TorusDesktop: Pointing via the Backdoor is Sometimes Shorter

Stéphane Huot, Olivier Chapuis, Pierre Dragicevic

TorusDesktop: Pointing via the Backdoor is Sometimes Shorter

Stéphane Huot[1,2]
huot@lri.fr

Olivier Chapuis[1,2]
chapuis@lri.fr

Pierre Dragicevic[2]
dragice@lri.fr

[1]LRI - Univ. Paris-Sud & CNRS
F-91405 Orsay, France

[2]INRIA
F-91405 Orsay, France

ABSTRACT

When pointing to a target on a computer desktop, we may think we are taking the shortest possible path. But new shortcuts become possible if we allow the mouse cursor to jump from one edge of the screen to the opposite one, i.e., if we turn the desktop into a torus. We discuss the design of TORUSDESKTOP, a pointing technique that allows to wrap the cursor around screen edges to open this pointing backdoor. A dead zone and an off-screen cursor feedback make the technique more usable and more compatible with everyday desktop usage. We report on three controlled experiments conducted to refine the design of the technique and evaluate its performance. The results suggest clear benefits of using the backdoor when target distance is more than 80% the screen size in our experimental conditions.

Author Keywords

Pointing Technique, Cursor Wrapping, Torus.

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation: User interfaces—*Graphical user interfaces.*

General Terms

Human Factors, Experimentation.

INTRODUCTION

When flying from New-York to San Francisco, one usually does not fly around the globe across the Atlantic and the Pacific Oceans. Yet we often do it on our computers: we routinely move our mouse pointer from one side of the screen to the opposite side – e.g., to select a tool or invoke a menu command – ignoring potential trajectory shortcuts. Such shortcuts would only require a small modification to the mouse behavior: when the pointer goes past a screen edge it re-appears on the opposite side, as in the Asteroids or Pac-Man video-games (see Figure 1).

We introduce TORUSDESKTOP, a pointing technique which opens these shortcuts on our computer desktops. Although many pointing facilitation techniques have been already pro-

S. Huot, O. Chapuis and P. Dragicevic. TorusDesktop: Pointing via the Backdoor is Sometimes Shorter. In CHI '11: Proceedings of the SIGCHI Conference on Human Factors and Computing Systems, 829-838, ACM, May 2011.
Authors Version
doi: http://doi.acm.org/10.1145/1978942.1979064

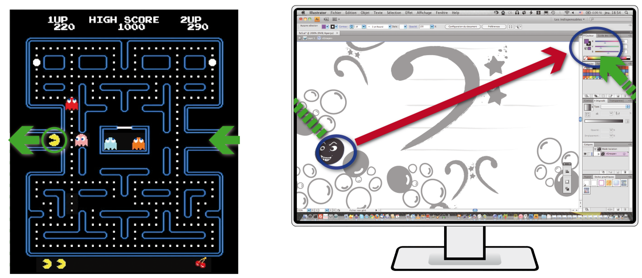

Figure 1. The Pac-Man video-game and a case scenario where pointing through screen edges could be beneficial.

posed, most of them are *target-aware* [26, 3], i.e., they require knowledge of all the potential targets the user may acquire. These techniques can be extremely efficient but they are sensitive to distractors and are difficult to integrate to existing systems. Only a few *target-agnostic* pointing facilitation techniques have been introduced and the results have been mixed. TORUSDESKTOP is target-agnostic, making it easy to integrate to existing systems and compatible with most existing pointing facilitation techniques.

TORUSDESKTOP teleports the mouse cursor to the opposite side of the screen when it goes past one of the screen's edges. This technique is sometimes referred to as *cursor wrapping*. One consequence of this wrapping behavior is that the shortest path between two points is not necessarily the on-screen segment that connects them. Although this may evoke a sphere topology, wrapping the cursor around screen edges actually turns the computer desktop into a torus.

The idea of wrapping the mouse cursor around screen or window edges is not new. In addition to video games from the early 80's, a few system tweaks and mouse drivers support this technique. But current implementations are all under-designed as the cursor immediately jumps when it reaches a screen edge. This can yield several problems: first, it is easy to trigger the wrapping inadvertently. Second, it might be difficult to find the new location of the cursor. Third, the technique prevents the user from using the border to acquire targets that are located on screen edges. TORUSDESKTOP addresses these issues by introducing a wrapping dead zone and visual feedback to anticipate cursor jumps.

As cursor wrapping has never been studied experimentally, it is not clear whether it should be supported natively by operating systems and better publicized among end users, or simply abandoned. Our initial Fitt's Law simulations (considering all possible pointing tasks on a 2560x1600 display with 40-pixel targets) suggest that cursor wrapping should outperform direct pointing in more than 40% of all possible pointing tasks. But it is unlikely that the question can be adequately answered by a naive Fitts' Law simulation: choosing to use cursor wrapping or not might have an impact on efficiency, and large cursor jumps might be distracting to users and could result in a drop in performance.

Thus we conducted three controlled experiments to refine our design and evaluate its performance. The results of the two first experiments identify the best off-screen feedback, and suggest that a dead zone of $5 - 10\%$ the size of the screen should be provided to enable edge pointing. Our final experiment confirms that our naive Fitts' Law simulation is overly optimistic as it does not account for factors such as the distraction produced by cursor teleportation or the cost of having to chose whether or not to use the backdoor. Nevertheless, our experiment reveals that TORUSDESKTOP is still faster than direct pointing for targets whose distance is greater than 80% the width of a 2560-pixel wide display. This suggests that enabling cursor wrapping is worthwhile, especially in situations where commonly-accessed widgets are located close to the edges of the screen (Figure 1) or when going back-and-forth between two very distant targets.

RELATED WORK

A fundamental tool in the area of target acquisition is Fitts' law [20]. This law models the movement time to acquire a target of size W at distance D as a linear function of an index of difficulty ID usually defined as $\log_2\left(\frac{D}{W}+1\right)$. According to this law, techniques that try to facilitate pointing increase W, reduce D or do both [3]. They are either target-aware or target-agnostic.

Target-Aware Techniques

Most techniques that increase W are target-aware. They either expand the targets themselves [21] – sometimes in the motor space only [27, 9] – or expand the cursor's activation area [15, 12]. Target-aware techniques for reducing D try to predict the target(s) the user wants to acquire. They then bring the cursor closer to the target [2, 16] or bring potential targets closer to the cursor [4]. Another way to reduce D is to use a grid of cursors and a target-aware algorithm that tries to select the appropriate cursor [19].

However, target-aware techniques fail when there are a large number of potential targets, and they are difficult to implement at a system-wide level because they require access to target information that is solely available at a system-level.

Target-Agnostic Techniques

Effective target-agnostic pointing facilitation techniques are relatively rare. Speed-adaptive C-D gain has been modeled as a technique that increases W in motor space, but experiments did not confirm the improvements predicted by the model [11]. Angle Mouse adapts C-D gain to trajectory curvature, but it has been shown to only benefit motor-impaired users [26]. Finally, visual and motor-space uniform magnification (i.e., W and D increased in the same proportion) have been shown to improve pointing performance, but only for very small targets [24].

