Improving the Acquisition of Small Targets

Andy Cockburn & Andrew Firth

Human-Computer Interaction Lab, Department of Computer Science, University of Canterbury, Christchurch, New Zealand

Tel: +64 3 364 2987
Fax: +64 3 364 2569
Email: {andy, apf23}@cosc.canterbury.ac.nz

This paper describes the design and comparative evaluation of three methods that aid the acquisition of small targets. The first method, called 'bubble targets', increases the effective width of the target as the pointer approaches. The second method uses a form of 'stickyness' to restrict movement as the pointer passes over an object. In the third method, called 'goal-crossing', the user simultaneously presses two mouse buttons before passing the pointer over the item. Goal-crossing overcomes the need for the user to decelerate the mouse when acquiring the target. Two evaluations were conducted, with the first (n=37) based on the acquisition of abstract targets for Fitts' Law modelling, and the second based on an ecologically oriented window resizing task (n=11). Both showed that goal-crossing allowed the fastest target acquisition, but that it produced high error rates and was unpopular with participants. The 'bubble' and 'sticky' techniques also allowed faster target acquisition than the traditional approach, and users were enthusiastic about them. Fitts' Law accurately modelled all techniques. Implications of the results for general user interface design are briefly discussed.

Keywords: Target acquisition, Fitts' Law, expanding targets, sticky icons, goal-crossing.

1 Introduction

As the resolution of computer displays increases, designers of graphical user interfaces can increasingly rely on accurate and precise depiction of small user interface components. It is now common to find direct manipulation interface controls such as window borders, drop-down menus, and margin-markers that are smaller than 10 pixels (≈2mm on typical displays) on one or both dimensions. Although small components may be readily discernible (by those with normal eyesight), acquiring them with a mouse-driven cursor can be slow and frustrating. Making interface components larger decreases the acquisition time but reduces the number of items that can be placed in the display and adversely affects visual design.

Recently three separate schemes have been proposed for reducing target acquisition time without demanding increased screen space: expanding targets, sticky icons, and goal-crossing. McGufin and Balakrishnan (2002) showed that expanding targets, which enlarge as the cursor moves towards them, improve performance even when the expansion begins very late in the overall movement towards a target. Sticky icons effectively 'grab' the cursor as it moves over them (Worden et al 1997). Large movements 'snap' away from the icon, but small movements remain inside the item. Worden et al's evaluation suggested that sticky icons were particularly effective for older users when targeting small items. Goal-crossing interfaces (Accot and Zhai 1997, 2002) allow items to be selected by passing the cursor over the target area. They improve selection times because users do not need to decelerate and stop the cursor over the item. Details of these studies, and other related work, are provided in Section 2.

This paper describes the design and comparative evaluation of variants of these three schemes for aiding mouse-driven selection of small targets. The evaluation focuses on three factors: the comparative efficiency of the techniques, the degree to which Fitts' Law models their use, and the subjective preferences for the schemes. Although we expect our results to support those of related prior studies, direct comparison of the methods has not been possible due to differences in experimental methods. In particular, subjective preferences for the methods have not previously been compared.

Two separate evaluations were conducted. The first used an abstract selection task to generate data independent of any particular usage scenario. The second is more ecologically oriented, using a realistic window-resizing task to investigate the limitations of each technique in a more natural setting.

The following section describes related work on modelling and enhancing target acquisition in graphical user interfaces. Section 3 then describes the three targeting interfaces evaluated. Section 4 describes the first evaluation based on abstract target acquisition tasks, and Section 5 details the window resizing evaluation. Results are discussed and directions for further work are presented in Section 6. Section 7 concludes the paper.

2 Related Work

2.1 Fitts' law

Fitts' Law (1954) is commonly used to predict the time to move a mouse pointer from one location to another. Using the "Shannon formulation" of Fitts' Law (MacKenzie 1992), cursor movement time, *MT* increases linearly with the Index of Difficulty (IoD), which relies on the logarithm of the distance moved (the amplitude), *A*, over the width of the target, *W*. The two constants, *a* and *b*, are determined experimentally and depend on cognition and motor preparation time, and on hand-eye coordination, respectively. Fitts' Law also provides a measure of human processing of movement tasks, called the 'Index of Performance' (IoP) or

'bandwidth' (measured in 'bits/second') which is calculated from the reciprocal of the constant *b*.

$MT = a + b \times IoD$ where $IoD = \log_2\left(\frac{A}{W} + 1\right)$. Also, $IoP = \frac{1}{b}$ **Equation 1.**

Fitts' law was originally proposed for one-dimensional motion tasks. For movements in two dimensions, the target width *W* is normally measured using the smallest value of the width and height dimensions (MacKenzie and Buxton 1992).

