DynaSpot: Speed-Dependent Area Cursor

Olivier Chapuis, Jean-Baptiste Labrune, Emmanuel Pietriga

DynaSpot: Speed-Dependent Area Cursor

Olivier Chapuis[1,3]
chapuis@lri.fr

Jean-Baptiste Labrune[2]
labrune@media.mit.edu

Emmanuel Pietriga[3,1]
emmanuel.pietriga@inria.fr

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

[2]MIT Media Lab
Cambridge, MA, USA

[3]INRIA
Orsay, France

ABSTRACT
We present DynaSpot, a new technique for acquiring targets based on the area cursor. DynaSpot couples the cursor's activation area with its speed, behaving like a point cursor at low speed or when motionless. This technique minimizes visual distraction and allows pointing anywhere in empty space without requiring an explicit mode switch, thus enabling users to perform common interactions such as region selections seamlessly. The results of our controlled experiments show that the performance of DynaSpot can be modeled by Fitts' law, and that DynaSpot significantly outperforms the point cursor and achieves, in most conditions, the same level of performance as one of the most promising techniques to date, the Bubble cursor.

ACM Classification Keywords
H.5.2 Information Systems: Information Interfaces and Presentation – User Interfaces, Input Devices and Strategies

Author Keywords
Area cursor, Bubble cursor, DynaSpot, Fitts' Law

INTRODUCTION

The increase in both resolution and size of computer displays requires users of desktop interfaces based on the ubiquitous WIMP paradigm to make highly precise pointing movements to acquire small interface components over possibly long distances when using a conventional point cursor. Several techniques have been proposed to make this fundamental task easier. Many have been shown to perform better than the point cursor in experimental settings that were consisting of isolated targets on fairly sparse desktops [1, 3, 5, 10]. However, these techniques are very sensitive to the layout and density of interface components, and difficulties arise when selecting one target among multiple objects that are spatially close together. As noted by Baudisch et al. [4], non-uniform target distributions with clusters of small targets are commonplace in GUIs. In such configurations, these techniques do not provide a significant advantage and some can actually degrade performance.

Other promising techniques have been proposed recently that work better in a wider range of configurations, including many variations on expanding targets [6, 7, 20, 21], the Ninja cursor [15] and Starburst [4]. One of the most promising technique, the Bubble cursor [9], is a variation on the Area cursor [14, 26] that dynamically adapts its activation area to encompass the closest object only. This is achieved by expanding the boundaries of each target based on a Voronoi tessellation that fills the empty space surrounding each potential target thus maximizing their effective size. While this optimizes pointing performance, problems arise when considering interaction beyond the acquisition of a single interface component. First, as with several of the above-mentioned techniques, selecting a position in the "empty" space between targets requires a mode switch. Yet empty space selection is crucial to many common interactions, e.g., to select groups of objects. The mode switch solution results in *"a slightly less than seamless interaction style"* [2] for these essential object manipulation features [15]. Second, rapid and large changes of the bubble size in non-uniform target distributions may distract the user and hinder user acceptance [9, 17, 12], a crucial factor [2] that is sometimes overlooked.

In this paper, we present DynaSpot, a new type of area cursor that couples the cursor's activation area with its speed, as illustrated in Figure 1. The activation area grows as a function of speed up to a maximum size, typically set to a few dozen pixels, thus minimizing visual distraction. At low speed and when motionless, DynaSpot behaves as a regular point cursor, making all conventional point cursor interactions, including empty space selection, possible without the need for an explicit mode switch.

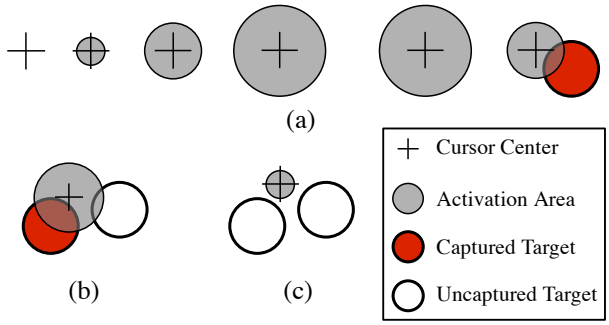

Figure 1. (a) DynaSpot's activation area is coupled to cursor speed. (b) Multiple objects intersect the area: the target closest to the cursor center is highlighted and selected. (c) Empty space selection is possible whenever the activation area is not intersecting any object.

After a review of related work, we discuss the design and implementation of DynaSpot, and report the results of two controlled experiments. Results show that DynaSpot significantly outperforms the point cursor and achieves levels of pointing performance similar to the Bubble cursor in most layout configurations, including densely populated scenes. We then show that its performance can be modeled with Fitts' law. We conclude with a discussion of our findings and directions for future work.

RELATED WORK

Fitts' law is the fundamental tool used to study pointing in human-computer interfaces [18, 25]. It makes it possible to predict movement time MT with the following equation:

$$MT = a + b \times log_2(\frac{A}{W} + 1)$$

where A is the distance to the target (amplitude of movement), W the width of the target, and a, b are two coefficients determined empirically, depending on factors such as input device and population of users. Techniques developed to facilitate pointing in virtual worlds try to decrease movement time either by reducing A, increasing W, or a combination of both. We direct interested readers to a survey by Balakrishnan [2] and an overview by Grossman et al. [9] that follow this categorization to review existing techniques. In the following, we consider existing techniques from a slightly different perspective, considering not only performance but compatibility with conventional cursor interactions beyond single target acquisition, and user acceptance.

Sticky icons [26] and Semantic pointing [5] dynamically adapt the control-display ratio, slowing down the cursor as it approaches a potential target. These techniques support conventional point cursor interactions. They are, however, very sensitive to the layout and density of potential targets; while they work well in sparsely populated workspaces, intervening distractors on the path to the actual intended target in denser workspaces slow down cursor movements, possibly degrading performance compared to a regular point cursor. Cockburn and Firth [7] propose to enable the control-display adaptation on one axis only depending on the widget's orientation, thus partially solving the problem for some types of widgets such as scrollbars.

Drag-and-pop [3] reduces amplitude of movement (A) when dragging an object by temporarily bringing potential targets, closer to the cursor. As such, the technique efficiently solves one particular type of pointing-based interaction, but is not a general desktop pointing technique. Object pointing [10] takes a radical approach, ignoring the empty space between targets by making the cursor jump from one object to the nearest one in the direction of movement, thus considerably reducing A. The Delphian desktop [1] follows the same principle, taking into account peak velocity to determine the goal target, allowing to jump over potential distractors. Both techniques are very sensitive to the layout and density of objects, which can have a strong impact on the accuracy of the goal target prediction method. Lank et al., describe an enhanced endpoint prediction method [16] achieving 42% accuracy and an additional 39% of predictions falling on an adjacent target, with one third of gesture time remaining. Still, wrong predictions can be frustrating, and the behavior of the cursor, jumping from object to object, can be annoying. By skipping empty space, these techniques do not allow the user to perform some useful point cursor interactions, such as region selection, without an explicit mode switch. Kobayashi and Igarashi propose another promising way to reduce the amplitude of movement (A) by having multiple cursor instances all synchronized with the same input device: by distributing the cursors over the screen, Ninja cursor [15] reduces the average distance to any given target using interactive, seamless disambiguation methods to activate the appropriate cursor. General point cursor interactions that require clicking in empty space are however not possible without mode switching, except for a restricted form of lasso selection.