Other techniques employ more than one input device at a time. For example, D can be reduced in a target-agnostic manner using eye tracking. MAGIC [28] uses eye tracking to define an area where the pointer is automatically warped. The Rake cursor uses a grid of cursors and eye tracking for cursor selection [10, 25]. These techniques benefit from the increase in input bandwidth provided by gaze tracking, but they cannot be implemented on standard computer hardware.

Adaptive [18] and adaptable [13] methods have also been considered: DirtyDesktops [18] creates magnetic fields around frequently-selected locations on the screen and UIMarks [13] lets users specify on-screen locations whose acquisition will be facilitated. However, adaptative techniques improve pointing only for frequently-selected targets and adaptable techniques require user intervention.

Edge and Displayless Pointing

HCI practitioners early noticed that targets on screen edges are easier to acquire because screen edges stop the cursor, effectively increasing W in the motor space. Edge pointing has been studied experimentally in [14, 1].

Edge pointing becomes problematic in multi-display environments: by default, desktop environments treat multiple displays as a single space, disabling edge pointing between them. Mouse Ether [5] takes into account the space between the displays as well as display size and resolution to compute a motor space – the *ether* – that lies between the displays. This re-enables edge pointing, since stopping the mouse in the ether warps the pointer to the closest display edge.

Mouse Ether is conceptually similar to our dead zones: they both add off-screen pointing space that (among other things) enable edge pointing. One problem with Mouse Ether is the absence of visual feedback when the cursor is in the ether. Several techniques have been proposed to visualize the location of off-screen objects: Halo [7] surrounds off-screen objects with rings large enough to reach the edge of the display, and Wedge [17] uses a triangle pointing towards the off-screen object. A recent study suggested that augmenting Mouse Ether with Halo helps, while also suggesting that Mouse Ether itself (with or without feedback) hurts performance when displays are sufficiently far apart [23].

Cursor Warping vs. Cursor Wrapping

Cursor warping refers to the sudden teleportation of the mouse cursor to a possibly distant place. It has been used to reduce pointing distance in some target-aware pointing techniques [2, 16] as well as in target-agnostic ones [28]. Manually-triggered cursor warping has also been used for rapidly switching between displays in multi-monitor environments [8]. However, it is also believed that sudden cursor jumps can be confusing to users and can slow them down [6].

Cursor wrapping should not be confused with cursor warping: wrapping the mouse cursor around screen edges involves a specific type of cursor warping, going from one edge of the screen to the opposite one. Several applications exist that support cursor wrapping. More than 15 years ago, the FVWM X Window Manager could be configured to en-

Figure 2. Wrapping dead zone (right) and expansion of targets located on the screen edge (left).

able it. Today, system-level tools provide the same feature[1]. But as discussed previously, none of these applications provide a dead zone or off-screen feedback. Moreover, to our knowledge, such techniques have never been evaluated.

THE TORUSDESKTOP TECHNIQUE

The TORUSDESKTOP extends direct cursor wrapping techniques with two additional features in order to make it usable and compatible with everyday desktop usage:

- a *wrapping dead zone* that adds a displayless pointing space around screen edges in order to help users anticipate cursor jumps and to re-enable edge pointing;
- a *wrapping feedback* that provides visual feedback on the cursor's location inside the dead zone to further increase user's control over cursor wrapping.

Wrapping Dead Zone

The *wrapping dead zone* is a displayless frame added around the screen edges. When the cursor reaches a screen edge, the user needs to cross this space before the cursor gets teleported to the opposite edge (Figure 2). This design presents three advantages:

Prevention of accidental triggering. In situations where users do not want to cross the screen, the wrapping dead zone prevents them from wrapping the cursor accidentally. Accidental wrapping can be distracting – especially repetitive wrapping when following a screen edge – and can slow users down since they have to bring the cursor back once they realize it has jumped. They may even lose the cursor altogether if they do not realize it has moved to the opposite side. The dead zone addresses this issue by making it more difficult to trigger the wrapping and allowing to cancel it.

Support for anticipation. In cases users want to cross the screen, the wrapping dead zone helps them anticipate the cursor jump and gives them time to switch their visual attention to the region where the cursor will re-appear. Additionally, it provides users with more flexibility, as they can adapt their mouse movement while crossing the dead zone to control where and when the cursor will re-appear.

Compatibility with edge pointing. As discussed previously, targets located on screen edges are faster to acquire, a feature that is now commonly used in window management systems and desktops (e.g., Mac OS' menu bars and MS Windows' task bar). While a naive implementation of the wrapping technique defeats edge pointing, using a large enough dead zone re-enables this feature as clicks within the dead zone are dispatched to the screen edge where the cursor comes from (Figure 2 left).

However, using a dead zone raises two issues. First, it increases the distance users have to cover during cursor wrapping so it may reduce the number of cases where the technique is useful. Second, it is not clear which dead zone sizes are small enough not to impede cursor wrapping, while being large enough to allow comfortable edge pointing. These questions will be later addressed in our experiment sections.

Wrapping Feedback

When crossing a dead zone, a standard mouse cursor would stop on the screen's edge and the user would have to blindly move a virtual cursor within the dead zone. It has been suggested that visual feedback about the position of an off-screen cursor helps pointing in displayless space [23], so we chose to augment the dead zone with visual feedback. Since there are many possible designs, we identified the three following requirements for TORUSDESKTOP visual feedback:

Position along the edge. The feedback needs to show where the cursor is located along the screen edge: for example if the exiting edge is vertical, users need to keep track of the cursor's y-coordinate to be able to predict where it will re-appear on the opposite edge.

Position within the dead zone. The feedback also needs to show the cursor's position in the orthogonal direction, i.e., how deep the cursor is in the dead zone. This is necessary for users to be able to predict when the cursor will be teleported and better anticipate its arrival. This also allows users to see how far they can go before the cursor jumps to prevent accidental triggering, especially during edge pointing.

Feedback mirroring. The two pieces of information above should be shown both near the edge where the cursor exits the screen and near the opposite edge. Thus, users can use the feedback whether they are focusing on the exiting side – i.e., when moving close to the edge or when doing edge pointing – or on the reentering side – i.e., when using cursor wrapping to point to a distant target.

We experimented with three feedback methods: *Halos*, *Arrow* and *Ghost*. Figure 3 explains these three techniques in detail: DZ is the dead zone size, d is the cursor's distance to the dead zone entrance and the constant k is Halo's intrusion distance. Gray arrows depict how cursor movements map to movements of visual feedback.

Halos. *Halo* [7] is a technique for providing on-screen feedback for off-screen objects, e.g., showing the location of points of interest in a map on a handheld device. It shows an arc of circle next to the screen edge; the circle is centered on the off-screen object in order to convey its direction and distance. In our case the off-screen object is the cursor itself, so when it enters the dead zone, we display a Halo both on the exit and on the entrance sides of the screen (Figure 3a). As in the original technique, the arcs stick out from the displayless space with a fixed intrusion distance k.

