Automatic Bars with Single-Switch Scanning for Target Selection

Mathieu Raynal[1(✉)] and I. Scott MacKenzie[2]

[1] ELIPSE team, IRIT lab, University of Toulouse, Toulouse, France
mathieu.raynal@irit.fr
[2] Electrical Engineering and Computer Science, York University, Toronto, Canada
mack@yorku.ca

Abstract. We present and evaluate two point-select methods using automatic bars or regions and single-switch scanning. The first method, iButton, uses horizontal and vertical bars that move in sequence following switch selections with a final selection at the point of intersection. An additional feature is the scanning of three buttons to set the direction the bars move: left/up, right/down, or none. The second method, CSSS for circular single-switch scanning, rotates a $\frac{\pi}{8}$ radian arc region, a line within the arc region, and a pointer along the line. Three switch selections are required to select a target. In a Fitts' law user study with 10 participants, both methods exhibited similar throughput, about 0.35 bps. The iButton method was about 11% faster with selections taking on average 7.80 s compared to 8.74 s with CSSS. However, the CSSS method was about 35% more accurate with a 4.24% error rate compared to a 6.56% error rate with iButton. Eight of ten participants preferred the CSSS method. All participants reported that they felt faster with the CSSS method, even though the iButton method was objectively faster.

Keywords: motor disability · pointing technique · single-switch scanning · assistive technologies · Fitts' law

1 Introduction

On our desktops and laptops, WIMP interfaces remain the most frequently used. Selecting elements using a pointing device and pointing technique is one of the main pillars of these interfaces. The pointer is usually manipulated using a mouse or touchpad. However, users with severe motor impairments are generally unable to use these devices, preferring instead to use adapted devices. A common alternative is a setup using eye-tracking [10]. However, these techniques require the user to fixate on the element he wishes to select. Slightly less restrictive techniques exist, such as head tracking [8] or face tracking [4], or devices using EEG signals [6], or voluntary muscle contraction signals using EMG [5]. In this article, we are not proposing a new physical device, but simply two pointing techniques based on automatic scanning and requiring only a single input switch for the user. These techniques are inspired by scanning keyboards, where the cursor moves automatically across the keyboard and the user only activates a single input switch when the cursor is on the desired character.

K. Miesenberger et al. (Eds.): ICCHP 2024, LNCS 14751, pp. 76–83, 2024.
https://doi.org/10.1007/978-3-031-62849-8_10

2 Automatic Bars with Single-Switch Scanning

The work presented here extends earlier work on two-bar single-switch scanning (TBS[3]) with two new techniques that also use single-switch scanning.

2.1 Optimized Two Bars Single-Switch Scanning

The first technique is an improved version of the "Button" technique which was the best of the two versions previously proposed and tested [7]. In the initial version, before the start of each line, two soft buttons are displayed in the center of the bar to allow the user to choose the movement direction. Button focus automatically alternates between the two buttons until the user makes a selection to set the direction of movement. This done by pressing the single input switch when the desired direction button is highlighted. In the follow-on study described herein, we made improvements from lessons learned during the first experiment: namely, when choosing the direction, a central button is added to allow the user not to move the bar if it is already positioned at the desired location (Fig. 1). In addition, we add a timeline on the buttons, so users are aware of when the cursor is about to change buttons. We call this variation "Improved button" or "iButton" for short.

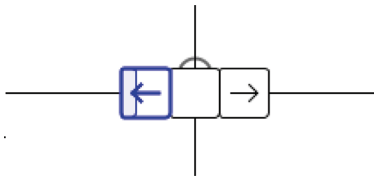

Fig. 1. Soft buttons are scanned to choose the movement direction, with a central button if the line is not to move.

3 Circular Single-Switch Scanning

The second pointing technique, "Circular Single-Switch Scanning" (CSSS), is based on circular scanning of the screen. Pointing occurs in three main steps: First, an arc region of $\frac{\pi}{8}$ radians advances counter-clockwise around an origin (Fig. 2, left). Initially, the origin is at the center of the screen, then it is positioned at the coordinate of the last selection. When the arc covers the desired zone, the user presses the input switch. Then, a line appears at the base of the arc and rotates around the same origin (Fig. 2, right). Once the arc stops, a line restarts the scan from the position of the arc in the same direction of rotation. Finally, the user stops the line by pressing the input switch, wherein a pointer automatically moves along the line. The user again presses the input switch to perform a click selection.

