Multi-Monitor Mouse

Hrvoje Benko **Steven Feiner**
Department of Computer Science, Columbia University
500 West 120th Street, 450 CS Building, New York, NY
{benko, feiner}@cs.columbia.edu

ABSTRACT
Multiple-monitor computer configurations significantly increase the distances that users must traverse with the mouse when interacting with existing applications, resulting in increased time and effort. We introduce the Multi-Monitor Mouse (M^3) technique, which virtually simulates having one mouse pointer per monitor when using a single physical mouse device. M^3 allows for conventional control of the mouse within each monitor's screen, while permitting immediate warping across monitors when desired to increase mouse traversal speed. We report the results of a user study in which we compared three implementations of M^3 and two cursor placement strategies. Our results suggest that using M^3 significantly increases interaction speed in a multi-monitor environment. All eight study participants strongly preferred M^3 to the regular mouse behavior.

Author Keywords
Multi-monitor, mouse pointer, interaction technique.

ACM Classification Keywords
H.5.2. [**User Interfaces**]: Graphical User Interfaces, Input Devices and Strategies.

INTRODUCTION
Increased display size and resolution and the proliferation of multiple-monitor display configurations have significantly extended the amount of desktop space available to computer users. However, increased desktop space forces users to move their mouse cursor over larger distances. To compensate, users can increase pointer speed or acceleration, which have drawbacks pointed out by Baudisch et al. [3]. In addition, tiling several displays in a row often results in one dimension (typically width) being drastically larger than the other, causing excessive "clutching" when traversing multiple displays with a mouse.

While the operating system considers the multi-monitor desktop as one seamless environment, previous research by Grudin [6] clearly suggests that users treat multi-monitor systems as means of partitioning their desktop space. This is primarily due to the physical gaps between monitors, but differences in resolution, size, and apparent mouse speed can also contribute towards this mental partitioning. As Grudin points out, users have a tendency to distribute tasks among monitors, treating them as separate, but connected, spaces, and only rarely do they straddle an application window across multiple physical displays.

CHI 2005, April 2–7, 2005, Portland, Oregon, USA.
ACM 1-59593-002-7/05/0004.

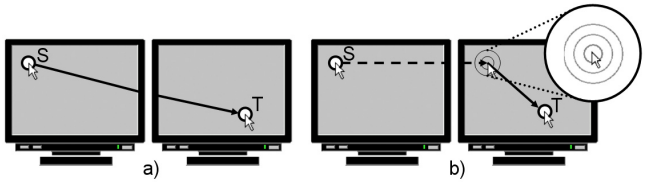

Figure 1: a) Using standard pointer movement to move between monitors from S to T. b) M^3 warps the cursor (dashed line) to the next screen, reducing the distance traversed by conventional mouse movement. "Sonar" circles are displayed to increase cursor visibility.

This research inspired us to create *Multi-Monitor Mouse* (M^3), a pointer interaction technique that warps the pointer between screens in a multi-monitor system configuration. M^3 simulates having one independent mouse pointer per screen, using a single physical mouse device.

RELATED WORK
Improving target acquisition across multiple monitors has been explored in context with eliminating warping effects caused by mismatched monitor alignment and differing screen resolutions with *mouse ether* [1], as well as avoiding the need to cross the bezels by bringing the targets closer to the current cursor location with *drag-and-pop* [2]. Baudisch et al. also proposed visual enhancements, such as *high-density cursor* [3], that increase visibility of cursors at high speeds. Interactions that warp the pointer closer to a target location have previously been explored on a single monitor in combination with eye gaze (e.g., [8] and *MAGIC pointing* [9]) or hand gestures (e.g., *flick* [5]). Zhai, Smith, and Selker compared one- and two-handed techniques for compounding tasks of scrolling and pointing [10].