[1] E.g., www.networkactiv.com/SoundyMouse.html, www.digicowsoftware.com/detail?_app=Wraparound

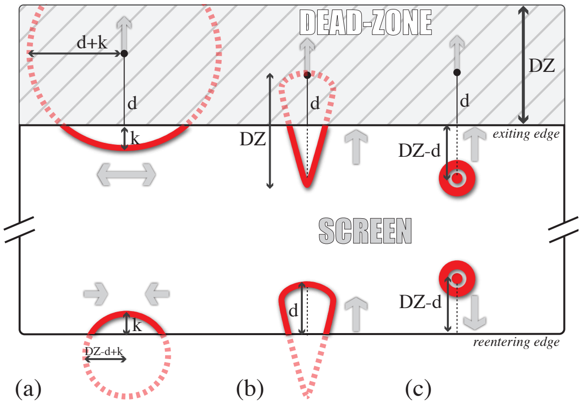

Figure 3. Halos, Arrow and Ghost TorusDesktop feedback techniques for a top-to-bottom cursor wrapping.

Arrow. The Arrow feedback is inspired from a variant of Halo called *Wedge* [17]. Arrow's triangular shape is similar to Wedge's but unlike Wedge, its intrusion distance is not fixed and its shape does not change. Instead, it is a solid triangle of constant size, always perpendicular to the screen edge, that sticks out on both sides (Figure 3b). On the exit side, its flat end is attached to the cursor and its tail sticks out. On the entrance side, its tail is attached to the cursor and its flat end sticks out. The angle formed by the flat end conveys the cursor's distance in a way similar to Wedge.

Ghost. Finally, we propose a simpler visual feedback called *Ghost*, specifically designed for wrapping dead zones. Next to the dead zone's exit (bottom of Figure 3c), a circular shape is displayed whose distance to the edge is the same as the cursor's. In other terms, the edge acts like a mirror and the circular shape is like the cursor's reflection on that mirror. The same circular shape is displayed at the same distance near the dead zone's entrance.

Even if these wrapping feedback techniques fulfill the requirements we identified, it is not clear how much they help, if they help at all. We investigate this question in our study.

Corners

In a torus topology, the four screen corners are equivalent. So when the cursor reaches a corner, it is not clear where it should exit. Besides, the behavior of the cursor in the vicinity of a corner can be disturbing. For example, a cursor going to the top-right corner will re-appear either on the top-left or on the bottom-right. These two locations are nevertheless close to each other on the torus, so if the user approaches a corner with a 45-degree angle, the cursor will eventually appear on the opposite corner no matter which path it takes. However, the cursor will rapidly jump twice on the screen, which can be visually disturbing. To address these issues, we added four *corner zones* of 20 pixels each. When the cursor reaches one of these zones, it simply re-appears on the diagonally opposite corner after the dead zone crossing.

PRELIMINARY EXPERIMENTS

We conducted two preliminary experiments in order to refine the design of the technique before comparing it with direct pointing. The first experiment compares the feedback techniques and provides a first sense of the impact of the dead zone on movement time. The second one investigates the compatibility of TorusDesktop with edge pointing.

Apparatus & Participants

The two experiments were conducted on a workstation running Mac OS X and with a 2560×1600 30" LCD monitor [2]. Such large displays are becoming more and more common and are likely to become a standard once their price drops. The TorusDesktop software was implemented in Java. The mouse was a standard optical mouse with 500 dpi resolution and default system acceleration.

Eight unpaid volunteers, all male and right-handed, participated in the experiments. Participants were experienced mouse users with ages ranging from 24 to 31 (median 26.5). Each participant took about 60 minutes to complete each experiment after which they were given a short questionnaire.

Experiment 1: Feedback & Dead Zone

This experiment addresses the following questions:

- *Q1: Which wrapping feedback (including no feedback) is the best, with and without a dead zone?*
- *Q2: Does dead zone size affect movement time?*

Task & Design. A trial was a TorusDesktop pointing task requiring subjects to cross either the left or the right edge of the screen. Subjects had to click on a *start target* at a distance DB1 to its closest edge and then acquire a *goal target* at a distance DB2 to the opposite edge by crossing the closest edge. Both start and goal targets were circles of 40 pixels. Targets were lying on the screen's horizontal centerline or placed above and below the centerline at a distance of 300 pixels, depending on the factor Align (see Figure 4a). Task direction was either left to right (Dir = LR) or right to left (Dir = RL).

The experiment was a within-subject design with the main factors: (i) Feedback: FB $=$ *None*, *Halos*, *Arrow*, *Ghost*; and (ii) Dead zone size: DZ $= 0, 125, 250, 500$.

Auxiliary factors were: (i) Distance of the start target to its closest edge, DB1 (Distance to Border 1) $= 50, 150$; (ii) Distance of the goal target to the opposite edge, DB2 (Distance to Border 2) $= 50, 150, 300$; (iii) Align and Dir.

Concerning the values we chose for dead zone size, 0 is the baseline condition implemented in former cursor wrapping techniques. 125 and 250 seem to be realistic values for edge pointing [1]. We added 500 for completeness although we expect it to be too large to be used in practice. Note that for DZ = 0, the feedback condition is irrelevant and we only need to test the condition for feedback = *None*.

We grouped trials into blocks according to DZ $\times$ FB. We used 2 orders of presentation for DZ, increasing and decreasing, and counterbalanced the presentation order of FB. Before each DZ $\times$ FB condition, participants did one block of 24 practice trials then 2 blocks of measured trials. We hence collected 8 (Participant) $\times$ 48 $\times$ (4(DZ)$\times$1(FB=*None*) + 3(DZ=125,250,500)$\times$3(FB)) = 4992 trials for analysis.

[2] Yielding 100.63 ppi and a pixel size of 0.025 cm.

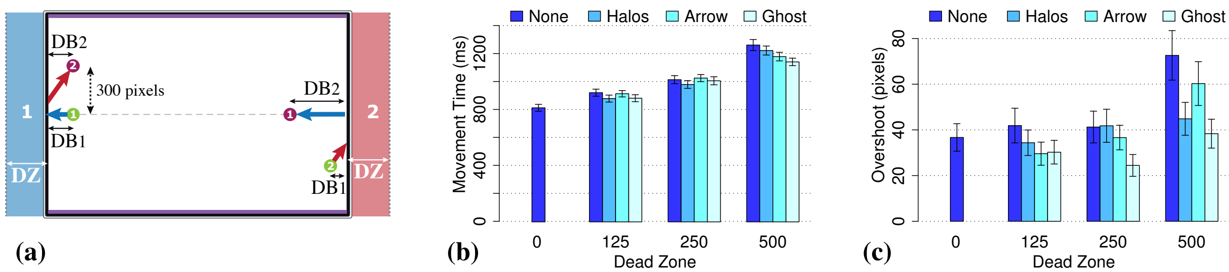

Figure 4. (a) Examples of target placement in experiment 1. Start targets are green, goal targets are red. Case 1: ALIGN = yes, DIR = RL, DZ on the left. Case 2: ALIGN = no, DIR = LR, DZ on the right. (b) MT per deadzone and feedback. (c) *OverShoot* per deadzone and feedback.

We collected three measures: (i) *MT*, the time from the click on the start target to a successful click on the goal target; (ii) *Error*, whether or not there was a click outside the target; and (iii) *OverShoot*, the distance in pixels of the furthest point reached by the pointer to the goal target.