Several techniques focus on increasing target width (W). Techniques based on lenses coupled with the cursor magnify objects but usually operate in the original, unmagnified, motor space, thus providing no actual advantage in terms of pointing facilitation [11, 23]. Ramos et al.'s Pointing lenses [24] are an exception, increasing target size in both visual and motor space for the acquisition of small targets with a stylus. Another solution consists in expanding targets dynamically when a point cursor approaches them. McGuffin and Balakrishnan [20] have found that users can still benefit from expansions that occur as late as after 90% of the movement has been completed. They were further studied in [21], and experiments by Cockburn and Brock suggest that visual expansion plays a more important role than motor expansion [6]. They also note that *"enlarged motor-spaces actually make the targets appear smaller than they really are"*, as empty space around objects is actually empty in visual space only, not in motor space, meaning that it cannot be used for interactions such as region selection.

Fitts' law can accurately model pointing to thin targets using area cursors with a simple modification to the equation: instead of representing the target width, the term W represents the cursor width [14]. This implies that cursors with larger activation areas make pointing easier, but such larger areas are more likely to encompass several objects, thus creating ambiguities. These can be resolved by using a secondary point cursor [26] or by interactively adjusting the cursor area on multi-point touchpads [22]. The Bubble cursor [9] improves upon the area cursor by partitioning empty space so as to maximize the activation area of each target. Starburst [4] relies on a different partitioning of space, better adapted to non-uniform target distributions. As mentioned earlier, this optimizes pointing performance, but prevents point cursor interactions that require clicking in empty space. The Bubble cursor's growing/shrinking area has also been reported to cause visual distraction in some situations [9, 17, 12]. Several variations on the technique have been designed [17, 12], but have had limited success both in terms of performance and user acceptance. The lazy bubble [17] makes it possible to point in some areas of empty space, but these are severely limited and difficult to identify, making interactions such as region selection impractical.

DYNASPOT

In his survey of pointing facilitation techniques [2], Balakrishnan identifies the final acceptability of a technique by end-users as a critical measure, seen as a complement to quantitative performance measures such as selection times and error rates. The visual distraction caused by some techniques and the mode switches required by earlier-mentioned techniques hinder their acceptance for many types of applications and environments. DynaSpot has been designed to facilitate pointing while taking this more qualitative measure into account. It was not designed to perform better than all other techniques under all conditions, but to strike a balance between performance, end-user acceptance and implementation in a realistic context.

DynaSpot builds upon area cursors. It uses the dynamic characteristics of the pointer to adapt the size of the cursor's activation area and facilitate difficult pointing tasks while behaving as a conventional point cursor when appropriate, without the need for an explicit mode switch. DynaSpot takes inspiration from other techniques that have successfully made use of the cursor's dynamic characteristics, such as Speed-Dependent Automatic Zooming [13], Sigma Lenses [23] and the Speed-coupled flattening lens [11].

As shown in Figure 2, the size of the activation area (which we term *spot* from now on) starts to increase as a function of cursor speed past a given threshold, and up to a maximal size SPOTWIDTH. When the cursor comes to a full stop, reduction of the spot starts after a certain duration LAG, and takes REDUCTIME to complete. As with regular area cursors, the spot is made translucent so as to avoid obscuring screen information relevant to the task [26].

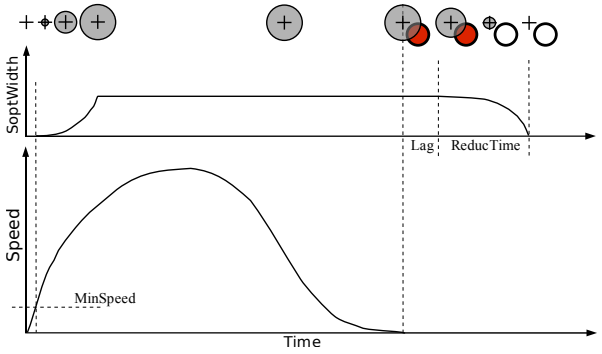

Figure 2. DynaSpot: spot width as a function of cursor speed.

A target can be selected as soon as the spot overlaps it. While early area cursor designs [14, 26] used a square shape, DynaSpot's activation area takes the shape of a circle, as does Bubble cursor's, so as to ensure that the nearest target is captured first. Still, as opposed to the latter technique, there can be situations where the spot overlaps more than one potential target, creating ambiguities regarding the one to select. To resolve such ambiguities, DynaSpot always selects the target closest to the cursor center (see Figure 1-b). This implies that the system should provide feedback about which target is currently selected (if any), as is the case for Bubble cursor. If the spot does not intersect any potential target, then the background (or "empty space") is selected, no matter the current spot width, allowing the user to perform any action initiated by a button press in empty space, such as a region selection (see Figure 1-c).

According to Fitts' law, DynaSpot should facilitate pointing because the *potential effective width* of a target can be larger than its actual width. For instance, if we consider an isolated circular target of width W and a spot width of SW at the time of actual target selection (i.e., when clicking), then the potential effective width is $EW = W + SW$. Thus, when the user clicks on the target before the spot starts shrinking (before the end of LAG in Figure 2), the effective width of the target is $EW = W +$ SPOTWIDTH, as illustrated in Figure 3-a. If we consider a target surrounded by other targets with empty space between them of width IS, as in Figure 3-b, then the potential effective width depends on the spot width and what we term the *interspace* between targets, IS. If $IS \leq$ SPOTWIDTH, then the potential effective width is $EW = W + IS$. The optimal SPOTWIDTH will depend on a number of factors: interface type, display resolution, input device, but also on each user. In a typical desktop environment, a SPOTWIDTH between 16 and 32 pixels represents a good compromise: it is large enough to facilitate the acquisition of small targets, yet small enough to prevent size variations from causing too much visual distraction.

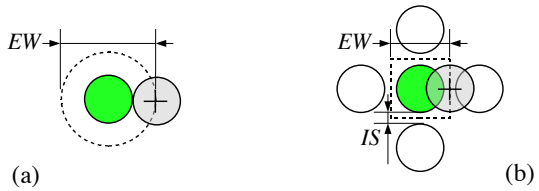

Figure 3. (a) Isolated target: potential effective width $EW = W +$SPOTWIDTH. (b) Small interspace between targets $IS \leq$ SPOTWIDTH: potential effective width is $EW = W + IS$.