MULTI-MONITOR MOUSE
We have implemented M^3 as a Windows application that runs in the background, minimized to the system tray. When M^3 is launched, it reads the system's information about the size, number and relative location of attached screens and forms a corresponding set of virtual frames to represent the screens. When the user issues a frame switch command, M^3 warps the mouse pointer to the new frame (screen). The new location of the cursor is signaled to the user by invoking the "mouse sonar" animation around the pointer (Figure 1b), a built-in Windows option that enhances pointer visibility. Otherwise, pointer movement is

completely unaffected by M³. (For example, the user is still free to move the pointer across screen boundaries by physically moving the mouse, but now has the option of directly warping to a different screen.)

M³ segments the pointer space according to screen space divisions, thus allowing for pointer warping across screens. While the techniques presented in this paper have been applied to switching between screens in a multi-monitor configuration, the same techniques, without modification, can be applied to any desktop space by dividing it into a set of virtual rectangular frames. These frames can be of arbitrary number and size, and can even overlap. For example, a large high-resolution monitor could be divided into several virtual frames, each containing the windows for one application. M³ would in this case switch the mouse pointer between different applications.

M³ FRAME SWITCH ALTERNATIVES

We have experimented with several switch designs. Two of the final four designs (*mouse button* and *keyboard* switches) require only standard computer peripherals, thus making them easy for most computer users to adopt. The other two (*head orientation* and *mouse location* switches) support more direct switching, but require extra equipment.

Mouse Button Switch

The *mouse button* switch command is issued by pressing one of the two side buttons (XButtons) on the five-button Microsoft IntelliMouse Explorer mouse. Since multi-monitor configurations are typically side-by-side arrangements, we decided to map the top side button to advance the frames forward (clockwise), and the bottom side button to advance the frames backward (counterclockwise). This decision is technically arbitrary, but we believe that it has ecological validity in that it mimics the behavior of the mouse itself when those side buttons are pushed: Pushing the top button would tend to rotate the whole mouse clockwise, while pushing the bottom button would have a counterclockwise effect. The virtual frames form a loop, making it possible to cycle through all the screens using just one of the buttons.

Keyboard Switch

The *keyboard* switch is modeled after a built-in Windows task-switch command (ALT+TAB). We mapped the forward frame switch to the ALT+"~" key combination, and backwards to ALT+SHIFT+"~". As with the mouse button switch, looping is supported and it is possible to use just one key combination to switch among all screens. This mode, although implemented, was not evaluated in our tests because of its similarity to mouse button switch.

Head Orientation Switch

By observing the work of several individuals in a multi-display environment, we noticed that a user's head position does not change much, but their head orientation changes continuously, depending upon the screen on which they are working. At a constant working distance from the user, the larger the screens, the larger the horizontal angle subtended by each screen that can be reliably measured with an absolute orientation sensor. We outfitted a pair of headphones with a 3DOF orientation sensor (InterSense InertiaCube2) and measure the user's head orientation to determine the screen at which they are looking. When the user turns their head towards another screen, the *head orientation* switch performs a frame switch.

While 6DOF tracking combined with eye-gaze tracking would reduce errors, we wanted to design a minimally invasive switch that would perform well with minimal calibration time. We have noticed that while eye gaze alone is sometimes used to glance at another screen, users tend to align their head with a monitor when performing tasks on its screen (especially with the larger, 24" diagonal, monitors we used), which supports our head orientation switch solution. In general, our technique is similar to MAGIC pointing [9], insofar as it warps the pointer to the area "in focus" (in our case, a screen), followed by fine selection by mouse movement alone. However, while MAGIC pointing warps the pointer within one screen whenever the eye gaze changes, we warp only across frame boundaries and leave all mouse manipulation within a single screen unchanged.

Mouse Location Switch

The last switching method we implemented is based on the idea that every screen could have a corresponding mouse-pad. The user still manipulates only one physical mouse, but physically placing the mouse on a different pad warps the cursor to the screen corresponding to that pad. We implemented *mouse location* switch using a touch-sensitive surface (MERL DiamondTouch table) on which the user can define any axis-aligned rectangle as a pad for a given screen. To aid the user in remembering the locations of the virtual mouse pads, we provided paper mouse-pad cutouts to be placed on the surface (Figure 2). Since the table operates through electrostatic coupling, the mouse is wrapped in aluminum foil to allow it to be tracked by the DiamondTouch surface when held by the user.