Quantitative Results. We removed 0.75% outliers (trials with a *MT* that is 3 standard deviations apart from the mean *MT* within the condition) and duplicated the data for DZ = 0 for each FB in order to perform a full factorial analysis: FB × DZ × Random(PARTICIPANT) with *MT*, *Error* and *OverShoot*.

An analysis of variance reveals an effect of DZ on *MT* ($F_{3,21} = 80.0, p < 0.0001$). A Tukey post-hoc test shows a significant difference in means between all DZ, with *MT* increasing with DZ (Figure 4b). We observed no significant effect of FB on *MT*. However, we found a significant interaction FB × DZ ($F_{9,63} = 2.62, p = 0.0123$), which can be observed in Figure 4b: the difference between mean *MT*s for each FB value is the largest for DZ = 500. Indeed a post-hoc test shows no significant difference between the FBs for DZ $\leq$ 250, whereas for DZ = 500, *Ghost* is significantly faster than *Halos* and *None*, and *Arrow* is significantly faster than *None*.

We found an average error rate of 7.9%. An analysis of variance using a nominal logistic test for the model *Error* $\sim$ FB × DZ reveals no significant effect, error rates being very close for each FB × DZ (min 5.0%, max 9.9%).

For *OverShoot*, we found a significant effect of both FB ($F_{3,21} = 6.32, p = 0.0032$) and DZ ($F_{3,21} = 8.01, p = 0.0010$) (see Figure 4c). Post-hoc Tukey tests show that *Ghost* exhibits significantly less *OverShoot* than other feedback and that *OverShoot* is significantly larger for DZ = 500. However, there is a significant interaction FB × DZ ($F_{9,63} = 3.82, p = 0.0007$), which can be observed in Figure 4c: *OverShoot* is significantly lower for *Ghost* than for all other FB when DZ = 250. For DZ = 125, the only significant difference is between *Ghost* and *None*. For DZ = 500 we observe more *OverShoot* for *None* than for other feedback and less for *Ghost* than for *Arrow*.

Qualitative Results. In the post-experiment questionnaire, participants were asked to rank the feedback techniques globally and for each dead zone size. Among the eight participants, five globally ranked *Ghost* first, and each of the three other techniques was ranked first by one participant (*Ghost* was ranked second, third and last in these cases). Rankings by DZ are consistent with global ranking. Only three participants ranked *None* higher for DZ = 125.

Summary. Back to our first question *Q1*, *Ghost* seems to be the best choice for TORUSDESKTOP for all dead zone sizes: even if it does not exhibit a significantly better performance – except for the limit case DZ = 500 – it yields the smallest *OverShoot* and was preferred by participants. Concerning *Q2*, it is confirmed that *MT* increases with DZ.

Experiment 2: Edge Pointing

The questions this second experiment addresses are:

- *Q1: Does a dead zone help users performing edge-pointing tasks? If yes, is there an optimal dead zone size?*
- *Q2: Does wrapping feedback help or impede users during edge pointing?*

Stimuli & Design. A trial consisted in an edge pointing task where the subject had to click on a circular *start target* and then acquire a *goal target* on a screen edge. The goal target was located to the left or right end of the screen, and was vertically centered (Figure 5a). It had a width of 40 pixels and two possible heights H $= 40, 125$ – a size comparable to buttons on typical task bars and menu bars. Start targets were located on a 3×3 grid designed to cover several angles of approach. Their location was defined by DB $= 200, 1200, 2200$, their distance to the edge where the goal target was, and DH $= 0, 600, -600$, their distance to the horizontal centerline of the screen.

The experiment was a within-subject design with the same main factors as the first experiment: (i) Feedback: FB = *None*, *Ghost*; and (ii) Dead zone: DZ $= 0, 125, 250, 500,$ inf. Given the findings of the first experiment, we only tested the *None* and *Ghost* feedback techniques in this experiment. We used the same dead zone sizes as in the first experiment and added an 'infinite' size (i.e., no cursor wrapping) as a baseline condition to test standard edge pointing.

Trials were grouped into blocks by DZ × FB. For each DZ × FB condition, participants started with one practice block of 2 × 2(H) × 3(DB) × 3(DH) = 36 trials, then proceeded with two measured blocks. Thus, for each participant, we collected 2 × 36 × (2(DZ=0,inf) + 2(FB) × 3(DZ=125,250,500)) = 576 trials for analysis.

In addition to *MT* and *Error* (defined as in previous experiment), we measured the dead zone distance effectively used *UseDistDZ* – i.e., the maximum horizontal travel distance inside the dead zone – and the number of times the cursor went past the dead zone *DZOverShoot*.

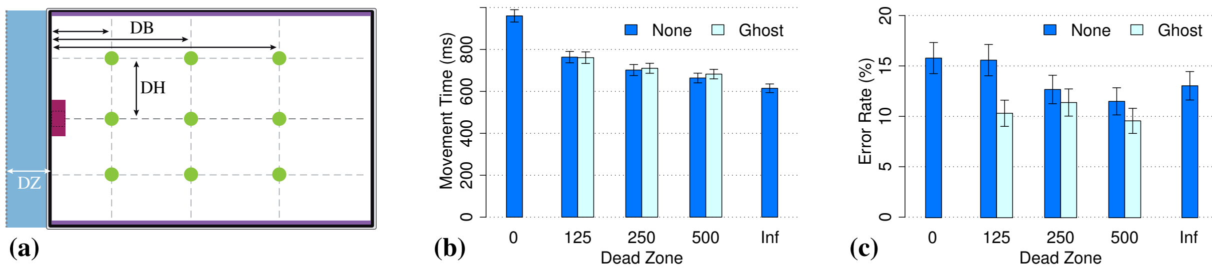

Figure 5. (a) Target placement in experiment 2. (b) MT per deadzone and feedback. (c) Error rate per deadzone and feedback.

Quantitative Results. We removed 0.97% outliers and duplicated the data for DZ = 0 and DZ = inf with the *Ghost* feedback to be able to perform a full factorial analysis FB × DZ × Random(PARTICIPANT).

An analysis of variance reveals an effect of DZ on *MT* ($F_{4,28} = 38.6, p < 0.0001$). As expected *MT* decreases as DZ increases (Figure 5b). A post-hoc Tukey test shows that (i) DZ = 0 is significantly slower than DZ $\geq$ 125; (ii) DZ = 125 is significantly slower than DZ $\geq$ 500; (iii) DZ = 250 is significantly slower than DZ = inf; and that (iv) difference is not significant for other DZ pairs. Indeed, we observe that the biggest improvement is from DZ = 0 to DZ = 125 (a 20% speed up).

As Figure 5b suggests, an analysis of variance reveals no effect of FB ($F_{1,7} = 0.42, p = 0.5463$) and no interaction FB × DZ ($F_{4,28} = 0.55, p = 0.6989$) on *MT*. Practical equivalence tests with a threshold of 20 ms (less than 3% of the grand mean) give positive results ($p \leq 0.02$), confirming there is no difference in MT between *None* and *Ghost*.

