Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Int. J. Human-Computer Studies 61 (2004) 857–874
International Journal of Human-Computer Studies
www.elsevier.com/locate/ijhcs

"Beating" Fitts' law: virtual enhancements for pointing facilitation

Ravin Balakrishnan*

Department of Computer Science, University of Toronto, 10 King's College Road, Toronto, Ont., Canada M5S 3G4

Available online 31 October 2004

Abstract

We survey recent research into new techniques for artificially facilitating pointing at targets in graphical user interfaces. While pointing in the physical world is governed by Fitts' law and constrained by physical laws, pointing in the virtual world does not necessarily have to abide by the same constraints, opening the possibility for "beating" Fitts' law with the aid of the computer by artificially reducing the target distance, increasing the target width, or both. The survey suggests that while the techniques developed to date are promising, particularly when applied to the selection of single isolated targets, many of them do not scale well to the common situation in graphical user interfaces where multiple targets are located in close proximity.

1. Introduction

Over the past two decades, as graphical user interfaces have largely superceded the command line interface, the act of pointing to various graphical elements such as icons, menus and buttons has emerged as the arguably most fundamental elemental task in human–computer communication. Given the pervasiveness of pointing throughout all applications using graphical interfaces, even a slight improvement in

*Tel.: +416 978 5359; fax: +416 978 5184.
E-mail address: ravin@dgp.toronto.edu (R. Balakrishnan).

1071-5819/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ijhcs.2004.09.002

pointing performance can have substantial impact on overall user productivity. Thus, it is highly worthwhile for interface researchers and designers to attempt to optimize pointing performance to the greatest extent possible.

In contrast to pointing to physical objects in the real world, pointing in the virtual world is typically achieved via an input device that acts as an intermediary between the human and the graphical objects being pointed to. Despite the presence of this intermediary, however, Card et al. (1978) showed in their seminal paper that virtual pointing can be accurately modelled using Fitts' law (Fitts, 1954; MacKenzie, 1992), which asserts that the movement time MT to acquire a target of width W which lies at a distance D is governed by the relationship

$$MT = a + b \log_2\left(\frac{D}{W} + 1\right),$$

where a and b are empirically determined constants, the logarithmic term is called the index of difficulty (ID) measured in "bits", and the reciprocal of b is the human rate of information processing for the task at hand and is often referred to as the index of performance (IP) or bandwidth. Note that the above Shannon formulation of Fitts' law is the widely preferred alternative amongst several from both theoretical and practical perspectives (see MacKenzie, 1992, for a discussion on these alternative formulations).

In addition to demonstrating the applicability of Fitts' law to modelling virtual pointing, Card et al. (1978) and other researchers (e.g. MacKenzie, 1992; Douglas and Mithal, 1997) have also clearly shown that virtual pointing using input devices like the mouse or stylus can result in performance very similar to intermediary-free physical pointing. Based on this well replicated finding, one might argue that pointing in the virtual world is "as good as it can possibly get". However, in recent years HCI researchers have realized that since virtual pointing does not have to be constrained by the laws of the physical world, it may be possible to actually "beat" Fitts' law and make virtual pointing *easier* than its physical counterpart. Assuming that the input device used is optimal in that it enables virtual pointing performance equivalent to physical pointing, Fitts' law indicates two possible approaches for further optimization: reduce D or increase W. Directly changing these two parameters obviously does nothing more than change the size and position of onscreen graphical elements, which are presumably already laid out in a reasonably optimal fashion due in part to the interface designer's basic appreciation of Fitts' law. The challenge is to indirectly affect further changes in D and/or W in ways that do not substantially alter the overall visual appearance of the graphical interface, but nonetheless result in shorter pointing times.

In this paper, we survey the recent research on attempts at creating virtual enhancements to improve pointing performance. We begin with some background on the current understanding of the underlying human motor actions that are believed to be modelled by Fitts' law. In light of this foundational knowledge, we then discuss the various pointing facilitation techniques that have been developed to

date, roughly grouping them in three categories: those that (1) primarily attempt to decrease D, (2) primarily attempt to increase W, and (3) both decrease D and increase W. The situations where these new techniques succeed in improving performance are discussed, but perhaps more importantly we focus on the situations where they fail to provide any gains or worse still result in poorer performance than ordinary pointing thus suggesting where more work is needed to refine the relevant techniques. In addition to providing the reader with a concise overview of the research in this fascinating area, we hope that this paper will help spur further research in several promising directions which we identify as a result of our integrative survey.

We note that this survey focuses on the virtual interaction techniques that have been developed to facilitate pointing, and not on new or improved input technologies such as haptic feedback devices that may also be beneficial to pointing performance. It is also important to acknowledge that the choice of input device can profoundly impact the performance of a particular technique (Jacob et al., 1994; Zhai, 1995), but a detailed treatment of this issue is beyond the scope of this paper.

2. Background

In order to gain insight into the directions that can be taken in designing virtual techniques to facilitate pointing, it is useful to understand the possible underlying motor control models that are likely explanations for Fitts' law.

One explanation, called the *iterative corrections model* (Crossman and Goodeve, 1963/1983; Keele, 1968), attributes the law entirely to closed-loop feedback control. This model states that the whole movement consists of a series of discrete sub-movements, each of which takes the user closer to the target and is triggered by feedback indicating the target is not yet attained.

Another explanation, called the *impulse variability model* (Schmidt et al., 1979), attributes the law almost entirely to an initial impulse delivered by the muscles, flinging the limb towards the target. The last part of the movement consists of the limb merely coasting towards the target.

It has been pointed out (Wing, 1983; Rosenbaum, 1991), however, that neither of these two explanations adequately accounts for all the effects shown in the large body of experimental data in the literature.