M³ POINTER-PLACEMENT STRATEGIES

In addition to deciding on how to trigger the frame switch, there are several possibilities for where to warp the mouse cursor in the target frame after the frame switch has oc-

Figure 2: M³ test setup consisting of four monitors and four corresponding "mouse pads" used by mouse location switch. The background is set to an inactive spreadsheet image to simulate a typical noisy working environment.

curred. After some preliminary experimentation, three strategies emerged as plausible candidates: *fixed location*, *frame-relative*, and *frame-dependent*.

Fixed-Location Placement Strategy

Our initial implementation of M^3 used the single *fixed-location* placement strategy of always warping the cursor to the center of the next frame (Figure 3a). While the center location is somewhat arbitrary, it does ensure that the maximum mouse traversal distance after the frame switch will always be at most half of the frame's diagonal. This can be beneficial if users distribute their tasks equally around the center, which is often the case with active working windows. However, it can be a nuisance when the target is located near an edge, which is the case for some frequent selection tasks, such as accessing the taskbar. In our current M^3 implementation, it is possible to select any fixed location as the warping target, as long as that location is available on all frames.

Frame-Relative Placement Strategy

Frame-relative placement works by translating the pointer to the next frame at the same location relative to the new frame's upper left corner as it was relative to its old frame's upper left corner (Figure 3b). This strategy essentially collapses the entire available space into one frame of mouse movement and is the only strategy we implemented in which the effect of pointer movement prior to the frame switch will not be negated by the switch itself.

Frame-Dependent Placement Strategy

If all frames are considered as completely independent spaces, the system can remember the last location of the cursor in each frame and warp the incoming cursor to that location. Thus, the last position of the cursor when the user warps out of the frame, becomes the starting location when the user eventually warps back to that frame (Figure 3c). This *frame-dependent* placement strategy, while preferred by some initial test users in the two-monitor setup, was not formally evaluated for four monitors, due to its increasing difficulty as the number of frames increases. This is presumably due to the memory load imposed on the user having to remember each frame's cursor position.

USER STUDY

Eight right-handed participants (6 male, 2 female, ages 23–32), all unfamiliar with the techniques, participated in a target-selection experiment, with a counterbalanced within-subject design. We decided to test regular unassisted mouse interaction (CTRL mode) with three M^3 frame-switch modes: mouse button (MB), head orientation (HEAD), and mouse location (ML). We tested each switch mode using two pointer-placement strategies: frame-relative (FR) and center fixed-location (C). This resulted in a total of seven different conditions. Each user was allowed to familiarize themselves with all mouse behaviors, and performed a block of 10 practice trials before completing the block for each condition. Each block consisted of five trials for each

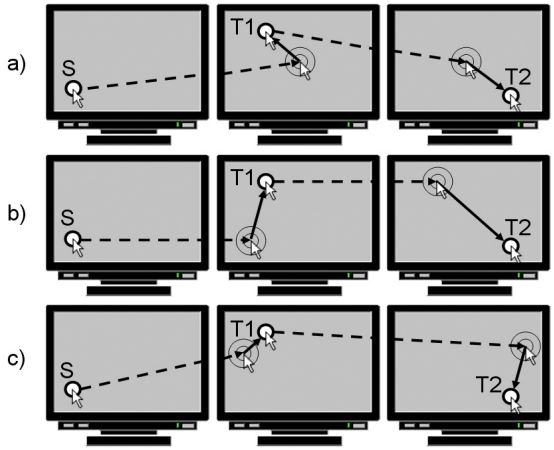

Figure 3: Traversing between S, T1 and T2 locations using different M^3 pointer placement strategies: a) fixed-location (center), b) frame-relative, c) one of many possible frame-dependent scenarios. Dashed lines indicate warping; solid lines indicate conventional movement.

of three different start-target distances and two directions (right and left) for a total of 30 movements per block.

