Controlled Experiment on Different Mouse Pointer Locating Techniques

Chufeng Hu
University of Waterloo
Waterloo, ON
c55hu@uwaterloo.ca

Yu Gu
University of Waterloo
Waterloo, ON
crispherg@gmail.com

ABSTRACT

Finding a "lost" mouse pointer is part of everyone's experience. Operating systems offer features to help with this issue, e.g. macOS has the "shake to locate cursor" option, while Windows offers the option to play a sonar wave animation around the pointer when Ctrl key is pressed. In this study, we designed an experiment to test whether some of these techniques are indeed helpful. We developed a web application and published it online. 12 participants finished our experiment. We compared the effect of two techniques: shaking-and-resizing (as in macOS) and Ctrl-and-resizing (pointer is magnified when pressing Ctrl key) with baseline setting (no technique available). In all three settings, we measured the time used to and recorded the trajectories of pointer. Our study shows that neither technique reduces the completion time of pointing tasks, despite the significant difference in pointer trajectories.

INTRODUCTION

Finding the tiny mouse pointer can be a challenging task, especially on nowadays super high resolution monitors, where losing the pointer happens from time to time. When using the computer, as the amount of information displayed increases users are more likely to find that locating a mouse pointer is a very common and frustrating experience[2]. The problem can be even worse when using multiple screens. As the total screen size increases, people move their mouse faster and set higher mouse pointer speed settings to quickly traverse the screen[14]. The faster the mouse moves, however, the more likely users are to lose track of it.

Several promising interaction techniques have been introduced to increase the accuracy of users' actions. In some implementations[12], the size of interface object or viewing region dynamically changes to provide users with a larger target area to interact with. The dynamic changes can enhance the accuracy of target acquisition, as well as notice users the rough location of the pointer. In macOS, the mouse pointer will be enlarged when it is moved back and forth quickly. (See figure 1). Windows also offers options to add additional animations around the mouse pointer to overcome this annoyance.

Figure 1: The techniques implemented in macOS and Windows. In macOS, the size of the mouse pointer will be enlarged when users move the pointer back and forth quickly. In Windows, a "Sonar" circle will be shown around the pointer

As software get more complex with increasing amount of content, we need an effective strategy to give visual feedback when users try to find the mouse pointer. This would improve the user experience and reduce the effort spent on designing user interface of software applications. While many existing research concerning "lost" mouse pointer aim at some special population[5][8], according to [**?**, 11], the problem exists universally no matter for those special population or for normal people.

Figure 1 shows two current implementations. Increasing the size of the mouse pointer may, however, result in spatial disorganization even though the pointer eventually shrinks to the normal size. Also, the mechanism that triggers the mouse pointer expansion by shaking may not be the optimal solution. Since some users may not find it intuitive to find the pointer by shaking it, or sometimes the shaking is not fast enough to trigger the expansion.

In this project, we implement the shaking-and-resizing technique inside a web application. In addition we implement Ctrl-and-resizing technique, where the same magnification is triggered by pressing Ctrl key. We design and conduct experiment to verify their effect on the time used to find the pointers.

RELATED WORK

Software features

macOS El Capitan introduced a relatively minor new feature called "shake to locate cursor". This technique temporarily

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

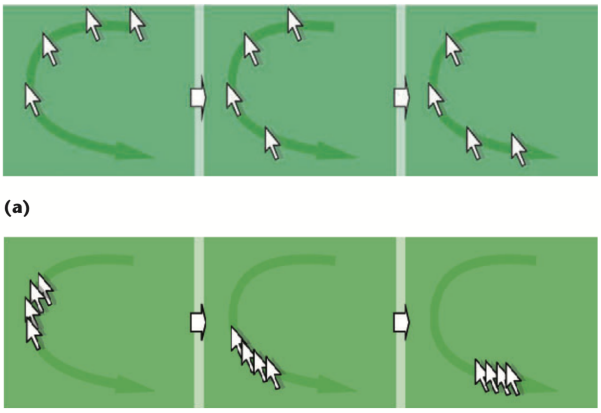


Figure 2: Figure 1 from [14]. High-density cursor versus mouse trail. Unlike the similar mouse trail technology (a), the high-density cursor (b) uses temporal super sampling to bridge gaps in the cursor‘s position without lag.

makes the user‘s mouse pointer much larger when shaken back and forth, making it easier to locate if the user loses track of it. This can be helpful for users who have large and high resolution displays. Although it is common for most of people to wiggle the pointer a bit to find it on the screen, some users find the zoom-in animation annoying and distracting, especially when they intend to rapidly move the mouse pointer in some particular applications (e.g. game, editing and drawing apps).