2.2 Expanding targets

One approach to easing acquisition of small targets is to enlarge targets when they are needed. The commercial MacOs X Dock[1] demonstrates the technique, providing an icon panel in which the icons expand as the cursor approaches. Unfortunately, the MacOs X implementation can frustrate targeting because the expansion causes the icons to move if the cursor approaches the panel from a non-perpendicular angle. This problem is eased if expanded icons overlap one another (McGuffin and Balakrishnan 2002) or if the expansion only takes effect when the cursor velocity decreases on final target approach (Gutwin 2002).

McGuffin and Balakrishnan (2002) closely examined the degree to which Fitts' Law modelled targeting expanding targets in one-dimensional tasks. They found that Fitts' Law accurately models performance, and that movement time is primarily governed by the final expanded target size. This result held even when the targets began expanding after most (90%) of the movement towards the target was complete. McGuffin and Balakrishnan's study examined selection of a single object with no surrounding objects, so the influence of distraction due to neighbour-object motion was not examined.

2.3 Sticky targets

Another approach to aiding target acquisition uses a metaphor based on gravity, magnetism, or stickiness. Worden et al (1997) implemented 'Sticky Icons' by decreasing the mouse control-display gain (MacKenzie and Riddersma 1994) when the cursor enters the icon (control-display gain determines the mapping between physical mouse movement and resultant cursor movement). In this way, the user must move the mouse further to escape the boundary of the icon, effectively making the icon larger without using extra screen space. Worden et al's evaluation showed Sticky Icons to be efficient for selecting small targets.

Langdon et al (2000) performed an evaluation of a similar 'force feedback' concept. Users were required to select the inner circle of a pair of concentric circles, either using or not using force feedback. In the force feedback condition, when the cursor entered the outer circle, a 'force' warped the pointer toward the inner one. The force feedback condition was 30% to 50% faster than the normal condition. The utility of this technique is limited because of the undesirable impact on selectging near-neighbour interface components. A scrollbar, for example,

[1] www.apple.com/macosx/theater/dock.html

would be difficult to use if the pointer continually warped toward the window border.

2.4 Goal-crossing targets

With goal-crossing (Accot and Zhai 1997, 2002) the user selects an item by passing the pointer over the target area with some modifier key pressed. The motion is fluid and rapid in comparison to the normal selection process of moving the cursor, decelerating on approach to the target, stopping on the target, and finally clicking. Their studies were based on input using a stylus rather than a mouse.

Accot and Zhai (1997) found that selection through goal-crossing conforms to Fitts' Law, but differs from pointing techniques in the value of *W*, which is effectively infinite because there is no need to stop the cursor within the target's border. Rather than using a 'Smallest-of' model (see Section 2.1) for width, goal-crossing moves toward a 'Largest-of' model. For elongated interface controls such as window borders, the 'Largest-of' model is many magnitudes larger than 'Smallest-of'.

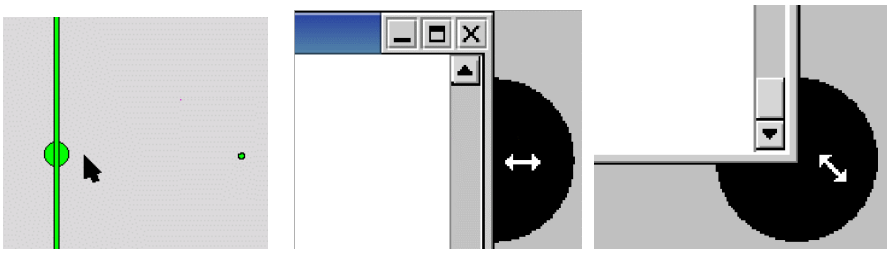
MacKenzie (1992) described a 'Stoke-through' technique similar to goal-crossing, where an icon is selected by depressing the mouse beside the icon, followed by a dragging motion over the icon, finishing with a button release on the other side. This technique was shown to be 40% faster than a standard point and select approach.

The main difference between moving the mouse for pointing tasks and for goal-crossing is that the mouse buttons are depressed, resulting in a dragging state. MacKenzie (1991) found that dragging times are slower than pointing times, and concluded that this was caused by interference from the additional task of holding down a mouse button. Thus, the Fitts' law values of *a* and *b* are increased for goal-crossing, but they would need to increase considerably to counteract the additional efficiency arising from a higher effective width value *W*.

3 Three Targeting Interfaces

The purpose of our evaluation is to directly compare the efficiency and subjective preferences for sticky, goal-crossing and expanding techniques in acquiring small targets. This section describes the three techniques evaluated.

All three interfaces, and the control 'normal' setting, were implemented in Tcl/Tk for Experiment 1 and Java Swing for Experiment 2. The experiments were run on a 1.4 GHz AMD Athlon computer running the Linux operating system, with a 19-inch display of 1600×1200 pixels. Control-display gain was set to a constant ratio of approximately 1:1.6 in both experiments. Input was provided through a three-button Logitech mouse.