The most successful and complete explanation to date (Rosenbaum, 1991), called the *optimized initial impulse* model (Meyer et al., 1988), is a hybrid of the *iterative corrections model* and the *impulse variability model.* This suggests that the process modelled by Fitts' law (Fig. 1) is as follows:

An initial movement is made towards the target. If this movement hits the target, then the task is complete. If, however, it lands outside the target, another movement is necessary. This process continues until the target is reached. Since the goal is to reach the target as quickly as possible, in an ideal case the subject should make a single high-velocity movement towards the target. In reality, however, the spatial accuracy of such movements can be quite poor. It can be shown (Meyer et al., 1988;

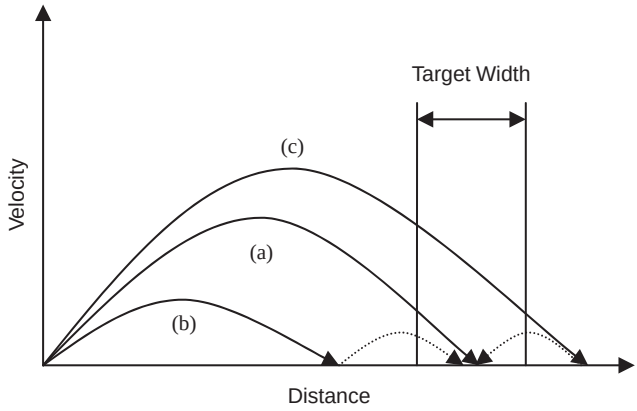

Fig. 1. Possible sequence(s) of submovements toward a target as described by the *optimized initial impulse* model (Meyer et al., 1988). (a) Is the case where a single movement reaches the target. (b) and (c) are the more likely cases where the initial movement under or over shoots the target, requiring subsequent corrective movements.

Rosenbaum, 1991) that the standard deviation (S) of the endpoint of any movement increases with the distance (D) covered by that movement, and decreases with its duration (T):

$$S = k\left(\frac{D}{T}\right),$$

where k is a constant. Thus, a movement with a long distance and short duration could be executed, but would result in a high standard deviation and therefore a low probability of actually hitting the target. Conversely, a series of long duration and short distance movements could be executed, hitting the target with certainty, but the total movement time would be extremely long. The solution, therefore, is to find the optimal balance of D's and T's that minimizes the total movement time (Rosenbaum, 1991). In essence, this means that most aimed movements consist of an initial large and fast movement that gets the subject reasonably close to the target, followed by one or more shorter, and slower, corrective movements that are under closed-loop feedback control.

Based on this explanation, we can hypothesize that virtual enhancements for improving pointing performance that attempt to decrease D should concentrate on the initial large and fast movement phase that covers the bulk of the distance towards the target. Conversely, techniques that attempt to decrease W would likely be able to reap almost all their benefit if they focused on the final corrective movement phase, since although W may play a part in the planning and execution of the initial large and fast movement, its effect is most apparent when the user is homing in on the target under closed-loop feedback control.

In Fitts' original work and the initial follow-up experiments in the motor control literature, there was typically a one-to-one correspondence between the human's

visual and motor spaces in that physical targets were selected by direct indication using the human hand. Pointing in the virtual realm of computers, however, typically involve an intermediary device (e.g. mouse, joystick, touchpad) that converts human motor actions into movements of a virtual cursor. There are thus three major factors that come into play and can affect performance (Graham and MacKenzie, 1996) in virtual pointing: motor or control space, visual or display space, and the control–display (C–D) transfer function that links the two spaces. Changes in D and/or W could occur in either the motor or visual spaces, or both. In the rest of this paper, we use the terms D and W to denote distance and width in both spaces, D_v, W_v and D_m, W_m to denote distance and width in the visual (v) and motor (m) spaces, respectively, when a distinction between the two is necessary.

3. Facilitating pointing by primarily reducing *D*

3.1. Designing widgets that minimize D

A somewhat trivial optimization is to simply move the targets close to the cursor where feasible. One instantiation of this idea are the contextual linear pop-up menus seen in many applications where the menus items are displayed right by the cursor when the menu is activated. Whereas the linear layout of these menus put some items further away from the cursor than others, pie-menus (Callahan et al., 1988) additionally improve the situation by arranging all items in a circle around the cursor thus making all items equidistant with a very small and constant D (Fig. 2). Although pop-up linear and pie menus are demonstrably effective (Callahan et al., 1988), they are only one of the many types of targets that are typically selectable in

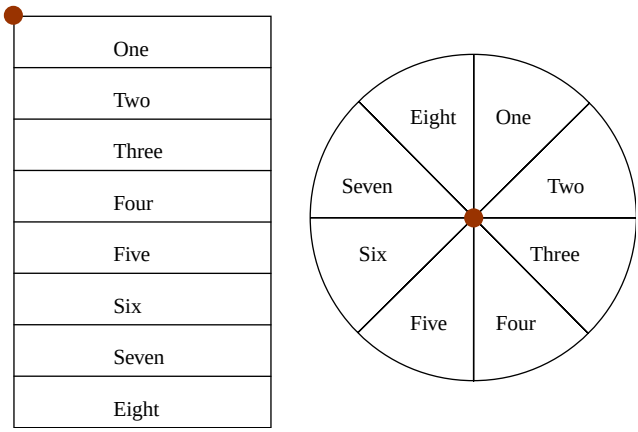

Fig. 2. Linear vs. pie menus. Distance of menu items from red starting point varies in linear menus (left), but is constant in pie menus (right).

graphical interfaces. Other common interface elements such as icons, buttons, and scroll-bars are not easily transformed to reduce their distance to the cursor, without drastic redesign of the overall interface layout. Note, however, that the use of hot-keys or dedicated scrolling wheels or joysticks on mice are able to effectively reduce D_m and/or increase W_m without altering D_v or W_v.

