows mean *MT* as a function of task difficulty for each pointing condition.

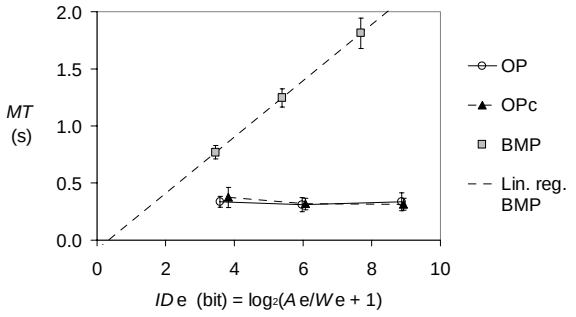

Figure 3. Effect of effective task difficulty on movement time, for each pointing technique (Exp.1A). Error bars on the Y axis represent α=.05 confidence limits based on between-participant standard deviations (SD).

The variable on the *x* axis is the *effective ID*, $ID_e=\log_2(A_e/W_e +1)$, where A_e denotes effective amplitude (the average amplitude of cursor motion) and W_e denotes effective target width (the value of *W* which, given the observed dispersion of clicks, would have yielded 4% errors). Recourse to the ID_e, computed on a participant-by-participant and block-by-block basis before averaging, makes it possible to assess Fitts' law satisfactorily even if the correlation between movement-endpoints dispersion and prescribed *W* is less than perfect [15]. Note, however, that this technique amounts to expressing Fitts' law as a scatter plot linking two dependent measures, *MT* on the *y* axis and ID_e on the *x* axis. Unlike the nominal (i.e., prescribed) value of *ID*, a systematic variable (i.e., a 'factor'), ID_e is a stochastic variable, which precludes the use of the analysis of variance to evaluate the impact of task difficulty. Note that the *ID* was computed identically for BMP and OP from onscreen length measurements.

Figure 3 shows that performance was always much faster for OP (mean *MT*=0.331s) than standard BMP (mean *MT*=1.274s), a 74% time-saving effect which the confidence limits show to be highly reliable. Second, whereas *MT* faithfully obeyed Fitts' law for BMP (for the best-fitting equation, $MT=0.246\ ID_e - 0.081$), it no longer did for OP, where the mean slopes were 0 and -0.01s/bit. Finally, Tim's visibility had no incidence on OP performance.

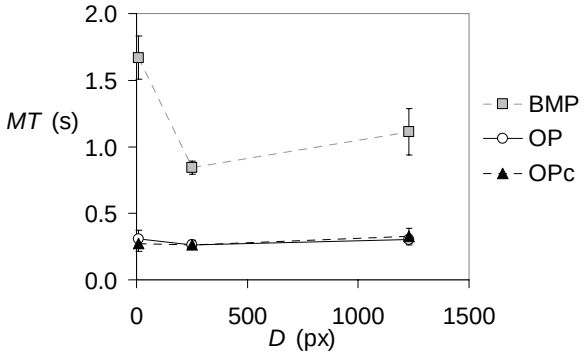

Figure 4. Effect of task scale on movement time, for each pointing technique (Exp.1B).

Figure 4 shows mean *MT* as a function of task scale for each pointing condition, as measured in Exp.1B. Again, OP yielded considerably faster performance than BMP overall, the mean time saving amounting to 76% of *MT*.

Note the interaction between scale and pointing mode ($p<.0001$): whereas for BMP performance speed was scale dependent, exhibiting the expected V-shaped relationship, for OP no scale effect was apparent. Newman-Keuls pair-wise comparisons revealed that for BMP both task miniaturization and task magnification significantly impaired performance ($p<.05$). For OP or OP_c, in contrast, neither pair-wise difference approached significance.

Tablet Amplitudes vs. Screen Distances. The data of Exp.1B, which contrasted very small and very large task scales on the screen, are well suited to illustrate the way in which the participants actually managed in tablet

space to cover screen-space distances. Figure 5 shows, for the three pointing techniques, how the amplitude actually covered by the puck on the tablet varied with the distance separating the two targets on the screen. Although a slight scale effect was still observed with OP, the amplitude of the puck movement required in OP mode to handle the three levels of *D* remained very short, in comparison with the BMP mode, revealing a drastic input-device footprint reduction.