(a) Exp. 1, bubble on line. (b) Exp. 2, edge bubble. (c) Exp. 2, corner bubble.
Figure 1: Bubble targets used in Experiments one and two. The target line or edge expands.

3.1 Bubble targets implementation

Based on expanding targets, bubble targets increase their effective size as the cursor approaches. In experiment one (Figure 1a), the bubble appeared as a circle centred on the x-axis of the line and on the y-axis of the cursor location. In Experiment two, the bubble appeared on the outside edge (Figure 1b) or corner (Figure 1c) of the target window.

Two design decisions with bubble targets involve trade-offs between visual distraction and the timing and size of the bubble display. McGuffin and Balakrishnan's (2002) showed that expanding targets remain efficient even when the expansion starts very late in the movement towards the target. This means that the bubbles need not be displayed until the cursor is very close to the target, reducing visual distraction. In our implementations, bubbles were only displayed when the cursor came within 15 pixels (experiment one) and 50 pixels (experiment two) of the targeted item. Similarly, larger bubbles are theoretically faster, but they are likely to increase visual distraction. Experiment one used a circular bubble with a 10-pixel radius, and experiment two used a radius of 40 pixels.

3.2 Sticky targets implementation

To provide a sense of 'stickyness', cursor motion must be constrained while within a sticky target. The primary design decision is in calibrating the level of 'stickyness' so that it aids targeting while not interfering with pointing elsewhere, including passing the cursor over the top of a sticky item.

We implemented sticky targets differently in experiments one and two, but the resultant interaction was similar in both implementations. In experiment one we used the Tcl/Tk motion event bindings to determine whether the pointer had moved sufficiently far to 'snap out' of a sticky target. When inside a sticky target, each time the movement event was registered (at most, once every 20ms on our machine), the software would calculate whether the cursor had moved more than a threshold distance (20 pixels) since the last motion event. If not, the cursor would warp back to the centre of the item. The cursor snapped out of the item once the threshold was exceeded. The resultant effect was that the cursor would stay motionless in the middle of the target when the mouse is moved slowly, but a slight

acceleration would snap out of the target. Calibrating a threshold level that provided a seemingly subtle and natural behaviour was straightforward.

In experiment two, like Worden et al (1997), we implemented sticky targets by reducing the mouse control-display gain to approximately 10% (1:0.16) of its original value when inside a sticky target. The interaction difference of this implementation to experiment one's is that continual slow motion within a sticky target will eventually leave the target's boundary.

The primary theoretical disadvantage of sticky targets stems from the risk of the cursor being 'grabbed' en-route to another target. One way to reduce this problem is to implement stickyness on only one axis. In experiment two, for example, where the tasks involved window resizing, the sticky effect was implemented across the window border, but not along it. Consequently, vertical window edges were horizontally sticky, and horizontal edges were vertically sticky. Beyond this, we have found that small sticky targets are surprisingly effective in coincidentally discriminating between ballistic motion to a different target (when stickyness is not desirable) and the decelerated motion of final acquisition (when it is). The reason is 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. The resultant effect, from our anecdotal experience, is a very natural effect that the cursor snaps onto targeted items, but passes over untargeted ones without disruption.

3.3 Goal crossing targets implementation

In our implementations, goal-crossing selection was achieved by dragging the cursor over a target while holding down the left and right mouse buttons. In experiment one, acquiring the target caused the cursor to lock onto the stationary target (by warping) until either mouse button was released. In the window resizing tasks of experiment two, the target window border followed the cursor, continually resizing the window border, until either mouse buttons was released. Two simultaneous mouse buttons were used because dragging tasks with one mouse button are commonly used by applications.

Theoretically, goal-crossing should allow the fastest target acquisition of the three techniques because the user does not have to decelerate the cursor on target approach. However, we wanted to inspect subjective preferences for a technique that we believed felt awkward with the mouse due to the unusual rapidly dragging action.

4 Experiment One: Abstract Task

The first experiment compared the three methods and the traditional method in an abstract one-dimensional task similar to that used by Accot and Zhai (2002). The experimental interface consisted of a six-pixel diameter dot placed some distance from the six-pixel wide grey target line in the middle of the window (Figure 1a). Participants clicked on the dot to start each task and then acquired the line as quickly as possible. Software logged the time between clicking the dot and

acquiring the target. Acquiring the target involved clicking on the line (for normal and sticky targets), clicking on the line or expanded bubble (for bubble targets), or goal-crossing the line with the left-mouse button held down. Software also logged all missed clicks and all mouse movement.

All participants used all four interfaces, with the order of exposure randomised to control learning effects. Five training selections and thirty-five logged selections were used per interface, with the same randomly determined distances to the left and right of the target line reused with each interface. The amplitudes between the starting circle and the line were between 10 and 390 pixels.