Regarding error, a nominal logistic ANOVA for the model FB × DZ ∼ *Error* (on the data set where 125 $\leq$ DZ $\leq$ 500) shows a significant effect of FB ($\chi^2 = 6.12, p = 0.0134$) but no effect of DZ ($\chi^2 = 2.79, p = 0.2477$) and no interaction FB × DZ ($\chi^2 = 1.99, p = 0.3697$). Figure 5c shows that *Ghost* is less error-prone than *None*.

For *DZOverShoot*, the percentage of trials with accidental cursor wrapping is significantly higher without a dead zone (24.37% for DZ = 0 and less than 7% for DZ > 0). We noticed small differences between *Ghost* and *None* – *Ghost* always yielding less overshoots – but these are not significant.

Regarding *UseDistDZ*, i.e., the distance covered in the dead zone, we observed that the 90% quantile is close to half the dead zone size for all DZ < inf. It is close to 600 pixels for DZ = inf, a result consistent with previous studies [1].

Qualitative Results. After the experiment, participants were asked to tell (i) whether the *Ghost* feedback helped them select the target and (ii) whether they found the feedback distracting. Six participants out of eight agreed or strongly agreed that the feedback helped (one was neutral and one disagreed). However, half the participants agreed or strongly agreed that the feedback was also distracting.

Summary. Back to our first question *Q1*, this study confirms that when cursor wrapping is enabled, users are more efficient at selecting targets on the screen edges if a dead zone is provided. Not only a dead zone expands these targets ($W = 40 + DZ$), but it also prevents accidental cursor wrapping that can be time-costly to recover from. The high cost of accidental wrapping is confirmed by participants' conservative use of dead zones when doing edge pointing. As Figure 5b does not exhibit an asymptote for DZ < inf, the study does not suggest an optimal dead zone size. It however reveals that a small deadzone (125 pixels) is enough to reduce movement time by 20%.

Regarding *Q2*, we observe that the *Ghost* feedback does not impair performance but does not improve it either. However, it significantly reduces errors, suggesting that feedback makes users more accurate. Since in real systems pointing errors can have a high cost in terms of time and user frustration, this further confirms that *Ghost* feedback should be provided. Some users might however find the feedback distracting, as suggested by answers to our questionnaire.

COMPARING DIRECT POINTING & TORUSDESKTOP

In the two previous experiments, we validated and refined the design of TORUSDESKTOP by confirming the benefits of a wrapping dead zone and by identifying the best wrapping feedback technique. The goal of this third experiment is to evaluate TORUSDESKTOP by comparing it with conventional pointing (i.e., is it worth opening the backdoor?).

To this end, we presented subjects with various pointing tasks and had them either use direct pointing only (condition *Direct*) or use the backdoor only (condition *Wrapping*). The goal was to assess if TORUSDESKTOP can help, and when. But since in real settings deciding whether or not to use cursor wrapping may take time and/or yield suboptimal choices, we added a more realistic condition where it was up to the subject to go through the backdoor or not (condition *Torus*).

Figure 6b qualitatively illustrates our initial expectations. Using *Direct*, the further apart the start and the goal targets are, the higher the movement time. *Wrapping* is likely to show the opposite trend since the further apart the targets are, the closer they are on a torus topology (but note that Fitts' law cannot account for possible distracting effects of cursor wrapping). We hypothesized that performance under the *Torus* condition would roughly follow the minimum of *Direct* and *Wrapping*, plus a possible penalty due to choice.

Apparatus & Participants

The apparatus was the same as in the previous experiments. We recruited a total of 18 participants (5 female), all right-handed and experienced mouse users, with ages ranging from 23 to 35 (median 27.8). 14 of them participated in at least one of the previous experiments or pilot studies.

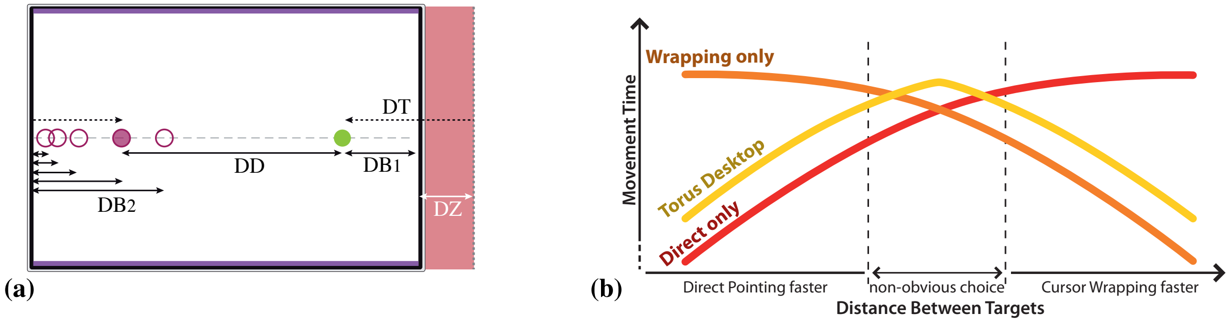

Figure 6. (a) Targets placement in experiment 3. The start target is green and potential goal targets are outlined in red. When the start target is acquired, the actual goal target appears with a solid color and other targets disappear. (b) Expected performance model. TORUSDESKTOP should behave as the most efficient technique according to the distance, though a penalty may be considered because of the choice between the techniques.

Stimuli & Design

Given the results of previous experiments, we used the *Ghost* feedback and a dead zone of 125 pixels for *Wrapping* and *Torus*. Recall this dead zone size yields a reasonable trade-off that meets the demands of both edge pointing and Torus pointing (i.e., neither of them is strongly penalized). As before, subjects had to click on a start target and acquire a goal target as fast as possible. Both targets were located on the horizontal centerline of the screen and were 40-pixel large[3].

At the beginning of a trial, all potential goal targets were shown. When the subject acquired the start target, the actual goal target appeared with a solid color and non-targets disappeared (see Figure 6a). This design was motivated by the inclusion of the *Torus* condition. In real settings, users might or might not know exactly where to click when they initiate a pointing movement. Our design is a trade-off between these two situations, since it reminds users of the possible target locations, but does not give them complete information about the task to prevent them from carefully deciding whether or not use the backdoor before the timing starts.

In addition to the pointing conditions TECH, the experiment included the factor DB1, the distance from the start target to the closest screen edge; and DB2, the distance from the goal target to the opposite edge. These two factors fully define the pointing tasks, whose direct pointing distance is DD = $2560 - (\text{DB1}+\text{DB2})$, where 2560 is the screen width; and whose torus pointing distance is DT = $\text{DB1}+\text{DB2}+125$, where 125 is the size of the dead zone (Figure 6a). Both DB1 and DB2 values were $\{50, 125, 250, 500, 750\}$. We chose these values according to an extensive pilot study suggesting that among all possible pointing tasks defined by these DB1×DB2 pairs, 7 clearly favor *Wrapping*, 7 clearly favor *Direct* and the remaining 11 yield comparable performances.