Speed-dependent Behavior

Figure 2 gives a general idea of the speed coupling between cursor speed and spot width. The details of this coupling play a fundamental role in the overall usability of the technique, and are described in this section. The behavior rules are as follows:

- when the cursor is moved fast enough (beyond a threshold speed $S \geq$ MINSPEED pixel.s^{-1}), the width of the spot is increased, provided that it has not yet reached its maximal value SPOTWIDTH;
- when the cursor comes to a full stop and does not move for a period of time equal to LAG, the spot shrinks to a point (1 pixel) over a period of REDUCTIME, provided the user does not move it again, in which case it would grow again;
- for slow movements below the speed threshold, the spot width remains constant.

Threshold speed MINSPEED allows the user to perform small, precise pointing movements using a conventional point cursor, without being distracted by a growing spot. When the cursor is moved faster, beyond this threshold, the spot grows, facilitating distant target acquisition. We have found 100 pixel.s^{-1} to be a reasonable value for MINSPEED.

The transitions from point cursor to area cursor and conversely can be achieved in various ways. We tested several possibilities through trial and error, and made the following observations. The spot should grow quickly once MINSPEED has been reached, but the growth profile does not seem to play an important role. We found that an exponential growth (up to SPOTWIDTH) by a factor of 1.2 at each input event works well.

The reduction transition, controlled by LAG and REDUCTIME (see Figure 2), is more complex because it has a direct impact on the potential effective width at the time of target selection. Higher values for both parameters should make the task easier. However, too high values imply that the user will potentially have to wait longer before she can perform interactions initiated by an implicit selection in empty space. In addition, the reduction profile applied during REDUCTIME also plays a role.

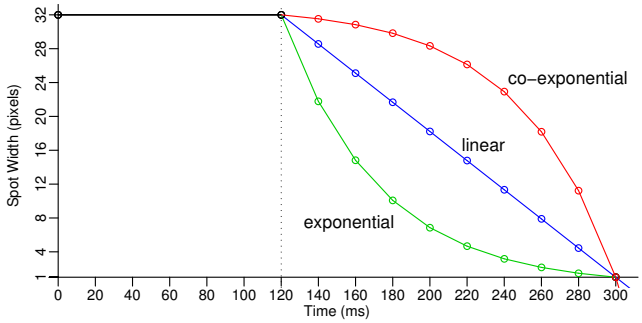

Figure 4. Spot width as a function of time for the three reduction methods (LAG = 120 ms, REDUCTIME = 180 ms).

We informally tested three methods to perform this reduction, as illustrated in Figure 4: (i) an "exponential" one where the spot width is reduced by a given percentage at each step; (ii) a "linear" one where the spot width is reduced by a given constant at each step, and (iii) a "co-exponential" one that mirrors the first method. We found that the exponential reduction yields more target acquisition errors, probably because of the abrupt transition after the LAG period, due to the fast reduction of the spot. The co-exponential method starts reducing the spot at a lower pace, providing a smoother transition than the linear and exponential methods.

PRELIMINARY STUDY: LAG AND REDUCTION TIME

Before comparing DynaSpot with other pointing techniques, we ran a preliminary experiment in which we formally evaluated different values of LAG and REDUCTIME for the co-exponential transition in order to fine-tune the technique.

Apparatus

We used a 3.2 GHz Pentium4 PC running X Window under Linux, equipped with an NVidia Quadro FX 1500 graphics card, a 1600 x 1200 LCD monitor (21"), and a standard optical mouse (400 dpi) with the default X Window acceleration function. Our program was written in Java using the OpenGL pipeline for graphics rendering. We carefully checked the refresh rate (50 fps), ensuring that timers were matching the lag and reduction set for each condition.

Participants

Eight unpaid adult volunteers (7 male, 1 female), from 22 to 41 year-old (average 26.6, median 24), all right-handed, experienced mouse users, served in the experiment.

Procedure and Design

The task was a simple reciprocal pointing task. The two targets were represented as circles 8 pixels in diameter, painted with a green fill color and outlined in black. They were centered horizontally, with a distance of 512 pixels between them, and were each surrounded by four distractors of the same size, painted with a white fill color and outlined in black. These four distractors were laid out so that the interspace IS between a distractor and the target would always match the SPOTWIDTH set for the current trial, as illustrated in Figure 8-b. We focused on small targets in this preliminary study as DynaSpot is expected to be most useful in this type of configuration. The object captured by the cursor (distractor or actual target, if any) was filled with a red color. Each target had to actually be selected before proceeding to the next: clicks outside the current target were counted as errors but did not end the task.

Our experiment was a $2 \times 3 \times 3$ within-participant design. Each participant had to perform several trials using two spot widths: SPOTWIDTH $\in \{16, 32\}$ with three durations for both lag and reduction time: LAG $\in \{60ms, 100ms, 140ms\}$ and REDUCTIME $\in \{100ms, 180ms, 260ms\}$.

We grouped trials into two blocks, one for each spot width. Four participants started with the small DynaSpot (16 pixels) while the four others started with the larger one (32 pixels). Within a block, trials were grouped by LAG $\times$ REDUCTIME condition presented in a pseudo-random order, each sub-block containing three series of 16 reciprocal pointing tasks. The first series was used for training, allowing participants to adapt to the new parameters before we measured their performance. They were then instructed to be as accurate and as fast as possible. The 16 pointing tasks of a series had to be performed in a row, but participants were allowed to rest between trials. The first targeting task of each trial was ignored. A total of 4,320 actual pointing tasks were thus taken into account in the analysis (240 measures for each SPOTWIDTH $\times$ LAG $\times$ REDUCTIME condition). The experiment started with a 3 minute training session where the experimenter explained DynaSpot's behavior and how to operate it to the participant. The experiment lasted approximately 20 minutes.

Results

Repeated measures analysis of variance reveals a significant simple effect on movement time for SPOTWIDTH ($F_{1,7} = 97.0$, $p < 0.0001$), LAG ($F_{2,14} = 11.5$, $p = 0.0011$) and REDUCTIME ($F_{2,14} = 7.5$, $p = 0.006$). The only significant interaction is for LAG $\times$ REDUCTIME ($F_{4,28} = 3.3$, $p = 0.0238$).

Mean movement time is 884 ms for SPOTWIDTH = 16 and 760 ms for SPOTWIDTH = 32. The LAG $\times$ REDUCTIME effect can be observed on Figure 5 (left): LAG seems to have an

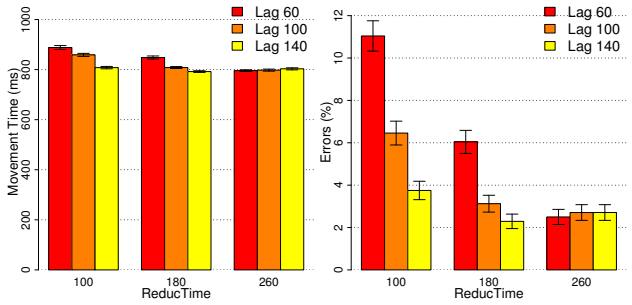

Figure 5. Movement time (left) and error rate (right) as a function of LAG, grouped by REDUCTIME.

effect for REDUCTIME equal to 100 and 180 ms but not for REDUCTIME = 260 ms. This is confirmed by post-hoc tests, which show a significant difference in mean between all LAG values for REDUCTIME = 100, between LAG 60 and 100 for REDUCTIME = 180, but no significant difference for REDUCTIME = 260. The overall error rate is 4.5%. Repeated measures analysis of variance reveals no effect on error rate for SPOTWIDTH ($F_{1,7} = 0.4$, $p = 0.522$), but a significant effect for LAG ($F_{2,14} = 10.8$, $p = 0.0014$) and REDUCTIME ($F_{2,14} = 17.7$, $p < 0.0001$). As for movement time, we observe a significant interaction for LAG × REDUCTIME only ($F_{4,28} = 3.4$, $p = 0.0227$), as illustrated in Figure 5 (right).