Our hypothesis was that participants would acquire targets faster when using M^3 relative to the control mode as the number of screen bezels that needed to be crossed increased. In addition, we speculated that the users would be faster using the frame-relative strategy rather than the center fixed-location strategy because of the utility of movement prior to the frame switch in the former strategy.

The experiment was conducted on a Dual Xeon (2.6GHz, 2GB RAM) computer running Windows XP, with four monitors tiled in a horizontal arrangement, driven by two ATI Radeon 9800 and 9000 graphics cards. All four monitors were Samsung SyncMaster 240T (24" diagonal, 1920×1200 resolution, 60Hz refresh), for a total desktop space of 7680×1200 pixels. The monitors were arranged in a semicircle of 80cm radius, with 12cm horizontal separation (including bezels) between each monitor's display, and the user was seated in the center to ensure equal distance and viewing angle to all monitors (Figure 2). Thus, each display occupied a 35° horizontal viewing angle with 8° separation. The mouse speed was set to the medium setting.

The task was based on a Fitts' Law target acquisition task [7], but without any variation of start and target sizes (fixed at 30 pixels square). To eliminate target discovery overhead, we presented the participant with both start and target buttons at the same time, asked them to locate both before starting a trial, and recorded the time it took between clicking on the start button and clicking on the target button. We selected distances of 2134, 4047, and 5956 pixels, such that each required crossing one, two, or three bezels, respectively. To eliminate strategy bias, each target was located exactly halfway between the center of the screen and the frame-relative location of the start button on that target screen (Figure 4). The direction varied between left-to-right and right-to-left.

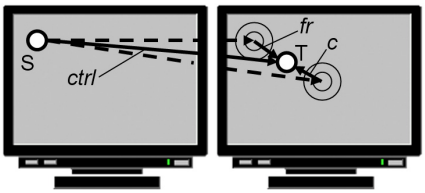

Figure 4: Start (S) and target (T) button layout in our user study. To eliminate strategy bias, T is located at the half-way point between the warping locations for frame-relative (*fr*) and center fixed-location (*c*) strategies. Control path (*ctrl*) is also shown.

Results

Movement times were first cleared by removing outliers (movement times more than two standard deviations larger than the mean for each condition), which accounted for less than 3.5% of all the trials. All data analysis was performed on a median movement time for each participant, distance, direction, and condition combination. We performed a 7 (Condition) × 2 (Direction) × 3 (Distance) ANOVA, with our subjects as a random variable. There were significant main effects for all three factors. The Direction factor contained a significant main effect, $F(2,14)=82.72$, $p<<0.001$: transitioning left-to-right was on average 0.2s faster than going right-to-left, which is consistent with previous research showing that right-handed users performed mouse movements faster from left-to-right than from right-to-left when using their right hand [4].

The mean movement times across Condition, $F(6,42)=3.88$, $p<0.01$, are shown in Figure 5(a). The interaction of Distance and Condition (Figure 5b) also had significant effects, $F(12,84)=9.509$, $p<<0.001$. As predicted, in all cases the FR strategy outperformed the C strategy by an average of 0.1s, which we believe is due to the C strategy discarding pointer movement prior to the frame switch. Of the M^3 modes using the FR strategy, MB (1584ms) was the fastest compared with CTRL (1906ms), presenting a 17% improvement in performance. MB was followed by HEAD (1698ms) and ML (1710ms), which were both significantly faster than CTRL. It is interesting to notice that for the shortest distance (2134 pixels) there were no significant differences between movement times across conditions. All of the M^3 performance gains come from time saved when traversing two bezels or more, with the biggest gains (up to 29%) being present at the largest distance (5965 pixels).

In the post-experiment questionnaire, all subjects strongly preferred some M^3 mode to the CTRL condition, and 6 out of 8 users preferred the FR strategy over the C strategy. While 7 out of 8 users preferred the MB mode over all the other modes, some mentioned that they would actually prefer a combination of HEAD and MB modes, in which the screen switch is triggered with the mouse button, but the cursor warps directly to the screen at which they are looking. This combination would resolve the "Midas touch" issue of pure HEAD mode, and also eliminate the multiple clicks necessary to switch across more than one screen.