3.2. Temporarily bringing potential targets to the cursor

One recent interesting attempt to directly reduce D for icons without redesigning their basic behaviour and layout is the drag-and-pop technique developed by Baudisch et al. (2003). In this technique, the system responds to directional cursor movements by temporarily bringing a virtual proxy of the most likely potential set of targets towards the cursor (Fig. 3). The overall interface is unchanged, but the user can now manipulate the proxy icons at a much closer distance. In a user study comparing drag-and-pop to direct drag-and-drop on large displays, Baudisch et al. (2003) found drag-and-pop to be up to 3.7 times faster for very large D, albeit at a slightly higher error-rate.

Although the drag-and-pop technique was designed for use with icons, one could imagine extending this idea to operate other interface elements by bringing them closer to the cursor when required. The challenge would be to design the interaction such that the proxies do not overly clutter the region near the cursor or obscure valuable information around it. Furthermore, it can be tricky to implicitly determine when the user intends to select the remote elements versus items that are already in the nearby vicinity. If the proxies are activated when not required, they will in the best case simply annoy the user, and in the worst case actually interfere with the

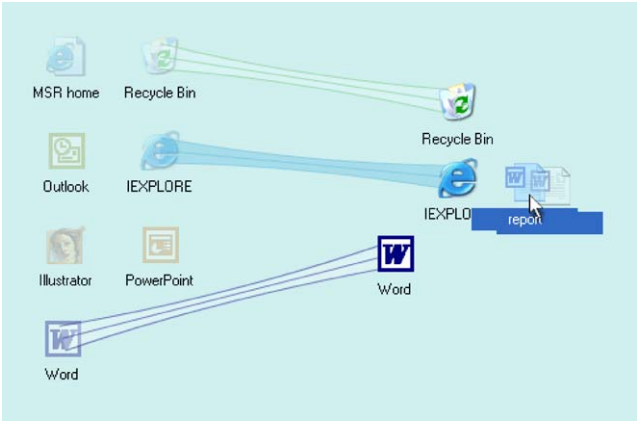

Fig. 3. Drag-and-pop (Baudisch et al., 2003). Virtual proxies of icons on the far left of the screen are brought closer to the cursor to facilitate quick pointing by reducing D. The relationship between proxy and actual icon is indicated by the stretched lines. The proxies only exist for the duration of the cursor drag action, thus not affecting the overall interface. (Picture taken from Baudisch et al., 2003).

user's ability to select an existing nearby target. Indeed, this technique like many others we will encounter in this survey, tends to work best on a fairly sparse virtual desktop. In dense desktops with many closely packed potentially selectable items, the problems with occlusion and false activation can obviate the potential benefits.

One potential solution to the false activation problem is to use an explicit trigger, thus trading-off seamlessness in interaction for repeatability and accuracy. Another solution that addresses both the false activation and occlusion problems may be to design an on-screen widget with parameters that users can manipulate to specify which interface items are brought closer to the cursor and how they are to appear (e.g. using transparency to mitigate interference with existing items (Harrison et al., 1995), or perhaps scaling the proxies to fit in blank space around the existing items near the cursor). This approach has the benefit of providing the user with significant additional control, at the expense of perhaps more complex and less transient interaction.

3.3. Removing empty space between the cursor and targets

Most efforts at improving pointing performance to date have considered the traditional situation of a 2D cursor pointing at targets represented as pixels, where all pixels of the display contribute equally to the overall space of possible selectable objects. However, Guiard et al. (2004) recently noted that in most real graphical user interfaces, there are a significant number of pixels that serve no useful function other than providing a pleasing interface layout. For example, if there were 50 selectable interface elements averaging $20 \times 20 = 400$ pixels each on a 1600×1200 pixel display, only $50 \times 400 = 20000$ pixels would be selectable out of a total of $1600 \times 1200 = 192000$ pixels. Thus 98.9% of the available pixels provide no useful information to the task of selecting the 20,000 pixels of interest.[1] Worse, they actually hinder that task by increasing the distance between the selectable targets. Based on this simple yet important observation, Guiard et al. (2004) proposed that since we are typically only interested at pointing to selectable targets, these useless pixels could be removed from consideration within the pointing task. Thus, the overall space is collapsed such that all targets are effectively next to one another, reducing D very significantly.

Obviously, it would be quite undesirable to actually remove all that empty space and move the targets together, since it would utterly ruin the layout and usability of the graphical interface. Instead, Guiard et al. (2004) designed an interaction technique, called *object pointing*, where the cursor essentially skips across the empty space, jumping from one selectable target to another. Thus, D_m is reduced while keeping D_v unchanged. In practical terms, when the cursor leaves a selectable object and its velocity exceeds a threshold, it jumps to the next available target in the

[1] In general, Guiard et al. (Guiard et al., 2004) show that the wasted information is $\log_2(S_s/S_0) - \log_2 N$, where S_s is the total surface area of the display, S_0 is the surface area of all selectable objects, and N is the number of selectable objects.

direction of the cursor's current movement. In a controlled experiment, they showed that object pointing was on average 74% faster than regular pointing for a reciprocal pointing task—a very significant improvement in the base case. A second experiment, where the task more closely resembled pointing in real user interfaces which have varying target densities, showed that the degree to which object pointing outperformed regular pointing depended upon the target density. As might be expected, when the display was sparsely populated and distances between targets large, the benefits of object pointing was very significant. Conversely, in very dense displays where targets were close together to begin with, regular pointing was preferable.