These results show that for a long-enough REDUCTIME, LAG can be set to any value within the considered range. For shorter REDUCTIMES, the duration of LAG has a significant effect on both movement time and error rate, and has to be chosen carefully. Overall, the fastest and least error prone condition evaluated was LAG = 140 ms and REDUCTIME = 180 ms. For our implementation of DynaSpot, we did not want the full reduction phase to last longer than 300 ms, as longer delays can be frustrating. We thus used the following values: LAG = 120 ms and REDUCTIME = 180 ms.

MAIN EXPERIMENT: DYNASPOT VS. BUBBLE VS. POINT

Having fine-tuned DynaSpot's parameters, we ran a second experiment to evaluate the quantitative performance of DynaSpot and get the subjective impressions of participants. We compared two DynaSpots with different spot widths (16 and 32 pixels) against a regular point cursor, serving as a baseline, and the Bubble cursor [9], one of the most efficient general pointing techniques to date (Figure 6).

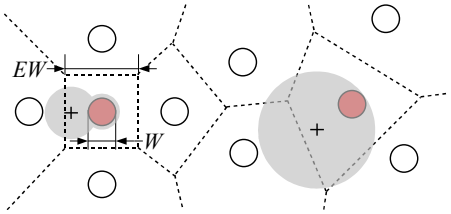

Figure 6. The Bubble cursor captures the target closest to its center. The shape of the targets are expanded to a maximal shape obtained by a Voronoi tesselation. The bubble's effective width, EW, is thus defined by the corresponding shape.

Apparatus

We used a workstation running X Window under Linux, equipped with two double core 64-bits 2.4 GHz processors, an NVidia Quadro FX4500 graphics card, a 1600 x 1200 LCD monitor (21") and a standard optical mouse (400 dpi) with the default X Window acceleration function. Our program was written in Java using the OpenGL pipeline for graphics rendering, thus ensuring a minimum frame rate of 50 fps even for large alpha-blended Bubble cursor areas (something impossible with the default Java2D rendering pipeline).

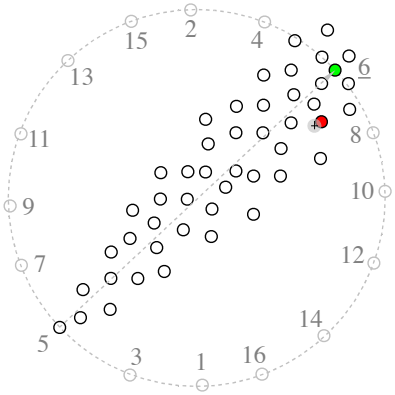

Figure 7. Sixth pointing task of a series (ISO 9241-9 circular layout)

Participants

Twelve unpaid adult volunteers (all male), from 21 to 33 year-old (average 25.2, median 25), all right-handed, experienced mouse users, served in the experiment.

Task and Procedure

We followed the same general procedure as the one used by Grossman and Balakrishnan to compare Bubble cursor with object pointing and a point cursor [9]: participants had to select a target rendered as a solid green circle outlined in black, surrounded by a set of distractors. Additional distractors were placed on the path from the trial start point to the target. As illustrated in Figure 7, all distractors were the same size as the target and were rendered as black outlined circles. As in our preliminary experiment, the object captured by the cursor (if any) was painted red. The bubble cursor area and the DynaSpot disc were both rendered with a semi-transparent gray. As in the Bubble cursor paper's experiment, four main distractor targets were positioned to control the interspace IS around, and thus the effective width EW of, the goal target[1]. Two were placed along the direction of movement, one on each side of the target, while the other two were placed perpendicular to the direction of movement (see Figure 3-b). The remaining distractors were laid out so as to match the density condition DD on the path to the target. For DD = 0, there were no additional distractors on the path to the target. For DD = 1, additional distractors were packed from the start point to the closest main distractor, and offset in the direction perpendicular to the line of movement by a pseudo-random length, keeping them within a 20 degree slice centered in this line of movement. Additional distractors outside this slice were placed pseudo-randomly to match the density within the slice. For DD = 0.5, there were half as many distractors.

[1]The original Bubble cursor experiment controlled the interspace in terms of EW/W ratio (Figure 6), which we will also use here for cross-experiments comparisons.

We made the following adjustments to the original design. Instead of making the next target appear in an unpredictable location, we laid out all 16 targets of a trial series in a circular manner. The order of appearance followed the recommendations of the ISO 9241-9 standard forcing participants to perform pointing tasks in every direction [8]. We chose this more predictable behavior of targets, encountered in several pointing experiments, e.g., [6, 11, 23, 25], as it better simulates situations where users have a rough idea about the direction of the target they are aiming at before starting the pointing task. Each target had to actually be selected before proceeding to the next: clicks outside the current target were counted as errors but did not end the task.

Design

Our experiment was a $4 \times 3 \times 3 \times 3 \times 3$ within-participant design with the following factors: *(i)* four techniques TECH: Bubble, DynaSpot16 (SPOTWIDTH = 16), DynaSpot32 (SPOTWIDTH = 32) and Point Cursor; *(ii)* three target widths W: 8, 16 and 32 pixels; *(iii)* three amplitudes A: 256, 512 and 768 pixels; *(iv)* three EW/W ratios: 1.5, 2 and 3; *(v)* three distractor densities DD: 0, 0.5 and 1.

We grouped trials into four blocks, one per technique. Each TECH block was divided into 3 sub-blocks, one per EW/W condition. Each of these sub-blocks was composed of 3 W $\times$ 3 A series of 16 pointing tasks where each DD was used 5 times (the first task of a series was not recorded). An additional sub-block at the beginning (W and A random) was used for training. To counterbalance the presentation order of conditions, we computed a Latin square for TECH and a Latin square for EW/W and crossed them, obtaining 12 orders, one for each participant. The order of the W $\times$ A conditions, as well as the density DD, were chosen randomly but the same order was used for each TECH across participants for the 15 recorded tasks of a series.

The experiment started with a training session consisting of 4 TECH $\times$ 3 EW/W series, each with W = 16 and A = 512. The experimenter introduced each technique to the participant during the first series of each corresponding TECH block; the two remaining blocks being used as actual training. For the series actually recorded, participants were instructed to be as accurate and as fast as possible. The 16 pointing tasks of a series had to be performed in a row, but participants were allowed to rest between series. A total of 19,440 actual pointing tasks were thus taken into account in the analysis (60 measures for each unique condition). The experiment lasted approximately 45 minutes.