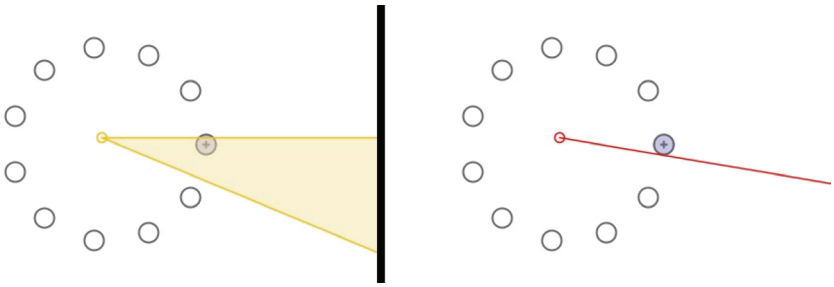

Fig. 2. CSSS method. At the left, the arc region of $\frac{\pi}{8}$ radians; at the right, a red line appears and moves within the arc when is stopped. See text for discussion. (Color figure online)

4 Method

The goal of our user study was to compare the two pointing methods described above, "iButton" and "CSSS". We used the ISO 9241-411 2D Fitts' law task.

Participants - We recruited 10 participants from the first author's university campus. Six were male, four were female. Ages ranged from 18 to 33 yrs.

Apparatus - The experiment was conducted on a Dell Latitude 5490 laptop with a resolution of 1920 × 1080 pixels. The experiment tasks were presented using a modified version of GoFitts[1], a Java application implementing 2D Fitts' law task described in the ISO standard (see the target arrangement in Fig. 2). GoFitts includes additional utilities such as FittsTrace which plots the cursor trace data captured during trials.

The 2D task presented 11 targets per sequence, combining three movement amplitudes (100, 200, 400 pixels) with three target widths (20, 40, 80 pixels). Amplitude and width were included to ensure the conditions covered a range of task difficulties. The result is nine sequences for each test condition with IDs ranging from $\log_2(\frac{100}{80}+1) = 1.17$ bits to $\log_2(\frac{400}{20}+1) = 4.39$ bits.

Procedure - The software and experiment were explained and demonstrated, following which testing began. Testing took a little under one hour per participant. Before each method, participants could test the method as much as they wanted (they generally performed two or three sequences of target selections).

Design - The experiment was a 2 × 3 × 3 within-subjects design with the following independent variables and levels: Pointing method (iButton, CSSS), Amplitude (100, 200, 400 pixels), and Width (20, 40, 80 pixels). For each sequence,

[1] https://www.yorku.ca/mack/FittsLawSoftware/.

11 targets appeared. The dependent variables were throughput (bps), movement time (ms), and error rate (%).

Participants were divided into two groups for counterbalancing. One group started with the CSSS method and the other started with the iButton method. The total number of trials was 1980 ($= 2 \times 3 \times 3 \times 11 \times 10$).

5 Results

We present below the results of our experiment, organized by dependent variable, with follow-on analyses for path traces and Fitts' law models.

Throughput - Throughput (TP) [9], measured in bits per second (bps), is calculated over a sequence of trials as the ID-MT ratio: $TP = ID_e/MT$. The ISO 9241-411 standard [2] specifies calculating throughput using the effective index of difficulty (ID_e) which involves adjusting the accuracy to reflect the spatial variability in responses: $ID_e = \log_2(A_e/W_e + 1)$ with $W_e = 4.133 \times SD_x$ (see [3] for details).

The mean throughputs were very similar between the two pointing methods, 0.345 bps for iButton and 0.348 for CSSS (Fig. 3, left). Consequently, the effect of pointing method on throughput was not statistically significant ($F_{1,8} = 0.023$, ns). These performance measures are inline with those obtained in the earlier user study [7] with the first version of the Button technique (0.356 bps).