Of all the techniques surveyed in this paper, object pointing is arguably the one with the most significant performance gains, at least as demonstrated by the controlled laboratory experiments. However, there are a few practical considerations that may lower its overall applicability and value, and thus hinder widespread adoption. First, as Guiard et al. (2004) themselves note, object pointing is not *always* applicable in that users may sometimes want to select and manipulate any of the individual pixels, or one of many small groups of pixels (e.g. a text character in a word processor), on a display. Also, what can seem like "blank" space in an interface is sometimes used as implicit targets to facilitate interaction. For example, rubber-band selection requires pointing and clicking on a blank part of the interface, and then moving the pointer such that the bounding box encompasses the desired set of targets. Thus, it would be necessary to provide both object and regular pointing in an interface, with some mechanism to switch between them, resulting in a slightly less than seamless interaction style. Second, again identified in Guiard et al. (2004), is the fact that in many graphical interfaces, the selectable objects are often tiled together (e.g. menus, toolbars, rows of icons). Since such tilings have no empty space between targets, object pointing clearly has no benefit, although it is not detrimental in that it simply regresses to regular pointing. Third, on the basis of evidence that eye gaze precedes hand movements (Abrams et al., 1989; Jacob, 1991), Guiard et al. argue that "there should be no difficulty for the user to follow the jumping motion of the highlight across the layout of objects" (Guiard et al., 2004). Their experimental results clearly demonstrate that this jumping motion is not an impediment to raw quantitatively measurable performance, at least within the strictures of performing a controlled experiment. However, in the context of real usage in everyday computing environments where subjective impressions of an interface can play a much larger role than in controlled experiments, the visual discontinuities inherent in such jumping motions may prove to be overly annoying to the user and thus be a barrier to acceptance. It would be interesting to explore this issue in a field study of object pointing in real use.

It is worth noting that a more rudimentary discrete form of object pointing already exists in many interfaces where the cursor keys can be used to discretely move between targets in any of the four main directions. Similarly, some interfaces use the tab key to cyclically select between targets in some predefined ordering. Both these techniques focus only on selectable targets, and bypass the less interesting blank space surrounding those targets.

4. Facilitating pointing by primarily increasing W

4.1. Area cursors

An interesting twist to the pointing facilitation problem is suggested by the work of Kabbash and Buxton (1995) who investigated the use of area cursors that have an active area or "hot spot" that is larger than the single pixel of standard cursors (Fig. 4). Kabbash and Buxton showed that selection using area cursors could be accurately modelled using Fitts' law, with W being the width of the cursor rather than the width of the target (assuming that the target width is smaller than cursor width). Thus, very small targets which would traditionally have a high index of difficulty when selected by a point cursor would have a much lower index of difficulty when selected by an area cursor. Zhai et al. (1994) also showed that 3D volume cursors could improve performance when selecting objects in 3D space, although their focus was on the effects of translucency in the cursor, rather than a Fitts' law analysis per se.

While the basic Fitts' law analysis of Kabbash and Buxton is sound and indicates that area cursors can be a promising way to improve pointing performance, Worden et al. (1997) point to two significant problems with area cursors: large area cursors can obscure underlying data, and it can be difficult if not impossible to use area cursors to select one target from several targets closely grouped together. The first problem is largely mitigated if the area cursors are rendered semi-transparent as in Zhai et al. (1994). The second problem, however, requires more creative handling. Worden et al. (1997) propose an enhanced area cursor that has two hot spots: the area encompassed by the whole area cursor, and a second single point hot spot within the area cursor. When targets are far apart, the cursor behaves like the default area cursor. However, when more than one target is within the area cursor, the point hot spot is used to discriminate between those targets. In a controlled experiment, they showed that this enhanced area cursor performed identically to regular point cursors when targets were close together, and outperformed point cursors when targets were far apart thus reaffirming the results of Kabbash and Buxton (1995).

Given the demonstrated benefits of area cursors, it would be interesting to explore further enhancements that would make them work in a facile manner in real interfaces. For example, one could imagine an area cursor that morphs into a point

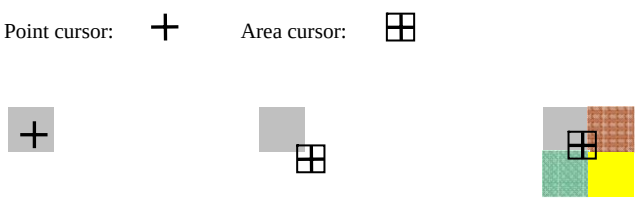

Fig. 4. Point vs. area cursors. (left) Selection with a point cursor is achieved when centre of the cross-hair is within the desired target. (middle) Selection with an area cursor is achieved when any part of the area cursor intersects the desired target. (right) When the area cursor intersects multiple targets, the target under the cross-hair is selected as in Worden et al. (1997).

cursor when over clusters of targets such as toolbars and menus, thus removing any of the occlusion effects inherent even when transparency is used to render the area cursor.

4.2. Expanding targets

Recently, several interaction techniques (Furnas, 1986; Mackinlay et al., 1991; Bederson, 2000) have been developed where the size of the interface widget or viewing region dynamically changes to provide the user with a larger target area to interact with at their focus of attention. Dynamically sized widgets are also found in the Apple MacOSX "dock" where the icons in the desktop toolbar expand when the cursor is over them. These expanding widgets that are displayed at significantly reduced size by default and dynamically expanded to a usable size only when required could be an effective strategy for maximizing the use of screen real estate without sacrificing target selection performance.