Combined Width and Hypotheses

One of the main factors used in the experiment comparing Bubble cursor to other techniques [9] was the EW/W ratio (effective width of the goal target by its actual width). The combination of this and factor W controls the distance between the goal target and the four distractors surrounding it. This abstraction, well-adapted to the former experiment, is however not best suited to analyze the different conditions with DynaSpot.

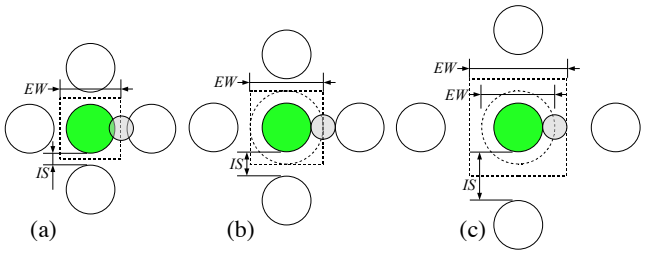

Figure 8. (a) $IS <$ DynaSpot Width: Bubble and DynaSpot have the same EW. (b) $IS =$ DynaSpot Width: Bubble and DynaSpot have the same EW, but different effective target shapes. (c) $IS >$ DynaSpot Width: Bubble's EW is greater than DynaSpot's EW.

Since the distractors are uniformly placed around the target, the ratio can be expressed in terms of interspace IS between the goal target and the distractors: $EW = W + IS$ and $(EW/W) = (W + IS)/W$. As illustrated in Figure 8, this formulation helps identify the three main conditions for DynaSpot: the spot width can be (a) larger than, (b) equal to, or (c) smaller than, the interspace IS. The factors W and EW/W can be grouped into one factor that we call the *combined width* CW. In the remainder of this paper, we use the following notation for each pair of conditions W $\times$ EW/W:

$$(BW, DW_{16}, DW_{32}, W)$$

where BW is the Bubble cursor's effective width, DW_{16} and DW_{32} are the potential effective widths for the two DynaSpot sizes, and W is the target's width (which corresponds to the effective width of the target for the point cursor). The factors described in the previous section yield nine combined widths CW, listed in Table 2. When the interspace IS is equal to one of the DynaSpot TECH's potential effective width (case (b) of Figure 8), we underline the corresponding DynaSpot (16 and 32 pixels).

Our main hypothesis is that for a given combined width CW, the effective width EW for each technique should determine the performance ordering among techniques: if the effective width for technique a is larger than for technique b (for a given combined width), then a should be faster than b (for this combined width). When the effective widths of two techniques are equal, we do not expect to find significant differences in terms of performance. However, we expect a performance degradation when DynaSpot is at its limit effective width (underlined width in the CW notation). Indeed, in this particular case, the spot reduction and small intersection between the target and the spot may forbid the user to use the full potential effective width of DynaSpot. Additionally, we hypothesize that density DD will have a similar effect on point cursor and both DynaSpots, as the behavior of all three techniques is not directly impacted by density. On the contrary, we expect a performance degradation for low densities in the case of Bubble cursor, consistent with Grossman and Balakrishnan's observations regarding visual distraction in this condition [9].

Results

Results of the repeated measures analysis of variance are reported in Table 1. We verified that there was no effect of TECH presentation order and observed that learning effects

Factors	DF	DFDen	F	p
TECH	3	33	115.5	< 0.0001
CW	8	88	509.0	< 0.0001
DD	2	22	28.5	< 0.0001
A	2	22	409.6	< 0.0001
TECH × CW	24	264	12.6	< 0.0001
TECH × DD	6	66	9.0	< 0.0001
TECH × A	6	66	2.3	0.0470
CW × DD	16	176	5.6	< 0.0001
CW × A	16	176	2.1	0.0085
DD × A	4	44	1.8	0.1435
TECH × CW × DD	48	528	1.0	0.3746
TECH × CW × A	48	528	1.2	0.1628
TECH × DD × A	12	132	2.3	0.0123
CW × DD × A	32	352	1.1	0.2637
TECH × CW × DD × A	96	1056	1.2	0.0904

Table 1. Results of the ANOVA for MT $\sim$ TECH × CW × DD × A**.**

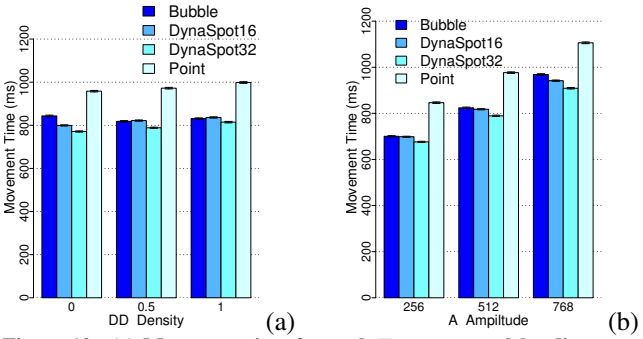

Figure 10. (a) Movement time for each TECH **grouped by distractor density. (b) Movement time for each** TECH **grouped by amplitude.**

were not significant. As expected CW and A have a significant effect on movement time MT. We also observe an effect of TECH on movement time. Mean movement time is 976 ms for Point cursor, 831 ms for Bubble, 819 ms for DynaSpot16 and 791 ms for DynaSpot32. However, the ANOVA also reveals significant interactions: TECH × CW, TECH × DD and TECH × A. A thorough comparison between techniques must thus take into account combined width, density of distractors, and amplitude.

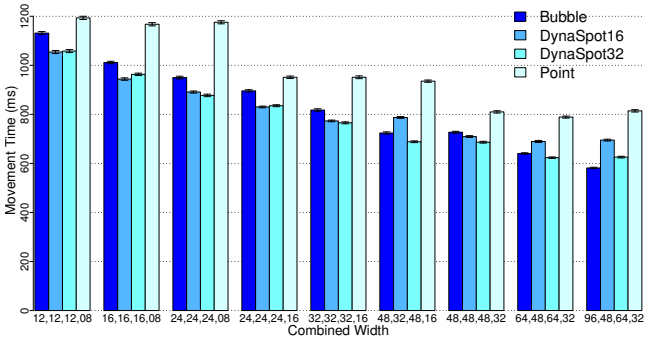

Figure 9. Mean movement time per TECH**, grouped by** CW**.**

	Tukey HSD test ($\alpha = 0.05$)		
CW	All DD conditions	DD = 0	DD = 1
(12,12,12,08)	P < D32,D16 ; B < D32,D16	-	B $\not<$ D32,D16
(16,16,16,08)	P < B,D32,D16	B < D32	-
(24,24,24,08)	P < B,D16,D32	B < D32	-
(24,24,24,16)	P < B,D32,D16	B < D32,D16	-
(32,32,32,16)	P < B,D16,D32	B < D16,D32	-
(48,32,48,16)	P < D16,B,D32 ; D16 < D32	-	D16 < B
(48,48,48,32)	P < B,D16,D32	-	-
(64,48,64,32)	P < D16,B,D32	-	-
(96,48,64,32)	P < D16,D32,B ; D16 < B	-	D32 < B

P = Point cursor, B = Bubble, DX = DynaSpotX

Table 2. Significant differences for mean movement time MT **between** TECH**, by** CW**. The two rightmost columns show how the results are modified if we restrict our analysis to distractor densities 0 and 1.**

Figure 9 shows the mean movement time for each TECH by combined width CW. Table 2 gives the results of the Tukey HSD *post-hoc* test for differences in mean between techniques by combined width (where $a < b$ means that TECH b is significantly faster than TECH a). The test shows that Bubble and DynaSpot are both significantly faster than Point and that there is little difference between Bubble and DynaSpot.