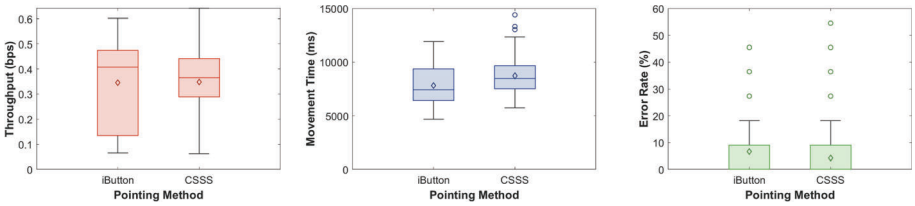

Fig. 3. Results for pointing method. Left: throughput (bps); center: movement time (s); right: error rate (%). Horizontal lines in the boxes represent the median, and markers show the means of the responses

Movement Time and Error Rate - By using the effective index of difficulty in the calculation, throughput takes into account both input speed and selection variability. Although throughput is almost identical for our two techniques, it is interesting to study these two criteria separately. See Fig. 3 for movement time (left) and error rate (right).

We see in Fig. 3 (middle) that the speed is dependent on the technique: participants were faster with iButton than with CSSS (7801 ms vs. 8737 ms,

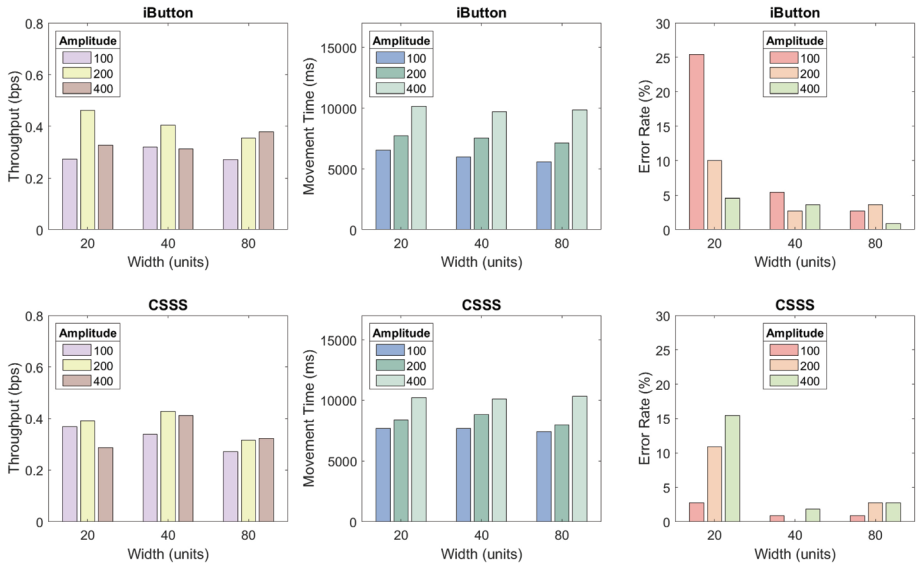

Fig. 4. Throughput (bps), movement time (s), and error rate (%) by width and amplitude for the iButton (top) and CSSS (bottom) methods.

respectively, for a reduction of 11% for iButton), and this in a significant way ($F_{1,8} = 21.506, p < .005$). However, our improved version, iButton, is slower than its previous version (7085 ms for the previous version). Moreover, we can see in Fig. 4 (middle) that width has no effect on the movement time for both methods ($F_{2,16} = 0.970$, ns). On the other hand and as expected, the distance between the targets influences the movement time to reach the target ($F_{2,16} = 5.997, p < .05$).

The box plots for error rate (Fig. 3, right) look unusual due to the high number of error-free sequences. Of the 90 trial sequences for each pointing method, the number of error-free sequences was 51 (56.7%) and 67 (74.4%) for iButton and CSSS, respectively. Consequently, for both pointing methods, the median error rate was 0%.

Participants were about 35% more accurate with CSSS than with iButton (4.24% errors vs. 6.56% errors, respectively), but the effect of selection method on error rate was not statistically significant ($F_{1,8} = 3.023, p > .05$). Both techniques are more accurate than the previous version (8% of errors for the previous version). Moreover, we can see in Fig. 4 (right) that width has an impact on the error rate ($F_{2,16} = 17.048, p < .0005$): The participants many more errors with the small targets.