McGuffin and Balakrishnan (2002) investigated the potential performance benefits of such expanding targets. Based on the optimized initial impulse model of movement (Meyer et al., 1988) discussed earlier, they argued that in the situation where the target's width expands at some point during the movement, it can be expected that the first large and fast movement towards the target is planned and executed with the initial, unexpanded, target width as the input parameter to the user's motor control system. However, subsequent corrective sub-movements should, according to this model, be able to respond to changes in the target's size since these sub-movements are under closed-loop feedback control. Thus, based on this explanation of Fitts' law, they hypothesized that in most cases target acquisition time would depend largely on the final target size and not the initial one at the onset of movement. They verified this hypothesis in a controlled experiment, and further showed that users were able to take advantage of the larger expanded target width even when expansion occurred after 90% of the distance to the target was already traversed. They also showed that the overall performance could be accurately modelled by Fitts' law using the expanded target width as the governing size parameter. Their study, however, focused only on the case where there was a single isolated target to be selected. When multiple targets are present in close proximity, expanding only one target can affect its neighbours either by occluding them or pushing them out of the way. Either approach is likely to lead to undesirable side effects. In a design exercise, they explored several alternatives approaches for addressing the tiled multiple target scenario, with limited success (Fig. 5).

McGuffin (2002) mathematically analysed the multiple target situation and showed that performance gains would only be possible if one could reasonably predict the trajectory of the cursor such that only the few targets in the general vicinity where the cursor is headed are expanded, leaving other widgets unchanged. Unfortunately, to date, accurate cursor prediction is difficult to achieve in practice.

Zhai et al. (2003) subsequently pointed out that in McGuffin and Balakrishnan's (2002) study, users performed a large block of trials during which target expansion *always* occurred and as such any observed performance gains could be due to users

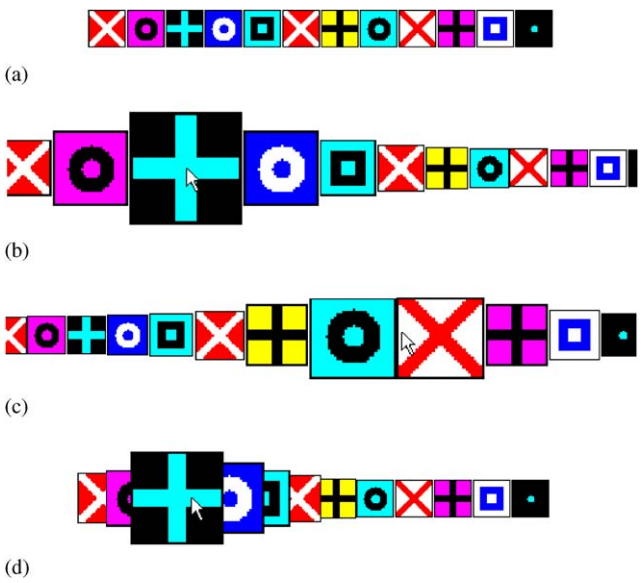

Fig. 5. Expanding target designs (McGuffin and Balakrishnan, 2002). (a) The buttons are un-expanded when the pointer is far away. (b) Expanding target design that fully expands targets underneath the cursor, while partially expanding and pushing neighbouring targets sideways. (c) A user starting in the state shown in (b) may try to move to the right to select the button with the light X on the dark background. By the time the cursor reaches the desired button's location, the button has moved to the left and the user is now over a different button (one with a *dark* X on a *light* background). Such horizontal movement of targets in tiled layouts can negate the benefits of target expansion. (d) Expanding target design that differs from that in (b, c) by allowing for some overlap between adjacent buttons thus alleviating the problems caused by sideways motion. In this case, however, benefits due to expansion are only realized when the cursor enters the target from outside the tiled targets. Sideways movement of the cursor result in target expansion in visual space, but not in motor space.

anticipating the expansion and thus a priori planning their entire movement sequence to account for the larger W, instead of actually responding to *changes* in W during the movement. They replicated the experiment with an additional condition where expansion occurred in a pseudo-random fashion—thus users were unable to anticipate the expansion ahead of time. Their results showed that even when users could not anticipate the expansion, similar performance gains were realized. Zhai et al. (2003) also attempted to design a suitable tiled multi-target expanding widget, but as with (McGuffin, 2002) they had to rely on cursor prediction to achieve satisfactory results.

An interesting counterexample to the benefits of expanding targets is indicated by Gutwin's (2002) study of target selection in a fisheye view. Here, as with many of the tiled multi-target designs explored by McGuffin (2002) and Zhai et al. (2003), not only did the targets expand in size, but they also shifted in x–y space. Gutwin (2002) showed that the distortion and movement of these fish-eye views actually made it *more* difficult to select targets, even though their size had increased. He also showed

that by "flattening" the distortion based on cursor velocity and acceleration, performance was significantly improved. He did not, however, analyse his results in terms of Fitts' law, thus making generalizable comparisons with the other work on expanding targets (McGuffin, 2002; Zhai et al., 2003) difficult. This study highlights the importance of careful interaction design, and the importance of not relying solely on the results of controlled experiments that look at only one aspect of a new pointing technique's performance (such as the studies in McGuffin, 2002; McGuffin and Balakrishnan, 2002; Zhai et al., 2003). It also indicates that the dynamic characteristics of the pointer, and not just its position, should be factored into the equation when designing interaction techniques that attempt to "beat" Fitts' law.

5. Facilitating pointing by both decreasing *D* and increasing *W*

5.1. Dynamically changing the control–display gain

The ratio of the amount of movement of an input device and the controlled objects (i.e. typically a cursor) is refereed to as the C–D gain. In early graphical interfaces and in even earlier control systems, this gain was typically a constant. The effect of gain on performance has been quite extensively explored in the literature (Buck, 1980; Arnaut and Greenstein, 1990; Jellinek and Card, 1990; MacKenzie and Riddersma, 1994) but the results are still somewhat inconclusive. It can be argued that very high gains result in poor performance due to the difficulty in making the precise movements required at the final phases of target acquisition, although the initial coarse-grained movement can be achieved more rapidly. Conversely, very low gains improve precise positioning at the expense of poorer initial coarse-grained movement time. Some researchers (Gibbs, 1962; Zhai, 1995) have argued and provided experimental evidence that this trade-off typically results in a U-shaped gain-performance curve, with optimal performance achieved at moderate gain levels. Others (Hammerton and Tickner, 1966; Jellinek and Card, 1990), however, have argued differently. In a classic study that investigated if and how power mice that had variable control gain affected performance, Jellinek and Card (1990) found that performance decreased as gain was increased. Nevertheless, they argued that this was due to limitations in resolution of the mouse sensing technology, and not to inherent human performance limitations. They further suggested that if gain affected performance, this would violate Fitts' law—a difficult argument to accept given that Fitts' law does not address variables other than distance and width. In more recent work, MacKenzie and Riddersma (1994) showed that in a Fitts' selection task medium gain resulted in highest performance times but also the highest error rates, thus casting doubt on the existence of an optimal gain level (MacKenzie, 1995).