Figure 10-a shows mean movement time for each technique grouped by distractor density. We see that movement time increases as density increases for Point cursor and DynaSpot, but not for Bubble cursor. For Bubble cursor, a post-hoc Tukey test reveals that it is faster for DD=0.5 than for DD=0, confirming the results of [9]. The test also reveals that each of DynaSpot16, DynaSpot32 and Point cursor is faster for DD=0 than for DD=1. Moreover, as shown in Table 2, Bubble cursor is slower than DynaSpot for DD=0 in most conditions where the effective widths are equal, while Bubble cursor is faster than DynaSpot for DD=1 when the Bubble's effective width is larger than the DynaSpot's effective width.

Figure 10-b shows mean movement time for each technique grouped by movement amplitude. We see that the difference between Bubble cursor and DynaSpot increases with amplitude. A post-hoc Tukey test shows that DynaSpot32 is faster than Bubble for an amplitude of 768, but no such difference is detected for smaller amplitudes. Moreover, removing the data for which DD=0 makes this significant difference disappear (a cause of the TECH × DD × A interaction).

These results show that the effective width determines the performance ordering among techniques only under certain conditions regarding distractor density. Our hypothesis is thus only partially verified. Distractor density affects Bubble cursor performance, especially for large movement amplitudes. As predicted, a significant degradation is observed when DD=0, i.e., when the bubble's envelope varies most during movement, causing visual distraction. Finally, distractor density also affects Point cursor and DynaSpot in a similar way, degrading performance as it increases.

Regarding errors, we find an overall error rate of 6.5%. Repeated measures analysis of variance shows a significant effect on error rate for TECH ($F_{3,33} = 11.6$, $p < 0.0001$), CW ($F_{8,88} = 8.3$, $p < 0.0001$) and A ($F_{2,22} = 5.7$, $p = 0.0097$). Interestingly, there is no significant effect of DD ($F_{2,22} = 1.4$, $p = 0.262$). Error rate was 9.7% for Point cursor, 6.5% for Bubble cursor, and 4.9% for both DynaSpot16 and DynaSpot32. As usual in pointing task experiments, error rate decreases as the (effective) width grows and the amplitude decreases. Again, we find a significant interaction between TECH and CW ($F_{24,264} = 2.6$, $p < 0.0001$). Figure 11 shows error rate for each TECH grouped by combined width CW.

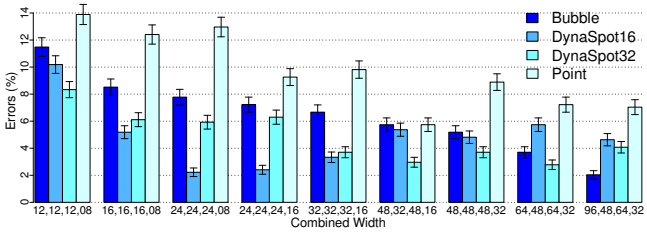

Figure 11. Error rate for each TECH grouped by combined width CW.

Removing errors trials or taking the time of the first click (instead of first *successful* click) does not change the results. We checked the number of outliers by counting the number of trials where the time is 3 standard deviations away from the mean movement time (by participant, technique, combined width, and amplitude). The data contains only 0.99% such outliers; 87% of them are errors, and none of them is more than 3.6 standard deviations away. Again the analysis without these outliers yields the same results.

Performance results for Point and Bubble cursors are consistent with those in [9]. However, our participants perform faster overall: 10.6% faster for Bubble cursor and 9.6% faster for Point cursor. This can be explained by the use of a ratio of 1.5 instead of 1.3 for the smallest value of EW/W, a larger error rate in our experiment, and by the details of the task: the location of the next target in our case was predictable, whereas it was not in [9].

Qualitative Results

Participants were asked to rank the techniques by subjective preference in a post-hoc questionnaire. All participants ranked DynaSpot (either 16 or 32) as their preferred technique, followed by the other DynaSpot in second. Only two participants chose another technique than the other DynaSpot as the second best. One ranked the Bubble cursor second, the other DynaSpot third and Point last, while the other participant ranked the Point cursor second, the other DynaSpot third, and Bubble cursor last. Most participants complained about the visual distraction caused by the Bubble cursor envelope's strong variations under certain conditions, leading seven of them to rank that technique last. This is again consistent with earlier results [9, 12]. For instance, one participant said that *"Bubble cursor is distracting when the target is far away because the bubble has a big size"*.

Fitts' Law and Effective Width(s)

Figure 12 plots movement time as a function of IDE, the index of difficulty computed with the target's potential effective width. We take the mean for each combined width, amplitude and technique, fitting 27 points for each technique. Table 3 gives the intercept, the slope and the adjusted r^2 for both IDE and ID, the latter being computed using the actual width of the target. We see that using the effective width yields higher r^2 values and improves the fit. When fitting all the data, we obtain the equation $MT = 85 + 181.IDE$ with an adjusted r^2 of 0.962 (for 108 points). This shows that the potential effective width for DynaSpot provides a "definition" of the width appropriate for applying Fitts' Law.

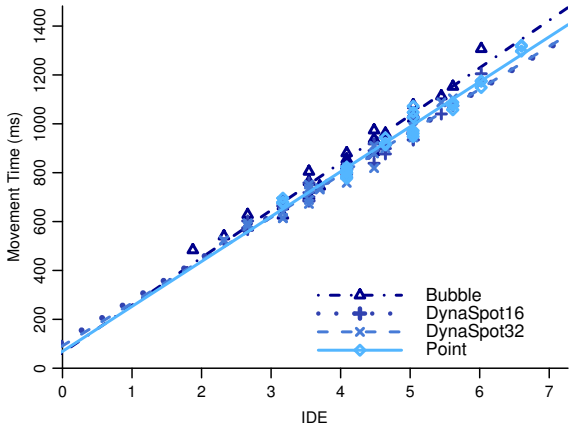

Figure 12. Linear fit: index of difficulty computed with effective width.

Models \ Techniques	Using width W $MT = a + b.ID$			Using effective width EW $MT = a + b.IDE$		
	a	b	Adj. r^2	a	b	Adj. r^2
Bubble	-96	188	0.855	65	194	0.970
DynaSpot16	104	145	0.894	108	171	0.966
DynaSpot32	2	160	0.857	88	176	0.973
Point	69	183	0.969	69	183	0.969

Table 3. Linear fit: intercept, slope and adjusted r^2 using ID or IDE.