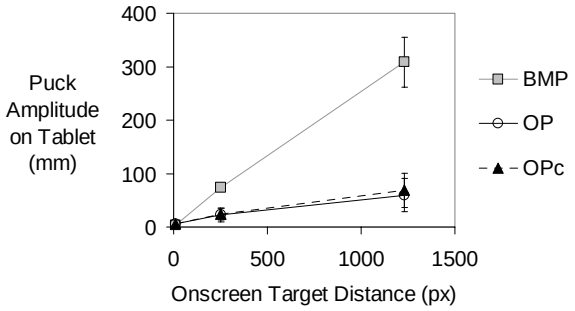

Figure 5. Puck amplitude on the tablet as a function of onscreen target distance, for each pointing technique.

Subjective Difficulty Scores. The rating, on a 5-point scale, of the difficulty experienced at pointing in Exp.1 was about the same for the two variants of OP (1.42 and 1.47). But pointing was judged far less difficult with the OP technique (mean±*SD*=1.44±0.74) than the usual BMP technique (3.55±0.44). All but one participant (who equally scored OP and BMP), assigned a reduced score of difficulty to OP (sign test $p<.001$, one-tailed).

5 Experiment 2: Serial Pointing in 2D Space

This experiment aimed at evaluating OP in 2D space, using a more realistic simulation of a GUI. Because the results of Exp.1 had confirmed that the visibility of Tim was superfluous, in Exp.2 the pointing-mode variable became binary, contrasting BMP vs. the variant of OP that never displays Tim. Pointing mode was crossed with another factor, onscreen target density, with the display exhibiting 6, 12, or 60 objects.

5.1 Methods

Equipment and Display. The input device consisted of the same A3-format tablet as in Exp.1, but the puck was replaced by a standard mouse on which a stylus had been fixed. The mouse served to control onscreen cursor motion in relative mode, so that not only position, but also direction were defined in the 'egocentric' frame of reference of the hand. However, the attached stylus allowed the tablet to track the hand's absolute position so as to allow footprint measurements.

The screen resolution was set to 1024x768 pixels. The layout of the 6, 12, or 60 objects, all identical (32x32 pixel icons), was randomly drawn by the program on a black background. By default, the displayed objects were dark blue, but the object that was the current pointing target was shown in a more conspicuous light-green color. Highlighting of the currently visited object, whether light-green or dark-blue, was obtained by surrounding this object with a white outline that enlarged the object to 52x52 pixels.

Task. The pointing paradigm was serial, but no longer reciprocal in the sense of a back and forth (stationary) movement. To track the green target, which every successful click made to jump to another unpredictably determined object, the participant had to produce a series of clicks to follow the path dictated by the program across graphical objects. This task was designed to mimic a rather common situation in GUIs (e.g., reaching and clicking the FILE menu, then the OPEN item, then browsing a tree, etc.). Within each display, 11 successive clicks had to be made, yielding blocks of 10 measurements of *MT*. If a target object was missed by the click, it remained the target, waiting for a correct click, and so a 0% error rate was imposed on all participants.

Participants and Procedure. Twelve new unpaid volunteers participated. Each of the 6 cells of the design (2 modes x 3 object densities) was explored 6 times for each participant, yielding a set of 36 blocks each including 10 movements overall, with possible order effects being counterbalanced with Latin squares. The experiment was completed in a single session that lasted about 40 minutes.

5.2 Results and Discussion

Performance Speed. Figure 6 shows the effect of movement difficulty on *MT* for BMP and OP. The two curves were computed by averaging the coefficients of Fitts' law estimated in each participant. Above 3.6 bits (the abscissa of the intersection between the curves), the more difficult the task, the better OP relative to BMP.

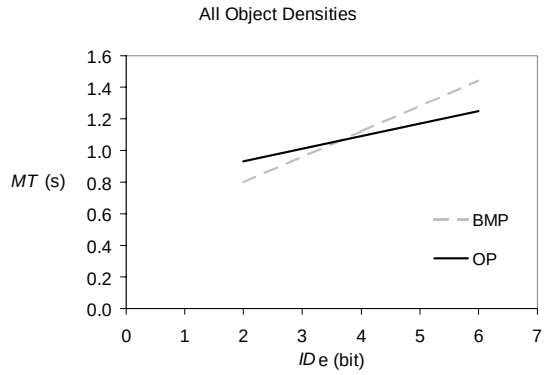

Figure 6. Movement time in Exp.2 as a function of effective task difficulty for the OP and BMP modes.

However, the curve intersection that specifies the critical level of difficulty above which OP began to surpass

BMP depended on object density: the abscissa of this intersection (for the current version of our OP algorithm) was 2.7, 3.5, and 7.0 bits for the 6, 12, and 60 object condition, respectively. So, in keeping with our expectations, the less dense the display and the more difficult the task, the greater the benefit brought about by OP.