Windows also has a feature to help users find stationary mouse pointer. After user enabling the “auto-locator cursor”, an animation of circle appears over the cursor whenever the user press control key. Another implementation in Windows focuses on the situation that users lose mouse pointers during movement. [14] believes the key reason is that the mouse pointer is rendered only once per frame, which makes it visually jump from one rendering position to the next. To address this problem, Windows users can activate the “High-density cursor”, which fills the space between the mouse pointer’s current position and previous position with additional pointer images. Some users complain the “ghost” trails can be messy and sometimes even cause sudden confusion.

Finding the lost pointer

The most relevant works to our research problem include [8, 11]. Nic Hollinworth and Faustina Hwang [8] investigated the effect of three different techniques among two age groups. They compared the following techniques:

1. Touch centering (their proposed method): the mouse is relocated to the center of the screen when the users touches the mouse again. This is achieved with a special hardware that detects release and re-touch of the mouse.

2. Shaking: different from shaking-and-resizing discussed in our work, the “shaking” in [8] refers to simply shaking a pointer in normal environment, without resizing animation.

3. Windows sonar, as mentioned before.

The findings of [8] are that young people are better at detecting moving pointer and they find shaking (under normal condition, no technique available) is the most easy to use among the three. Among the older adults, shaking is more often rated as “difficult to use”.

Chuanyi Liu and Rui Zhao [11] studied the effect of different visual enhancements in helping people find the lost pointer. They compared the time people use to locate the pointers when pointers have different shapes and when the visual enhancement are different. In the following table, we list the different factors that they have taken into consideration in [11]:

Pointer shapes	E-resize, arrow, I-Beam, CrossHair and dot
Background	Simple (e.g. solid color), medium (e.g. landscape photo) and complex (e.g. doodle art)
Animation	Baseline (no enhancement), trailing, resizing, color change and others

The experiment of [11] was to divide the screen into different regions, each with a single-letter label. The pointer is hidden in one of these regions and visual effects are played to participants without interaction. By passively watching the screen, the participants try to identify the region the pointer was in and type the labeling letter. This experimental setup does not agree with daily scenario, where people usually move randomly to make the pointer more conspicuous.

There are also other works that discuss in general how to improve usability of pointer for low-vision users [5]. However, to the best of our knowledge, there is no work that investigates into the combined effect of shaking and resizing. [8] studied the effect of shaking, but resizing is unavailable under their setting; [11] discussed about resizing without user interaction. In our work, we compare the effects of different techniques in an experimental setting where participants can use the pointer interactively, as in their daily usage.

Multi-mornitor

The losing pointer problem can be even worse in multi-monitor environment. The use of multiple monitors is becoming popular as some people believe it can boost productivity, but this increases multitasking and puts strains on window management [12].

[1] presented a system that combines head tracking and mouse input to allow users to control GUI that spans multiple monitors, and the reason why they used head tracker is that eye tracking does not work well in the presence of larges in head position. Their results showed that the mouse movement is significantly decreased and users preferred their tracking system, although task time actually increased.

Another approach M^3 introduced in [4] virtually simulates having one mouse pointer per monitor when using a single physical mouse device. Whenever the user issues a frame switch command, M^3 warps the mouse pointer to the new frame with a “sonar circle” animation. Their study confirmed

that M^3 can improve the target acquisition performance, but there were no significant differences between movement time across techniques.

Similar to M^3, Microsoft Research proposed mouse ether [3], which solves the problem when user uses multiple monitors with different resolution and size. Mouse ether applies appropriate transformations to all mouse move events in order to eliminate the warping effects. According to their study, mouse ether can improve the target acquisition performance and therefore saves time for multiple monitor users.