Figure 13 shows the position of user clicks relative to the target, and the potential/effective target width for one combined width: (64,48,64,32). We observe that clicks are scattered across a larger area for Bubble cursor than for DynaSpot32, even though both have the same effective width for this combined width. We explain this by the fact that, for DynaSpot32, this corresponds to the case described in Figure 8-b, with the interspace equal to the spot's width, preventing users from fully taking advantage of the effective width. Interestingly, we observe that users do not use the full potential of the Bubble cursor either, as there are very few clicks in the corners of the target's activation area.

In an effort to formalize these observations, we measured the distance to the center of the target for all clicks by combined width, and analyzed the 95% quantile of these distances. As expected, effective widths are reflected in these distances. But other observations can be made. For instance when DynaSpot is at its limit potential effective width (underlined conditions in CW), as for DynaSpot32 in Figure 13, we do find a significant difference between DynaSpot and Bubble cursor, as observed above, but also between DynaSpot32 and DynaSpot16 when the latter is at its limit potential effective width. Another interesting observation is that none of the 95%-quantile distances are larger than the effective width, confirming our initial observation that the corner of the Bubble's activation area are seldom used.

Another type of "effective width", that we call the *a posteriori* effective width and denote W_e, was introduced by Crossmann in his 1956 doctoral dissertation and advocated by MacKenzie and others in the field of HCI [18, 25, 27]. This *a posteriori* effective width comes from the idea of performing an "adjustment for accuracy": the width of the target is corrected so that, under certain hypotheses, the data

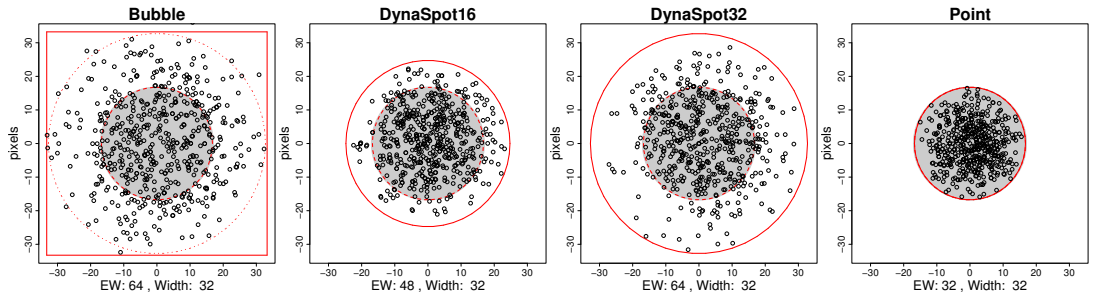

Figure 13. Position of user clicks (after a coordinates change to match a right-to-left horizontal target acquisition) for each technique for combined width CW (64,48,64,32) – Bubble effective width: 64, DynaSpot16 (resp., DynaSpot32) max. effective width: 48 (resp., 64), and target width: 32.

gives raise to an error rate of 4%. A priori, this normalization process leads to more robust results, allowing for better comparisons between experiments. In the following, we check that this definition of effective width can be used to model DynaSpot movement time.

We refer the reader to [25] for details about the computation of W_e. This involves removing outliers, using the time at first button press, computing by participant and full condition, with $W_e = 4.133 \cdot sd$ where sd is the standard deviation of the *oriented distance* from the click to the target's center divided by $\sqrt{2}$. Mean values of W_e for each technique TECH at each combined width CW are given in the table below.

CW	Bubble	DynaSpot16	DynaSpot32	Point
(12,12,12,08)	15.37 (+0.36)	13.89 (+0.21)	13.94 (+0.22)	11.51 (+0.52)
(16,16,16,08)	20.04 (+0.32)	17.12 (+0.10)	17.26 (+0.11)	11.47 (+0.52)
(24,24,24,08)	27.31 (+0.19)	20.38 (-0.24)	26.41 (+0.14)	11.30 (+0.50)
(24,24,24,16)	26.27 (+0.13)	23.53 (-0.03)	25.38 (+0.08)	18.54 (+0.21)
(32,32,32,16)	36.15 (+0.18)	29.85 (-0.10)	32.68 (+0.03)	19.25 (+0.27)
(48,32,48,16)	52.41 (+0.13)	30.74 (-0.06)	43.35 (-0.15)	17.22 (+0.11)
(48,48,48,32)	50.03 (+0.06)	46.31 (-0.05)	48.12 (+0.00)	36.20 (+0.18)
(64,48,64,32)	68.61 (+0.10)	48.51 (+0.02)	57.19 (-0.16)	34.82 (+0.12)
(96,48,64,32)	91.57 (-0.07)	45.80 (-0.07)	59.96 (-0.09)	35.69 (+0.16)

Zhai's index of occupation [27], $I_u = log_2(W_e/EW)$, is given in parentheses. I_u indicates the degree to which the participants over-utilize (positive I_u) or under-utilize (negative I_u) the potential effective target. We observe that the index of occupation is systematically higher for Point cursor, and then for Bubble cursor, and that it globally decreases as the width grows. This can be explained by the error rate (see Figure 11) especially for Point cursor. When comparing Bubble cursor and DynaSpot, which have similar error rates, this confirms that participants better use the full effective target width with Bubble rather than with DynaSpot.

The counterpart of W_e, the *a posteriori* effective amplitude A_e, is computed as the mean of the distance from the movement start point to the point where the user clicks. We can compute the effective index of difficulty:

$$ID_e = log_2(A_e/W_e + 1)$$

Table 4 gives Fitts' law equation parameters for ID_e, and the throughput in $bit \cdot s^{-1}$, computed using either the slope of Fitts' equation, or the formula recommended in [25]. This throughput has the advantage of taking the intercept into account and to be less dependent both on the ID range used [25] and on the users' nominal pointing speed [19]. Thus, it

	$MT = a + b.ID_e$			TP = 1000/b	TP mean of mean
	a	b	adj. r^2	ID_e (IDE)	[25]
Bubble	46	186	0.825	5.38 (5.15)	5.09
DynaSpot16	85	161	0.781	6.21 (5.85)	5.56
DynaSpot32	51	169	0.809	5.92 (5.68)	5.55
Point	49	179	0.756	5.59 (5.46)	5.34
All	57	174	0.792	5.75 (5.52)	5.38

Table 4. Fitts' law equation parameters for ID_e and throughputs.

may be a good index of performance to compare techniques among papers. Note that the r^2 values obtained here do not look as good as with the other method, but in this case we fit 324 points by technique, which are subject to participant performance variability.