Mouse Footprint. We obtained an estimate of the mouse footprint for the BMP and the OP mode by computing on the *x* and *y* axis the *SD* of the absolute mouse coordinates delivered by the tablet. Since, with a normal spread of hand positions in both dimensions, a rectangle with its vertical and horizontal sides equal to 2 *SD* should include about 96% x 96% = 92% of all positions, mouse footprint can be estimated as the length of the rectangle's diagonal, as SQRT(2 SD_x^2+ 2 SD_y^2), or as the rectangle's surface area, as 2 SD_x * 2 SD_y. Even though this method can occasionally lead to overestimations of the foot print in case of a gross repositioning of the mouse, it ought not bias the OP vs. BMP footprint comparison. Using the rectangle's diagonal, mouse footprint was 23.8% for OP relative to BMP (mean±*SD*=37±0.6mm vs. 155±23mm). In terms of surface area, footprint in OP amounted to a minute 6.2% of BMP (695±234mm² vs. 11,192±3,191mm²).

Satisfaction Scores. Overall, all three object densities mixed, OP and BMP scored about the same in terms of satisfaction concerning ease of use (12.8 vs. 12.3, respectively, $t(11)=0.53$, $p>.1$), but the scores depended on the density, in keeping with the speed data. No significant differences were found for the intermediate and higher density conditions. In the low-density condition, however, OP scored 15.8 and BMP 10.8 ($t(11)=4.46$, $p<.001$, one tailed), with all individual judgments, save a tie, favoring OP over BMP.

6 Overview

To sum up, input flows in current GUIs suffer a serious waste of information because users are always asked to select pixels regardless of the fact that the system, more often than not, needs just the specification of a discrete object. We have introduced OP, a new pointing mode that dispenses the user of producing redundant cursor moves across the voids of graphical displays. With formal experiments in 1D and 2D space, we have shown that OP indeed facilitates target acquisition, both objectively and subjectively. The data corroborate our hypotheses that the benefit of OP increases (1) as pointing becomes more difficult—either because of a higher *ID* or because of a more marked departure, in either direction, from optimal interface size—and (2) as object density in graphical space decreases.

7 Implications for HCI Design

OP shares with semantic pointing [5] a valuable characteristic: it reduces the difficulty of target acquisition without perturbing the layout of graphical objects. However, the two techniques differ. Whereas semantic pointing affects the law's intercept, with the whole curve being shifted downward, OP affects the *slope* of Fitts' law, virtually canceling it. This means that, regardless of the spatial arrangement of objects in the display, OP makes pointing about as easy as though all objects were tightly packed together, thus forming a soft keyboard.

Since pointing is a basic building block of all our GUIs, the potential implementation scope of the OP mode seems to be fairly broad. Our demonstration that the advantage of OP over the current BMP technology is enhanced when pointing difficulty increases is obviously important—indeed, it is for difficult tasks that we need pointing optimization. Current trends in HCI technology raise two major challenges that seem tailored to the OP technique. First, with the spread, sooner or later, of high-resolution screens, pointing difficulty will tend to increase in the future. The other challenge arises from the current broadening of the range of interface scale. Since interface scale is known to dramatically affect the bandwidth of the interaction [11], the OP technique seems particularly promising in the face of dramatically miniaturized and magnified displays.

One spectacular characteristic of the OP mode is a considerable reduction of input device footprint. This feature may be useful for standard users, if only because desks are often cluttered, or touchpad area is limited. But it should be of special help to disabled users for whom the production of normal amplitude movements is a problem. OP makes it possible to control cursor motion in normal, zero-order mode (e.g., with a standard mouse) with strongly reduced hand moves.

The OP technique is more than a mouse counterpart of the keyboard direction keys. One important difference is that the definition of the jumping direction with OP is far finer than up, down, left, and right, making it possible to move one's selection, not simply through rows and columns, but through any arbitrary layout of objects. Moreover, unlike the set of direction keys on a keyboard, OP does not require a time consuming switch from one input device to another.

Rather than an alternative to BMP, OP should be viewed as an *optional mode* made available to users beside the usual BMP mode. One should be allowed to switch opportunistically from BMP to OP, via some simple mouse command, to occasionally cope with pixels (e.g., in a drawing task). So the interface should provide the user with some convenient means to switch back and forth between OP and BMP.