The presentation order of the techniques was counterbalanced. Prior to the experiment, the *Direct* and *Wrapping* techniques were introduced to the participants with two short practice sessions. Then, the experiment was divided in two parts. First, participants performed 4 series of 25 trials per technique. A series of trials included all the possible combinations of DB1 and DB2 and was fully randomized. This part was exclusively a training session. Then, participants performed 1 series of practice trials followed by 5 series of measured trials per technique. Thus, for each participant, we collected a total of 3 (TECH) × 5 (repetition) × 5 (DB1) × 5 (DB2) = 375 trials for analysis.

We collected movement time (*MT*) and errors (*Error*) defined as in previous experiments. The experiment lasted about 45 minutes after which participants were given a short questionnaire and were interviewed about the strategies they developed in the *Torus* condition.

Quantitative Results

We removed 0.76% outliers defined as in previous experiments and performed a full factorial analysis with the model TECH×DD×Random(PARTICIPANT) and the finer model TECH×DB1×DB2×Random(PARTICIPANT). We found no learning effect and no significant difference in performance between the 14 subjects who were involved in preliminary experiments and the 4 new subjects.

Average Performance. The ANOVA reveals no effect of TECH on *MT* ($F_{2,34} = 0.245$, $p = 0.7836$ for the DD model and $F_{2,34} = 0.612$, $p = 0.5481$ for the DB1×DB2 model). Mean *MT* are close for *Direct* ($1091\,ms$), *Wrapping* ($1094\,ms$) and *Torus* ($1108\,ms$). These similarities confirm that we chose well-balanced pairs of DB1×DB2 for *Wrapping* and *Direct*, but also suggest that the improvement promised by *Torus* may have been outweighted by the cost of choice. This will be discussed later.

We found a significant effect of TECH on *ErrorRate* ($F_{2,34} = 7.74$, $p = 0.0017$ for the DD model and $F_{2,34} = 7.40$, $p = 0.0021$ for the DB1×DB2 model). A post-hoc Tukey test shows that *Wrapping* and *Torus* are significantly less error-prone than *Direct*, with an error rate of about 6.6% for *Wrapping* and *Torus* versus 9.2% for *Direct*. A possible explanation is that participants were more careful with *Wrapping* and *Torus*, as they were less familiar with these techniques than with *Direct*.

Effect of Direct Distance. The ANOVA reveals a significant effect of DD ($F_{13,221} = 14.0$, $p < 0.0001$) and a significant interaction TECH×DD on *MT* ($F_{26,442} = 12.5$, $p < 0.0001$). We found no significant effect or interaction on *ErrorRate*.

Figure 7 shows *MT* as a function of DD for the three techniques. In accordance with our first intuitions, *Direct* gets slower as DD increases and *Wrapping* gets faster, although the

[3]Pilots studies did not show any effect of target size when comparing TORUSDESKTOP and direct pointing.

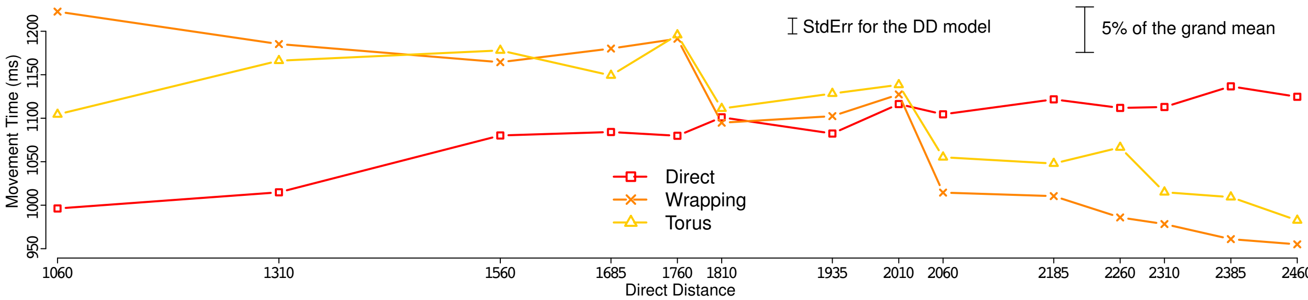

Figure 7. ***Direct*** **vs.** ***Wrapping*** **vs.** ***Torus*** **as a function of** DD**. Since the y-axis has its origin at 950, scale bars of error and grand mean are shown.**

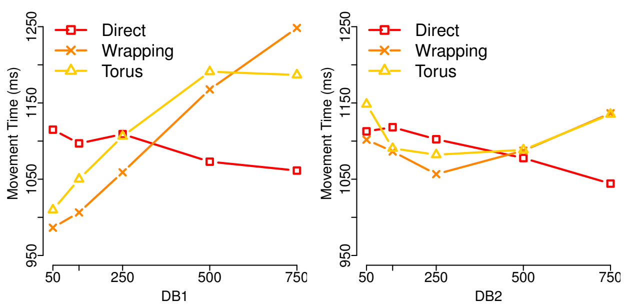

Figure 8. ***MT*** **as a function of** DB1 **(left) and** DB2 **(right) per** TECH**.**

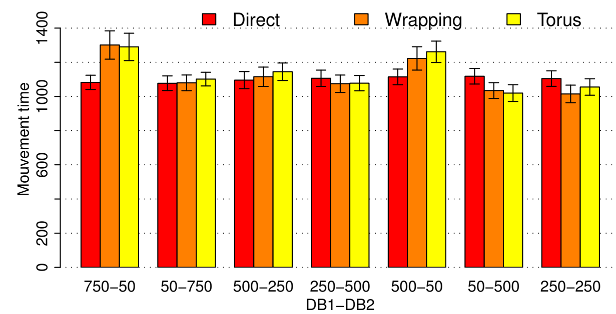

Figure 9. ***MT*** **as a fct. of certain critical** DB1-DB2 **couples per** TECH**.**

curve exhibits some irregularities (which will be explained when analyzing DB1 and DB2). The two techniques are comparable where the two curves cross, i.e., between DD=1810 and 2010 pixels. This corresponds to a Torus travel distance of only DT=675 to 875 pixels, suggesting that *Wrapping* is slower than what Fitts' law would have predicted. Taking trials where DD$\sim$DT, we estimate this penalty to about 200 ms [4]. This penalty is likely due to the difficulty in reacquiring the mouse cursor, but far from invalidating the whole approach, it merely increases the target distance above which *Wrapping* starts to be beneficial. Indeed, post-hoc tests show clear benefits for *Wrapping* above DD=2010 pixels, i.e., 80% the screen size in our experimental setup.

Figure 7 shows that the behavior of *Torus* is similar to *Wrapping* for DD>2010 pixels, where it exhibits a choice penalty of about 50 ms but still clearly outperforms *Direct*. For DD=1810 to 2010 the 3 conditions exhibit similar performance. The left part of the curve is however less consistent with our initial expectations: for DD<1810, the performance with *Torus* is close to *Wrapping* instead of being close to *Direct* as in Figure 6b. One explanation is that participants failed to choose direct pointing when it was more efficient (our later experimental data confirms this). However, *Torus* also gets closer to *Direct* as DD decreases, which suggests that subjects might still favor *Direct* when it is clearly beneficial.