Cursor Trace Examples - Figure 5 provides examples of the pointer traces for both methods. On the left, the traces for the iButton technique are straight horizontal or vertical lines, which reveal the functioning of the pointing method where the movements of the pointer are guided by successive rectilinear move-

ments of the vertical then horizontal bars. On the right, the CSSS traces are straight lines between the starting target and the target. These traces do not reflect the entire selection process of the CSSS technique, as the pointer only moves once the arc and then the line have stopped.

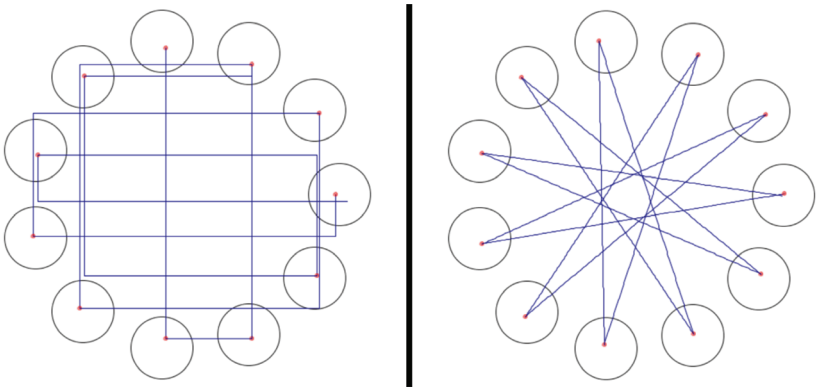

Fig. 5. Example pointer traces for the iButton (left) and CSSS (right) methods.

Fitts' Law Models - To test for conformance to Fitts' law, we built linear regression models for each pointing method. We can see in Fig. 6 that the slope of the iButton line is higher than that for CSSS, with the two lines crossing at about 4.5 bps. So if the iButton technique is faster for index below $ID = 4.5$ bits, the CSSS technique should be faster for higher *IDs*.

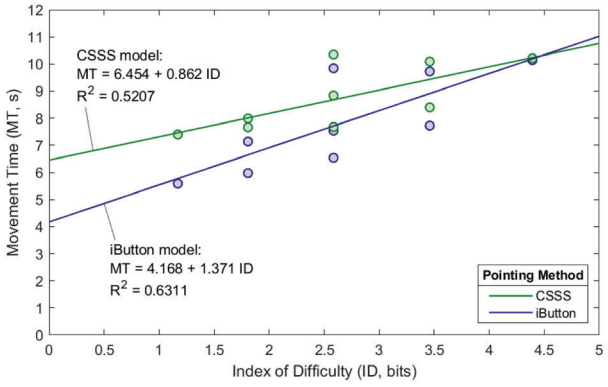

Fig. 6. Fitts' law models for iButton (blue line) and CSSS (green line) methods (Color figure online)

However, there are differences for the same index of difficulty. This is because the index of difficulty depends on width and amplitude. Thus, the combinations 20/100, 40/200 and 80/400 yield the same index of difficulty. In the Sect. 5, we saw that movement time depends on the amplitude but not on the size of the targets. This is why we have here distinct values for the same *ID*. This difference generates a lower value of the correlation coefficient (r and also R^2).

Preference - At the end of the experiment, we asked participants for their preference between the two techniques. Eight of ten preferred CSSS. All participants reported that they felt faster with the CSSS method event though objectively this was not the case.

6 Discussion

The changes made to the initial version of the Button technique resulted in an increase in the movement time. Indeed, participants were 10% slower with the new version (7801 ms vs. 7085 ms in the previoius version). This difference is due to the additional central button. The button is presented first to the user and remains active for 500 ms before the highlight moves to the next button. As this button is presented on both bars, it increases the pointing time by one second when the user has to move both bars. On the other hand, this button also makes it possible not to move a bar when the movement doesn't require it. This avoids the user having to press the single input switch quickly to stop the bar. This has helped to reduce the error rate (6.56% compared with 8% initially).

Our second technique, CSSS, was slower than iButton, but the participants also made fewer pointing errors with this technique. So, if we take throughput into account, our two techniques are equivalent in terms of performance (albeit iButton is faster but CSSS is more accurate).