After using each interface the participants were asked to respond to the questions "Selecting the line was physically demanding (high effort)" and "Selecting the line was efficient (fast)" using five point Likert scales, from one (disagree) to five (agree). After completing the tasks with all of the interfaces, they were asked to rank the four methods, from their favourite to their least preferred, and to provide any additional comments.

Thirty-seven undergraduate Computer Science students took part in the experiment. They were rewarded with a $5 shopping voucher. Participants were encouraged to rest and flex their wrists between tasks.

4.1 Data Analysis

The experimental data is inspected in two ways. First, regression analysis is used to determine whether Fitts' Law holds for each of the techniques. 'Bandwidth' measures (Section 2.1) are also calculated from the regression analysis to determine the psychophysical throughput of each method. Second, a 4×4 repeated-measures analysis of variance (ANOVA) is conducted for factors 'interface type' and 'distance'. The factor 'distance' allows us to inspect whether any of the techniques favoured short or long distances. Its four levels are 'short' (10-100 pixels), 'low-medium' (101-200 pixels), 'high-medium' (201-300 pixels) and 'high' (301-390 pixels). Error rates are also inspected.

4.2 Results

4.2.1. Conformity to Fitts' Law

In calculating the Index of Difficulty of each task, we used the visible thickness of line target, which was constant at 6-pixels. It is important to note, however, that the three new selection schemes are designed to increase the actual target size, through expansion, stickyness or goal-crossing, reducing the IoD.

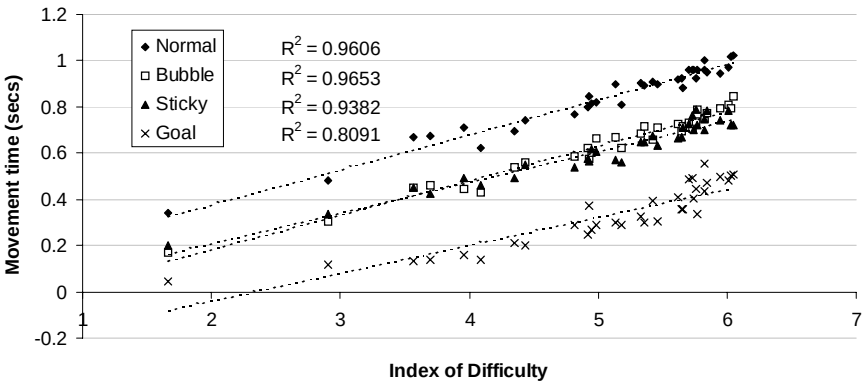


Figure 2: Movement time plotted against Index of Difficulty for the four interfaces. Linear regression lines of best fit are shown.

Linear regression analysis of the relationship between movement time and IoD shows that Fitts' Law accurately models all four methods—lines of best fit are shown in Figure 2. The r^2 values for the four techniques were 0.96, 0.97, 0.94 and 0.81 for the normal, bubble, sticky and goal-crossing techniques (r^2 values exceeding 0.8 are normally deemed to be accurate in experiments involving human participants). The lines of best fit, r^2 values, and Index of Performance measures for the four techniques are shown in Table 1.

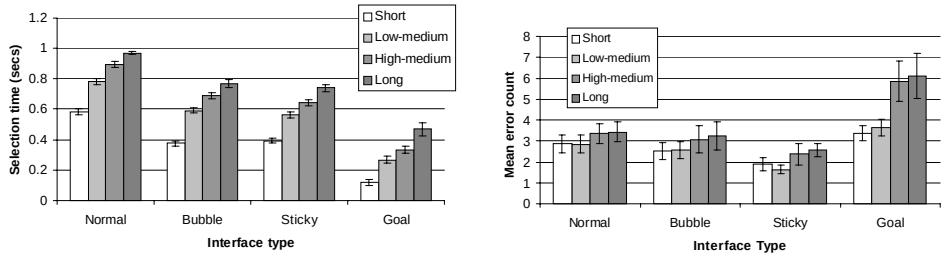
4.2.2. Differences between the methods

All three of the new target selection techniques were significantly faster than the traditional approach, as shown in Figure 3a. The mean target acquisition times for normal, bubble, sticky and goal-crossing interfaces were 810ms (standard deviation 185), 607ms (s.d. 186), 585ms (s.d. 180ms) and 298ms (s.d. 216) respectively, producing a reliable difference $F_{3,108}=199.8$, $p<0.001$. In pairwise comparison, a Tukey test produces an Honest Significant difference of 111ms, revealing a reliable difference ($p<.05$) between all interface types except bubble and sticky targets, which performed similarly. It is telling that the slowest of the new techniques, bubble targets, reduced targeting time by 25% of the normal method.