Effects of DB1 *and* DB2*.* We found significant effects of both DB1 ($F_{4,68} = 68.9$, $p < 0.0001$) and DB2 ($F_{4,68} = 3.68$, $p < 0.0090$) on *MT*. We also found significant interactions DB1×DB2 ($F_{16,272} = 6.29$, $p < 0.0001$), DB1×TECH ($F_{8,136} = 28.5$, $p < 0.0001$) and DB2×TECH ($F_{8,136} = 6.67$, $p < 0.0001$). We found no significant effect or interaction on *ErrorRate*.

[4]This value is consistent with a Fitts' law analysis we conducted on an extensive pilot study comparing *Wrapping* with *Direct*.

The interaction DB1×TECH can be observed in Figure 8 left. As DB1 increases, *MT* decreases for *Direct* (because DD decreases) and increases for *Wrapping* (because DT increases). For *Torus*, *MT* behaves like *Wrapping* up to DB1=750, suggesting *Direct* was preferred when the start target was very far from the edge. Ideally, *MT* should have followed *Direct*'s trend starting from DB1=500, but both the cost of the choice and the overuse of the backdoor seem to have prevented this.

Surprisingly, the interaction DB2×TECH is quite different (see Figure 8 right). For the same reasons as above, *MT* decreases with DB2 for *Direct*. But for *Wrapping*, *MT* follows a catenary curve with a minimum at DB2=250. Since *Wrapping* should normally increase with DB2, this suggests an issue with goal targets being very close to the edge. This issue also impacts the *Torus* condition, which exhibits the same minimum at DB2=250.

The asymmetric effects of DB1 and DB2 are further detailed in Figure 9, which shows *MT* by TECH for each DB1-DB2 pair that yields a DD value of 1760, 1810, 2010 or 2060. These values correspond to the irregularities we previously observed in Figure 7. Figure 9 confirms that *Wrapping* does poorly when DB2 is small. For example, *Wrapping* does much worse with the pair 750-50 than the pair 50-750, despite these pairs yielding the same DD=1760. This is the cause for the peak in Figure 7. This peak is followed by a sharp decline at DD=1810 which involves the more balanced pairs 500-250 and 250-500. *Torus* exhibits the same irregularities.

This asymmetry can be explained in the light of the optimized dual sub-movement model [22] and by considering *when* in the pointing movement cursor warping occurs. Using *Wrapping*, the mouse cursor first travels the distance DB1 + DZ, then warps and re-appears on the opposite side, after which it travels a distance DB2 before reaching the target. If DB2 is small compared to DB1 + DZ, the warping occurs at

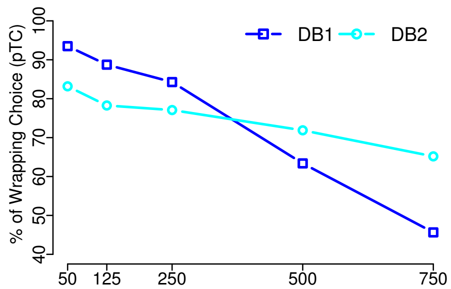

Figure 10. Percentage of *Wrapping* choice for DB1 and DB2.

the end of the movement – i.e., the corrective phase where visual feedback is the most crucial. Cursor warping requires attention shift, which likely disrupts the corrective process and slows users down. Conversely, if DB2 is large compared to DB1 + DZ, the warping happens during the initial ballistic phase of the movement where visual feedback is not used, thus its impact on performance is less severe.

Note that we could have ran a post-hoc analysis, but doing so with such a large number of data points is subject to methodological issues (high risks of type I or type II errors) and a correct analysis would have required a fair amount of space to justify and report. Since the significant interactions we found and the Figures 7, 8 and 9 are already quite informative we chose not to perform theses analyses.

Choice Strategies

So far, our results show that using the backdoor was beneficial when more than 80% of the screen had to be traveled. However, we observed mixed results when subjects had to make a choice, especially when direct pointing was the best choice. Therefore, we further analyze the choices made in the *Torus* condition. We consider the measure *pTC*, i.e., the % of the time where subjects chose wrapping, the model DD×Random(PARTICIPANT) and the finer model DB1×DB2×Random(PARTICIPANT) for this condition.

Unsurprisingly, DD has an effect on *pTC* ($F_{13,221} = 25.2$, $p < 0.0001$) and *pTC* increases with DD: the larger the distance to travel, the more often the backdoor was used.

DB1 and DB2 also have an effect on *pTC* ($F_{4,68} = 28.0$, $p < 0.0001$ and $F_{4,68} = 9.61$, $p < 0.0001$ respectively), with no DB1×DB2 interaction. As can be seen in Figure 10, the closer to the edges the targets were, the more often cursor wrapping was used. The dissimilar slopes further suggest that subjects gave more weight to the distance of the start target when they had to make a choice. This might be due to the fact that this information was available before DB2.

During the post-experiment interviews, participants reported using different strategies that can be summarized as:

- START: only wrap when the start target is close to the edge.
- GOAL: only wrap when the goal target is close to the edge.
- WRAP: always wrap the mouse cursor.
- DIRECT: always use direct pointing.
- NOCLUTCH: take the path that minimizes mouse clutching.
- RANDOM: choose more or less randomly.

10 participants reported relying mostly on START and 3 reported using it as a secondary strategy. 5 participants reported using WRAP as their primary strategy and 1 mentioned it as a secondary strategy. All the other strategies have been mentioned as a main strategy only once. GOAL was mentioned as a secondary strategy 3 times. Reported strategies were consistent with mean *pTC* per participant and with our analyses of the effects of DB1 and DB2 on *pTC*, except for two participants who reported using START and DIRECT but actually chose wrapping 93% and 76% of the time.

Overall, participants overused the backdoor: the global mean of *pTC* is 75% (std dev. 15%, median 74%). Even in the worst case scenario (DB1=DB2=750, DD=1060 and DT=1625), wrapping was used about 30% of the time. This trend could be partly due to a "good user" effect. It is also likely that participants were not accurate enough at estimating when cursor wrapping would beat direct pointing. It could be that with more training, users would develop a habit of the technique and start making close-to-optimal choices. But since we used an extensive training session and did not find a learning effect – even for subjects who were not involved in preliminary experiments – TORUSDESKTOP could probably benefit from visual clues that help users make optimal choices and develop more rational strategies.

Another question concerns the cognitive load associated with the choice. Although we did not measure cognitive load formally, we gave a post-experiment questionnaire where we asked subjects if they found it difficult to choose between direct pointing and wrapping. Out of 18, 4 strongly disagreed and 9 disagreed, suggesting cognitive load is moderate.

CONCLUSION AND FUTURE DIRECTIONS

Despite being an old idea, cursor wrapping is a simple and target-agnostic way of reducing target distance in pointing tasks. We discussed how such a technique should be designed and proposed the TORUSDESKTOP technique: it includes a dead zone that prevents accidental cursor warping and facilitates edge pointing, and a visual feedback that helps keeping track of the cursor inside the dead zone.

We tested several variations over this design and found that our *Ghost* off-screen feedback reduces overshoots during both edge pointing and cursor wrapping and is well-received by end-users, and that a 125-pixel dead zone (5% the screen size[5]) yields good performance for edge pointing while not sacrificing cursor wrapping performance. Recall the optimal dead zone size is infinite for edge pointing and is zero for cursor wrapping. However, a 125-pixel dead zone size is a reasonable trade-off where neither task is strongly penalized.