Assistive pointers

Partially sighted people can not use graphic user interface without alteration[9]. Furthermore, according to the study of [6], partially sighted user will not choose to use devices that do not allow them to use their "residual visual processes". Therefore, the graphic user interface can present a significant barrier for low vision users. Some liturratures have already addressed this issue and proposed several implementations.

Fraser [5] proposed a framework of assistive pointers to fully articulate the design space of assistive pointers for low vision users. Their framework is organized into four dimensions: mode, stage, dependence, and pervasiveness. The framework is built to classify and organize existing solutions of assistive pointers. In their study, locating lost mouse pointer is describe in the second dimension: stage, which refers to three phases. They are presented in table 1. Although it is necessary to consider different assitance for each phase[5], we mainly focus on the phase of locating, as we believe that locating the pointer is the most challenging step in this process.

Phase	Definition
Locating	Users try to locate the lost pointer on screen
Moving	Users move mouse pointer to the target
Acquiring	User precisely place the pointer over the target

Table 1: Three phases organized in the framework of assistive pointers

The study of [15, 13, 10] showed that losing the mouse pointer is common among older people. There are a few studies considered the problem of older adult computer users such as the study of Nic and Faustina [7, 8], which has already been discussed in previous section.

EXPERIMENT

We conduct experiment to compare the effect of the following techniques:

- Shaking and resizing: the mouse pointer is temporarily magnified when the user shakes the mouse pointer around.
- Pressing Ctrl and resizing: the mouse pointer is temporarily magnified when the user presses the Ctrl key.
- Baseline: nothing happens when the user shakes the pointer or presses the Ctrl key.

Participants

All 12 participants are graduate students who are enrolled in the HCI course. They are all experienced users of mice and touchpads.

Implementation

The experiment is developed as a web application mainly written in JavaScript. Participants' states and records are maintained by React and Redux. Due to the concern of security, there is no API to manipulate the position of user's mouse pointer. In our implementation, we lock the actual mouse pointer and replace it with a dummy pointer. During each task, we track the trajectory of the dummy pointer in every 100 milliseconds. After the experiment, participants' records were sent to the server and stored with a unique id.

To emulate the shaking-and-resizing effect of macOS, we use a zigzag detection algorithm that detects sharp changes in the direction of pointer movements in the past 700ms. If the velocity vector has changed direction at least two times (determined by finding samples of v_t s.t. $v_{t_1} \cdot v_{t_2} < 0$), a zoom in animation of the pointer will be triggered. The duration of the zoom-in animation is 300ms, and the zoomed size of the pointer is 4x of the original. The pointer will shrink to its normal size in 1s of reaching the zoomed size, if zigzag detection has not detected other activation during the time period. The during of the zoom-out animation is 300ms, too.

In the Ctrl-and-resizing setting, the same zoom-in animation is triggered by pressing and hold Ctrl key, and on release of Ctrl key, the zoom-out animation is triggered.

We have made our code available online [16], readers may read into our code for other implementation detail.

Task Design

Task unit

To measure the time used to find the pointer, we design the following pointing task.

1. The participant is informed of the position of a target.
2. On hitting space key, a background image is unveiled, and the pointer will be relocated to some place that is irrelevant of its previous path.
3. The participant tries to find the pointer, and then click on the target shown to him in step 1.

The time from the unveiling of the background image to the successful click on the target is measured. It is the sum of two parts: the time used to find the pointer, and the time used to navigate to the target. Since there is no obvious way of extracting only the first part, we ask participants to complete the same pointing task in all three settings. Our assumption is that in all settings, the time used to navigate to the target should be the same.

Background images

In most daily situations, pointers are not hard to find. To simulate the scenarios where these techniques might be necessary, we choose background images that are overwhelmingly rich in detail, like those shown in Fig 3.

Figure 3: Screenshot after the start of a task. The pointer is hidden somewhere in the background. The target (pink) is near the center of the screen.

Starting and target positions

Since we need each participant to perform the same task in all three settings, it is necessary to have different starting positions for the same background image, otherwise the user may associate the image with his previous experience and have some prior knowledge when the same image appears the second or third time. In our experiment, we set four different starting positions for each image, so that when the user comes across the same image in a different setting, he still has no clue where the pointer might be hidden.