Despite the inconclusiveness of the research on the effects of gain, in practical terms interface designers have found that efficiencies could often be obtained by dynamically varying the C-D gain based on input device speed. Often referred to as "mouse acceleration", this technique was based on the assumption that when a user

moved the device fast they intended to cover a greater distance. Thus, simply decreasing the C-D gain as device movement speed increased would allow greater virtual distances to be traversed with smaller device movements.

This basic adaptive C-D gain technique, however, is based entirely on device movement speed and does not take into consideration the location and size of potentially selectable targets. More recently, several researchers (Keyson, 1997; Worden et al., 1997; Cockburn and Firth, 2003; Blanch et al., 2004) have investigated dynamically adjusting the C-D gain based on knowledge about the targets: increasing gain when the cursor is outside targets and decreasing it when inside. The resulting "sticky" targets make it easy to get into known targets and harder to leave them. In a controlled experiment, Keyson (1997) demonstrated that using a target aware adaptive C-D gain resulted in significantly decreased target acquisition times. Worden et al. (1997) demonstrated a similar result and interestingly showed that older users benefited more from the adaptive technique than younger users. Cockburn and Firth (2003) showed in particular that small targets were easier to select with a target aware adaptive C-D gain. Most recently, Blanch et al. (2004) systematically analysed target aware C-D gain adaptation in terms of Fitts' law and showed that performance could be accurately predicted based on the resulting larger W_m and smaller D_m in motor space, rather than the size and location of the invariant visual target. They also present some nice redesigns of standard graphical widgets (Fig. 6) to take advantage of what they refer to as "semantic pointing".

While target aware C-D gain adaptation undoubtedly improves pointing performance for single isolated targets, problems arise when multiple targets are present in that intervening targets act as traps that can hinder movement along the way to the desired target. Drawing upon the *optimized initial impulse* model (Meyer et al., 1988) of movement discussed earlier, Worden et al. (1997) theorized that since high-velocity movements are indicative of the initial impulse, the C-D gain should remain high during high-velocity movements even when over potential targets. Only when the cursor velocity decreased, indicating corrective sub-movements towards a desired target, did they reduce the C-D gain. Thus, intervening targets could be skipped over as long as the user moved fairly quickly over them. In their experiment, they included a single intervening target between the start position of the cursor and the final desired target, and their results indicate that performance was generally unaffected by this distracter target. Cockburn and Firth (2003) attempt to mitigate the problem of intervening targets by implementing C-D gain adaptation only along one axis. Further, for the small targets that they were concerned with, they incidentally observed that when "the cursor passes rapidly over an interface component, the motion is often too rapid for the window system's event model to trigger an event on the underlying widget" (Cockburn and Firth, 2003), resulting in a quick and dirty solution to the problem. Blanch et al. (2004) recognize this problem, but did not suggest solutions in their paper.

We note that while the solutions employed by Worden et al. (1997) and Cockburn and Firth (2003) appear reasonable in the situation where there are a few intervening distracter targets, most real user interfaces would likely have many such intervening