We also compared TORUSDESKTOP with direct pointing and uncovered the following potential sources of difficulties with the cursor wrapping approach:

- Cursor teleportation adds a time penalty of ~200ms,
- Targets very close to the edges are harder to acquire,
- Chosing whether or not to use the backdoor has some cost.

[5]All figures are given according to our experimental setup that involves a 30" 2560x1600 display.

In our study, the cost of choice took the form of a small time penality (∼50ms) when cursor wrapping was the most beneficial, and of suboptimal choices (overuse of the backdoor) when direct pointing was the best option.

However, rather than invalidating the whole approach, these difficulties merely increase the travel distance above which TORUSDESKTOP starts being beneficial. Indeed, our study shows that cursor wrapping outperforms direct pointing above a travel distance of 2010 pixels (80% the screen size), and these benefits are preserved when users have to choose between direct pointing and cursor wrapping. These benefits can translate to much higher gains when one needs to regularly acquire targets close to an edge (e.g., toolbar buttons), or when going back-and-forth between two very distant targets (e.g., toolbars placed at opposite sides of the screen). However, despite extensive training, our study participants were not very accurate at estimating which technique will be the most efficient under a given condition. We are investigating how to augment TORUSDESKTOP with visual clues and feedforward techniques to help users make optimal choices and develop better strategies in the long run.

Further design is also required to support TORUSDESKTOP in multi-display environments. Several strategies can be considered such as disabling cursor wrapping on adjacent screen edges, restricting wrapping to the active screen or supporting on-demand wrapping/screen jump.

Finally, a field study of TORUSDESKTOP is clearly needed to validate the approach and ensure that cursor wrapping can be adopted and effectively used by end users in their everyday desktop usage. As a first step towards this goal, we implemented an application that enables TORUSDESKTOP at the system-level on Mac OS X and that is freely available at `http://insitu.lri.fr/TorusDesktop`.

REFERENCES

1. C. Appert, O. Chapuis, and M. Beaudouin-Lafon. Evaluation of pointing performance on screen edges. In *AVI '08*, 119–126. ACM, 2008.
2. T. Asano, E. Sharlin, Y. Kitamura, K. Takashima, and F. Kishino. Predictive interaction using the delphian desktop. In *UIST '05*, 133–141. ACM, 2005.
3. R. Balakrishnan. "Beating" Fitts' law: virtual enhancements for pointing facilitation. *IJHCS*, 61(6):857–874, 2004.
4. P. Baudisch, E. Cutrell, M. Czerwinski, D. Robbins, P. Tandler, B. Bederson, and A. Zierlinger. Drag-and-pop and drag-and-pick: techniques for accessing remote screen content on touch- and pen-operated systems. In *Interact '03*, 57–64. IOS, 2003.
5. P. Baudisch, E. Cutrell, K. Hinckley, and R. Gruen. Mouse ether: accelerating the acquisition of targets across multi-monitor displays. In *CHI '04 EA*, 1379–1382. ACM, 2004.
6. P. Baudisch, E. Cutrell, and G. Robertson. High-density cursor: a visualization technique that helps users keep track of fast-moving mouse cursors. In *Interact '03*, 236–243. IOS, 2003.
7. P. Baudisch and R. Rosenholtz. Halo: a technique for visualizing off-screen objects. In *CHI '03*, 481–488. ACM, 2003.
8. H. Benko and S. Feiner. Pointer warping in heterogeneous multi-monitor environments. In *GI '07*, 111–117. ACM, 2007.
9. R. Blanch, Y. Guiard, and M. Beaudouin-Lafon. Semantic pointing: improving target acquisition with control-display ratio adaptation. In *CHI '04*, 519–526. ACM, 2004.
10. R. Blanch and M. Ortega. Rake cursor: improving pointing performance with concurrent input channels. In *CHI '09*, 1415–1418. ACM, 2009.
11. G. Casiez, D. Vogel, R. Balakrishnan, and A. Cockburn. The impact of control-display gain on user performance in pointing tasks. *HCI*, 23(3):215–250, 2008.
12. O. Chapuis, J.-B. Labrune, and E. Pietriga. Dynaspot: speed-dependent area cursor. In *CHI '09*, 1391–1400. ACM, 2009.
13. O. Chapuis and N. Roussel. UIMarks: Quick graphical interaction with specific targets. In *UIST '10*, 173–182. ACM, 2010.
14. J. Farris, K. Jones, and B. Anders. Factors affecting the usefulness of impenetrable interface element borders. *Human factors*, 44(4):578, 2002.
15. T. Grossman and R. Balakrishnan. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *CHI '05*, 281–290. ACM, 2005.
16. Y. Guiard, R. Blanch, and M. Beaudouin-Lafon. Object pointing: a complement to bitmap pointing in guis. In *GI '04*, 9–16. CHCC Society, 2004.
17. S. Gustafson, P. Baudisch, C. Gutwin, and P. Irani. Wedge: clutter-free visualization of off-screen locations. In *CHI '08*, 787–796. ACM, 2008.
18. A. Hurst, J. Mankoff, A. Dey, and S. Hudson. Dirty desktops: using a patina of magnetic mouse dust to make common interactor targets easier to select. In *UIST '07*, 183–186. ACM, 2007.
19. M. Kobayashi and T. Igarashi. Ninja cursors: using multiple cursors to assist target acquisition on large screens. In *CHI '08*, 949–958. ACM, 2008.
20. S. MacKenzie. Fitts' law as a research and design tool in human-computer interaction. *HCI*, 7:91–139, 1992.
21. M. McGuffin and R. Balakrishnan. Fitts' law and expanding targets: experimental studies and designs for user interfaces. *ACM ToCHI*, 12(4):388–422, 2005.
22. D. Meyer, J. Smith, S. Kornblum, R. Abrams, and C. Wright. Optimality in human motor performance: Ideal control of rapid aimed movements. *Psych. Review*, 95(3):340–370, 1988.
23. M. Nacenta, R. Mandryk, and C. Gutwin. Targeting across displayless space. In *CHI '08*, 777–786. ACM, 2008.
24. G. Ramos, A. Cockburn, R. Balakrishnan, and M. Beaudouin-Lafon. Pointing lenses: Facilitating stylus input through visual- and motor-space magnification. In *CHI '07*, 757–766, 2007.
25. K.-J. Räihä and O. Spakov. Disambiguating Ninja cursors with eye gaze. In *CHI '09*, 1411–1414. ACM, 2009.
26. J. Wobbrock, J. Fogarty, S.-Y. Liu, S. Kimuro, and S. Harada. The angle mouse: target-agnostic dynamic gain adjustment based on angular deviation. In *CHI '09*, 1401–1410. ACM, 2009.
27. A. Worden, N. Walker, K. Bharat, and S. Hudson. Making computers easier for older adults to use: area cursors and sticky icons. In *CHI '97*, 266–271. ACM, 1997.
28. S. Zhai, C. Morimoto, and S. Ihde. Manual and gaze input cascaded (MAGIC) pointing. In *CHI '99*, 246–253. ACM, 1999.