There was a marginal interaction between interface type and distance ($F_{9,324}=2.16$, $p=0.024$), meaning that performance with at least one interface deteriorated differently to the others as distance increased. One probable explanation is the comparatively high performance deterioration with goal-crossing between high-medium (201-300 pixels) and long (301-390 pixel) amplitudes of motion. We believe this effect is due to the difficulty of maintaining a dragging

Table 1: Linear regression equations, r^2 values, and Indexes of Performance for the four techniques.

Method	**Line of best fit**	r^2	**IoP (bits/sec)**
Normal	MT= 74 + 152×IoD	0.96	6.59
Bubble	MT= -115 + 149×IoD	0.97	6.70
Sticky	MT= -53 + 132×IoD	0.94	7.55
Goal	MT= -282 + 121×IoD	0.81	8.26



(a) Movement time. (b) Errors.

Figure 3: Mean movement times and error counts for the four interfaces by distance. Error bars show ±1 standard error around the mean.

state over long distances. This explanation is supported by the analysis of errors, reported below.

The number of off-target clicks was also analysed. Any mouse press between selecting the initial dot and clicking the target line was added to the 'missed clicks' total. The mean number of misses across the 35 logged tasks were significantly different ($F_{3,108}$=7.8, p<0.001) at 10.4 (s.d. 15.7) with the normal method, 7.9 (s.d. 11.9) with bubble targets, 6.1 (s.d. 7.3) with sticky targets, and 16.3 (s.d. 15.7) with goal-crossing, as shown in Figure 3b.

The 'miss' count for goal-crossing is an upper-bound measure because the software logged any mouse press after selecting the initial dot as a miss. With goal-crossing, however, it is legitimate for the user to initiate the task by clicking the dot, but only acquire goal-crossing mode (by pressing both mouse-buttons) on final approach to the target. The fastest goal-crossing users initiated goal crossing mode at the same time as selecting the initial dot, but many users moved most of the distance towards the target before pressing both buttons. Consequently, the high 'error' count for goal-crossing is suspect.

One final measure suggests that goal-crossing became more difficult and error prone than other methods as the distance increased. When analysing errors using the same two-factor ANOVA as that used for movement time, there is a marginal interaction between interface type and distance: $F_{9.324}$=1.9, p=0.05. This interaction is visible in the comparatively large step between goal-crossing errors in the short to low-medium distances versus the high-medium and long distances. The participants' comments support the explanation that this is due to the difficulty of maintaining two-button dragging states over long distances. Possibly, with more experience users would learn to only initiate the goal-crossing mode when finally approaching the target.

4.2.3 Subjective preferences

The participants indicated that targeting with all of the methods demanded relatively low effort. Responses to question one "Selecting the target was physically demanding (high effort)" ranged from low values with bubble and sticky targets (both means 2.5, s.d. 1,1) through goal-crossing (3.0, s.d. 1.5) to the most demanding normal interface (3.1, s.d. 1.3). Although these ratings do not produce a reliable difference (Friedman χ^2=6.5, p=0.09), the participants' comments reflected the trend in the ratings. For example, one participant commented "goal-crossing is

Table 2: Preference rankings for the four interface types.

	Preference Ranking			
Method	**1st**	**2nd**	**3rd**	**4th**
Normal	4	6	12	15
Bubble	8	12	13	4
Sticky	17	12	6	2
Goal	8	7	6	16

more physically demanding because holding down the mouse buttons locks up your wrist more making it harder to scroll around with the mouse, reducing fine control and increasing fatigue."

Interestingly, despite the substantially faster performance with goal-crossing, the participants rated it relatively poorly for efficiency. Mean responses to question 2 "Selecting the line was efficient (fast)" for the bubble, sticky, goal-crossing and normal interfaces were 3.8 (s.d. 1.2), 3.9 (s.d. 0.9), 3.4 (1.6) and 2.6 (1.1) respectively, producing a reliable difference: Friedman $\chi^2=23.8$, $p<0.01$.

The overall rankings of the four techniques show a strong preference for the sticky technique, as shown in Table 2. Forty-six percent of the participants ranked sticky targets first, and a further 32% ranked it second. Goal-crossing, in contrast, was ranked first by 22% of the participants, and last by 43%.

Bubble targets were consistently in the middle rankings. Negative comments about bubble targets concerned the visual distraction that they caused and the fact that they encouraged the participants to become sloppy in their targeting. For example, one participant commented: "I became over-confident and sometimes missed even the bubble" and another stated "I wonder if my times were actually worse with bubbles because I changed my attitude. I didn't worry about accuracy because the target was so big, so I often overshot".

4.3 Summary

The results support prior work showing that Fitts' Law accurately models all four techniques. Although goal-crossing allowed extremely rapid performance, the subjective responses, comments, and error rates suggest that the users found it awkward and clumsy. It must be stressed that this finding applies only to mouse-driven goal-crossing, and does not apply to stylus input methods.