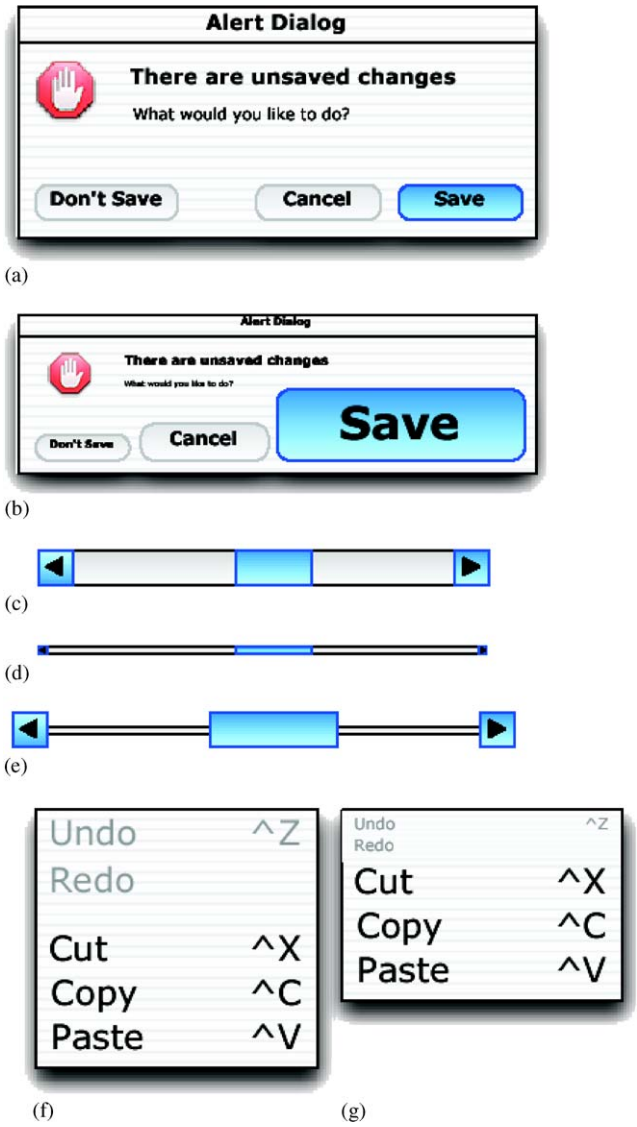

Fig. 6. Semantic pointing widget designs from Blanch et al. (2004). (a) Visual space appearance of buttons in a dialogue box. (b) Motor space version of button design in (a) with much larger targets for certain buttons. (c) Standard scroll-bar design. (d) Visual space appearance of scroll-bar redesigned to occupy smaller screen space. (e) Motor space version of scroll-bar design in (d) with larger targets for active areas. (f) Visual space appearance of menu. (g). Motor space version of the menu design in (f) with the distance D_m to more important items reduced by compressing the size W_m of less important items.

targets. Those intervening targets that are closer to the desired target would be difficult to skip over using the suggested solutions since the user would have slowed down, iteratively performing corrective sub-movements while homing in on the

desired target, and in the process encountering interference from the nearby distracters.

6. Conclusions

Our survey has shown considerably research activity in developing, analysing and evaluating new techniques for artificially enhancing pointing performance in graphical user interfaces. Unfortunately, while some of the techniques, such as object pointing (Guiard et al., 2004), are particularly promising none of them has yet to be demonstrated to work uniformly well in all situations encountered in typical graphical user interfaces.

An interesting problem that appears in many of the techniques surveyed is that while the techniques tend to produce significantly improved pointing performance when selecting isolated targets, difficulties arise when they are used for selecting one of multiple targets that are spatially close together. In particular, those techniques that attempt to facilitate pointing by increasing W do not scale well to selection from multiple targets. Attempts to effectively utilize target expansion in groups of multiple tiled targets (McGuffin, 2002; Zhai et al., 2003) have only been marginally successful, and analysis of the problem (McGuffin, 2002) indicates that this may be a fundamentally intractable problem unless cursor trajectory can be accurately predicted. Techniques that rely on C-D gain adaptation also tend to be less effective when intervening targets lie in the cursor's path towards the desired target. The few solutions explored to date (Worden et al., 1997; Cockburn and Firth, 2003) can alleviate but do not entirely solve the problem. It is interesting to note that this problem with C-D gain adaptation is analogous to the problem faced by designers of haptic-feedback interfaces that physically "attract" or "repel" the cursor towards and away from interface elements (Oakley et al., 2001; Oakley et al., 2002). Thus, as advances are made towards solving this problem in the haptic-feedback community, it may be possible to apply the approaches used there towards the general virtual pointing domain.

As alluded to in the introduction, the choice of input device can significantly alter the effectiveness of a particular pointing enhancement technique. In particular, the techniques we have surveyed that rely on altering the C-D mapping will typically not work well with input devices such as pens/styli which essentially merge the motor and visual spaces. Given the increased use of PDAs and tablet computers, it may be worth asking if further efforts in developing pointing enhancement techniques that rely on C-D gain adaptation are even worthwhile.

Apart from a few notable exceptions (Worden et al., 1997; Cockburn and Firth, 2003), most of the research in this area have tended to follow the classic route of developing a new technique and comparing it with the status quo un-enhanced pointing technique. Thus, from surveying the literature alone, while we can ascertain how the new techniques perform relative to the default, it is somewhat difficult to directly compare the various new techniques relative to one another. Although many of the papers surveyed do analyse their results in terms of Fitts' law which thus

provides an analytical base for comparison, the subtle differences between the various experiments tend to make such cross experiment comparisons unreliable. The work by Cockburn and Firth (2003) and Worden et al. (1997), are the exceptions that do provide us with some sense of how expanding targets, area cursors, and C-D gain adaptation compare. In general, however, it would be worthwhile to conduct further studies comparing these various techniques to one another directly. Another interesting avenue for further research is in exploring combinations of the various techniques. For example, combining expanding targets or area cursors with C-D gain adaptation could result in interesting new techniques for pointing facilitation.

Finally, it is worth noting that in almost all cases the techniques surveyed have been evaluated in terms of quantitative performance measures such as selection times and error rates. While these are valuable measures, a perhaps more critical measure is the final acceptability of a technique by end-users. While this measure is often difficult to quantify directly, indirect observation can often be useful. For example, Blanch et al. (2004) observed that their users did not even seem to notice the distortions in motor space of the semantic pointing enhanced widgets, yet performed better with semantic pointing. In a sense, perhaps the best measure of all is whether or not a new pointing facilitation technique even elicits a comment from users.

References

Abrams, R., Meyer, D., Kornblum, S., 1989. Speed and accuracy of saccadic eye movements: characteristics of impulse variability in the oculomotor system. Journal of Experimental Psychology: Human Perception and Performance 15, 529–543.

Arnaut, L., Greenstein, J., 1990. Is display/control gain a useful metric for optimizing and interface. Human Factors 32 (6), 651–663.

Baudisch, P., Cutrell, E., Robbins, D., Czerwinski, M., Tandler, P., Bederson, B., Zierlinger, A., 2003. Drag-and-pop and drag-and-pick: techniques for accessing remote screen content on touch- and pen-operated systems. Proceedings of Interact, pp. 57–64.

Bederson, B., 2000. Fisheye menus. Proceedings of the ACM UIST Symposium on User Interface Software and Technology, pp. 217–225.

Blanch, R., Guiard, Y., Beaudouin-Lafon, M., 2004. Semantic pointing: improving target acquisition with control–display ratio adaptation. Proceedings of the ACM CHI Conference on Human Factors in Computing Systems, pp. 519–525.