DISCUSSION AND FUTURE DIRECTIONS

The results of our experiments are very encouraging. They show that DynaSpot provides an average speed-up of 18% over a conventional point cursor, and that for equivalent effective widths it achieves the same level of performance as the Bubble cursor, one of the more promising techniques to date. DynaSpot is slightly more efficient for low object densities and slightly less efficient for high ones. But most importantly, DynaSpot provides these quantitative performance benefits without departing too much from the conventional point cursor technique. This has at least three significant practical consequences. First, end-users are more likely to adopt the new technique in their daily use of GUIs because DynaSpot behaves "almost like" a point cursor and does not cause a strong visual distraction. Second, DynaSpot is compatible with all point cursor interactions such as region selections initiated by clicking in empty space *without requiring an explicit mode switch*. Finally, implementing DynaSpot does not require significant changes to existing GUI frameworks to support the technique: we implemented support for DynaSpot in the ZVTM Java toolkit[2] in less than 500 lines of code, and a lazy version of the technique[3] was implemented in the Metisse windowing system[4], relying solely on the accessibility API to make the technique work across unmodified applications.

Quantitative and theoretical analyses of performance results show that DynaSpot performance can be modeled with both

[2] `http://zvtm.sf.net`
[3] Which only needs to know the position and shape of interface components at click time, but does not feature target highlighting.
[4] `http://insitu.lri.fr/metisse`

the *a priori* and *a posteriori* effective widths. Somewhat unexpectedly, DynaSpot proves to be on a par with Bubble cursor in most targeting situations. If Bubble cursor's effective width is sufficiently larger than DynaSpot's, then Bubble cursor is faster. However, this happens mostly in configurations where the bubble cursor size is likely to vary dramatically, causing visual distractions that both hinder performance and user acceptance of the technique.

As future work we would like to evaluate area selection. We can predict what should happen with DynaSpot: (i) if empty space between targets is sufficiently large compared to the maximum spot size, the time it takes to initiate a selection in empty space should be similar to the time it takes with a point cursor; (ii) in a dense layout, we expect DynaSpot to be penalized because of the lag+reduction time (300ms). In this particular situation, an explicit mode-switching mechanism might represent an interesting compromise. DynaSpot would then have to be compared, for dense layouts, to Bubble and Area cursors augmented with such an explicit mode-switch, but also to an augmented DynaSpot featuring both time-based (implicit) and explicit mode-switching. We also plan to investigate the use of speed coupling in other pointing techniques, as this seems to be an efficient way of adapting a technique's behavior.

REFERENCES

1. T. Asano, E. Sharlin, Y. Kitamura, K. Takashima, and F. Kishino. Predictive interaction using the delphian desktop. In *Proc. UIST '05*, 133–141. ACM, 2005.
2. R. Balakrishnan. "Beating" Fitts' law: virtual enhancements for pointing facilitation. *Inter. J. of Hum.-Comput. Stu.*, 61(6):857–874, 2004.
3. P. Baudisch, E. Cutrell, D. Robbins, M. Czerwinski, 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 *Proc. Interact '03*, 57–64, 2003.
4. P. Baudisch, A. Zotov, E. Cutrell, and K. Hinckley. Starburst: a target expansion algorithm for non-uniform target distributions. In *Proc. AVI '08*, 129–137. ACM, 2008.
5. R. Blanch, Y. Guiard, and M. Beaudouin-Lafon. Semantic pointing: improving target acquisition with control-display ratio adaptation. In *Proc. CHI '04*, 519–526. ACM, 2004.
6. A. Cockburn and P. Brock. Human on-line response to visual and motor target expansion. In *Proc. GI '06*, 81–87, 2006.
7. A. Cockburn and A. Firth. Improving the acquisition of small targets. In *Proc. British HCI '03*, 181–196, 2003.
8. S. A. Douglas, A. E. Kirkpatrick, and I. S. MacKenzie. Testing pointing device performance and user assessment with the ISO 9241, part 9 standard. In *Proc. CHI '99*, 215–222. ACM, 1999.
9. T. Grossman and R. Balakrishnan. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *Proc. CHI '05*, 281–290. ACM, 2005.
10. Y. Guiard, R. Blanch, and M. Beaudouin-Lafon. Object pointing: a complement to bitmap pointing in GUIs. In *Proc. GI '04*, 9–16, 2004.
11. C. Gutwin. Improving focus targeting in interactive fisheye views. In *Proc. CHI '02*, 267–274. ACM, 2002.
12. M. Hertzum and K. Hornbæk. Input techniques that dynamically change their cursor activation area: A comparison of bubble and cell cursors. *Int. J. Hum.-Comput. Stud.*, 65(10):833–851, 2007.
13. T. Igarashi and K. Hinckley. Speed-dependent automatic zooming for browsing large documents. In *Proc. UIST '00*, 139–148. ACM, 2000.
14. P. Kabbash and W. Buxton. The "prince" technique: Fitts' law and selection using area cursors. In *Proc. CHI '95*, 273–279. ACM/Addison-Wesley, 1995.
15. M. Kobayashi and T. Igarashi. Ninja cursors: using multiple cursors to assist target acquisition on large screens. In *Proc. CHI '07*, 949–958. ACM, 2008.
16. E. Lank, Y.-C. N. Cheng, and J. Ruiz. Endpoint prediction using motion kinematics. In *Proc. CHI '07*, 637–646. ACM, 2007.
17. J. Laukkanen, P. Isokoski, and K.-J. Räihä. The cone and the lazy bubble: two efficient alternatives between the point cursor and the bubble cursor. In *Proc. CHI '08*, 309–312. ACM, 2008.
18. I. S. MacKenzie. Fitts' law as a research and design tool in human-computer interaction. *Hum.-Comput. Interact.*, 7:91–139, 1992.
19. I. S. MacKenzie and P. Isokoski. Fitts' throughput and the speed-accuracy tradeoff. In *Proc. CHI '08*, 1633–1636. ACM, 2008.
20. M. McGuffin and R. Balakrishnan. Acquisition of expanding targets. In *Proc. CHI '02*, 57–64. ACM, 2002.
21. M. J. McGuffin and R. Balakrishnan. Fitts' law and expanding targets: Experimental studies and designs for user interfaces. *ACM Trans. Comput.-Hum. Interact.*, 12(4):388–422, 2005.
22. T. Moscovich and J. F. Hughes. Multi-finger cursor techniques. In *Proc. GI '06*, 1–7, 2006.
23. E. Pietriga and C. Appert. Sigma lenses: focus-context transitions combining space, time and translucence. In *Proc. CHI '08*, 1343–1352. 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 *Proc. CHI '07*, 757–766. ACM, 2007.
25. R. W. Soukoreff and I. S. MacKenzie. Towards a standard for pointing device evaluation: Perspectives on 27 years of Fitts' law research in HCI. *Int. J. Hum.-Comput. Stud.*, 61(6):751–789, 2004.
26. A. Worden, N. Walker, K. Bharat, and S. Hudson. Making computers easier for older adults to use: area cursors and sticky icons. In *Proc. CHI '97*, 266–271. ACM, 1997.
27. S. Zhai, J. Kong, and X. Ren. Speed-accuracy tradeoff in Fitts' law tasks: on the equivalency of actual and nominal pointing precision. *Int. J. Hum.-Comput. Stud.*, 61(6):823–856, 2004.