Sticky targets were extremely popular, rated first or second favourite by 78% of participants. They also performed well, with mean selection times 28% faster than the normal method, and with 41% fewer target misses.

5 Experiment Two: Window Resizing Evaluation

The second experiment examines how the four techniques compare in a more ecologically oriented task. The aim is to explore the design decisions necessary to deploy the methods in support of real tasks, and to validate the results of the abstract tasks in experiment one.

5.2 Window Resizing

Resizing windows is a common action in graphical desktop environments—Gaylin (1986) found that it comprised 2% of all windows commands and 12.4% of commands when a user logs onto a computer. Despite the frequency of window resizing, the traditional method requires a high level of precision in acquiring and dragging the window border (normally only a few pixels wide). Some window management systems, such as Sawfish[2], allow customisable key-bindings to remove the need to acquire the window border, but this approach suffers the disadvantages that the user must customise and memorise the keybinding, and that the keybindings can clash with those used by application software.

5.2 Interface modifications

The bubble and goal-crosssing methods were modified for the resizing task, based on the design decisions that we felt would be necessary to create commercially viable implementations.

Bubbles only appeared on the outside window edges (see Figure 1b,c). This change would be necessary in commercial deployment to avoid targeting collisions between bubbles and interface controls such as scrollbars that are close to the window edge. Also, to reduce visual distraction, the bubbles only appear when the cursor is inside the window. Consequently, when resizing a window that the cursor is initially outside, the user would have to sweep the cursor inside the window, then move outwards towards its edge. The radius of the part-bubble was 40 pixels, and the bubbles appeared when the cursor was less than 50 pixels from the window edge.

Like bubble borders, goal-crossing mode could only be initiated from inside the window being resized. This meant that when the cursor was initially outside the target window, the user would have to sweep the cursor inside the window to acquire goal-crossing mode, and then move outwards across the border with mouse buttons one and three held down. We felt that this design decision would be necessary in a commercial implementation to avoid problems associated with passing the cursor over other windows in the display. If goal-crossing mode could be acquired from outside a window boundary, then the user would need to carefully coordinate the timing of their mouse button presses to ensure that windows lying between the initial cursor location and the target window were not mistakenly resized.

5.3 Method

Five window-resizing tasks were carried out with each of the four interfaces. The window borders used to resize the windows were 6 pixels wide, and the corners extended 25 pixels along the border. The tasks differed by whether the window was being enlarged or diminished, whether a side border or corner was used, and

[2] sawmill.sourceforge.net

Table 3: Window resizing tasks used in Experiment 2.

Task	Resize	Border used	Cursor location
1	Enlarge	Right	Inside window
2	Diminish	Right	Inside window
3	Enlarge	Bottom-right corner	Inside window
4	Enlarge	Left	Outside window
5	Diminish	Left	Outside window

whether the cursor started inside or outside the window (see Table 3). The different resize directions, borders used and cursor locations were included to illuminate differences between the techniques—they were not intended to be independent experimental factors. Each task consisted of five subtasks that varied the distance the pointer started from the border. The interface order was randomly assigned to each participant to control learning effects.

Tasks were initiated by clicking on a red dot positioned 20, 50, 100, 200 or 500 pixels from the target border. The participants then acquired and dragged the window border into a large blue-coloured region that was 50 pixels from the border in the resizing direction. The software automatically logged task times.

Eleven participants took part in the experiment. All were Computer Science graduate students.

5.4 Results

Timing data from the experiment was analysed in a 4×5×5 repeated measures analysis of variance (ANOVA). The three factors (all within subjects) were Interface type (normal, bubble, sticky and goal-crossing), Task (one to five, as shown in Table 3), and Distance (20, 50, 100, 200 and 500 pixels).

The results support those of experiment one. Across the 1100 subtasks, the mean window resizing time was relatively fast at 855ms, with a standard deviation of 438ms. ANOVA revealed a significant main effect for Interface type ($F_{3,30}$=27.0, $p < .01$), with goal-crossing fastest (678ms, sd 460ms), and progressively slower through sticky borders (797ms, sd 274ms), bubble borders (907ms, sd 502ms), and normal borders (1041ms, sd 398ms). Figure 4 shows the mean task times for each of the interfaces across the five tasks.

As expected, there were significant main effects for factors Task ($F_{4,40}$=4.5, $p < .01$) and Distance ($F_{4,40}$=44.8, $p < .01$). Mean task times increased through Tasks one (mean 779ms, sd 465ms) to five (mean 933ms, sd 441ms). The increasingly complex movements required to complete the tasks explain this increase (discussed below). Similarly, mean subtask times increased as the distance increased: from a mean of 689ms (sd 348ms) at 20 pixels to 1101ms (sd 522ms) at 500 pixels.