Buck, L., 1980. Motor performance in relation to control–display gain and target width. Ergonomics 23 (6), 578–589.

Callahan, J., Hopkins, D., Weiser, M., Shneiderman, B., 1988. An empirical comparison of pie vs. linear menus. Proceedings of the ACM CHI Conference on Human Factors in Computing Systems, pp. 95-100.

Card, S., English, W., Burr, B.J., 1978. Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT. Ergonomics 21, 601–613.

Cockburn, A., Firth, A., 2003. Improving the acquisition of small targets. Proceedings of the British HCI Conference, pp. 181–196.

Crossman, E., Goodeve, P., 1963/1983. Feedback control of hand-movement and Fitts' law. Quarterly Journal of Experimental Psychology 35A, 251–278.

Douglas, S., Mithal, A.K., 1997. The Ergonomics of Computer Pointing Devices. Springer, Berlin.

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

Furnas, G., 1986. Generalized fisheye views. Proceedings of the ACM CHI Conference on Human Factors in Computing Systems, pp. 16–23.

Gibbs, C.B., 1962. Controller design: interactions of controlling limbs, time-lags, and gains in positional and velocity systems. Ergonomics 5, 385–402.

Graham, E., MacKenzie, C., 1996. Physical versus virtual pointing. Proceedings of the ACM CHI Conference on Human Factors in Computing Systems, pp. 292–299.

Guiard, Y., Blanch, R., Beaudouin-Lafon, M., 2004. Object pointing: a complement to bitmap pointing in GUIs. Proceedings of Graphics Interface, pp. 9–16.

Gutwin, C., 2002. Improving focus targeting in interactive fisheye views. Proceedings of the ACM CHI Conference on Human Factors in Computing Systems, pp. 267–274.

Hammerton, M., Tickner, A.H., 1966. An investigation into the comparative suitability of forearm, hand, and thumb controls in acquisition tasks. Ergonomics 9, 125–130.

Harrison, B., Kurtenbach, G., Vicente, K., 1995. An experimental evaluation of transparent user interface tools and information content. Proceedings of the ACM UIST Symposium on User Interface Software and Technology, pp. 81–90.

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

Jacob, R., Sibert, L., McFarlane, D., Mullen, M., 1994. Integrality and separability of input devices. ACM Transactions on Computer–Human Interaction 1 (1), 3–26.

Jellinek, H., Card, S., 1990. Powermice and user performance. Proceedings of the ACM CHI Conference on Human Factors in Computing systems, pp. 213–220

Kabbash, P., Buxton, W., 1995. The "Prince" technique: Fitts' law and selection using area cursors. Proceedings of the ACM CHI Conference on Human Factors in Computing Systems, pp. 273–279.

Keele, S.W., 1968. Movement control in skilled motor performance. Psychological Bulletin 70, 387–403.

Keyson, D., 1997. Dynamic cursor gain and tactual feedback in the capture of cursor movements. Ergonomics 12, 1287–1298.

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

MacKenzie, S., 1995. Input devices and interaction techniques for advanced computing. In: Barfield, W., Furness, T. (Eds.), Virtual environments and advanced interface design. Oxford University Press, Oxford, pp. 437–470.

MacKenzie, S., Riddersma, S., 1994. Effects of output display and control–display gain on human performance in interactive systems. Behaviour and Information Technology 1 (3), 328–337.

Mackinlay, J., Robertson, G., Card, S., 1991. The perspective wall: Detail and context smoothly integrated. Proceedings of the ACM CHI Conference on Human Factors in Computing Systems. pp. 56–63.

McGuffin, M., 2002. Fitts' law and expanding targets: an experimental study and applications to user interface design. M.Sc. Thesis, Department of Computer Science, University of Toronto.

McGuffin, M., Balakrishnan, R., 2002. Acquisition of expanding targets. Proceedings of the ACM CHI Conference on Human Factors in Computing Systems, pp. 57–64.

Meyer, D.E., Smith, J.E., Kornblum, S., Abrams, R.A., Wright, C.E., 1988. Optimality in human motor performance: ideal control of rapid aimed movements. Psychological Review 95, 340–370.

Oakley, I., Brewster, S., Gray, P., 2001. Solving multi-target haptic problems in menu interaction. Extended Abstracts of the ACM CHI Conference on Human Factors in Computing Systems, pp. 357–358.

Oakley, I., Adams, A., Brewster, S., Gray, P., 2002. Guidelines for the design of haptic widgets. Proceedings of the British HCI Conference, pp. 195–212.

Rosenbaum, D., 1991. Human Motor Control. Academic Press, San Diego, CA.

Schmidt, R.A., Zalaznik, H.N., Hawkins, B., Frank, J.S., Quinn, J.T., 1979. Motor output variability: a theory for the accuracy of rapid motor acts. Psychological Review 86, 415–451.

Wing, A., 1983. Crossman and Goodeve (1963): Twenty years on. Quarterly Journal of Experimental Psychology 35A, 245–249.

Worden, A., Walker, N., Bharat, K. and Hudson, S., 1997. Making computers easier for older adults to use: area cursors and sticky icons. Proceedings of the ACM CHI Conference on Human Factors in Computing Systems, pp. 266–271.

Zhai, S., 1995. Human performance in six degree-of-freedom input control. Ph.D. Thesis, Department of Industrial Engineering, University of Toronto.

Zhai, S., Buxton, W., Milgram, P., 1994. The "Silk Cursor": Investigating transparency for 3D target acquisition. Proceedings of the ACM CHI Conference on Human Factors in Computing Systems, pp. 459–464.

Zhai, S., Conversy, S., Beaudouin-Lafon, M., Guiard, Y., 2003. Human on-line response to target expansion. Proceedings of the ACM CHI Conference on Human Factors in Computing Systems, pp. 177–184.