On the other hand, several errors were observed on small targets. The error rate is quite high for targets close to the iButton and for small targets in particular. When the bars are too close to the target, and the target is small, users are unable to start and stop the bar quickly enough. On the contrary, the error rate increases with distance for CSSS. This is because the farther the target is from the starting position, the more the angle of selection of the target is small during circular movements. This makes the task more difficult.

Finally, it is important to mention that the scan intervals for the two techniques (the two bars for iButton; the arc, the line and then the pointer for CSSS) were determined empirically. Each user can set these according to their motor skill, adjust as their skill develops, and find the best compromise between speed and precision.

7 Conclusion and Future Work

Regardless of the technique used, one problem persists: the error rate remains high, especially when selecting small targets. This problem is linked to the movement speed, which is constant throughout the movement. Some errors could

certainly be avoided if the speed were slower at the start of the movement and increased with distance. This would also make it possible to adapt the speed and reduce the time needed to reach distant targets. A follow-on study will therefore focus on a movement speed function in order to optimise pointer speed according to the distance travelled.

A "discrete" version of CSSS will also be studied. Instead of a continuous circular movement, the selector would move from zone to zone, then from target to target. This could be useful, for example, in configurations with targets are close together, such as in menus, where this type of discrete movement has already been studied with pointing devices [1].

References

1. Dubois, E., Serrano, M., Raynal, M.: Rolling-menu: rapid command selection in toolbars using roll gestures with a multi-DoF mouse. In: SIGCHI Conference on Human Factors in Computing Systems (CHI 2018), pp. 1–12 (2018)
2. ISO: Evaluation methods for the design of physical input devices - ISO/TC 9241-411: 2012(E). Report Report Number ISO/TS 9241-411:2102(E), International Organisation for Standardisation (2012)
3. MacKenzie, I.S.: Fitts' law. In: Norman, K.L., Kirakowski, J. (eds.) Handbook of Human-Computer Interaction, pp. 349–370. Wiley, Hoboken (2018). https://doi.org/10.1002/9781118976005
4. Magee, J., Felzer, T., MacKenzie, I.S.: Camera mouse + ClickerAID: dwell vs. single-muscle click actuation in mouse-replacement interfaces. In: Antona, M., Stephanidis, C. (eds.) UAHCI 2015. LNCS, vol. 9175, pp. 74–84. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20678-3_8
5. Perez-Maldonado, C., Wexler, A.S., Joshi, S.S.: Two-dimensional cursor-to-target control from single muscle site sEMG signals. IEEE Trans. Neural Syst. Rehabil. Eng. **18**(2), 203–209 (2010). https://doi.org/10.1109/TNSRE.2009.2039394
6. Pinheiro, C.G., Naves, E.L., Pino, P., Losson, E., Andrade, A.O., Bourhis, G.: Alternative communication systems for people with severe motor disabilities: a survey. Biomed. Eng. Online **10**(1), 1–28 (2011). https://doi.org/10.1186/1475-925X-10-31
7. Raynal, M., MacKenzie, I.S.: TBS^3: two-bar single-switch scanning for target selection. In: Miesenberger, K., Kouroupetroglou, G., Mavrou, K., Manduchi, R., Covarrubias Rodriguez, M., Penáz, P. (eds.) ICCHP-AAATE 2022. LNCS, vol. 13341, pp. 338–346. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-08648-9_39
8. Rodrigues, A.S., da Costa, V.K., Cardoso, R.C., Machado, M.B., Machado, M.B., Tavares, T.A.: Evaluation of a head-tracking pointing device for users with motor disabilities. In: Proceedings of PETRA 2017, pp. 156–162. ACM, New York (2017). https://doi.org/10.1145/3056540.3056552
9. Soukoreff, R.W., MacKenzie, I.S.: Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts' law research in HCI. Int. J. Hum. Comput Stud. **61**(6), 751–789 (2004). https://doi.org/10.1016/j.ijhcs.2004.09.001
10. Zhang, X., MacKenzie, I.S.: Evaluating eye tracking with ISO 9241 - part 9. In: Jacko, J.A. (ed.) HCI 2007. LNCS, vol. 4552, pp. 779–788. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73110-8_85