There was an interesting interaction between Task and Interface type ($F_{12,120}$=7.0, $p < .01$), visible in Figure 4. Unlike the other interfaces, goal-crossing performance was dramatically different between tasks that involved window enlarging (one, three and four) and shrinking (two and five). This effect is explained by the need to change mouse direction while holding down the mouse button when goal-crossing. In task two, the cursor begins inside the window, so the user must issue the crossing action rightwards over the right hand window border before changing

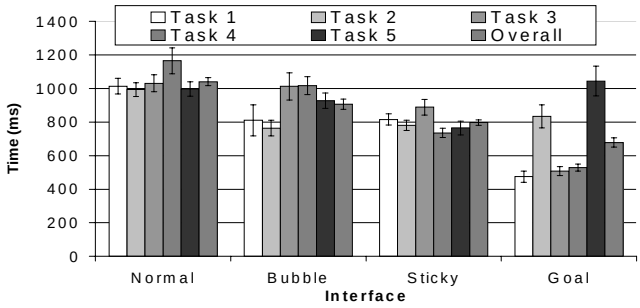


Figure 4: Mean movement times for the four interfaces by task. Error bars show ±1 standard error around the mean.

direction to drag the border leftwards. The advantage of goal-crossing in avoiding deceleration is reduced, and its performance is similar to the other interfaces. In Task five, which involves decreasing the window size starting outside the target window, goal-crossing performs even more poorly. Again, this is explained by the mouse direction changes required while holding the mouse button down. The user must first move inside the window, press and hold the mouse button while sweeping outward to acquire the border, and then change direction for a second time to shrink the window. Interestingly, goal crossing remained efficient in Task 4, even through a direction change is required. We suspect that the reason for this is that the mouse button is not held down while changing direction because users move the cursor into the window before changing direction, pressing the mouse button, and sweeping outward over the border.

5.5 Comments

Many of the participants were surprised by how poorly they performed with the normal interface. One stated “you don’t realise how slow the normal technique is until you try some of these other methods”.

Although seven of the participants stated that they liked the rapid speed of goal-crossing, several noted that the high speed was at the cost of accuracy. When goal-crossing, most participants made rapid sweeping motions with the mouse, moving the cursor much further than the minimum movements required. The experimental tasks promoted this rapid and crude movement because the participants were informed that the timing stopped when the window border entered the target blue-coloured region. The experiment fails to indicate whether goal-crossing would remain as efficient if specific window sizes were required.

6 Discussion and Further Work

Although the results show that goal-crossing is a highly efficient method of mouse-driven target acquisition, the participants’ subjective preferences indicate that it should be used with caution in mouse driven graphical user interfaces. In general,

users felt a lack of control with goal-crossing due to the in accuracy of rapid dragging. It is highly likely that this negative perception only applies to mouse-driven input and would not be observed with the more dextrous input available with a stylus. However, stylus input devices remain specialist hardware that is unusual on everyday desktop computers.

Bubble targets resulted in improved performance when compared to the normal method, but many participants stated that they found the appearance of the bubbles distracting. The bubbles also induced a 'sloppy' targeting behaviour in some participants who, anticipating the appearance of the bubble, targeted the region around the target rather than the target itself. This behaviour adversely affected their effectiveness due to hunting effects. In the window-resizing tasks of experiment two, the hunting effect was exacerbated because the bubbles only appeared on the outside edge of the window that the cursor was inside. Consequently, having overshot the bubble, it would disappear, and would only reappear once the cursor had re-entered the window. Due to the risks of visual distraction and of encouraging hunting, we recommend that expanding targets be used with caution.

Sticky targets were efficient and popular. Participants liked the fact that they created no visual distraction, yet simplified targeting. Interestingly, we did not observe a sticky target learning effect equivalent to that with bubbles, where the participants became more 'sloppy' in anticipation of eased targeting. We suspect this is explained by the absence of the bubble's affordance—the participants would decelerate towards the small target as normal, but the stickyness caused the cursor to snap to the line, reducing overshooting.

Despite the efficiency and popularity of the sticky method, it should still be used with caution, and further research is necessary to determine how closely sticky components can be placed without causing interference between targets. By 'interference', we mean situations where the cursor unintentionally locks onto a sticky target nearby another sticky component. The user would have to 'snap' the cursor away from the item, with the consequent risk of overshooting the target.

We intend to further investigate how sticky targeting can be enhanced through feedback in other modalities. Brewster et al (1994), for example, showed that scrollbar use was enhanced through the addition of auditory feedback. Similarly, Fraser and Gutwin (2000) show that in visually stressed situations, redundant visual and auditory feedback reduced targeting time. Oakley et al (2001) examined how haptic feedback affected simple user interface controls, finding that it did not reliably influence task time, but significantly reduced the number of errors.