Obviously, OP does not yield any benefit for selection within display regions that are *tiled* with graphical objects, because such regions offer no voids. For example, following a horizontal path to reach the next

hierarchical level in a cascading menu costs time [2], but for little or no gain: the probability that the cursor aims at the next (previous) level of the hierarchy when it starts moving to the right (left) is about 1. One solution is to make the menu highlight jump by discrete steps depending on the initial direction of cursor motion [13]. Facilitating pointing across tiled spaces with such a technique might nicely complement OP.

8 Future Work

One problem that remains to be addressed is the design of an appropriate command to switch back and forth between OP and BMP. Among the possibilities, we may think of a combination of mouse-button presses or a brief oscillation of the cursor. The latter option seems particularly worth examining. Quick, small amplitude oscillations of the hand, very easy to produce [9], might usefully enrich the vocabulary of input commands.

One possible development would be to also allow, in OP mode, the object highlight to switch from objects in the active window to objects hosted by surrounding windows, perhaps by setting different velocity thresholds for jumping within and between windows.

Our preliminary investigation used a newborn version of the OP algorithm for 2D space, with presumably sub-optimal parameter settings. So, although OP already worked better than BMP, its efficiency can certainly be improved. One feature that will require special care is the setting of the initial value and the expansion dynamics of the angular sector that serves in OP mode to detect the target of current cursor motion.

Exp.2 is likely to have underestimated the performance benefit of OP, as novel techniques are inevitably penalized in evaluation experiments: OP was far less familiar to our participants than BMP. So more evaluation work is needed not only to optimize OP, but also to reevaluate the technique with practiced users.

Acknowledgments

Thanks to Matthieu Langet, who did the software development and kindly helped us run the experiments.

References

1 Abrams, R.A., Meyer, D.A., & Kornblum, S. (1989). Speed and accuracy of saccadic eye movements: Characteristics of impulse variability in the oculomotor system. *J. Exp. Psychol.: HPP, 15*, 529-543.

2 Accot, J. & Zhai, S. (1997). Beyond Fitts' law: Models for trajectory-based HCI tasks. *Proc. of CHI'97*, pp. 295-302.

3 Accot, J. & Zhai, S. (2001). Scale effects in steering law tasks. *Proc. of CHI'01*, pp. 1-8.

4 Baudisch, P., Cutrell, E., Robbins, D., Czerwinski, M., Tandler, P., Bederson, B., & Zierlinger, A. (2003). Drag-and-Pop and Drag-and-Pick: Techniques for accessing remote screen content on touch- and pen-operated systems. *Proc. of INTERACT'03*, pp. 57-64.

5 Blanch, R., Guiard, Y., & Beaudouin-Lafon, M. (in press). Semantic pointing: Improving target acquisition with control-display ratio adaptation. *Proc. of CHI'04.*

6 Card, S.K., English, W.K., and Burr, B.J. (1978). Evaluation of mouse, rate-controlled isometric joystick, step-keys, and text keys for text selection on a CRT. *Ergonomics, 21*, 301-613.

7 Fitts, P.M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *J. Exp. Psychol., 47*, 381-391.

8 Gutwin, C. Improving focus targeting in interactive fisheye views. *Proc. of CHI'02*, pp. 267-274.

9 Guiard, Y. (1993). On Fitts' and Hooke's laws : simple harmonic movement in upper-limb cyclical aiming. *Acta Psychologica, 82*, 139-159.

10 Guiard Y. (2001). Disentangling relative from absolute movement amplitude in Fitts' law experiments. *Ext. Abstracts of CHI'01*, pp. 315-316.

11 Guiard, Y., Beaudouin-Lafon, M., Bastin, J., Pasveer, D., & Zhai, S. (in press). View size and pointing difficulty in multi-scale navigation. *Proc. of AVI 04*. Gallipoli (Italy), May 25-28, 2004.

12 Jacob, R.J.K. (1991). The use of eye movements in human-computer interaction techniques: What you look at is what you get. *ACM Transactions on Information Systems 9(3)*, 152-169.

13 Kobayashi, M. & Igarashi, T. (2003). Considering the direction of cursor movement for efficient traversal of cascading menus. *Proc. of UIST'03*, pp. 91-94.

14 McGuffin, M. & R. Balakrishnan, R. (2002). Acquisition of Expanding Targets. *Proc. of CHI'02*, pp. 57-64.

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

16 Welford, A. T. (1968). *Fundamentals of skills*. London: Methven.

17 Zhai, S., Conversy, S., Beaudouin-Lafon, M., & Guiard Y. (2003). Human on-line response to target expansion. *Proc. of CHI'2003*, pp. 177-184.