For target positions, we set only one target position for each image, target positions for all images are near the center of the screen. This is out of the concern that participant may spend extra time finding the target (despite he is shown the position of the target before starting).

All tasks

All tasks are grouped as shown in the table:

Settings (shuffled)	Tasks
Shaking	24 tasks
Ctrl-key	(6 images $\times$ 4 start positions
Baseline (no technique)	$\times$ 1 target position, shuffled)

Tasks in the same setting are completed in a contiguous block in order to avoid possible confusion.

Procedure

Our web application is published online [17] so every participant can do the experiment using their own computer whenever or wherever they feel like. Experiment data is collected upon completion and is written to our server.

Following the link, participants will first be shown a few slides, describing the context of the problem and the experiment set up. The participants are then directed to the three groups of tasks. The first task in the group is always an additional sample task. In the sample task, the participant must use a different way of starting the task: to click on a "Next" button below a brief description of available technique. This is to make sure that participant are aware of the changing settings. The participant will test out the technique in this sample task. The sample task is not timed and has no effect on the final result.

After completing all tasks, the participant will fill in a questionnaire before they leave.

RESULT

Triggering frequency

Although we asked our participants to use shaking or Ctrl-key in every task, we did not enforce this. Participants can proceed to next task without interruption if they do not trigger the available technique. As expected, techniques are not always triggered. Shaking-and-resizing is triggered in 54.2% of the times and Ctrl-and-resizing 84.7%.

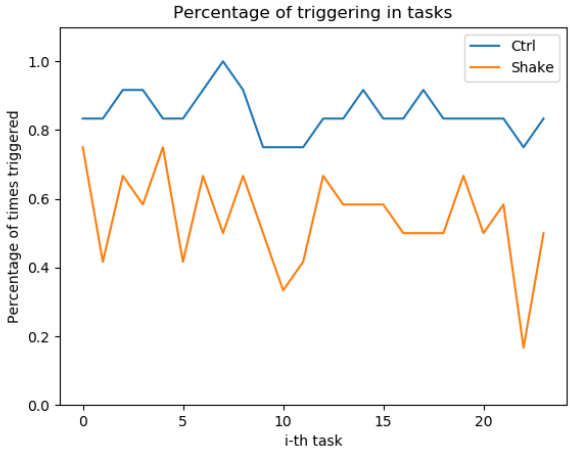


Figure 4: Triggering frequency vs. number of tasks finished.

For shaking-and-resizing, we also observed a decline in triggering frequency with respect to time. Participants are more likely to shake their mice in the first shaking task than in the last, as shown in Figure 4. This trend together with the overall lower triggering frequency suggest that shaking-and-resizing might requires more effort to trigger compared to pressing Ctrl key.

Time

We consider completion time as the main dependent measure and it is defined as the time from pressing the space key, to the successful click on the target. To eliminate influence of the familiarization with tasks, we randomly shuffled the order of three settings. As the participants are likely to find and move mouse pointer faster in later tasks.

According to the result of repeated-measures ANOVA, there is no significant effect of techniques on trial time ($F_{2,22} = 2.4908, p = 0.1059 > 0.05, \eta_p^2 = 0.18$). Pairwise comparison shows there is no significant difference between 3 settings (i.e. Ctrl, Shake, None).

The average completion time is 2011.134 ms for baseline setting, 2283.149 ms for Ctrl-and-resizing, and 2234.873 ms for

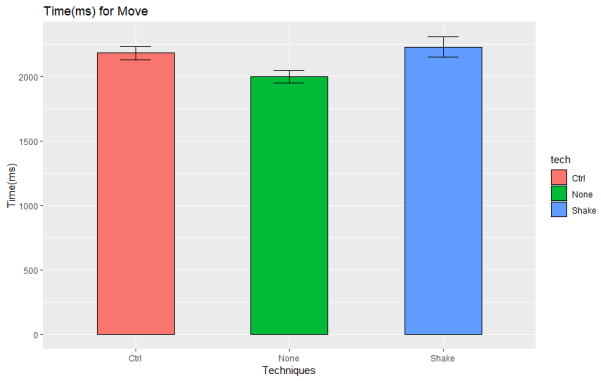


Figure 5: Average movement time for different settings, there is no significant difference between them

shaking-and-resizing. For each pair of settings, post-hoc analysis showed there is no significant difference between them. As a result, the hypothesis that techniques have significant impact to the completion time can not be validated through our experiment. (Figure 5)

Trajectory to target

We have recorded the trajectories of the pointer during all user experiments. By examining these trajectories, we get a few interesting observations.