We also intend to investigate how targeting aids such as these can be used to assist users who have visual or motor-coordination impediments.

7 Conclusions

It is increasingly common for graphical user interfaces to contain direct manipulation controls that measure only a few pixels on one or both dimensions. Acquiring small targets using a mouse demands a high level of precision and can be frustratingly error-prone and slow.

This paper described the design, implementation and evaluation of three alternative schemes to aid the mouse-driven acquisition of small targets. Bubble targets increase in size when the cursor is close. Sticky targets either alter the mouse control-display gain or warp the cursor once the target boundary is crossed to produce a sense of 'stickyness'. Goal-crossing targets are acquired by sweeping the cursor over the item while holding down a modifier key or mouse button. Although all three techniques are based on prior research, the methods have not previously been evaluated in direct comparison.

Measurements in an abstract targeting task confirmed that Fitts' Law accurately models all three techniques, and that they all reduced the normal target acquisition time by at least 25%. Although goal-crossing was extremely rapid, many of the participants felt it was impractical with a mouse due to a lack of accuracy. Sticky targets were popular due to their natural behaviour and their absence of visual distraction. Performance measures and subjective preferences in a more ecologically oriented window-resizing task confirmed the results and preferences of the abstract task.

The results are promising, and future work will focus on identifying the factors affecting the commercial deployment of sticky interface components. We also wish to investigate how these techniques can be used to aid those with visual and motor-coordination impairments.

References

Accot J, Zhai S, 1997 "Beyond Fitts' Law: Models for Trajectory-Based HCI Tasks", in *Proceedings of CHI'97 Conference on Human Factors in Computing Systems.* (Atlanta, Georgia, March 22-27) pp 295--302

Accot J, Zhai S, 2002 "More than Dotting the i's --- Foundations for Crossing-Based Interfaces", in *Proceedings of CHI'2002 Conference on Human Factors in Computing Systems* (Minneapolis, Minnesota, 20--25 April) pp 73--80

Brewster S, Wright P, Edwards A, 1994 "The design and evaluation of an auditory-enhanced scrollbar", in *Proceedings of CHI'94 Conference on Human Factors in Computing Systems* (Boston, April 24--28) pp 173--179

Fitts P, 1954 "The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement." *Journal of Experimental Psychology* **47** 381-391

Fraser J, Gutwin C, 2000 "The Effects of Feedback on Targeting Performance in Visually Stressed Conditions", in *Proceedings of Graphics Interface Conference* (May, Montreal, Canada)

Gaylin K, 1986 "How are Windows Used? Some Notes on Creating an Empirically-Based Windowing Benchmark Task", in *Proceedings of the CHI'86 Conference on Human Factors in Computing Systems III* pp 96--100

Gutwin C, 2002 "Improving Focus Targeting in Interactive Fisheye Views", in *Proceedings of CHI'2002 Conference on Human Factors in Computing Systems* (Minneapolis, Minnesota, 20--25 April) pp 267--274

Langdon P, Keates S, Clarkson P, Robinson P, 2000 "Using Haptic Feedback to Enhance Computer Interaction for Motion-Impaired Users", in *3rd International Conference on Disability, Virtual Reality and Associated Technologies (ICDVRAT 2000).* (Alghero, Sardinia, Italy.) pp 25--32

MacKenzie I, 1991 "Fitts' Law as a Performance Model in Human-Computer Interaction"

MacKenzie I, 1992 "Movement Time Prediction in Human-Computer Interfaces", in *Proceedings of Graphics Interface '92.*

MacKenzie I, Buxton W, 1992 "Extending Fitts' Law to Two-Dimensional Tasks", in *Proceedings of CHI'92 Conference on Human Factors in Computing Systems* (Monterey, May 3--7) pp 219--226

MacKenzie I, Riddersma S, 1994 "Effects of Output Display and Control-Display Gain on Human Performance in Interactive Systems" *Behaviour and Information Technology* **13** 328--337

McGuffin M, Balakrishnan R, 2002 "Acquisition of Expanding Targets", in *Proceedings of CHI'2002 Conference on Human Factors in Computing Systems* (Minneapolis, Minnesota, 20--25 April) pp 57--64

Oakley I, McGee M, Brewster S, Gray P, 2001 "Putting the Feel in 'Look and Feel'", in *Proceedings of CHI'2000 Conference on Human Factors in Computing Systems* (The Hague, The Netherlands, April 1--6) pp 415--422

Worden A, Walker N, Bharat K, Hudson S, 1997 "Making Computers Easier for Older Adults to Use: Area Cursors and Sticky Icons", in *Proceedings of CHI'97 Conference on Human Factors in Computing Systems.* (Atlanta, Georgia, March 22-27) pp 266-271