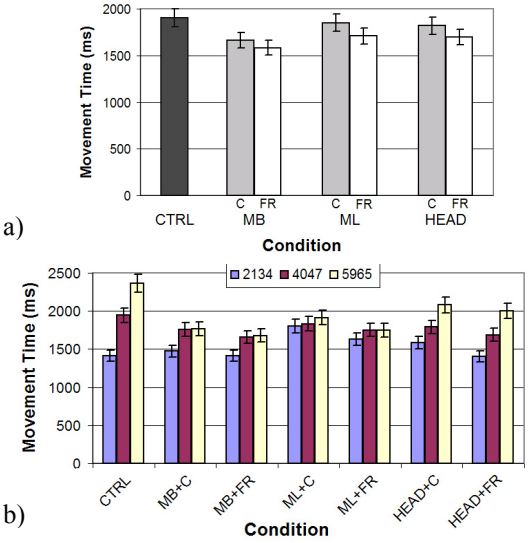

Figure 5: Aggregated movement times (ms) with 95% confidence intervals: a) Condition factor, b) Interaction of Distance and Condition.

CONCLUSIONS

Our user study confirmed that M^3 can improve target acquisition performance in multi-monitor systems; as predicted, the effect was higher with increased distance and FR strategy. Subjective evaluations show a strong preference for M^3 over regular mouse pointing, and we believe that the performance advantage could significantly increase with experience. In future work, we would like to evaluate additional M^3 modes and strategies, as well as implement a hybrid mouse button + head orientation switch mode.

ACKNOWLEDGMENTS

This work is funded in part by NSF Grant IIS-01-21239, ONR Contract N00014-04-1-0005, and a gift from Mitsubishi Electric Research Labs.

REFERENCES

1. Baudisch, P., Cutrell, E., Hinckley, K. and Gruen, R., Mouse Ether: Accelerating the Acquisition of Targets Across Multi-Monitor Displays. *CHI '04 Extended Abstracts*, ACM Press (2004), 1379–1382.
2. Baudisch, P., Cutrell, E., Robbins, D., Czerwinski, M., Tandler, P., Bederson, B. and Zierlinger, Z., Drag-and-Pop and Drag-and-Pick: Techniques for Accessing Remote Screen Content on Touch- and Pen-operated Systems. *Proc. INTERACT '03*, (2003), 57–64.
3. Baudisch, P., Cutrell, E. and Robertson, G., High-Density Cursor: A Visualization Technique that Helps Users Keep Track of Fast-Moving Mouse Cursors. *Proc. INTERACT '03*, ACM Press (2003), 236–243.
4. Boritz, J., Booth, K.S. and Cowan, W.B. Fitts's Law Studies of Directional Mouse Movement. *Proc. of Graphics Interface* (1991). 216–223.
5. Dulberg, M.S., Amant, R.S. and Zettlemoyer, L.S., An Imprecise Mouse Gesture for the Fast Activation of Controls. *Proc. INTERACT '99*, IOS Press (1999), 375–382.
6. Grudin, J., Partitioning Digital Worlds: Focal and Peripheral Awareness in Multiple Monitor Use. *Proc. CHI '01*, ACM Press (2001), 458–465.
7. MacKenzie, I.S. Fitts' Law as a Research and Design Tool in Human-Computer Interaction. *Human-Computer Interaction*, *7* (1992). 91–139.
8. Sibert, L.E. and Jacob, R.J.K., Evaluation of Eye Gaze Interaction. *Proc. CHI '00*, ACM Press (2000), 281–288.
9. Zhai, S., Morimoto, C. and Ihde, S., Manual and Gaze Input Cascaded (MAGIC) Pointing. *Proc. CHI '99*, ACM Press (1999), 246–253.
10. Zhai, S., Smith, B.A. and Selker, T., Dual Stream Input for Pointing and Scrolling. *CHI '97 Extended Abstracts*, ACM Press (1997).