Is shaking intuitive

Before conducting the experiment, we thought about a possible advantage of the shaking-and-resizing technique: shaking might be the natural move to do even if the user is not aware of the technique being available. User may trigger the magnification unexpectedly. If this is true, we expect to see some zigzag pattern in the trajectories when no technique is available.

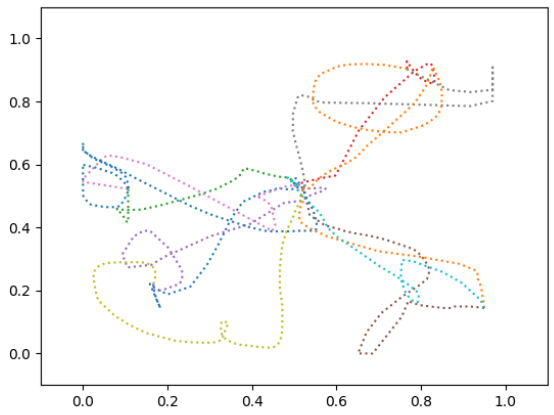


Figure 6: Sample pointer trajectories when no technique is available.

Figure 6 shows the trajectories of baseline tasks. We find only a few zigzag patterns when shaking-and-resizing is not available, and those moves are not likely to trigger the magnification.

Total moving distance

All pointer trajectories of three of the tasks that have the same background image are plotted in Figure 7. We see from these plots that trajectories in shaking-and-resizing setting are the most messy, and Ctrl-and-resizing seems to have most direct trajectories. We quantify this observation by measuring the ratio between distance and displacement:

$$\tau = \frac{\text{distance}}{\text{displacement}}$$

Repeated-measures ANOVA revealed a significant effect of techniques ($F_{2,22} = 16.7668, p < 0.05, \eta_p^2 = 0.6$) on τ. The average distance ratio is 1.89 for none locating enhancement, 1.55 for Ctrl-and-resizing, and 2.69 for shaking-and-resizing. According to the post-hoc analysis, the distance ratio of Ctrl-and-resizing is significantly ($p < 0.05$) less than the distance ratio of shaking-and-resizing.

Despite the longer travel distance, the average completion times are nearly equal, as discussed before. This may clear one doubt about shaking technique that the shaking adds overhead to the entire action therefore may increase the total time used.

By examining the replay of the mouse movement, we noticed that shaking can be done very quickly, and usually users start shaking without too much hesitation. Meanwhile in tasks where shaking is not available, there is not too much efficient mouse movement either.

User rating

In the questionnaire we asked our participants to rate how helpful these techniques are. We give the histogram plot in figure 9. In accordance with our statistics, users are mostly neutral on this question. An interesting observation is that while voting for shaking-and-resizing has a nice bell shape, the Ctrl-and-resizing receives a more bipolar perception. One possible explanation is that, participants often end up using both hands in Ctrl-and-resizing tasks, and this might be an important factor that divides people.

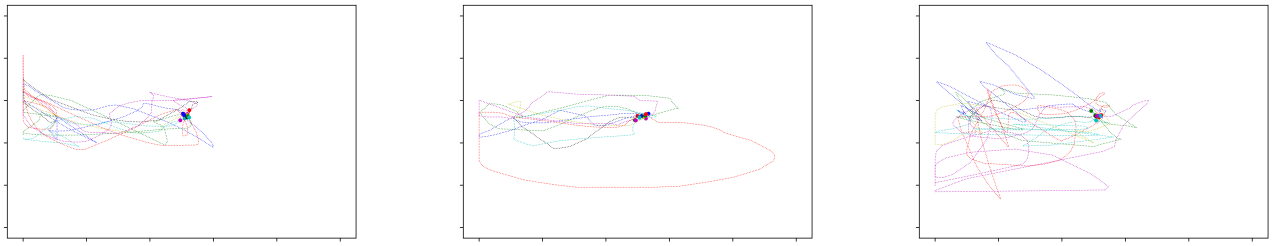
DISCUSSION

Tasks are too easy

Some participants has commented that they felt the tasks are too easy:

- *It was actually easy for me to locate the cursor without any magnification help.*
- *Personally, I found I didn't really need either technique to identify where my cursor was.*
- *Finding the cursor has a lot to do with motion and not necessarily size. I found that when using the technique, I could pinpoint the cursor based on its motion in most of the images presented.*

Although we used complex background images to simulate scenarios where participants can not find the mouse pointer,



(a) Baseline (no technique available) (b) Ctrl-and-resizing (c) Shaking-and-resizing

Figure 7: Trajectories of pointer under different settings.

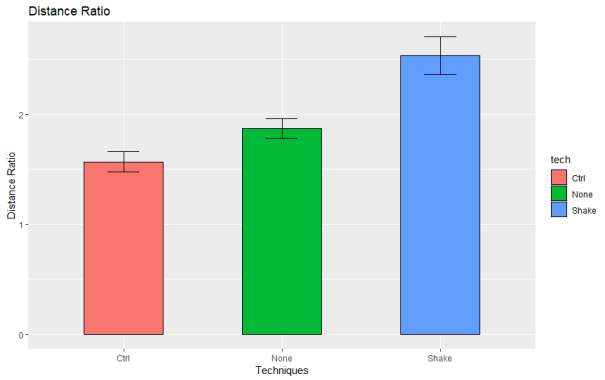


Figure 8: Average distance ratio (τ) for three techniques, ANOVA revealed that technique has significant effect on τ

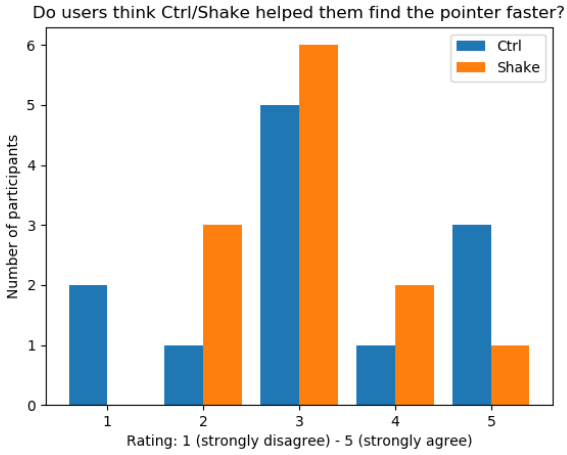


Figure 9: Distribution of votes for different techniques.

during the experiment, we observe that some participants can locate the pointer quickly. This agrees with the result of [8], that young people are in general very good at detecting moving object, and they usually find it is enough to have a normal pointer. To make tasks harder, we may use dynamic backgrounds (e.g. GIF, video) instead of stationary images, so that participants are less likely to find the pointer with simple movement. However, how often those scenario occur in daily life would be questionable.

Design flaw

Here we point out some flaws in our experiment we have noticed. We could have fixed them if we had more time and, more importantly, participants.

For each participant, we shuffled the order of 3 settings so that we can reduce the influence of familiarization. But due to the limited number of participants, our sample is unbalanced.

Before each task, we present the target with a light grey background so that participants are expected to locate the target without any effort. However, a few participants comment that they occasionally forgot the position of the target during the task. In our analysis, we exclude outliers that used very long time to click the target. To reduce this possibility, we may use different colors for targets in different images, so that there is always a significant contrast.

Future work

Shaking-and-resizing has the advantage of requiring no memorization of key combination, and can be done with one hand easily, however, our study suggests that to trigger by shaking is not as easy. One of the participants also suggests that it is hard to know how much he should shake the pointer. It may be worthy to invest time in tuning the triggering condition and animation effects.

In the implementation of shaking-and-resizing, we choose parameters (e.g. trigger condition, animation duration, zoomed size) based on our preference, as determining the optimal configuration for the shaking-and-resizing is not the concern of this project. Comparing different configurations for these two techniques remains interesting but is left as future work.

CONCLUSION

In this study, we conducted experiment to test whether shaking-and-resizing and Ctrl-and-resizing help people locate their mouse pointer faster. We find that there is no significant difference in the completion time of pointing tasks when technique is or is not available. We also studied the trajectories of pointers, and found the that the distance-displacement ratio is the largest for shaking-and-resizing and smallest for Ctrl-and-resizing, and the difference is statistically significant. Because our small sample size and the fact that all participants are young CS graduate students, it is hard to generalize our result to other groups.

REFERENCES

1. Mark Ashdown, Kenji Oka, and Yoichi Sato. 2005. Combining head tracking and mouse input for a GUI on multiple monitors. In *CHI'05 extended abstracts on Human factors in computing systems*. ACM, 1188–1191.
2. Robert Ball and Chris North. 2005. Effects of tiled high-resolution display on basic visualization and navigation tasks. In *CHI'05 extended abstracts on Human factors in computing systems*. ACM, 1196–1199.
3. Patrick Baudisch, Edward Cutrell, Ken Hinckley, and Robert Gruen. 2004. Mouse ether: accelerating the acquisition of targets across multi-monitor displays. In *CHI'04 extended abstracts on Human factors in computing systems*. ACM, 1379–1382.
4. Hrvoje Benko and Steven Feiner. 2005. Multi-monitor mouse. In *CHI'05 extended abstracts on Human factors in computing systems*. ACM, 1208–1211.
5. Julie Fraser and Carl Gutwin. 2000. A framework of assistive pointers for low vision users. In *Proceedings of the fourth international ACM conference on Assistive technologies*. ACM, 9–16.
6. Ephraim P Glinert and Bryant W York. 1992. Computers and people with disabilities. *Commun. ACM* 35, 5 (1992), 32–35.
7. Nic Hollinworth. 2010. Helping older adults locate'lost'cursors using FieldMouse. In *Proceedings of the 12th international ACM SIGACCESS conference on Computers and accessibility*. ACM, 315–316.
8. Nic Hollinworth and Faustina Hwang. 2011. Cursor relocation techniques to help older adults find'lost'cursors. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 863–866.
9. Julie A Jacko and Andrew Sears. 1998. Designing interfaces for an overlooked user group: considering the visual profiles of partially sighted users. In *Proceedings of the third international ACM conference on Assistive technologies*. ACM, 75–77.
10. Kerrie Laguna and Renée L Babcock. 1997. Computer anxiety in young and older adults: Implications for human-computer interactions in older populations. *Computers in human behavior* 13, 3 (1997), 317–326.
11. Chuanyi Liu and Rui Zhao. 2018. Find the "Lost" Cursor: A Comparative Experiment of Visually Enhanced Cursor Techniques. In *International Conference on Intelligent Computing*. Springer, 85–92.
12. Michael McGuffin and Ravin Balakrishnan. 2002. Acquisition of expanding targets. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 57–64.
13. Jessica Paradise, Shari Trewin, and Simeon Keates. 2005. Using pointing devices: difficulties encountered and strategies employed. (2005).
14. George Robertson, Mary Czerwinski, Patrick Baudisch, Brian Meyers, Daniel Robbins, Greg Smith, and Desney Tan. 2005. The large-display user experience. *IEEE computer graphics and applications* 25, 4 (2005), 44–51.
15. Nadine Vigouroux, Pierre Rumeau, Frédéric Vella, and Bruno Vellas. 2009. Studying point-select-drag interaction techniques for older people with cognitive impairment. In *International Conference on Universal Access in Human-Computer Interaction*. Springer, 422–428.
16. Chufeng Hu Yu Gu. 2018a. Pointer Locating Experiment. `https://github.com/ChesterHu/cursor-locating`. (2018). [Online].
17. Chufeng Hu Yu Gu. 2018b. Pointer Locating Experiment Application. `http://yugu233.com/cs889/cursor-locating/client/dist/index.html`. (2018). [Online].